A method for computer-aided symmetry detection in 3D CAD models

Burić, Mladen

University of Zagreb

Faculty of Mechanical Engineering and Naval Architecture

Mladen Burić

A METHOD FOR COMPUTER-AIDED SYMMETRY DETECTION IN 3D CAD MODELS

DOCTORAL THESIS

Zagreb, 2023

This page intentionally left blank.

University of Zagreb

Faculty of Mechanical Engineering and Naval Architecture

Mladen Burić

A METHOD FOR COMPUTER-AIDED SYMMETRY DETECTION IN 3D CAD MODELS

DOCTORAL THESIS

Supervisor:

Assoc. Prof. Stanko Škec, PhD

Zagreb, 2023

This page intentionally left blank.

Sveučilište u Zagrebu

Fakultet strojarstva i brodogradnje

Mladen Burić

METODA ZA RAČUNALNO POTPOMOGNUTU DETEKCIJU SIMETRIJE U 3D CAD MODELIMA

DOKTORSKI RAD

Mentor:

izv. prof. dr. sc. Stanko Škec

Zagreb, 2023.

This page intentionally left blank.

BIBLIOGRAPHY DATAS

UDC:	621
Keywords:	computer-aided symmetry detection, exact symmetry, partial symmetry, axisymmetry, reflectional symmetry, Computer-aided design (CAD), Boundary representation (B-rep)
Scientific area:	Technical sciences
Scientific field:	Mechanical engineering
Institution:	Faculty of Mechanical Engineering and Naval Architecture (FMENA), University of Zagreb
Supervisor:	Assoc. Prof. Stanko Škec, PhD
Number of pages:	261
Number of figures:	73
Number of tables:	23
Number of references:	151
Date of oral examination:	8th December 2023
Jury members:	Assoc. Prof. Stanko Škec, PhD, University of Zagreb, Croatia Prof. Nenad Bojčetić, PhD, University of Zagreb, Croatia Prof. Željko Ivandić, PhD, University of Slavonski Brod, Croatia
Archive:	FMENA University of Zagreb National and University Library in Zagreb

ACKNOWLEDGMENTS

I am grateful to my supervisor, Assoc. Prof. Stanko Škec, for providing me with the opportunity, guidance, and unwavering support throughout the research and the preparation of this doctoral thesis.

I would also like to express my gratitude to the Committee members, Prof. Nenad Bojčetić and Prof. Željko Ivandić, for their invaluable feedback and constructive criticism, which greatly contributed to the improvement of this dissertation.

Special thanks go to Assoc. Prof. Tina Bosner and Assist. Prof. Mario Brčić for their generous support and time dedicated to this project. I also thank all Chair of Design and Product Development members, particularly Prof. Emer. Dorian Marjanović. Special thanks to Prof. Zoran Lulić, who is always open to help.

Lastly, I want to acknowledge my family for their unwavering patience, understanding, and moral and financial support throughout my academic journey. My heartfelt thanks go to my wife, Monika, for her support and encouragement, and to my son, Krsto, whose presence always brings me joy.

ABOUT THE SUPERVISOR

Stanko Škec was born in Zagreb, Croatia. He holds PhD degree from the Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb (UNIZAG-FSB). He is currently an Associate Professor at the Chair of Design and Product Development at UNIZAG-FSB. He was appointed as a Visiting Assistant Professor (2018-2019) at the Engineering Systems Group, DTU – Technical University of Denmark, Lyngby, Denmark. Previously, he was a visiting researcher at several institutions abroad – Ecole Centrale Paris, the University of Bristol, and The University of British Columbia.

As a project team member, he previously participated in various domestic (i.e., two Croatian Science Foundation projects and one government-founded project) and international projects (i.e., one EUREKA project, five ERASMUS+ projects, and one HORIZON2020 projects). He also led the international ERASMUS+ project "E-learning Platform for Innovative Product Development" (http://www.elpid.org/).

Dr Škec has published his work in international journals (e.g., TFSC, AEI, Design Studies, Research in Engineering Design, Journal of Engineering Design) and peer-reviewed conferences (e.g., ICED, DESIGN, NordDesign). He acted as a reviewer for several international journals (e.g., Sustainability, IEEE Access, IEEE TEM, AI EDAM).

The primary research field and scientific focus of Dr Škec has been the multidisciplinary field of product-service systems design and development. His research interests are primarily focused on the management and monitoring of development processes and activities, as well as on studies of distributed work and virtual collaboration.

He has actively participated in the organisation of the biennial DESIGN series event since 2012, which regularly attracts more than 300 experts from more than 30 countries around the world. In addition, he was elected to serve as an Assistant Programme Chair of ICED17 (21st International Conference on Engineering Design), which was held at The University of British Columbia, Vancouver (http://iced17.org/).

Besides the scientific and professional work summarised above, Dr Škec is involved in teaching through UNIZAG-FSB undergraduate and graduate study programs in product development and design theories. As a part-time assistant lecturer, he held tutorials at the Polytechnics of Zagreb and the Faculty of Industrial Engineering Novo Mesto (Slovenia).

ABSTRACT

Geometrical symmetry represents a fundamental property employed in many aspects of mechanical engineering. Engineers typically rely on visual inspection to detect symmetry in 3D CAD models, but there is a requirement for computer-aided symmetry detection. While prior studies have focused on symmetry detection in 3D CAD models with analytic surfaces, there is a need for a method that can handle numeric surfaces as well. This doctoral thesis introduces a geometry-based computer-aided symmetry detection method for boundary representation 3D CAD models that utilise both analytic and numeric geometry. The method can detect global and partial reflectional symmetry and axisymmetry in single-part CAD models with one body and manifold geometry. It consists of six steps: (1) interpretation of the 3D CAD model, (2) analysis of the B-rep, (3) generation, (4) trimming, and (5) evaluation of planes of symmetry and axes of symmetry candidates, and (6) visualisation of detected planes of symmetry and axes of symmetry. The method is intended to be general and applicable to various input CAD models and solid modelling CAD systems. It has been implemented by proposing a computational environment using a CAD system's Application Programming Interface functionalities. The method and the computational environment were subjected to structural and performance validation using the Validation Square. The results indicate that the planes and axes of symmetry can be detected accurately and with linear time complexity.

Keywords:

symmetry detection, exact symmetry, partial symmetry, axisymmetry, reflection symmetry, Computer-Aided Design (CAD), Boundary representation (B-rep)

PROŠIRENI SAŽETAK

Geometrijska simetrija (u daljnjem tekstu simetrija) je svojstvo objekta koje postoji u prirodi i mnogim proizvodima koje je izradio čovjek. Simetrija je predmet istraživanja u područjima poput matematike, fizike, elektrotehnike, arhitekture, građevinarstva, biologije, te čak i filozofije. Iz perspektive strojarstva, mnogi dijelovi i sklopovi su simetrični kako bi zadovoljili zahtjeve funkcionalnosti i performansi. Na primjer, simetrija poboljšava stabilnost u sklopovima s rotirajućim dijelovima poput turbostrojeva i motora s unutarnjim izgaranjem. Tijekom montaže, mehanički dijelovi s višestrukom refleksijskom simetrijom obično zahtijevaju manje vremena za rukovanje i manje su podložni greškama pri sastavljanju uzrokovanim nepravilnom orijentacijom dijela. U numeričkoj analizi, simetrija se često koristi za smanjenje veličine računalnog modela, što posljedično rezultira smanjenjem računalnog zahtjeva same analize. U tehničkom crtanju simetrični dijelovi mogu se prikazati pola u presjeku, a pola u vanjskom pogledu, čime se pojednostavljuje i smanjuje broj potrebnih ortogonalnih projekcija na crtežu. Pored toga, kod kotiranja simetričnih dijelova ili značajki nije potrebno kotirati simetralu. Simetrija je korisna u proizvodnom procesu za određivanje ravnine razdvajanja kod štancanja i injekcijskog prešanja strojnih dijelova. Nadalje, iz perspektive oblikovanja pomoću računala, prisutnost ili nedostatak simetrije u pojedinačnim dijelovima ili sklopovima može biti važan čimbenik prilikom konstruiranja. Simetrično konstruirani dijelovi i (pod)sklopovi mogu znatno pojednostaviti proces oblikovanja pomoću računala. Na primjer, određene značajke ili cijeli geometrijski oblik trodimenzionalnog CAD (engl. Computer-aided design) modela, može se generirati značajkama zrcaljenja, kružnog ili linearnog uzorka. Drugi primjer je da se broj pojedinačnih zrcalno simetričnih dijelova u sklopnom CAD modelu može smanjiti zamjenom s jednim simetričnim dijelom. Na temelju navedenih primjera može se zaključiti da je simetrija važno svojstvo koje se vrlo često koristi u raznim područjima strojarstva, pa samim time i pri konstruiranju.

Tijekom oblikovanja pomoću računala, kod simetrično izrađenih CAD modela, nerijetko se javlja zahtjev za potvrdom postojanja određene vrste simetrije. Pri tome je sama informacija o simetriji vrlo rijetko izravno pohranjena u izvornim CAD modelima, dok u neutralnim formatima za razmjenu (npr. STEP, IGS, itd.) u ovome trenutku uopće nije podržana pohrana informacije o simetriji. U iznimnim slučajevima, informacija o

simetriji može biti neizravno pohranjena u izvornom CAD modelu. To se na primjer događa kada je konačni geometrijski oblik dobiven operacijom zrcaljenja u odnosu na ravninu simetrije. Zbog toga se u praksi još uvijek za otkrivanje informacije o simetriji uglavnom koristi vizualno prepoznavanje od strane inženjera. Međutim, vizualno prepoznavanje može biti zahtjevno i dugotrajno za složene geometrijske oblike ili pri analizi velikoga broja CAD modela pohranjenih u CAD repozitoriju. Isto tako, egzaktna simetrija u matematičkom smislu ne može se otkriti vizualnim prepoznavanjem niti u jednom CAD modelu, nego samo aproksimativna. Shodno tomu, drugi način za otkrivanje informacija o simetriji je upotrebom računalno potpomognute detekcije simetrije (RPDS), koja omogućuje automatsku identifikaciju ravnina, osi ili točaka simetrije u dvodimenzionalnim ili trodimenzionalnim digitalnim objektima. RPDS dobila je značajnu pozornost u strojarstvu, ali i u drugim područjima istraživanja poput matematike, računarstva, medicine, arhitekture i građevinarstva. U strojarstvu, RPDS je do sad korištena je u svrhu pronalaženja, kompresije i poravnanja CAD modela, oblikovanja proizvoda za montažu, te otkrivanje namjere konstruktora u skeniranim modelima tijekom procesa povratnog inženjerstva.

U matematičkom smislu, objekt je simetričan ako ostaje invarijantan u odnosu na određene geometrijske transformacije kao što su refleksija, rotacija, translacija ili njihova međusobna kombinacija (npr. diedarska simetrija je kombinacija refleksijske i rotacijske simetrije). Prema mjerilu, simetriju se može podijeliti na globalnu, parcijalnu (djelomičnu) i lokalnu. Globalna simetrija označava da je cijeli digitalni objekt simetričan, dok parcijalna simetrija upućuje na to da je digitalni objekt globalno simetričan, ali da postoje određeni lokalni dijelovi geometrije koji nisu simetrični. Konačno, lokalna simetrija označava da je relativno mali dio geometrije ili da su samo određeni dijelovi digitalnog objekta simetrični. Kao što je već ranije spomenuto, simetrija može biti egzaktna (točna) ili aproksimativna (približna). Za razlikovanje spomenutih dviju vrsta simetrije koristi se određena funkcija udaljenosti $d(M,T(M))$, koja mjeri udaljenost između geometrijskog oblika M i njegove transformacije $T(M)$. Funkcija udaljenosti se uspoređuje s određenom greškom ε, tako da je $d(M,T(M))<\varepsilon$. Ako je $\varepsilon\cong 0$ tada se simetrija može smatrati egzaktnom, u suprotnom aproksimativnom. Da bi se iz perspektive strojarstva neki CAD model smatrao egzaktno simetričnim, greška ε bi trebala biti reda veličine minimalnih tolerancija izrade strojnih dijelova, odnosno $\varepsilon=10^{-6}$.

Digitalni objekti koji su predmet istraživanja u okviru ovoga doktorskog rada jesu kruti CAD modeli s rubnim prikazom (engl. boundary representation, B-rep). Postojeći pristupi za RPDS u CAD modelima s rubnim prikazom uglavnom su ograničeni na analitičke površine (ravne, cilindrične, stožaste, sferne i toroidalne), iako isti često sadrže i numeričke površine (primjerice B-spline, ekstrudirane površine, zarotirane površine, itd.). Općenito se u kontekstu trodimenzionalnih digitalnih objekta pristupi RPDS-a temelje se na analizi geometrije ili prikaza. Pristup temeljen na analizi geometriji za detekciju simetrije koristi određena geometrijska svojstva digitalnog objekta (primjerice pozicije točaka u prostoru, zakrivljenost plohe, itd.), te se najčešće kao ulazni digitalni objekti koriste kruti CAD modeli, rešetkaste konstrukcije, voksel modeli, NURBS modeli, oblaci točaka, mrežasti modeli, itd. U nekim slučajevima vrši se dodatna obrada inicijalnog ulaznog digitalnog objekta, primjerice, inicijalni mrežni model se pretvara u voksel model ili oblak točaka. Pristup temeljen na geometriji omogućava detekciju aproksimativne i egzaktne simetrije. U pristupu temeljenom na pogledu, trodimenzionalni objekt se pretvara u dvodimenzionalni prikaz, kao na primjer, u sliku ili projekciju. Međutim, ovaj pristup ograničen je isključivo na otkrivanje aproksimativne simetrije što ga čini ne prikladnim za CAD modele s rubnim prikazom.

Nadalje, prema načinu detekcije ravnina ili osi simetrije u digitalnom objektu, postoje dva pristupa u RPDS-u: eksplicitni i implicitni. Kod eksplicitnog pristupa se ravnine ili osi simetrije u digitalnom objektu pronalaze direktno. Implicitni pristup podrazumijeva da se za dani ulazni digitalni objekt identificira određeni broj kandidata za ravnine i/ili osi simetrije. Potom se ti kandidati evaluiraju s obzirom na geometriju ulaznog digitalnog objekta u svrhu provjere predstavljaju li neki od kandidata u konačnici i prave ravnine ili osi simetrije. Kandidati za ravnine i osi simetrije identificiraju se na nekoliko načina, među kojima su analiza glavnih komponenti, uparivanjem određenih entiteta (na primjer točaka), analiza svojstava lokalnih površina, inkrementalna rotacija oko centroida, itd. Neke od prethodnih RPDS studija ograničene su isključivo na otkrivanje ravnine i/ili osi simetrije koje prolaze kroz neku referentnu točku poput ishodišta, težišta, ili središta mase, što je prikladno za detekciju egzaktne, ali ne i aproksimativne simetrije.

Kad je riječ o detekciji simetrije u krutim CAD modelima, one je proučavana iz dva aspekta: temeljem značajki ili rubnog prikaza. Prvi način koristi značajke modeliranja, Booleove operacije i povijesti modeliranja za detekciju egzaktne simetrije u pojedinačnim

dijelovima i sklopovima. Međutim, taj aspekt ograničen je na izvorne CAD modele koji su vezani uz neki CAD sustav, te može biti osjetljiv na određene loše navike konstruktora koje prakticira prilikom modeliranja (na primjer modeliranje korištenjem viška značajki ili modeliranje simetričnih oblika korištenjem značajki koje nisu izvedene simetrično). Drugi aspekt za ulaz koristi geometrijske i topološke informacije rubnog prikaza, što omogućuje upotrebu izvornih CAD modela kao i neutralnih formata za razmjenu. U svrhu otkrivanja globalne refleksijske i osne simetrije, neke studije predlažu pristup koji se temelji na petljama (zatvoreni krug rubova koji okružuju plohu). Kandidati za ravnine i osi simetrije dobiveni su kao rezultantni vektor dvaju jediničnih normalnih vektora iz identičnih parova petlji, dok su stvarne ravnine i osi simetrije dobivene međusobnom usporedbom kandidata prema kumulativnim površinama petlji. U drugom istraživanju se predlaže tzv. „podjeli pa vladaj“ pristup, koristeći kao ulaz plohe za otkrivanje egzaktne globalne i parcijalne refleksijske i osne simetrije. U prvoj fazi, lokalni kandidati za ravnine i osi simetrije dobivaju se preko lokalnih simetričnih svojstava ploha i njihovih sjecišta, kako bi se u drugoj fazi podudaranjem lokalnih kandidata utvrdila globalna simetrija. Druga studija predlaže pristup za otkrivanje cikličkih područja u parcijalno osno simetričnim modelima koristeći unaprijed dodijeljenu os simetrije za ulaz. Nadalje, u jednoj drugoj studiji korišten je pristup temeljen na grafu za izdvajanje simetričnih područja na različitim geometrijskim mjerilima. Predloženi pristup uključuje detekciju egzaktne refleksijske, rotacijske i translacijske simetrije.

Općenito, postojeće studije RPDS-u u CAD modelima s rubnim prikazom imaju nekoliko nedostataka: (1) većina studija je ograničena na analitičke površine, tj. ravne, cilindrične, stožaste, sferne i toroidalne površine, iako kruti CAD modeli često sadrže i numeričke površine, (2) iako su neke RPDS studije uključivale i numeričke površine (B-spline), one su razmatrane samo u kontekstu refleksijske i cikličke simetrije. (2) Stoga su potrebna daljnja istraživanja u tom području kako bi se uključili i drugi tipovi numeričkih površina koji se javljaju u CAD modelima, i (3) parcijalna simetrija je još uvijek nedovoljno istražena te postoji potreba za predlaganjem određene metrike u svrhu detekcije egzaktne globalne, parcijalne i nesimetrije u CAD modelima.

U kontekstu spomenutih nedostataka u prethodnim istraživanjima, glavni cilj ovoga doktorskog rada je predložiti metodu i razviti računalno okruženje za detekciju egzaktne globalne i parcijalne refleksijske i osne simetrije u CAD modelima s rubnim prikazom

koji sadrže analitičke i numeričke površine. Istraživanje je ograničeno na krute CAD modele s geometrijom mnogostrukosti, kao i na pojedinačne dijelove sa samo jednim tijelom. Shodno tomu, hipoteza istraživanja glasi: Primjenom metode temeljene na geometriji moguće je detektirati t egzaktnu globalnu i parcijalnu refleksijsku i osnu simetriju u CAD modelima s rubnim prikazom koji sadrže analitičke i numeričke površine. Očekivani znanstveni doprinosi istraživanja je: 1) Metoda temeljena na geometriji za detekciju simetrije u CAD modelima koji sadrže analitičke i numeričke površine, i 2) Računalno okruženje za detekciju simetrije u CAD modelima koji sadrže analitičke i numeričke površine.

Predložena metoda za RPDS koristi rubni prikaz kao ulaz za detekciju simetrije, dok kao rezultat pruža informaciju o prisutnosti egzaktne globalne ili parcijalne simetrije, te refleksijske ili osne simetrije. Korištenje rubnog prikaza kao ulaz predstavlja općeniti pristup primjenjiv i proširiv na različite CAD sustave i formate datoteka. Predložena metoda sastoji se od šest koraka: (1) interpretacija 3D CAD modela, (2) analiza rubnog prikaza, (3) generiranje kandidata za ravnine i osi simetrije, (4) reduciranje kandidata za ravnine i osi simetrije, (5) evaluacija kandidata za ravnine i osi simetrije i (6) vizualizacija otkrivenih stvarnih ravnina i osi simetrija.

Inicijalno je potrebno CAD model interpretirati pomoću odgovarajućeg CAD sustava. Nakon interpretacije, rubni prikaz se podvrgava analizi, koja uključuje klasifikaciju topoloških elemenata (ploha i rubova) prema njihovim osnovnim geometrijskim elementima (vrstama površina i krivulja). Temelj predložene metode detekcije simetrije su plohe, koje kao što je ranije spomenute uključuju i analitičke, također uključuju i numeričke vrste površina (B-spline, zarotirane plohe, itd.). Pri tome RPDS metoda nije ograničena na određene vrste numerički površina, nego zbog svoje općenitosti omogućuje prilagodbu prema vrstama površina koje su specifične za određeni CAD sustav ili CAD format. Osim klasifikacije, u koraku analize rubnog prikaza provodi se izračun specifičnih svojstava ploha. U tu svrhu, svaka ploha se predstavlja jedinstvenom točkom (centroidom ili njegovom projekcijom na plohu), a odgovarajući jedinični vektor normale ili jedinični vektor osi izračunava se u spomenuti točkama. Predložena metoda oslanja se na implicitni (indirektni) pristup detekcije simetrije koji podrazumijeva generiranje određenog skupa kandidata za ravnine i osi simetrije, te njihovu evaluaciju u svrhu detekcije stvarnih ravnina i/ili osi simetrije. Kandidati se u okviru predložene metode

generiraju iz glavnih osi inercije (po tri kandidata za ravnine i za osi simetrije), iz parova sličnih ploha, te iz pojedinačnih ploha. Kandidati generirani iz glavnih osi inercije pokrivaju detekciju potencijalno egzaktne globalne simetrije u CAD modelu. Međutim, ti kandidati nisu dovoljni za detekciju ravnina i osi simetrije koji se ne poklapaju s glavnim osim inercije. Primjerice, CAD model može biti višestruko egzaktno globalno ili parcijalno reflektivno simetričan. Da bi se pokrile te vrste simetrije, kandidati se također generiraju i iz parova sličnih ploha, a u uparuju se samo plohe istoga tipa. Metrika koji se koristi za utvrđivanje sličnih ploha je kosinusova sličnost, za čiji se izračun upotrebljavaju rubovi ploha. Svakom rubu se dodjeljuje posebni kôd koji sadrži informacije o vrsti petlji kojoj pripada (vanjskoj ili unutarnjoj), vrsti ruba (linija, kružnica, elipsa, B-spline, itd.), te duljina ruba. Svaki par ploha koji ima kosinusovu sličnost jednaku 1, njegove plohe smatraju se identičnim, a sličnima ako je ispod jedan, a iznad određene granične vrijednosti (npr. 0,7). U konačnici se iz parova simetričnih ploha generiraju kandidati za ravninu simetrije (položaj kandidata ravnine dobiva se iz središnja točke između ploha, dok orijentacija preko jediničnih normala ploha). Posljednja skupina kandidata generira se iz pojedinačnih ploha (cilindričnih, stožastih, itd.) i odnosi se na kandidate za osnu simetriju čime se pokriva eventualno postojanje parcijalne osne simetriju u CAD modelu. Orijentacija kandidata osi dobiva se iz jediničnog vektora osi plohe, a položaj je definiran točkom na osi plohi. Generirani kandidati se prosljeđuju sljedećem koraku.

U sljedećem koraku se reducira broj kandidata s ciljem uklanjanja duplikata i neprikladnih kandidata koji su značajno udaljeni od centra gravitacije CAD modela. Svaki od preostalih kandidata za ravnine i osi simetrije evaluira se pomoću vektorskog računa koristeći pri tome specifična svojstva ploha iz koraka analize rubnog prikaza. Ako proces evaluacije pokaže da određeni kandidat za ravninu ili os simetrije doista predstavlja i stvarnu ravninu ili os simetrije, isti se vizualizira u CAD modelu kako bi se inženjeru pružila informacija o simetriji. U okviru studije predložene su dvije metrike za evaluaciju kandidata za ravnine ili osi simetrije u CAD modelima s rubnim prikazom, SFI i GSI. SFI mjeri simetriju topologije, dok GSI mjeri simetriju geometrije u CAD modelu. Pri tome, GSI predstavlja mjeru za definiranje granica između egzaktne globalne, parcijalne simetrije i asimetrije.

U okviru doktorskog rada, predložena metoda je implementirana unutar komercijalnog CAD sustava koristeći funkcionalnosti njegovog aplikacijskog programskog sučelja, a

kao rezultat toga razvijeno je računalno okruženje. Računalno okruženje se sastoji od grafičkog sučelja, koje omogućuje upravljanje procesom detekcije simetrije kao i post procesuiranje i vizualizaciju simetričnih i nesimetričnih ploha. Također je dokazano da se, zbog svoje općenitosti, metoda može implementirati i u drugim CAD sustavim koji pružaju slične funkcionalnosti aplikacijskog programskog sučelja.

Predložena metoda i računalno okruženje za RPDS podvrgnuti su validaciji upotrebom *Validation square* metode, koja se sastoji od strukturne validacije (koja je kvalitativna) i validacije performansi (koja je kvantitativna). Sama validacija provedena je na reprezentativnom skupu od 1100 CAD modela. Na temelju validacije zaključeno je da metoda omogućuje detekciju simetrije na različitim formatima ulaznih CAD modela (izvorni, Parasolid i STEP). Također, metoda se pokazala robusnom s obzirom na interpretaciju CAD modela (koji mogu poticati iz različitih CAD sustava), što može imati ima minimalan lokalni utjecaj na rezultate detekcije simetrije. Na primjer, u rijetkim situacijama metoda ne može detektirati određene simetrične parove ploha ili samo-simetričnih ploha zbog numeričkih grešaka koje nisu unutar $\varepsilon=10^{-6}$. Nadalje, validacija je otkrila da je predložena metoda omogućuje detekciju simetrije uz visoku točnost od 87% te linearnom računalnom složenošću. Konačno, temeljem validacije, predložena RPDS metoda pokazuje mogućnost proširenja na CAD modele bez mnogostrukosti, pojedinačne CAD modele s višestrukim tijelima, sklopne CAD modele, druge CAD formate (ACIS i IGES), pa čak i druge trodimenzionalne digitalne objekte kao što su rešetkaste konstrukcije. Na temelju svega toga može se zaključiti da je potvrđena hipoteza da se metoda temeljena na geometriji može koristiti za otkrivanje egzaktne globalne i parcijalne refleksijske i osne simetrije u CAD modelima s rubni prikazom koji sadrže analitičke i numeričke površine. Isto tako, predložena RPDS metoda i implementacija metode u obliku razvoja računalnog okruženja, potvrđuju da su ostvareni očekivane znanstveni doprinosi postavljene u okviru ovoga doktorskoga rada.

Iako predložena metoda za RPDS predstavlja unaprjeđenje u odnosu postojeće studije, nekoliko je budućih smjerova istraživanja. Prije svega, metoda je ograničena na refleksijsku i osnu simetriju. Kod mehanički dijelova su prisutne i druge vrste simetrije, poput cikličke i diedarske simetrije. Trenutno je ciklička simetrija u kontekstu CAD s modela rubnim prikazom još uvijek nedovoljno istražena, te bi buduća istraživanja mogla ići u smjeru proširenja mogućnosti metode na detekciju cikličke simetrije. Osim toga,

metoda bi se mogla poboljšati uvođenjem detekcije lokalne simetrije. Predložena RPDS metoda pokazuje obećavajuće rezultate kada je u pitanju njezino proširenje na CAD modele bez mnogostrukosti, pojedinačne dijelove s višestrukim tijelima, sklopne CAD modele, druge CAD formate (ACIS i IGES), CAD modela sastavljenih predominantno od numeričkih ploha, pa čak i druge trodimenzionalne digitalne objekte kao što su rešetkaste konstrukcije. Iako metoda uključuje numeričke površine, potreban je daljnji nastavak istraživanja vezan uz analizu CAD modela sastavljenih u potpunosti od numeričke geometrije (primjerice karoserija automobila, trup zrakoplova, itd.). U tom kontekstu, dan je prijedlog poboljšanja inicijalno predložene metode. Na posljetku, buduća istraživanja bi trebala uključivati unaprjeđenje računalnog okruženja u svrhu razvoja samostalne aplikacije, kako bi detekcija simetrije postala neovisna o CAD sustavu.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACIS	Advanced Computerized Implementation of Standards
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
AAOS(s)	Actual axis(es) of symmetry
APOS(s)	Actual plane(s) of symmetry
AOS(s)	Axis(es) of symmetry
AOSC(s)	Axis(es) of symmetry candidate(s)
API	Application Programming Interface
B-rep	Boundary Representation
CAD	Computer-aided design
CAE	Computer-aided engineering
CAM	Computer-aided manufacturing
CAPP	Computer-aided process planning
CASD	Computer-aided symmetry detection
CAx	Computer-aided x
CGM	Convergence Geometric Modeler
COG	Centre of gravity
CSG	Constructive Solid Geometry
DMSC	Digital Metrology Standards Consortium
FC	Face class
FN	False negative
FP	False positive
FEA	Finite element analysis
GUI	Graphical user interface
IGES	Initial Graphics Exchange Specification
ISO	International Standard Organization
JT	Jupiter Tessellation
MBD	Model Based Definition
PDF	Portable Document Format
POS(s)	Plane(s) of symmetry
POSC(s)	Plane(s) of symmetry candidate(s)

PMI	Product Manufacturing Information
STEP	Standard for the Exchange of Product model data
TN	True negative
TP	True positive
VB	Visual Basic
VBA	Visual Basic for Application
PC	Pairwise comparison
QIF	Quality Information Frameworks
3D	Three-dimensional
2D	Two-dimensional

NOMENCLATURE

$\mathbf{a}_A$	[-]	AOSC orientation vector
a	[-]	component of the principal axis
AR	[-]	area ratio
A	[-]	point on POSC
A	[-]	binary feature vector
b	[-]	component of the principal axis
b	[-]	vector between the COG & a point A on the AOSC
B	[-]	binary feature vector
BBC	[-]	Braun-Blanquet coefficient
c	[-]	component of the principal axis
CS	[-]	Cosine similarity
CS(F_1,F_2)	[-]	Cosine similarity between 1st and 2nd face
d	[-]	degree
JI	[-]	Jaccard index
f	[-]	face vector (normal or axis vector)
$\mathbf{F}_1$	[-]	binary feature vector of 1st face
$\mathbf{F}_2$	[-]	binary feature vector of 2st face
f_p	[-]	symmetric face pairs
f_c	[-]	individual faces
GSI	[-]	Global symmetry index
GSI_E	[-]	Global symmetry index for edges
DOS	[-]	Degree of Symmetry
d_1	[-]	u degree
d_2	[-]	v degree
d_A	[m]	point-to-line distance between AOSC and COG
d_M	[m]	point-to-plane distance of between POSC and COG
$d_{C,k}$	[m]	point-to-plane distance POSC and stand-alone face
$\mathbf{I}_P$	[kg/m^2]	inertia tensor
K_1	[-]	upper index on u control points,
K_2	[-]	upper index on v control points
L	[kgm^2/s]	angular momentum vector
M	[mm]	midpoint
n	[-]	unit normal vector to the face at some point

$\mathbf{N}$	[-]	normal vector to the face at some point
$\mathbf{n}_{\mathrm{P}}$	[-]	unit normal vector to the POSC
n_{F}	[-]	number of faces in the CAD model
n_{E}	[-]	number of edges in the CAD model
n_{L}	[-]	number of loops in the CAD model
n_{FC}	[-]	number of faces in a face class
n_{FP}	[-]	number of symmetrical face pairs
n_{SF}	[-]	number of symmetrical stand-alone faces
n_{SP}	[-]	number of symmetrical points
n_{TP}	[-]	number of total sampled points
$N_i^{d_1}(u)$	[-]	B-spline basis functions
$N_i^{d_2}(v)$	[-]	B-spline basis functions
$\mathbf{P}$	[-]	point
$\mathbf{p}$	[-]	cross-product vector (POSC)
$\mathbf{q}$	[-]	cross-product vector (AOSC)
P_i	[-]	control points
P_{ij}	[-]	control points
R	[m]	radius
$\mathbf{r}$	[-]	position vector of a face
$\mathbf{r}_i$	[-]	position vector of the i-th face
$\mathbf{r}_j$	[-]	position vector of the j-th face
$\mathbf{s}_{ij}$	[-]	resultant vector of two subtracted position vectors
$\mathbf{t}_i$	[-]	position vector of the i-th face
$\mathbf{t}_j$	[-]	position vector of the j-th face
SFI	[-]	Symmetrical faces index
SLI	[-]	Symmetrical lines index
w_i	[-]	weights (curve)
w_{ij}	[-]	weights (surface)
S	[-]	average position of sampled points
SD	[-]	symmetry distance
$S(x,y)$	[-]	implicit equation of analytic curve
$S(x,y,z)$	[-]	implicit equation of analytic surface
SDC	[-]	Sørensen–Dice coefficient
SSC	[-]	Szymkiewicz–Simpson coefficient

$\mathbf{V}$	[-]	direction vector
$\mathbf{v}$	[-]	orientation vector
α	[º]	half-angle of conical surface
β	[-]	coefficient of symmetry
γ_u	[º]	periodicity angle of the face in u direction
γ_v	[º]	periodicity angle of the face in v direction
ε	[-]	computation error for CASD
ε_A	[-]	computational error for surface area
$\boldsymbol{\sigma}(u,v)$	[-]	parametric surface
$\boldsymbol{\lambda}(u)$	[-]	parametric curve
$\boldsymbol{\Omega}$	[-]	velocity vector
ω_{ij}	[-]	weights (surface)
ΔL_x	[m]	minimum bounding box length in x direction
ΔL_y	[m]	minimum bounding box length in y direction
ΔL_z	[m]	minimum bounding box length in z direction
δ_{max}	[m]	max. allowable distance of POSC or AOSC from COG

1 INTRODUCTION

This introduction chapter provides general information about symmetry, its significance, and various types of symmetries in mechanical engineering. The reader is introduced to the motivation for investigating symmetry detection and is provided with a concise overview of the thesis's research objective, hypothesis, methodology, and anticipated scientific contributions.

Geometric symmetry (hereinafter denoted as symmetry) is a property that exists in a wide range of natural and man-made objects. This property is present in the simplest of molecules [1], in human and biological organisms [2], and in complex galaxies in the universe [3]. In particular, symmetry is an integral part of science and engineering. It was the scope of research in biology [2], mathematics [4], physics [5], and even philosophy [6]. Many structures exhibit symmetry in architecture and civil engineering for aesthetic and functional reasons [7]. The symmetry of a lattice steel structure may have a distinct influence on its static and kinematic behaviour [8]. In electrical engineering, the symmetry of an electronic system can improve its robustness and reliability against noise, interference, and failure [9]. Further, symmetry can be utilised in robotics for visual object recognition [10].

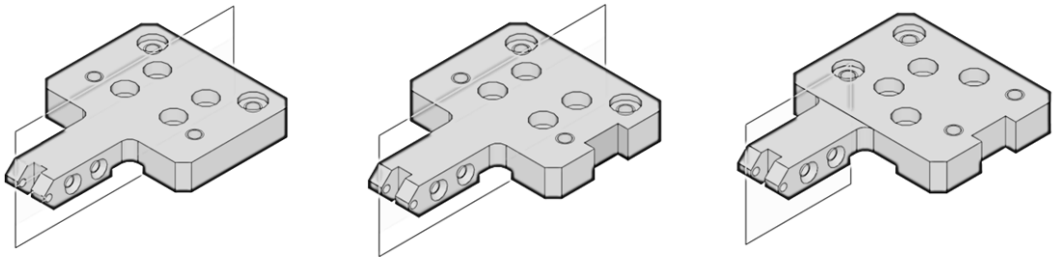
From the perspective of mechanical engineering, many parts and assemblies are symmetrical to meet functional and performance requirements [11]. For instance, it enhances the stability and balance of assemblies with rotating parts, such as turbomachines and internal combustion engines. Technical drawings can depict symmetrical parts by illustrating half in the section and half in the outside view, reducing the number of necessary orthogonal projections [12]. Additionally, dimensioning is not required for the line of symmetry of symmetric parts and features [13]. Computer-aided engineering (CAE) often utilises symmetry to reduce the size of the analysis model, which decreases computational demand and increases analysis accuracy [14,15]. Symmetry is also useful in manufacturing to determine the parting planes of mechanical parts in the stamping and moulding processes [16]. During assembling, mechanical parts with multiple reflectional symmetries require less handling time and are less likely to face assembly errors due to incorrect orientation [17]. Moreover, the symmetry of assemblies

can be used to build design knowledge and guide its application in engineering design [18].

From an engineering design standpoint, the presence or absence of symmetry in parts or assemblies can play a crucial role. Incorporating symmetry into the design of parts or (sub)assemblies can significantly streamline the computer-aided design (CAD) process. This is because certain features or the entire geometric shape of the part's CAD model can be mirrored or patterned (linear, circular, or translational). Additionally, replacing two mirrored parts with one symmetrical part can reduce the number of individual parts in the assembly CAD model [19]. Symmetry is an essential property extensively utilised in various fields, including engineering design, as evidenced by the examples.

In mathematical terms, an object is symmetrical if it remains unchanged (invariant) after undergoing geometric transformations such as reflection, rotation, translation, or their combinations [20,21]. Dihedral symmetry, for instance, involves both reflection and rotation transformations. Symmetry is categorised as either exact (perfect) or approximate (imperfect), depending on its level of accuracy. A distance function $d(M,T(M))$ is utilised to distinguish between the two. This function measures the distance between the geometric shape M and its transformation $T(M)$ and is evaluated within a specific computation error ε, such that $d(M,T(M))<\varepsilon$ [22]. When $\varepsilon\cong 0$, the detected symmetry is considered exact, otherwise approximate. From the perspective of mechanical engineering, the computation error should be at least within the manufacturing accuracy $\varepsilon=10^{-6}$ to qualify the mechanical part exactly symmetrical [16,23].

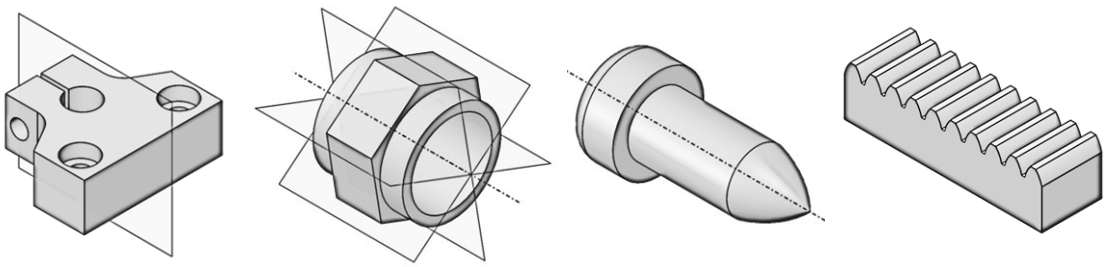
In terms of scale, mechanical parts exhibit either global [23], partial [16], or local [20] symmetry. Global symmetry implies that the 3D CAD model is fully symmetrical (Figure 1, a). Partial symmetry denotes that the 3D CAD model would be globally symmetrical if certain geometrical features would not disturb symmetry [23] (Figure 1, b). Hence, partial symmetry can be considered a subset of global symmetry with local non-symmetric features. In mechanical engineering, partial symmetry is also denoted as quasi-symmetry [14]. Finally, the local symmetry of the 3D CAD model refers to the case when only certain portions of the 3D CAD model are symmetrical (Figure 1, c). This doctoral thesis focuses on exact global and partial symmetry in 3D CAD models because of their relevance in mechanical engineering [14–19].



a) global symmetry b) partial symmetry c) local symmetry

Figure 1. Mechanical parts exhibiting reflectional symmetry at different scales

Mechanical parts commonly exhibit several basic types of symmetry, depending on the type of transformation (Figure 2). These types include reflectional (mirror or bilateral), rotational (cyclic symmetry and axisymmetry), and translational (repetitive) symmetry [11]. Reflectional symmetry denotes that the part remains unchanged when mirrored about a plane. (Figure 2, a). Rotational symmetric parts remain invariant under rotation about a central axis by an angle of $360°/N$, where N represents the number of repetitions at angular intervals (Figure 2, b). This type of symmetry is known under the term N-fold rotational symmetry, where N is at least $N=2$ because 1-fold symmetry does not represent symmetry (after 360° rotation, any object is mapped again onto itself). N-fold rotational symmetry about a central axis is denoted cyclic symmetry, which is very common in mechanical engineering [23]. Axisymmetry is a particular case of N-fold rotational symmetry where $N=\infty$. Hence, an axisymmetric part remains unchanged under all rotations about a central axis (Figure 2, c). Translational symmetric parts are invariant when sliding in some direction (Figure 2, d). Apart from the basic types of symmetry, mechanical parts may exhibit a combination of symmetries. For example, a mechanical part may be dihedral symmetric, which means that it is simultaneously reflectional and cyclic symmetric (Figure 2, b). The scope of this research is limited to reflectional and axisymmetric, as those two types are the most common in mechanical engineering [18].



a) reflectional symmetry b) cyclic symmetry c) axisymmetry d) translational symmetry

Figure 2. Mechanical parts with diverse types of symmetries

1.1 Research motivation

As highlighted in the introduction chapter, symmetry is an important geometrical property often employed in mechanical engineering. Engineers must often ensure that their 3D CAD models are designed symmetrically to achieve their intended design. While symmetry information is crucial, it is often not directly stored in the CAD model as planes and axes of symmetry [24]. The exception of indirectly stored symmetry information within the CAD model is when the final geometric shape is created by mirroring or pattern operations (linear, circular or translational) with respect to a plane or axis. However, such indirectly stored symmetry information is usually lost when exchanging the CAD model via neural or kernel CAD file formats (STEP[1], IGES[2], Parasolid, and ACIS[3]) because these formats currently do not support the exchange of reference geometry (planes and axes).

Engineers typically rely on visual inspection to detect symmetry in 3D CAD models, but this process can be time-consuming and challenging for complex shapes [23]. In addition, the engineer can visually detect only approximate symmetry [25], as exact symmetry needs additional mathematical verification. Computer-aided symmetry detection (CASD) aims to automatically identify planes, axes, and points of symmetry within digital objects, including 3D CAD models. Figure 3 compares qualitatively the time required for symmetry detection between the engineer and CASD, showing that as the number of digital objects increases, CASD becomes more efficient. Additionally, the mathematical verification of exact symmetry is faster and easier to compute through CASD. However, state-of-the-art CAD systems often do not provide proper CASD tools or require manual input from the user to verify symmetry [26]. Thus, one of the motivations of this research is the lack of proper symmetry detection tools in state-of-the-art CAD systems that could support or replace the need for visual detection of symmetry by the engineer.

[1] **St**andard for the **E**xchange of **P**roduct Model Data
[2] **I**nitial **G**raphics **E**xchange **S**pecification
[3] **A**dvanced **C**omputerized **I**mplementation of **S**tandards

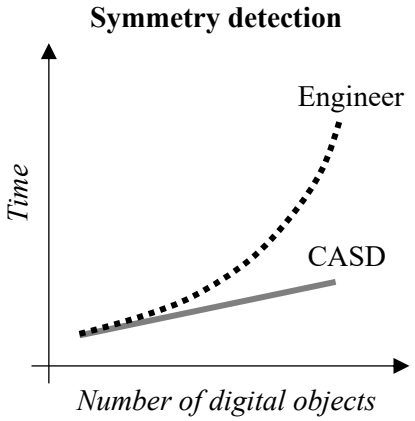


Figure 3. Symmetry detection – engineer vs CASD

The digital objects that are the focus of this doctoral thesis are 3D CAD models with boundary representation (B-rep or BREP), which are most often used in solid modelling [27,28]. B-rep CAD models consist of analytic or numeric geometry [28,31]. As outlined in Chapter 2, while previous CASD studies typically covered analytic geometry including plane, cylindrical, conical, spherical, and toroidal surfaces, there remains a gap in addressing numeric geometry. Until now, only spline surfaces and curves were considered [14,23]. This gap is a significant drawback since B-rep CAD models often require both geometry types. Therefore, this doctoral thesis is motivated by the limitations of prior CASD studies and aims to propose a CASD method to address these shortcomings. The following section outlines the research objective, hypothesis, and scientific contributions.

1.2 Research objective, hypothesis, and scientific contributions

The main research objective of this doctoral thesis is to introduce a CASD method and propose a computational environment for detecting exact global and partial reflectional and axisymmetry in B-rep CAD models containing analytical and numerical geometry. This research is limited to solid B-rep CAD models with manifold geometry and single parts with one body. The past CASD studies use either a geometry-based or view-based approach [29,30] (for more details, refer to Chapter 2). The geometry-based approach relies on the geometrical information of the input digital object to recognise symmetry.

The hypothesis of this research is as follows:
A geometry-based method can be used to detect exact global and partial reflectional symmetry and axisymmetry in B-rep CAD models containing analytical and numerical surfaces.

The expected scientific contribution of this doctoral thesis is:

1. A geometric-based method for symmetry detection in 3D CAD models that contain analytical and numerical surfaces.
2. A computational environment for symmetry detection in 3D CAD models that contain analytical and numerical surfaces.

Considering the defined research objective, hypothesis, and scientific contributions, the following section discusses the research methodology.

1.3 Research methodology

The research methodology in this doctoral thesis is based on reference [32]. It comprises four distinct phases (each phase is discussed in detail in the subsequent four paragraphs):

(1) the preliminary research,
(2) the proposed CASD method,
(3) implementing the CASD method into a computational environment, and
(4) validation of the proposed CASD method and computational environment.

(1) To establish a comprehensive understanding of the research area, the preliminary research phase involves an overview of existing scientific and expert literature. This literature overview entails identifying relevant sources and analysing and synthesising research findings. The primary focus of this phase is to gain a fundamental understanding of symmetry, types of symmetries, and previous CASD studies, particularly those concerning solid CAD models. Additionally, the preliminary research delves into the mathematical concepts underlying the B-rep technique, which is commonly used in state-of-the-art CAD systems [10]. Furthermore, this phase aims to explore suitable CAD file formats for use in CASD. Ultimately, the preliminary research phase defines the research methodology, goals, and hypothesis.

(2) After conducting preliminary research, a CASD method for solid B-rep CAD models that include both analytical and numerical surfaces is introduced. The CASD method aims

to provide a versatile solution through a geometry-based approach that allows for a range of input CAD models (including native, kernel, and neutral file formats) and can be adapted to various CAD systems. The proposed CASD method leverages the geometrical and topological data of the B-rep model to detect symmetry.

(3) The third phase involves proposing a computational environment designed to detect symmetry, allowing for the successful execution and validation of the proposed method. This is accomplished by selecting a suitable commercial CAD system, specifically Solidworks, and developing the computational environment through its Application Programming Interface (API). To demonstrate that the API serves solely as a tool for the development of the computational environment, the possibility of implementing the CASD method into two additional CAD systems, the commercial NX and the open-source FreeCAD, is explored.

(4) In the final stage of the research methodology, two datasets consisting of over 1000 solid CAD models in neutral STEP file format were gathered from two online CAD databases. These datasets contain a diverse range of mechanical parts and consist of common types of analytic and numerical surfaces present in solid CAD models. For validating the proposed CASD method and computational environment, the Validation Square [33] has been employed. Through this validation process, the advantages and drawbacks of the CASD method were identified and analysed. Finally, improvements to the method and computational environment were suggested, and future research directions were proposed based on the findings and conclusions.

1.4 Thesis organization

The doctoral thesis is divided into six chapters that, to some extent, align with the previously described phases of the research methodology.

Chapter 1 introduces symmetry and the types of symmetry and provides insights into the research motivation alongside a brief overview of the research aim, hypothesis, and expected scientific contributions.

Chapter 2 provides a comprehensive review of prior studies in CASD, with a special focus on mechanical engineering and symmetry detection in 3D CAD models. It also highlights the notable research gaps. Additionally, it covers CAD systems for CASD and the B-rep technique that represents the shape of the 3D CAD model. The chapter also includes the

basics of analytic and numeric geometry in the context of CAD exchange formats and existing geometric modelling kernels. This chapter corresponds to the preliminary stage of the research methodology.

In Chapter 3, a method for CASD in B-rep CAD models based on analytic and numeric geometry is proposed, considering the theoretical research background and previous CASD studies. The chapter is organised into sections that correspond to the proposed method's steps: (1) interpretation of the 3D CAD model, (2) analysis of B-rep, (3) generation, (4) trimming, and (5) evaluation of the planes and axes of symmetry candidates, and (6) the visualisation of the actual planes and axes of symmetry. This chapter focuses on the second phase of the research methodology.

In Chapter 4, the implementation of the proposed CASD method into a CAD system is discussed, with a focus on the computational environment. Additionally, the feasibility of applying the method in two other CAD systems is explored through their Application Programming Interfaces. This chapter represents the third phase of the research methodology.

Chapter 5 is devoted to validating the proposed CASD method and the computational environment. To accomplish this, the Validation Square [33] is used, which includes both qualitative and quantitative measures of structural and performance validity. This chapter also discusses the results within the framework of previous CASD studies.

Chapter 6 serves as the conclusion to the doctoral thesis. It reflects on the hypothesis and scientific contributions of the research and offers suggestions for future work. The last two chapters represent the fourth and final phase of the research methodology.

2 RESEARCH BACKGROUND

The second chapter summarises the research background for the conducted study. First, a review of past computer-aided symmetry detection research is given, with particular attention to mechanical engineering and 3D CAD models. Based on that, the research gaps are highlighted. Then, CAD systems for solid modelling and the underlying boundary representation technique most widely used for describing the 3D CAD model's shape are discussed. Next, the fundamentals of analytic and numeric geometry are examined from the perspective of data exchange files and geometric modelling kernels.

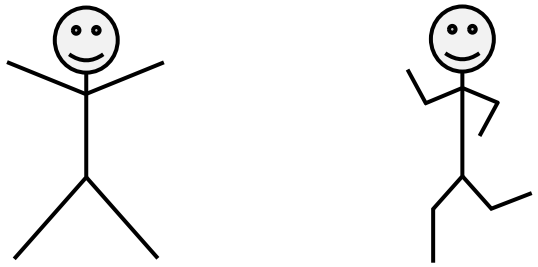
Computer-aided symmetry detection has been the scope of research for many years in different fields, including mechanical engineering [23], computer engineering [34], civil engineering [35], mathematics [36], and medicine [37]. As mentioned earlier, CASD aims to automate the detection of planes, axes, or points of symmetry in 2D or 3D digital objects. Typical examples of 2D digital objects used as input are images [38], computed thermography (CT) scans [37], and views [39]. On the other hand, the typical 3D digital objects exploited as CASD input are point clouds [40], mesh models [41], 3D CAD models [42], cable-strut structures [43], voxel models [41], and NURBS models [44]. CASD studies can be divided on whether they detect symmetry on complete [46] or incomplete [47] digital objects with considerable missing parts. Further, the inputs for CASD can be grouped into discrete (e.g., point clouds, mesh models, and voxel models) or continuous data (e.g., B-rep CAD model and NURBS model). In certain CASD studies, the initial input digital object is further processed and converted to make it more suitable for symmetry detection analysis. For that purpose, a mesh model was converted into a voxel model [48] or point cloud [38]. Also, a point cloud was subjected to meshing [49]. Furthermore, the initial input digital object may be represented by shape descriptors [50–53], which map the shape of an input digital object to a spherical domain. The main advantage of shape descriptors is that they exhibit the equivalent symmetry as the initial input digital object [50,51]. Another advantage of shape descriptors is that a complex shape is simplified, facilitating the CASD process. For instance, using spherical harmonic coefficients, the shape of a 3D mesh model with thousands of vertices may be represented

using only a few tens of coefficients [52]. The shape descriptors used so far are orientation histogram [50], reflective symmetry descriptor [47], generalised moment functions, and volumetric function [53]. The main drawback of using shape descriptors is that the plane and axes of symmetry often need to pass through the digital object's centre point or centroid, which is unsuitable for detecting approximate symmetry. In this doctoral thesis, the digital objects employed as input are B-rep 3D CAD models generated during solid modelling.

Apart from the input, the CASD generates specific output. The output of CASD can be classified based on various criteria: type of transformation, scale, accuracy, and distance metrics. The first three criteria were already highlighted in the previous chapter. As already indicated, the most common types of transformation are reflection [16], rotation [14,23], and translation [20]. As already stated, the possible CASD outputs in terms of scale are global, partial, or local symmetry, while in terms of accuracy, symmetry is exact or approximate. As previously outlined, the accuracy is distinguished based on the distance function $d(M,T(M))<\varepsilon$[4]. Usually, the correlation between the input and output in terms of accuracy is as follows. If the input is continuous data, then the expected output is most often exact symmetry, and if the input is discrete data, then the expected output is usually approximate symmetry. In the last criterion, distance metrics, the output can be classified into extrinsic [16] or intrinsic [37] symmetry. Extrinsic symmetry is often measured with Euclidean distance, while intrinsic symmetry with geodesic distance [54]. Most CASD studies address the detection of extrinsic symmetry (see Subsection 2.1.3). That is because, most often, the inputs for symmetry detection are rigid objects. Intrinsic symmetry, however, is focused on detecting symmetries in non-rigid or deformable objects. The difference between extrinsic and intrinsic symmetry can be explained by observing the representation of the human body, as illustrated in Figure 4. For instance, a human body is extrinsically symmetric when it stands still with raised hands (Figure 4, left). However, the body becomes intrinsically symmetric when in motion (Figure 4, right); only the distance metrics to measure the symmetry are different. An extrinsically symmetric object is also intrinsically symmetric, but an intrinsically symmetric object is extrinsically asymmetric. Hence, extrinsic symmetry is a subset of intrinsic symmetry.

[4] In mechanical engineering ε corresponds to the minimum manufacturing accuracy $\varepsilon=10^{-6}$ m [16,23].

The present doctoral thesis addresses extrinsic symmetry as the 3D CAD models from solid modelling are rigid.



extrinsic symmetric intrinsic symmetric

Figure 4. Sketch of a human body standing still (left) and in motion (right)

After introducing the inputs and outputs of CASD, the common approaches of CASD are further discussed. The CASD studies can be divided into geometry-based [40,43] and view-based [29,30]. Geometry-based CASD exploits the digital object's geometrical information for symmetry recognition. Geometry-based CASD can be further divided into those that only require the position information [55] (e.g., use only point clouds as input) and those that also use the surface information [47] (e.g., surface normal or Gaussian curvature) for the detection of symmetry. On the other hand, in the view-based CASD, symmetry detection is conducted by exploiting the information stored in 2D representations such as orthogonal views or images [29,56]. For instance, a set of viewpoints can be generated by setting a camera on the sphere pointing at the origin of the 3D digital object [30]. When it comes to the approaches for detecting the planes and axes of symmetry, the prior CASD studies use two main approaches: implicit (indirect) [46] and explicit (direct) [21]. In the implicit approach, for the given input model, a set of planes of symmetry candidates (POSCs) or axes of symmetry candidates (AOSCs) is generated and evaluated to determine the possible existence of the actual planes of symmetry (APOSs) or actual axes of symmetry (AAOS) among them. The explicit approach computes the APOS or AAOS directly without generating a set of candidates. First, the CASD studies with the implicit approach are discussed.

The implicit approach, as outlined in Subsection 2.1.3 of CASD, typically involves three steps. Firstly, a set of POSCs or axes AOSCs is generated based on the input model. Secondly, duplicates may be removed from the candidate set, although this step is optional. Finally, the candidates are evaluated to determine if any APOSs or AAOS are

potentially present among them. These steps are discussed in the following paragraph. The set of POSCs or AOSCs are generated using Principal Component Analysis (PCA), Random Sample Consensus (RANSAC), pairing of entities (e.g., points [46], viewpoints [29,30], etc.), incremental rotations [55], and so on. PCA was often exploited in CASD to identify the candidates (see Subsection 2.1.3). Generally, this is because APOS and AAOS of digital objects exhibiting exact symmetry may be aligned with the principal axes [59]. However, the drawback of PCA is that the computation of principal axes may be sensitive to the distributions of points on the objects [60]. In addition, only exact symmetries aligned with the principal axes can be detected, while other symmetries existing in the object may remain undetected. Also, the POSCs or AOSCs are usually constrained to pass through a reference point such as the centroid or centre of gravity/mass [29,30], which is not suitable for detecting approximate symmetry. Another common approach to generate candidates is RANSAC, which is an iterative method for robust model fitting of data containing many outliers. The basic idea is to generate the POSCs and AOSCs by fitting input points. The disadvantage is that with many input points, RANSAC usually produces many candidates, which can result in a computationally demanding CASD procedure [61]. Hence, the sampling of points can be exploited to reduce the number of input points [62]. Another common approach to generate candidates is by pairing entities. For instance, POSCs were generated by matching pairs of feature points [47]. Feature points were extracted from vertices with the highest Gaussian curvature. Another study generated the POSCs by matching pairs of points with a similar value of the heat accumulation function (a surface function based on a heat diffusion process) [46]. Another study identified the POSCs from pairs of viewpoints by matching the viewpoint entropies [29,30]. The viewpoint entropy was computed based on Shannon entropy, considering the projection area of each visible face and the number of visible faces. Next, the candidates were obtained by incremental rotations around a spherical surface to generate a set of AOSC with increments fixed to 0.5º [55]. Generally, the problem when dealing with incremental rotations is the appropriate choice of increment. A too-small increment can result in too many candidates and a computationally demanding CASD procedure, while a too-large increment can result in an insufficient number of candidates, increasing the risk of symmetry misdetection. In addition, the candidates usually must pass through some reference point,

i.e., centroid, which limits the CASD to exact symmetry. In the second step of the implicit CASD approach, which is optional, the generated POSCs or AOSCs are further eliminated. For instance, the study in [40] conducted candidate pruning to eliminate duplicates using a distance function to measure the distance between two POSCs. Another CASD study eliminated duplicate POSCs through a comparison process by checking for coincidence and parallelism between two POSCs [42]. In the third step, the generated candidates are evaluated to determine if a POSC or AOSC also represents an APOS or AAOS. Different approaches are used for that. The studies reported in [29,30] evaluated each POSC with all pairs of viewpoints with matching entropy to verify whether the remaining matching pairs are within a minimum number. If the number of symmetric pairs was large enough, the POSC was declared as the APOS. The research in [46] proposed evaluating the POSCs through a voting scheme. Only mesh points with mean curvature higher than a threshold value were considered in the voting. The POSC received more votes if more point pairs were reflectional symmetrical to it. In the end, the POSC with the highest vote count was selected as the APOS. Another voting process was proposed in [47]. Each POSC was tested against all feature point pairs, and the POSC with the highest vote count was declared as the APOS. The study described in [57] proposed a variation of Hausdorff distance for the evaluation of candidates, which measures the average distance between the original mesh and the mesh created by reflecting the original mesh over the candidate. Another study used gradient descent to obtain the local optimum and its error for each POSC [60]. The error was computed as the mean square between a point and its closest mirror reflection point. The POSC with the smallest error was selected as the APOS. In summary, the implicit CASD approach generates a set of candidates, while the explicit approach, which is discussed in the next paragraph, computes the APOS or AAOS directly.

The study in [21] sampled the input model and computed the principal curvatures and directions for each sampled point. Then, pairs of the sampled points were generated to create candidate transformations (one point of the given pair is transformed to align its position, principal directions, and normal direction with the second point in the pair). The candidate transformations, i.e., a set of points in the transformation space, were clustered using mean shift to detect partial symmetry. The study in [58] suggested looking for correspondences between symmetric points rather than performing symmetric

transformations to find symmetries. These correspondences were represented in a symmetry correspondence matrix that encoded symmetry relationships between pairs of points sampled from the input data. The symmetry correspondence matrix was derived from the inverse of a dissimilarity matrix obtained by pairing points with a randomised voting algorithm. The analysis of the correspondence matrix was based on its spectral properties. For this purpose, the authors introduced the Symmetry Factored Embedding (SFE) and the Symmetry Factored Distance (SFD). The SFE embedded the input shape into a higher-dimensional Euclidean space, and the SFD represented the Euclidean distance in that space. In addition to detecting global extrinsic symmetry, partial extrinsic and intrinsic symmetry were also investigated. Although the proposed study is general and robust to noise in the input, its time complexity seems relatively high. The study in [41] used volumetric 3D shapes as input for detecting approximate reflectional and rotational symmetries. The shape was described by a binary indicator function, which equals "1" for interior points and "0" for all points outside the shape. Next, the distortion was computed, representing the total mismatched volume between the original and transformed shapes. The transformation space was then carefully sampled to find symmetry with high probability. Each transformation was efficiently evaluated using a sub-linear sampling that randomly examined only a small number of points. The proposed CASD research appears to be fast and robust to noise in the data. However, the input shape's centroid requires alignment with the origin, and the detected plane or axis of symmetry must pass through the origin. Another explicit approach in [52] proposed a more straightforward solution by approximating the triangle mesh model's shape with spherical harmonics coefficients. Consequently, a shape with thousands of vertices may be represented using only a few tens of coefficients. The study concluded that the reflective symmetry in the 3D shape is equivalent to a linear phase structure in the corresponding spherical harmonic coefficients. In this way, the symmetry detection matches a compact set of descriptors, i.e., optimising a linear phase fit to the observed coefficients. The proposed study is also applicable to point clouds. However, its main disadvantage is that only reflectional symmetry can be detected passing through the origin.

The mentioned CASD studies have been discussed in the context of different research fields, such as computational computer engineering, civil engineering, etc. However, this

doctoral thesis deals with CASD in mechanical engineering, so the following subsection reviews previous CASD studies in this field.

2.1.1 Computer-Aided Symmetry Detection in Mechanical Engineering

From the perspective of mechanical engineering, CASD was exploited for retrieval [44], compression [64], and alignment [60] of 3D CAD models, design for assembly [42], and detecting design intent in scanned models from reverse engineering [65]. Most past research investigated symmetry detection on single parts [16,23,42,63] and less often on assemblies [66]. Further, the CASD studies were dominantly focused on detecting exact symmetry [16,23,42,63] and rarely on detecting approximate symmetry [67,68]. The common types of symmetries studied were reflectional [16], rotational, [42] axisymmetry [66], translational [65], and dihedral [68] symmetry. The studies addressed the detection of symmetry at various scales: global [16], partial [23], and local [20]. The early research related to CASD in CAD was focused on detecting the axis of symmetry in 2D polygons [36,37]. However, due to their complexity, this did not match the requirements of 3D CAD models. The studies on CASD in 3D CAD models can be divided into two groups. The first group studied CASD from the perspective of B-rep [14,20,23,42,65– 69], while the second group studied CASD from the perspective of design features [25,70]. The general disadvantage of using design features over the B-rep is the restriction to native CAD models, as the history tree and feature information can only be shared through those file formats. The B-rep CAD models studied in prior CASD studies were exact [20,42] and approximate [65]. Exact B-rep CAD models are those obtained from solid modelling, while approximate B-rep CAD models originated from reverse engineering.

First, an overview of CASD studies in approximate B-rep CAD models is presented. The research in [65,67,68,69] studied CASD for detecting geometric design intent in reverse engineering CAD models obtained by scanning. Hence, the inputs for CASD were approximate B-rep CAD models, and the outputs were approximate symmetry. The studies [67,68] used the vertices of the B-rep model to detect local incomplete symmetries under reflection, rotation, translation, rotation-reflection, and glide-reflection were detected using a cycle clustering algorithm. Incomplete symmetries were defined as a set of incomplete cycles constructed by a set of consecutive vertices of an approximately regular polygon [67,68]. The detection of approximate symmetry of the vertices point set

was turned into a permutation of the points and mapping distances between the points approximately onto each other (the tolerance was fixed to max. 5% of the longest distance between points in the input point set). The drawback of the proposed study is that it relied on vertices (although the input model does not necessarily need to have vertices) and addressed only local symmetry. Another research by the same authors [69] investigated the decomposition of approximate B-rep CAD models into regularity feature trees (RFTs). Regularity features are simple closed volumes that combined describe the CAD model's original shape. The detection of regularity features was based on recovering broken symmetries in the model by analysing symmetry breaks in faces (a face might be broken in its interior, across one or more edges, or surrounding one or more vertices). The decomposition exposed rotational symmetric and regular translational arrangements of regularity features. The drawback of the study is that it addressed only local symmetry. The same authors presented an extension of the work in [65]. First, the shape of the approximate B-rep model was described through leaf-parts, i.e., regularity features at the leaves of the RFT. The basic idea employed was to seek regularities (congruencies, incomplete symmetries[5], and symmetric arrangements) within these leaf-parts. The regularities were detected via consistent mappings between characteristic point sets of the B-rep model, i.e., leaf-part centroids, vertices, and other specific points which characterise edges and faces (e.g., a circular arc may be uniquely determined by its two endpoints and its midpoint). Congruencies in the leaf-parts were detected by a clustering algorithm that created congruence sets (each containing one or more congruent leaf-parts). Next, subsets forming incomplete symmetries and incomplete symmetric arrangements were obtained for each congruence set. The type of transformations addressed to detect symmetric arrangements in the set of leaf-parts were reflection, inversion, translation, rotation, dihedral, glide (mirror in a line followed by translation parallel to the line), and screw (rotation about an axis followed by a translation along the axis). The study used the concept of compatible symmetries, meaning that symmetries were merged and multiple subsets shared the same type of symmetry. The compatible symmetries shared by leaf-parts and their symmetric arrangements were further combined to detect regularities in the form of transformations matching sub-parts of the model. The geometric regularities of and between model sub-parts enable the detection of geometric

[5] This means that not all elements building the symmetry were present

design intent. The study addressed the detection of local (from symmetric arrangements of leaf-parts) and global symmetry (from compatible symmetry). The main limitation of both studies [65,69] was the consideration of only planar, spherical, cylindrical, conical, and toroidal surfaces. This restriction arises from the difficulty of extending the geometry to numeric surfaces involved in the RFT construction. In addition, using approximate B-rep CAD models limits the CASD only to approximate symmetry. To detect exact symmetry within the 3D CAD model, exact B-rep's from solid modelling were used as input, which is discussed in the next paragraph.

The research reported in [23,42] studied symmetry detection in the context of Design for Assembly. The research proposed a loop-based approach for detecting exact and partial reflection symmetry in exact B-rep CAD models[6]. In addition to the mentioned symmetries, primary axes (major and minor) were detected, which lie parallel and perpendicular to the longest dimension of the 3D CAD model's smallest bounding box. First, the loops were classified based on the surface type (plane, cylinder, sphere, torus, cone, and spline). Then, the POSCs and AOSCs were generated by pairing identical loops through their properties, such as loop type and area (i.e., the surface area bounded by the loop of a face) and the number of edges. The POSCs and AOSCs were generated from the loop centroid and normal vector. For that purpose, three loop pair types were distinguished: coplanar, parallel, and coaxial. The first and the second loop pair types were used to create the POSCs, while the third was used to create the AOSCs. The AOSCs were also generated from single loops of the cylindrical, spherical, and toroidal surface types. The peak number of the initially generated POSCs and AOSCs can be estimated using the following equation:

$$n_{\mathrm{C}} = n_{\mathrm{POSC}} + n_{\mathrm{AOSC}} = \sum_{k=1}^{6} \frac{n_{\mathrm{L},k}\,(n_{\mathrm{L},k} - 1)}{2} + n_{\mathrm{E}}\,, \tag{1}$$

where n_{E} is the total number of edges in the CAD model, and $n_{\mathrm{L},k}$ is the total number of loops of the particular loop type (plane, cylinder, sphere, torus, cone, or B-spline). The initially generated POSCs and AOSCs were rationalised by eliminating duplicates and assigning the associated loops to the rationalised candidates. The symmetry was considered "stronger" if more loops and loop areas were associated with a rationalised

[6] A loop is a closed circuit of edges bounding a face.

POSC or AOSC. Hence, the rationalised candidates were ranked according to the number of loops and total loop area, and those with the highest ranking were declared as the APOSs or AAOSs. The study, however, did not explicitly emphasise the criteria for detecting partial symmetry. The study also proposed to group all symmetric loops about a common APOS or AAOS and thereby isolate the asymmetric portions of the boundary for further consideration. The primary axes were defined from the intersection of several highest-ranked candidates by identifying maximally two orthogonal axes. The study addressed loops with underlying analytic surface types (plane, cylinder, cone, sphere, and torus) and numeric surface types (only spline surfaces). Calculating loop properties (loop area and normal vector) for the spline surface type may be sensitive and lead to invalid POSCs when matching identical loops. Hence, the authors propose excluding such loops if they do not form a considerable proportion of the total surface area of the input model. In the case of an input model composed dominantly or entirely of B-spline loops, the proposal was to calculate the loop properties by projecting their associated faces against a plane and calculating the properties of the resulting planar projection. A significant obstacle in the implementation was the imprecision of geometric and topological definitions of loops in the ACIS geometric modelling kernel. Hence, instead of loops, the following studies proposed using faces as input.

The research [16,63] explored the use of CASD in the product development process. They proposed the divide-and-conquer strategy for detecting exact global and partial axisymmetry and reflectional symmetry using faces of the B-rep as input. As a preliminary step, the B-Rep model's topology was first transformed into hypergraph data structures to generate maximal surfaces and curves by merging adjacent surfaces and curves. For instance, two adjacent half-cylinders can be merged into one closed cylinder to obtain a maximal surface. Then, in the divide phase, the POSCs and AOSCs were generated from one, two, or three adjacent faces using the intrinsic parameters of the underlying analytic surfaces and their intersections (vertices, edges, and loops). For instance, the intrinsic parameters of a plane surface are its base point and normal vector, those of a cylindrical surface its axis point, axis vector and radius, and so on. Five categories of POSCs were distinguished:

- An Orthogonal POSC is orthogonal to an edge and created by the two neighbouring faces.

- A Loop Bisector POSC is attached to vertex and reflects a symmetry of its two adjacent edges.
- A Loop Symmetry POSC is the symmetry information within a face containing multiple loops.
- A bisector POSC coincides with an edge in a bisector plane produced by its adjacent faces.
- An AOSC is a special case of an infinite number of Orthogonal POSCs. For instance, axisymmetric faces (e.g., a closed cylinder, sphere, torus, and cone) have infinite POSCs.

The highest possible number of candidates n_C (including POSCs and AOSCs) can be estimated using the following equation:

$$n_C = 2n_E + pn_V + n_L, \tag{2}$$

where n_E represents the total number of edges, n_V is the total number of vertices, and n_L is the total number of loops in the 3D CAD model. At the same time, parameter p describes the maximum number of adjacent faces around a vertex (assuming up to four faces share the same vertex p=4). The generated candidates are evaluated in the conquer phase, consisting of a two-level propagation process. In the first level, coincident POSCs and AOSCs are merged to form POSC chains, representing the intersections between POSCs and the 3D CAD model's boundary. In the second level, the propagation expands over the 3D CAD model's boundary on both sides of POSC chains until all surfaces are covered without asymmetry. In this way, exact global symmetry is detected. However, it was not specified how much boundary needs to be covered to consider the CAD model partially symmetric. The study addressed only analytical surfaces (plane, cylinder, cone, sphere, and torus). Another study drawback is that a combinatorial analysis was used to obtain the combinations of surfaces, their adjacencies, and intersections for identifying the POSCs and AOSCs. Consequently, if the input 3D CAD model has certain non-predicted combinations of analytical surfaces, the corresponding POSCs or AOSCs (i.e., APOSs and AAOSs later) may remain undetected.

The studies [14,66] investigated CASD in the context of CAE for the detection of symmetric regions in partial axisymmetric (i.e., quasi-axisymmetric) 3D CAD models and their decomposition for automated hex-mesh generation. The study was first introduced on single parts [66] and later extended to assemblies [14]. The basic concept

was to classify faces into axisymmetric, pseudo-axisymmetric, cyclic symmetric, and non-axisymmetric to identify axisymmetric regions, cyclic sectors, and non-cyclic regions. The study addressed analytical surfaces and NURBS. The faces were classified based on the following criteria:

- Axisymmetric faces may be cylindrical, conical, spherical, and toroidal surfaces if their axis of rotation is colinear with the AOSC. In addition, the surfaces were required to span the entire 360° rotation around the AOSC and to be bounded by two outer loops. A plane face was classified as axisymmetric if its normal vector was parallel with the AOSC and all axes of bounding edges were collinear with the AOSC.
- A pseudo-axisymmetric face implies an axisymmetric face containing either non-axisymmetric or cyclic symmetric inner loops and axisymmetric outer loops.
- Cyclic symmetric faces were identified by grouping equivalent faces into repetitive patterns.

Equivalent faces were identified based on the geometrical properties of their underlying surfaces and the topological and geometrical properties of their curves. The first condition of equivalent faces was that they were equally distanced from the AOSC. The further considered geometrical properties were the normal orientation for plane surfaces, the radius for cylindrical and spherical surfaces, the major radius and half angle for conical surfaces, and the major and minor radius for toroidal surfaces. For the cylindrical, conical, toroidal, and plane surfaces, the additional geometrical property compared was the scalar projection of the corresponding face vector (normal or axis vector) onto the AOSC. The geometrical properties of the B-spline surfaces were queried through the equivalence of knot vectors and control points. If the geometrical properties of surfaces were equal, then the geometrical properties of their curves were compared. The linear, circular, and elliptical curves were compared using their geometric properties (the authors did not specify which exactly). Like surfaces, the B-spline curves were compared by their knot vectors and control points. Intersection curves between two surfaces were considered equivalent if their respective pairs of surfaces were equivalent. Finally, all equivalent cyclic faces (including the rotation angle between them) were organised into groups of cyclic faces to identify the existence of a repeated cyclic pattern in each group. The total number of combinations to obtain the repeated cyclic pattern in a group was n^m, where n

represents the number of repetitions, and *m* the number of faces in the group. The proposed criterion for selecting a combination of cyclic faces was to maximize the number of shared edges between the faces. Instead of finding the complete list of combinations for each group, the authors propose a front propagation to find one combination containing connected faces and an optimization loop to converge to the list of faces having the maximum number of shared edges. The main advantage of this CASD study is that it deals with the detection of cyclic symmetry (although only at the local level), while its main disadvantage is that it required manual input of the AOSC.

The study in [20] proposed a graph-based approach to detect multi-scale (i.e., at different geometric scales) symmetric regions and extract symmetric relations among these regions. The CAD model's B-rep model was represented by a Congruence-labelled Adjacency Graph (CLAG) and Frequent Sub-graph Mining (FSM) was used to mine in CLAG, and to extract complete multi-scale congruence features. To overcome the problem of redundant congruence features, a set of geometric heuristic rules were used to filter meaningless congruence features. The remaining congruence features reflectional, rotational, translational symmetries and symmetry structures (i.e., compound symmetry relations) were detected. The construction of CLAG was done in three steps. First, the congruent faces of the model were clustered into the same set and every set received a different face label. Second, a corresponding vertex label to each vertex of CLAG was added for each face based on its associated face label. And third, the edge label for each edge in CLAG was determined based on its adjacent faces in the model (the same edge label was used if two edges shared the same adjacent faces, otherwise a new edge label was generated). The study proposed the application of CASD for Smart Direct Modelling and multi-scale model simplification. The study was also limited to analytic surfaces. Another significant drawback of this graph-based approach is that it is computationally demanding. Next, the group of researchers who studied symmetry detection from the perspective of design features is discussed. Those studies are restricted to native CAD models and are applicable only to a particular CAD system.

The studies presented in [25,70,71] used feature information as input to detect exact reflectional and rotational symmetry in parts [25,71] and assemblies [70]. In the context of single parts, the feature information used were feature types, parameters, sketches, references, Boolean operations, and the modelling history or feature tree [25]. The CAD

model's features were first classified into congruent feature sets. Congruent means that two features can be transformed into each other by an isometry (reflection or rotation). The feature sets were detected through the study on the relationship between feature information and the self-symmetry of features. Then, the feature sets were sorted into an ordered sequence, and global symmetry was derived by successively merging and verifying the symmetries of feature sets in the ordered sequence. The research assumed that the CAD models were designed free of the designer's bad modelling habits, which is not always the case in practice. In other words, the proposed CASD research is sensitive to redundant feature modelling and modelling of symmetric shapes using non-symmetric features. In addition, the research covered only features made up of analytic surfaces. Hence, an extension of the study covering numeric surfaces was briefly introduced in [71]. The study used a feature reference tree (FRT) to describe the design history of the surface model. To deal with non-unique design history, so called feature tuples were constructed from the FRT and the parent-child relationships of topological entities. The surface models' global reflectional and rotational symmetries were detected using feature information in feature tuples. Another extension of the study [25] from single parts to assemblies was proposed by the same authors in [70] to detect global reflectional and rotational symmetry in assembly CAD models. The symmetry between components is identified through geometric reasoning by taking full advantage of the assembly constraints and the geometric information of the components. The parts within the assembly were also constrained only to analytic surfaces.

Considering the mentioned CASD studies, the following points are interesting from the perspective of the CASD method within this doctoral thesis:

- The B-rep [16,20,42,65,66] of the CAD model was more frequently used as input in CASD than features [25,71]. This is because the B-rep offers greater generality and flexibility, not being limited solely to native CAD models.
- The common topological entities used in prior CASD studies are faces [16,20,66], loops [42], and vertices [67]. Faces are more convenient than loops and vertices as they provide more detailed geometrical information. Loops are sets of connected edges, and calculating their properties can be imprecise and computationally demanding, while vertices may not adequately describe the shape of the CAD model.

- Most previous CASD studies [16,20,23,65,66] utilise the B-rep definition and classify topological entities based on the types of underlying geometric entities.
- The prior CASD studies dominantly use the implicit symmetry detection approach, where the POSCs and AOSCs were generated from the self-symmetry of single topologic entities or by pairing of topologic entities.
- The implicit symmetry detection approach is predominantly used in prior CASD studies [16,20,42,66], where the POSCs and AOSCs are generated from the self-symmetry of single topological entities or by pairing of topological entities.

The following subsection presents and discusses the basic idea of symmetry measures from the perspective of several existing CASD studies.

2.1.2 Symmetry and similarity measures

Symmetry measures are used in the implicit approach of CASD to evaluate or quantify the symmetry in the digital object. These measures are usually expressed as a numerical value and calculated for each POSC or AOSC. Typically, symmetry measures fall between 0 and 1 [72–75], and the POSC or AOSC with the highest symmetry measure score is selected as the APOS or AAOS [62]. One study suggested maximising a defined symmetry measure to assess each POSC [62], as follows:

$$s_X(\boldsymbol{p}) = \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}\varphi\left(\left\|\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}_i) - \boldsymbol{x}_j\right\|\right) \tag{3}$$

where ω_{ij} represents the weights of given pairs of points, φ is a similarity function, the function $\boldsymbol{r}(\boldsymbol{p},\boldsymbol{x}_i)$ reflects a point $\mathbf{x}_i$ over a plane P represented by the vector $\boldsymbol{p}$, and $\boldsymbol{x}_j$ is any other point. Although the weights were initially set to 1, they can be utilised to incorporate additional information about the input model, such as the normal vectors or directions of principle curvatures in corresponding point pairs with respect to a given plane. As the proposed symmetry measure is continuous and differentiable, optimisation methods can be applied. Ultimately, the APOS with the highest symmetry measure was selected as the POSC. One limitation of this study is that it can only identify the most significant APOS, even if the input model has more than one APOS.

Researchers have endeavoured to develop symmetry measures that assign a numerical value between 0 and 1 to quantify the level of symmetry expressed in digital objects [72–

75]. Typically, a score of 0 indicates non-symmetry, while a score of 1 denotes full symmetry. One such measure is the coefficient of symmetry β, which assesses reflectional symmetry in planar images along a specified axis of symmetry [72]. The coefficient of symmetry β can be defined as follows:

$$\beta = \frac{\iint w(x,y)\; w(\bar{x},\bar{y})\; dxdy}{\iint w^2(x,y)\; dxdy}, \tag{4}$$

where $w(x,y)$ and $w(\bar{x},\bar{y})$ are the intensity functions of two symmetric points and with respect to the axis of symmetry. The coefficient of symmetry ranges between $0 \le \beta \le 1$, where β=1 indicates exact symmetry and β=0 non-symmetry. Another symmetry measure, the Symmetry Distance (SD) applies to reflectional or rotational symmetry in 2D shapes represented by a sequence of points (i.e., polygons) [73]. SD is defined as the minimum mean squared distance between the points of the two shapes:

$$\mathrm{SD} = \frac{1}{j}\sum_{i=0}^{j-1}\left\| P_i - \hat{P}_i \right\|^2, \tag{5}$$

where P represents the shape and $\hat{P}$ its symmetry transforms, while j is the sequence of points. The SD enables a comparison of the "amount" of symmetry of different shapes and the "amount" of different symmetries of a single shape. If SD=0, the shape is considered perfectly symmetric. The authors also present a way to extend the SD to 3D shapes represented by points sampled on a plane and projected on the object. The study in [74] proposed the Degree of Symmetry (DOS) to measure reflectional symmetry in 2D polygons:

$$\mathrm{DOS} = 1 - \frac{\mathrm{SD}(A,B)}{\mathrm{Area}(A) + \mathrm{Area}(B)} \tag{6}$$

$$\mathrm{SD}(A,B) = \mathrm{Area}(A \cup B) - \mathrm{Area}(A \cap B) \tag{7}$$

where $\mathrm{SD}(A,B)$ represents the symmetric difference, and A and B are two sets. If the shape is symmetrical, the DOS equals 1 since the symmetric difference is 0. The larger the value of SD, the greater the non-symmetry.

The study presented in [75] proposes using Jaccard similarity as a symmetry measure for detecting the axis of reflectional symmetry in binary images with black and white pixels.

Thereby, the Jaccard index $\mu(B)$ for a reflectional symmetric binary image was defined as:

$$\mu(B) = \frac{\left|S(B) \cap S(B_R)\right|}{\left|S(B) \cup S(B_R)\right|}, \tag{8}$$

where B is the binary image (the brightness of the black pixels denoted with 1 and those of white pixels with 0), B_r is the reflection of the binary image B with respect to a line, $S(B)$ is the set of pixels belonging to the image B, the brightness of which is equal to 1. If $\mu(B)$=1, the binary image B is considered fully symmetrical. Detecting reflection symmetry involves iterating through all possible lines that cross the binary image and finding the one with the highest Jaccard index. Although the presented symmetry measures focus on the 2D domain, extending them to the 3D domain is a challenging task due to their definition [72–75]. As such, an appropriate symmetry measure that applies to 3D CAD models with B-rep is necessary and is proposed in the CASD method in this doctoral thesis (Subsection 3.5.2).

Another type of measure often employed in CASD are similarity measures. Generally, the similarity is studied in engineering design to support designers in generating new designs [76] or in manufacturing to extract existing product information, such as cost estimations in machining [77]. Moreover, recognising similarities in 3D CAD models can be applied to reuse existing design solutions [78]. For this purpose, a given input CAD model (new design) is used to retrieve similar CAD models from the database (existing designs). Recognising similarity may also benefit the clustering of CAD models [79]. However, in the implicit CASD approach, similarity measures such as the heat accumulation function [46] or Gaussian curvature [47] were exploited to match similar point pairs for generating POSCs. In the context of this doctoral thesis, similarity measures are engaging for generating the POSCs from identical or similar pairs of topological entities in the B-rep CAD model. Existing CASD studies [23,42] related to B-rep CAD models identified identical loop pairs using their properties (loop type, loop area, and the number of edges) as a similarity measure. Similarity measures from statistics were considered and investigated for implementation in the CASD method within this doctoral thesis. They are generally used to compare the similarity between two finite

datasets. For instance, the Cosine similarity (CS) computes the cosine of the angle between two feature vectors **A** and **B** [80]:

$$\mathrm{CS} = \cos\ \theta\ = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{|X\cap Y|}{\sqrt{|X|\cdot|Y|}} = \frac{c}{\sqrt{a+c\ \cdot\ b+c}}. \tag{9}$$

The CS can also be expressed as a function of two datasets (X and Y). In that case, it is defined as the intersection size (the number of common elements) divided by the square root of the set's multiplied cardinalities. The cardinality of a set, $|X|$ or $|Y|$, represents the number of elements it contains. Alternatively, if two finite datasets are represented by two binary feature vectors, **A** and **B**, then in the above Equation a represents the total number of features with the value 1 in **A** and 0 in **B**, b the total number of features with the value 0 in **A** and 1 in **B**, and c the total number of features with value 1 in both **A** and **B**. For instance, CS was utilised to compute the similarity between two Opitz code vectors (the CAD model features were presented by alphanumerical digits) [81]. Another typical similarity measure, the Jaccard index (JI) already mentioned in Equation (8), is defined as the size of the intersection divided by the size of the union (the number of unique elements) of two finite datasets X and Y [82]:

$$\mathrm{JI} = \frac{|X\cap Y|}{|X\cup Y|} = \frac{|X\cap Y|}{|X|+|Y|-|X\cap Y|} = \frac{c}{a+b+c}. \tag{10}$$

The JI was employed for clustering purposes to measure the similarity between machines/parts and group them [83]. Alternative similarity measures related to the Jaccard index are the Sørensen–Dice coefficient (SDC), which is defined as twice the size of the intersection divided by the sum of elements in each set, the Szymkiewicz–Simpson coefficient (SSC) or Overlap coefficient, described as the ratio between the size of the intersection and the smaller cardinality of two datasets, and the Braun-Blanquet coefficient (BBC), which represents the size of the intersection divided by the larger cardinality of two datasets [75,82]:

$$\mathrm{SDC} = \frac{2\cdot|X\cap Y|}{|X|+|Y|} = \frac{|X\cap Y|}{0.5\ \ |X|+|Y|} = \frac{c}{0.5\ \ a+b+2c}, \tag{11}$$

$$\mathrm{SSC} = \frac{|X\cap Y|}{\min\ \ |X|,|Y|} = \frac{c}{\min\ \ a+c,b+c}, \tag{12}$$

$$\mathrm{BBC} = \frac{|X \cap Y|}{\max\ |X|,|Y|} = \frac{c}{\max\ a+c, b+c}. \tag{13}$$

All mentioned similarity measures from Equations (8) to (12) have a range of [0, 1]. Values close to 1 represent very similar sets, while values close to 0 represent very different sets. The referred similarity measures are explored in terms of their possibilities and applicability for detecting similar pairs of topological entities to generate the POSCs (for more details refer to Subsection 3.3.2).

2.1.3 Overview of CASD studies

Tables 1 to 4 offer an overview of past CASD studies, organised by year of publication. These tables present pertinent information about the various 2D and 3D digital objects utilised as input, as well as a summary of the typical CASD outputs, including the type of transformation (such as reflectional, rotational, and axisymmetry), scale (global, partial, or local), accuracy (exact vs approximate), and distance metrics (extrinsic vs intrinsic), and the approaches (geometry-based vs view-based and explicit vs implicit). Based on this data, certain conclusions can be drawn. Point clouds are the most common input for CASD studies, with reflectional symmetry being the most frequently studied type of symmetry. Symmetry detection is usually addressed at the global level, with most studies focusing on detecting approximate and extrinsic symmetry. The geometry-based and implicit approaches are the most widely used methods for detecting symmetry. Based on the reviewed CASD studies in the previous subsections, the following subsection identifies the relevant research gaps in the context of B-rep CAD models.

Table 1. Overview of prior CASD studies (Part 1)

RESEARCH	YEAR	INPUT										OUTPUT												APPR				Note
		2D				3D						TYPE OF TRANSF.						SCALE			ACC		D.M.						
		Image	View	CT or MRI	Wireframe	Point cloud	Mesh model	Voxel model	CAD model	NURBS model	Cable-struts	Reflectional	Rotational	Cyclic	Axisymmetry	Translational	Dihedral	Global	Partial	Local	Exact	Approximate	Extrinsic	Intrinsic	Geometry-based	View-based	Explicit	Implicit	
Sun and Sherrah [50]	1997	●		●		●						●	●					●				●	●		●	●		●	Shape descriptor: Orientation Histogram (Extended Gaussian Image); PCA
Tate [23]	2000								●			●			●			●	●		●		●		●			●	B-rep CAD model
Tate and Jared [42]	2003								●			●			●			●	●		●		●		●			●	B-rep CAD model
Kazhdan et al. [48]	2004					●	●	●				●						●				●	●		●		●		Shape descriptors: Reflective Symmetry Descriptor, Voxelization of 3D Models, candidates must pass through the COG/COM
Lucchese L [84]	2004	●												●			●	●				●	●			●	●		-
Thrun and Wegbreit [39]	2005		●									●	●		●					●		●	●			●	●		-
Podolak et al. [53]	2006	●					●					●						●		●	●	●	●			●		●	Shape descriptors: Planar Reflective Symmetry Transform (PRST), Mesh model transformed into a volumetric function, candidates must pass through the COG/COM, Vote scheme
Martinet et al. [51]	2006						●					●	●					●				●	●		●			●	Shape descriptors: generalised moment functions, candidates must pass through COG/COM
Mitra et al. [21]	2006					●	●					●	●			●			●			●	●		●		●		shape descriptors: curvature
Zou and Lee [85].	2006				●								●									●	●			●		●	Skews symmetries
Li et al. [69]	2006								●				●			●			●			●	●		●				-

Table 2. Overview of prior CASD studies (Part 2)

RESEARCH	YEAR	INPUT										OUTPUT												APPR				Note
		2D				3D						TYPE OF TRANSF.						SCALE			ACC		D.M.						
		Image	View	CT or MRI	Wireframe	Point cloud	Mesh model	Voxel model	CAD model	NURBS model	Cable-struts	Reflectional	Rotational	Cyclic	Axisymmetry	Translational	Dihedral	Global	Partial	Local	Exact	Approximate	Extrinsic	Intrinsic	Geometry-based	View-based	Explicit	Implicit	
Li et al. [67]	2007								●			●	●			●	●	●		●		●	●		●		●		Input: approximate B-rep CAD model. Cycle clustering algorithm.
Tedjokusumo and Leow [60]	2007					●	●					●						●				●	●		●			●	PCA, Bilateral symmetry plane.
Pauly et al. [86]	2008					●	●					●	●			●			●			●	●		●		●		-
Chang and Park [49]	2008					●						●						●				●	●		●			●	Triangulation of point clouds. Initial POSC needs to be specified by the user.
Li et al. [68]	2008								●			●	●			●	●	●		●		●	●		●		●		Approximate B-rep CAD model Cycle clustering algorithm
Xu et al. [87]	2009						●					●							●			●		●	●			●	Geodesic distance. For Segmentation and Vote scheme
Li et al. [65]	2010								●			●	●	●	●	●	●	●		●		●	●		●			●	Approximate B-rep CAD model Regularity Feature Trees
Raviv et al. [54]	2010					●	●					●	●					●	●		●	●		●	●			●	-
Lipman et al. [58]	2010					●						●	●					●	●			●	●	●	●		●		Symmetry Factored Embedding (SFE) Symmetry Factored Distance (SFD)
Zingoni [55]	2012										●	●	●	●			●	●			●		●		●			●	-
Kakarala et al. [52]	2013						●					●										●	●		●		●		Shape descriptors - spherical harmonics
Li et al. [16]	2013								●			●			●			●	●		●		●		●			●	B-rep CAD model

Table 3. Overview of prior CASD studies (Part 3)

RESEARCH	YEAR	INPUT										OUTPUT												APPR				Note
		2D				3D						TYPE OF TRANSF.						SCALE			ACC		D.M.						
		Image	View	CT or MRI	Wireframe	Point cloud	Mesh model	Voxel model	CAD model	NURBS model	Cable-struts	Reflectional	Rotational	Cyclic	Axisymmetry	Translational	Dihedral	Global	Partial	Local	Exact	Approximate	Extrinsic	Intrinsic	Geometry-based	View-based	Explicit	Implicit	
Jiang et al. [25]	2013								●			●	●					●			●		●		●			●	-
Sipran et al. [46]	2014						●					●							●			●	●		●			●	Vote-based scheme. Models with large missing parts. Candidates generated by point pair matching.
Dang et al. [44]	2014									●		●						●	●			●	●		●			●	Retrieval, PCA
Korman et al. [41]	2015					●	●	●				●	●						●			●	●	●	●		●		-
Dang et al. [45]	2015									●		●						●				●	●		●			●	For alignment, Shape function
Stephenson et al. [57]	2015					●	●					●						●				●	●		●			●	PCA, Two procedures: (I) variation of Hausdorff distance, and (II) ray casting
Li et al. [30]	2016						●					●						●				●	●			●		●	Candidates from Continuous PCA, Viewpoint entropy
Schiebener et al.[88]	2016					●						●							●			●	●		●			●	Incomplete models. POSC generated from RANSAC.
Hruda & Dvorák [47]	2017						●					●							●			●	●		●			●	Incomplete models. POSC generated by pairing feature points (with the highest Gaussian curvature). Voting process for selection of APOS.
Jiang et al. [70]	2017								●			●	●					●			●		●		●			●	Symmetry detection in assembly CAD models.
Boussuge et al. [66]	2017								●				●	●	●			●	●	●	●		●		●			●	-
Hruda [62]	2018					●						●						●	●		●	●	●		●			●	Symmetry detection in incomplete models.
Tierney et al. [14]	2018								●				●	●	●			●	●	●	●		●		●			●	Input: B-rep CAD model. Symmetry-based decomposition for quasi-axisymmetric assembly CAD models.
Chen et al. [43]	2018										●	●	●	●			●	●			●		●		●			●	-

Table 4. Overview of prior CASD studies (Part 4)

RESEARCH	YEAR	INPUT										OUTPUT												APPR				Note
		2D				3D						TYPE OF TRANSF.						SCALE			ACC		D.M.						
		Image	View	CT or MRI	Wireframe	Point cloud	Mesh model	Voxel model	CAD model	NURBS model	Cable-struts	Reflectional	Rotational	Cyclic	Axisymmetry	Translational	Dihedral	Global	Partial	Local	Exact	Approximate	Extrinsic	Intrinsic	Geometry-based	View-based	Explicit	Implicit	
Li et al. [20]	2019								●			●	●			●				●	●		●		●		●		Input: B-rep CAD model. Congruence-labelled Adjacency Graph (CLAG) Frequent Sub-graph Mining (FSM)
Ji et al. [38]	2019					●						●						●				●	●		●			●	RANSAC
Gnutti et al. [89]	2021	●										●						●				●	●			●		●	-
Fotouhi et al. [37]	2020			●								●						●				●				●	●	●	RANSAC
Gothandaraman et al. [56]	2020					●						●						●				●	●			●	●		-
Hruda et al. [40]	2022					●						●						●	●			●	●		●			●	RANSAC

2.1.4 Relevant research gaps

After reviewing existing CASD studies related to 3D CAD models, this thesis has identified several research gaps. Firstly, past CASD studies have mainly focused on analytic surfaces such as plane, cylindrical, conical, spherical, and toroidal surfaces [20,63]. While some prior studies have attempted to include numeric surfaces, such as spline surfaces, only reflectional and cyclic symmetry were considered [23,42,66]. However, 3D CAD models often contain other types of numeric surfaces, such as extruded, swept, surface of revolution, and blend surfaces, which require further research in CASD. Additionally, while some studies have addressed partial symmetry [42,63], there is still a need for an adequate measure to detect partial symmetry. Overall, a unified CASD method is required to process exact global and partial reflectional and axisymmetric 3D CAD models with both analytic and numeric surfaces.

To address these research gaps, this doctoral thesis proposes a novel CASD method that can detect exact global and partial reflectional and axisymmetry in 3D CAD models with both analytic and numeric surfaces. Additionally, a symmetry measure is proposed to classify the type of symmetry in the B-rep CAD model. Specific software and programming languages, such as MATLAB, are commonly used for implementing CASD [40,43]. When using 3D CAD models as input, the CASD is typically conducted within a CAD system, which is discussed in the following section.

2.2 CAD systems for CASD

The CAD systems used to design 3D CAD models are categorised into commercial and open-source. Popular commercial CAD systems are AutoCAD, CATIA V5, Creo Parametric, Solidworks, NX, Solid Edge, and so on, while open-source CAD systems are FreeCAD, Open CASCADE, etc. CAD systems are also categorised as either 2D or 3D, with 2D systems being drawing-based and 3D systems being model-based. Most 3D CAD systems use a feature-based, parametric modelling approach, where design features are defined by various parameters such as sketch dimensions, extrude length, and revolve angle. The final shape of the 3D model is obtained step-by-step by turning 2D sketches into 3D features or by modifying existing geometry (Boolean operations, transformations such as mirroring, patterning, and so on). Features have been used as input for symmetry

detection [25]. Additionally, 3D CAD models can be divided into single parts and assemblies. A single part consists of one or more bodies, while an assembly comprises two or more parts. To date, CASD has been applied to both single part [16,42] and assembly [66] CAD models.

The 3D CAD models generated in CAD systems can be further divided into manifold and non-manifold. Manifold models are manufacturable objects, while non-manifold models are artificial objects that lack material thickness at certain edges or points [27] (Figure 5). In manifold models, an edge can only be shared by two faces, whereas non-manifold models may have an edge shared by more than two faces (Figure 5). CAD systems such as NX, Solidworks, and Solid Edge rely on the Parasolid geometric modelling kernel and only allow for the design of manifold models. On the other hand, CAD systems like CATIA V5, Autodesk Inventor, and Creo Parametric enable the creation of both manifold and non-manifold models. In the context of CASD, the manifold [23] and non-manifold models [63] were used as input. This doctoral thesis focuses on detecting symmetry in solid 3D CAD models consisting of single parts with one body and manifold geometry.

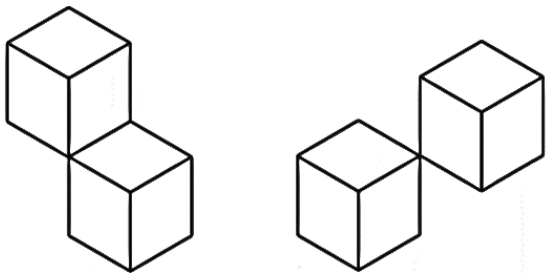


Figure 5. Examples of non-manifold models

CAD systems typically comprise several constituents [27], including graphical user interface, geometric modelling kernel, Application Programming Interface, databases, etc. (Figure 6).

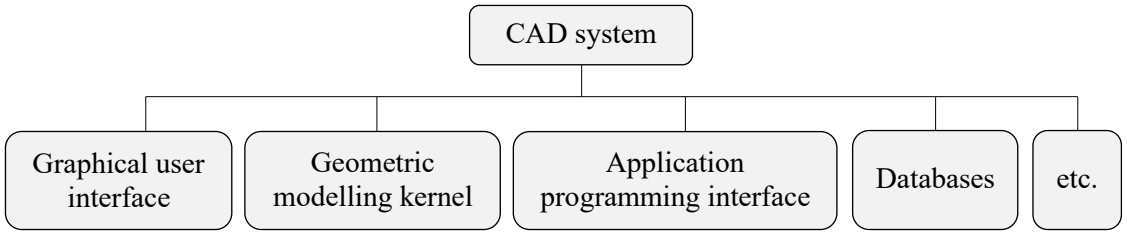


Figure 6. Basic constituents of CAD systems

The graphical user interface (GUI) enables the interaction between the user and the software. The layout and appearance of the GUI make every CAD software system unique and recognisable. The command toolbar, graphic window, and history tree are the main elements of the GUI. The command toolbar contains icons enabling access to different features and sketch operations. The graphic window provides the CAD models' visualisation and manipulation (translation, rotation, zoom-in and -out, etc.). The history tree records all modelling features, sketches, and operations undertaken to create the part, assembly, or drawing. In some CASD studies, the history tree was exploited for CASD [25,70]. The geometric modelling kernel is the heart of every CAD system, which creates, modifies, and maintains the shapes of the model [27]. There are several kernels in commercial CAD systems: ACIS, Parasolid, Convergence Geometric Modeler (CGM), ShapeManager, and C3D. Open CASCADE is a freely available open-source kernel besides commercial geometric modelling kernels. Until now, past CASD studies used CAD systems with ACIS [23,42], Parasolid [66], CGM [25], and Open CASCADE [16,63] geometric modelling kernels. Most CAD systems provide an API to access CAD system functionalities and develop custom applications that may enhance the CAD system possibilities. This doctoral thesis uses a CAD system API to implement the proposed CASD method. Thus, an overview of existing CAD system APIs is provided in the next subsection.

2.2.1 Application Programming Interface

The CAD system's Application Programming Interface can be used to develop custom applications and enhance the capabilities and functionalities of the CAD system. From the scientific research standpoint, the API can be utilised to implement methods and frameworks, write specific algorithms, and perform validations. For instance, the CAD systems APIs were used to develop tools for feature recognition in CAD models [90], idealisation of CAD models for finite element analysis (FEA) [91], automation of generating technical drawings [92], and so on. Three types of applications can be developed through the CAD system API: a) macros, b) stand-alone applications, or c) add-in applications. A macro is usually programmed using low-level programming languages, i.e., scripting languages such as VBA, VBScript, and CATScript. The programming of macros is relatively straightforward and quick. However, macros run in

the CAD system's memory space, which may affect the execution time of the macro. Add-in and stand-alone applications are developed using high-level programming languages, i.e., computer languages such as C#, C++, Fortran, and so on. An add-in application can be integrated within the CAD software environment and usually allows for the modification of the user interface of the CAD system. A stand-alone application has its own interface and does not require the user to work interactively with the CAD software. Most CAD system APIs are based on Microsoft's Component Object Model (COM) technology, where the basic API elements are various interfaces and objects with their properties and functions.

From the perspective of CASD, the APIs were frequently used for implementation purposes. For instance, the study reported in [63] utilised two CAD system APIs to develop a symmetry detection tool. The first was CATIA V5, and the second was Open CASCADE. The reason for having two APIs was that the API functionalities in CATIA V5 are relatively limited without special licensing. In particular, the lack of access to some of the data of the B-Rep should be highlighted. For instance, basic topological entities such as loops are inaccessible [91]. Within the study [63], researchers developed a CATIA V5 macro using Visual Basic for Application (VBA) to retrieve the basic parameters of analytic surfaces. The CATIA Application Architecture (CAA) is the API for developing user-specific applications in CATIA. CATIA V5 automation relies on several programming languages: CATScript, Visual Basic (VB), and C++. CATScript is a portable version and non-GUI-oriented programming language of Dassault Systemes, similar to VBScript (a subset of VB). An advantage of CATScript is its easy use and recording ability, but it has limited flexibility and difficulties in debugging. Several subsets of VB are provided within CATIA V5: Visual Basic for Application (VBA), Visual Basic 6.0 (VB6), and VBScript. VBA is directly hosted in CATIA V5 and provides a complete programming environment with an editor, debugger, and help object viewer. The scripting languages in CATIA provide limited access to automation objects. C++ is the primary language of CAA and is more powerful since it can overcome all the limitations of the scripting languages. The main disadvantage is its learning complexity, time-consuming development of applications, as well as a special licensing requirement [93]. Open CASCADE was the second CAD system used in the study [63] to overcome the mentioned limitations of CATIA V5 API functionalities in VBA. The initial input 3D

CAD model was exported as a STEP file, and the symmetry detection analysis was further conducted using Open CASCADE Technology (OCCT). The OCCT is a software development platform providing services for 3D surface and solid modelling, CAD data exchange, and visualisation [94]. The data structure of the OCCT library is compatible with STEP (ISO 10303-42) [95]. Most of OCCT functionality is available in the form of C++ libraries. The OCCT was used to develop specific applications in the CAD [96], Computer-aided manufacturing (CAM) [95], or CAE domain [97].

Instead of using two CAD systems, a more appropriate solution is to use one CAD system that provides complete access to the CAD model B-rep data structure. For example, the studies [14,66] used NX API to develop a CASD tool. The CAD system NX provides NX Open, a collection of APIs for creating custom applications through an open architecture using different programming languages (Visual Basic, Java, Python, C/C++, and C#) [98]. It allows the integration of third-party applications and customisation of the NX interface. For users with low-level programming skills for the automation of simple repetitive tasks, NX offers Simple NX Application Programming (SNAP) [99]. SNAP is based on the Visual Basic (VB.Net) language and can be used with the NX Journal Editor or Visual Studio. Some operations in NX may be automated with Graphics Interactive programming (GRIP). It can sometimes perform advanced, customised operations more efficiently than interactive NX [100]. GRIP uses a vocabulary of English-like words, which makes it similar in many ways to interpretive BASIC or FORTRAN. Knowledge Fusion [101] is an interpreted, object-oriented language that allows the addition of engineering knowledge to a part by creating rules, which are the basic building blocks of the language. The language is declarative rather than procedural, meaning the rules are generally only evaluated when referenced or demanded. In addition, external knowledge bases, such as databases or spreadsheets, may also be accessed.

The research in [23] developed a stand-alone application as an assembly-oriented CAD environment with an included symmetry detection tool. The environment was built directly at the ACIS geometric modelling kernel level using C++. Other CAD systems with API applicable for CASD are Creo Parametric, Autodesk Inventor, Solid Edge, and Solidworks. Creo Parametric does not have a built-in VBA for macros, but it offers VB libraries with limited functions to access the CREO functionalities. Thus, macros can be written in external environments such as MS Office VBA or Visual Studio VB.NET.

Another way to implement design automation is by using Mapkeys[7]. Maximum automation functionality in Creo Parametric can be accessed with the APIs Creo Parametric TOOLKIT, J-link, and Pro/Web.Link for Web Environment. While J-link and Pro/Web.Link are freely available, CP TOOLKIT requires special licensing to develop, build, and test applications (PTC Creo, 2016). TOOLKIT relies on C or C++ programming language, J-Link makes it possible to develop Java programs, while Web.Link uses the Netscape Web browser to build custom applications. The CAD system Autodesk Inventor exposes its programming interface using the Microsoft COM automation interface. The interface may be accessed using programming languages such as Microsoft Visual C++, VB, C#, and Delphi. Autodesk Inventor provides a built-in VBA editor to develop macros. Add-ins and stand-alone (EXE) applications are developed using the .NET-based languages and libraries with Visual Studio. The iLogic is a unique add-in based on the VB.NET language included within all Autodesk Inventor packages. The iLogic interface gives the user access to almost all API functionalities. However, it lacks direct debugging capabilities – stepping through the code line-by-line as it executes is impossible. In Solid Edge, macros can be developed through Excel VBA, while add-ins with the Microsoft frameworks VB.NET (C#) or Visual Studio (C++). Also, any programming or scripting language that supports COM can use the Solid Edge COM API. The different application types that can be developed using Solidworks API are macros, stand-alone, and add-in applications. The programming languages provided are VBA, Visual Basic .NET (VB.NET), Visual C++/CLI, Visual C# .NET, Visual C++ 6.0. One of the Solidworks API's main advantages compared to other CAD systems is the accessibility of almost all automation objects at the low-level programming languages without special licensing [102]. To summarise, the CAD system API functionalities were often exploited in CASD for different input 3D CAD models. Alternatively, when the input 3D CAD model is in STEP format, the information of the STEP file can be accessed through Java standard data access interface (JSDAI) [103], which is an open-source API to read, write and execute runtime manipulation of the EXPRESS-based data model [90]. For instance, JSDAI was used in the CAM domain to develop a STEP-based feature

[7] A Mapkey is a keyboard macro that maps frequently used command sequences to specific keyboard keys.

recognition system for recognising B-spline surface features [90]. It could also be used in the context of CASD as it provides access to the B-rep data structure.

Finally, several requirements should be considered when selecting an appropriate CAD system API. The first factor is the type of application that should be developed (macros, stand-alone applications, or add-in applications). Further, the type of programming language should be used and its availability within the API. Finally, the required accessibility of the CAD system functionalities via the API should be considered. In particular, the accessibility of the B-rep data structure is vital for CASD. Thus, the following subsection discusses the basic concept of B-rep.

2.2.2 Boundary representation

The B-rep derived from the 3D CAD model has been utilized as a reliable input for symmetry detection in numerous CASD studies [23,42,66]. Its versatility has been demonstrated in various other studies such as clustering of 3D CAD models [104], feature recognition for CAM [105], or model simplification for CAE [91]. The B-rep technique is dominantly used for representing and exchanging solid CAD models [27,28]. Alternative modelling techniques for representing solids, such as Octrees and Constructive Solid Geometry (CSG), have several disadvantages over B-rep. Octrees approximate the shape by many voxels, where larger-sized voxels make up the interior of the solid, while smaller-sized voxels make up the boundary. Although Octrees are easy to generate, they may be extremely inefficient for highly accurate models [106]. Thus, they are not accurate enough for exact symmetry detection. CSG models are compounds of simple primitive solids (cuboids, cylinders, prisms, pyramids, spheres, and cones) merged through Boolean operations (union, intersection, and difference). Although the CSG technique contains the geometry of a solid, the major drawback is that the topology is not present. The B-rep technique (Figure 7) overcomes this drawback as its data structure is composed of two main parts: topology and geometry [107].

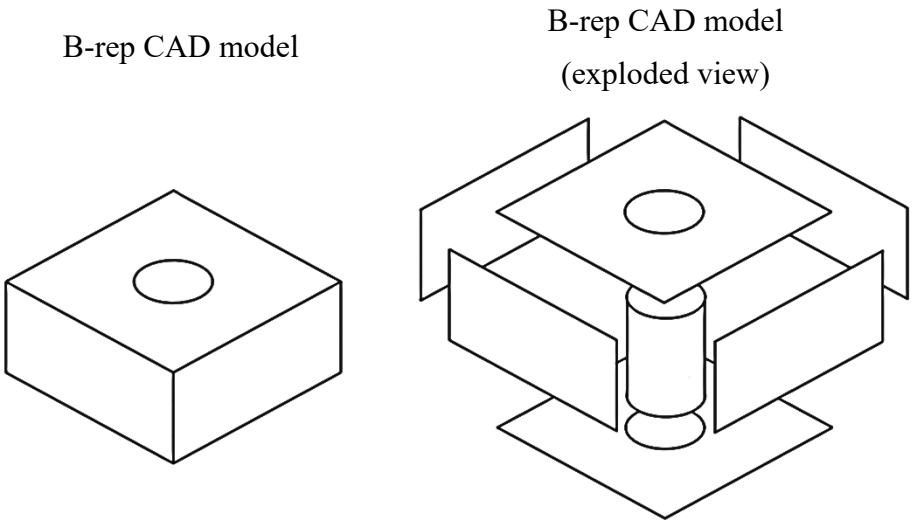
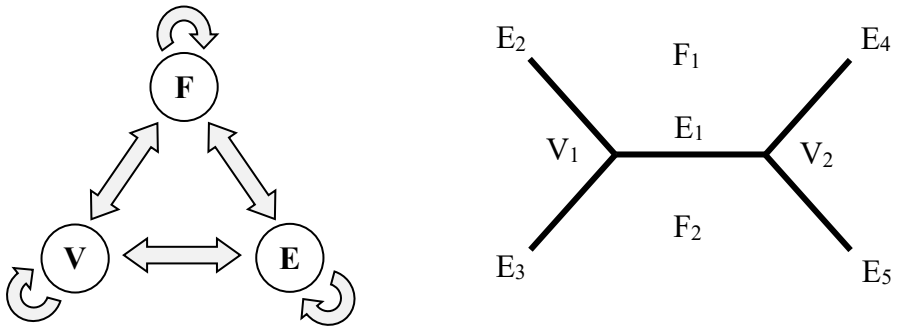


Figure 7. B-rep technique

The topology defines the structure of the model and provides information about the adjacency or connectivity between the three basic types of entities: faces, edges, and vertices. Edges bound faces, while vertices are points connecting several faces and edges. The topology of a solid has nine potential sets of pointers connecting vertices, edges, and faces (Figure 8, left). However, fewer pointers are stored as a trade-off between the speed of querying each topological entity and the space required to store the topological information. Hence, topology is most often described by the winged-edge data structure [108] (Figure 8, right). For solids bounded by two-dimensional manifolds, each edge lies on two faces, and each edge typically connects two distinct vertices.



F – face **E** – edge **V** – vertices

Figure 8. Potential set of pointers for a topological data structure (left) and winged-edge data structure (right)

The geometry defines the model's shape and consists of three basic types of entities (Figure 9): surfaces, curves, and points. Each surface is associated with a face, each curve with an edge, and each point with a vertex. The face is a portion of its associated surface,

the edge is a segment of its associated curve, and the vertex coincides with its associated point. Each geometric entity is defined by a set of properties describing the associated topological entity. For example, the point defines the position of its associated vertex in the modelling space.

Depending on the CAD system and its geometric modelling kernel, the B-rep data structure may contain other topological entities such as (e.g., shells, loops, and coedges). A shell represents a closed set of faces. A loop is a closed set of edges bounding a face. A coedge is an oriented edge whose orthogonal vector (cross product between the face normal vector and the edge's tangent vector) points in the direction of the interior of the face. In manifold CAD models, each edge can only be shared by two adjacent faces, which means there can only be two edges (each pointing in the opposite directions along the edge). In non-manifold CAD models, an edge can be associated with more than two faces and can have more than two coedges.

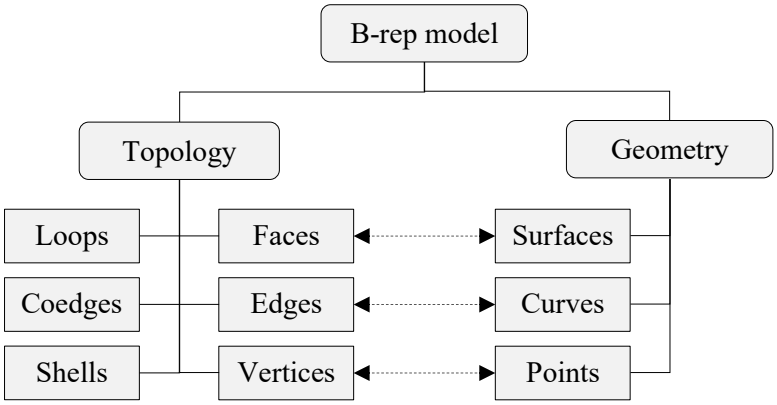


Figure 9. Example of a basic B-rep model data structure

Keeping the geometry and topology separate gives more flexibility during modelling operations [28]. Moreover, such an approach enables checking the B-rep model's topological integrity at any time using Euler's equation. In its simplest form, the Euler equation states that the number of vertices V and faces F is equal to the number of edges E increased by two:

$$V+F=E+2. \tag{14}$$

The extended version of the Euler equation applicable for the general case includes as well other contributors such as the number of genus G (the number of holes going through the solid), the number of inner face loops L, and the number of shells S:

$$V–E+F–L=2(S–G). \quad (15)$$

The B-rep data structure depends on the CAD system and the underlying geometric modelling kernel. CAD systems using the same geometric modelling kernel are likely to have a similar B-rep data structure (see Section 2.4). Figure 10 shows some of the basic types of geometric entities used in CAD systems. Typical types of surfaces are plane, cylinder, cone, sphere, toroidal, B-spline surface, etc., while typical curve types are line, circle, ellipse, B-spline curve, etc. As already mentioned, the geometry within a 3D CAD model can be divided into analytic and numeric [28,31], which is discussed in more detail in the following section.

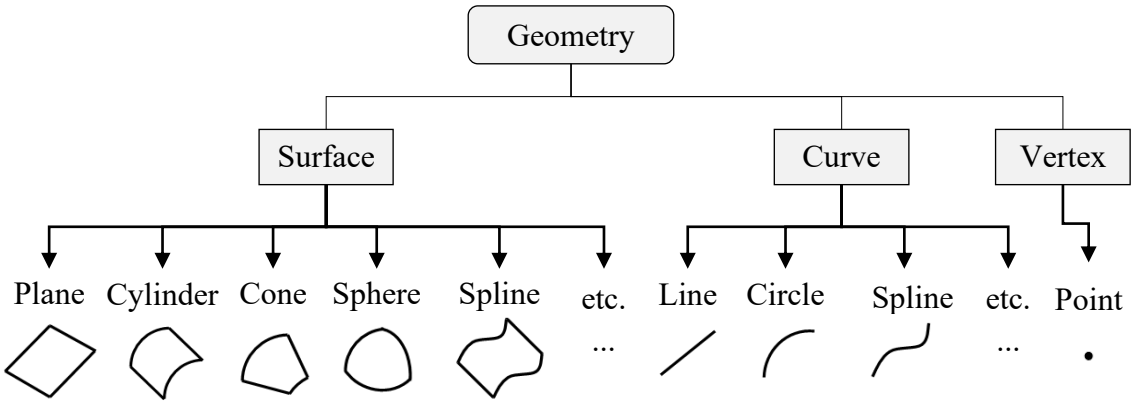


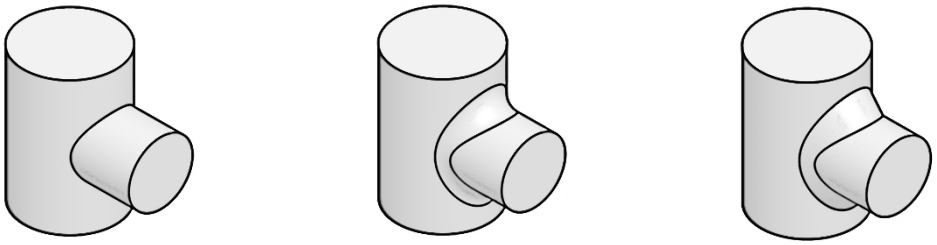
Figure 10. Basic types of geometric entities

2.3 Analytic and numeric geometry in B-rep

Analytic geometry can be divided into analytic surfaces and curves. There are five analytic surfaces: plane, cylindrical, conical, spherical, and toroidal surfaces. Typical analytical curves are line, circle, ellipse, hyperbola, and parabola. All other surfaces and curves that do not fall into the category of analytic surfaces and curves can be classified as numeric. Hence, typical numeric surfaces found in B-rep CAD models are spline surfaces, surfaces of revolution (or revolved surfaces), swept surfaces, blend surfaces, etc. The numeric curves are spline, p curve, trimmed curve, intersection curve, etc. The prior studies related to CASD in B-rep CAD models were mainly limited to analytic surfaces (plane, cylindrical, conical, spherical, and toroidal surfaces) [16,23,42,63,67,68,69]. Until now, numeric surfaces were briefly considered in the

context of CASD (as discussed in Subsection 2.1.1, only spline surfaces were considered). Therefore, further research is needed to include other types of numeric surfaces in CASD.

During feature-based modelling, analytic geometry is created within the CAD model from solid primitives, i.e., when analytic 2D curves are extruded, revolved, and so on. On the other side, numeric geometry may be created within CAD systems in two ways: automatically or intentionally. Automatically implies that the CAD system creates numeric geometry in the background during modelling without user's knowledge. A typical example is when fillet or chamfer features are used to round off or break edges. Consequently, by default, the CAD system may create either a blended or a spline surface (Figure 11). On the other hand, when the user needs to design 3D CAD models with complex shapes (fuselage or wing of an airplane, car body, turbine blade, etc.), free-form surface modelling is applied, and numeric geometry is created intentionally by the user.



(a) two cylinders with different diameters

(b) fillet/round feature (blend or spline surface)

(c) chamfer feature (spline surface)

Figure 11. A two-cylinder test – different features applied to the intersection edge.

There are three common methods to represent geometry in CAD systems [31]:

- implicit equations,
- parametric representation, and
- procedural definitions.

Using implicit equations, it is difficult to compute a point on a curve or surface while using parametric representation, it is difficult to determine if a given point lies on the curve or surface. Hence, all analytic curves and surfaces are also defined by parametric representations. The two most common techniques of representing curves and surfaces in geometric modelling are implicit equations and parametric representations [31,106,109]. The implicit equation of a surface in Euclidean space R^3 is defined by the equation:

$$S(x, y, z) = 0, \tag{16}$$

while those of a curve, respectively:

$$S(x, y) = 0. \tag{17}$$

On the other hand, the parametric representation of surfaces is defined as follows:

$$\boldsymbol{\sigma}(u, v), \tag{18}$$

where u and v are independent dimensionless parameters. The parametric representation of a curve is:

$$\boldsymbol{\lambda}(u). \tag{19}$$

A procedural definition of a curve or surface implies that a base curve or surfaces and a procedure or formula are used to compute points on the intended geometry from points on the base geometry [31]. For instance, offset and blend surfaces may be represented procedurally [106]. The main disadvantage of procedurally defined surfaces is that they usually cannot be transferred to other CAD systems that do not understand this definition [110]. The next subsection provides the implicit equations and parametric representation of surfaces and curves from the perspective of STEP and Parasolid [109,111]. The next two sub-section provide mathematical expressions for different types of analytic and numeric surfaces, which are important to compute surface and edge properties relevant for the proposed CASD method.

2.3.1 Surfaces

Surfaces are essential as they describe the shapes of the associated faces in the B-rep CAD model. The types of surfaces discussed in this section are the analytic surfaces plane, cylindrical, conical, spherical, and toroidal, and numeric surfaces spline, swept, and blend surfaces. A plane is an unbounded surface with a constant normal and is defined by a point on the plane and the normal direction. The plane surface has the following parameterisation:

$$\boldsymbol{\sigma}(u, v) = \mathbf{C} + u\mathbf{x} + v\mathbf{y}, \tag{20}$$

where **C** is the position location, while **x** and **y** represent the directions of the local coordinate system of the plane. The parametrisation ranges $-\infty < u, v < \infty$. The implicit form of the plane surface is defined as:

$$S(x, y, z) \equiv ax + by + cz + d = 0. \tag{21}$$

A cylindrical surface is a surface at a constant distance (the radius) from a straight line and is defined by its radius R and its orientation and location. The cylindrical surface has the following parameterisation:

$$\boldsymbol{\sigma}(u,v) = \mathbf{C} + R\left(\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y}\right) + v\mathbf{z}, \tag{22}$$

where R is the cylinder radius, $\mathbf{z}$ is the axis of the cylindrical surface. The parametrisation range is $0 \le u \le 360^{\circ}$ and $-\infty < v < \infty$. The surface unit normal vector can be obtained from the equation:

$$\mathbf{n}(u,v) = \left(\cos\left(u\right)\right)\mathbf{x} + \left(\sin\left(u\right)\right)\mathbf{y}. \tag{23}$$

The implicit form of the cylindrical surface is defined as:

$$S(x,y,z) \equiv x^2 + y^2 - R^2 = 0. \tag{24}$$

A conical surface is created when a line in 3D space is revolves around an intersecting line. It is defined by the half-angle α, location, orientation, and the radius R_c of the cone in the plane passing through the location point $\mathbf{C}$ normal to the cone axis. The conical surface has the following parameterisation:

$$\boldsymbol{\sigma}(u,v) = \mathbf{C} + \left(R_c + v\tan\alpha\right)\left(\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y}\right) + v\mathbf{z}, \tag{25}$$

where α is the half-angle, and R the radius. The parametrisation range is $0 \le u \le 360^{\circ}$ and $-\infty < v < \infty$. The surface unit normal vector can be obtained using the equations:

$$\mathbf{n}(u,v) = \frac{\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y} - \left(\tan\alpha\right)\mathbf{z}}{\sqrt{1 + \left(\tan\alpha\right)^2}}, \tag{26}$$

$$\mathbf{n}(u,v) = -\frac{\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y} - \left(\tan\alpha\right)\mathbf{z}}{\sqrt{1 + \left(\tan\alpha\right)^2}}. \tag{27}$$

The first equation above applies to the case when $\left(R_c + v\tan\alpha\right) > 0$, while the second equation applies when $\left(R_c + v\tan\alpha\right) < 0$. The implicit form of the conical surface is defined as:

$$S(x,y,z) \equiv x^2 + y^2 - \left(R_c + z\tan\alpha\right)^2 = 0. \tag{28}$$

A spherical surface is a surface that has a constant distance (radius) from a central point and is described by the radius, location, and orientation of the surface. The spherical surface has the following parameterisation:

$$\boldsymbol{\sigma}(u,v) = \mathbf{C} + R_{\mathrm{s}} \cos v\left(\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y}\right) + R_{\mathrm{s}}\left(\sin v\right)\mathbf{z}, \tag{29}$$

where **z** is the axis and R_{s} is the radius of the spherical surface. The parametrisation range is $0 \le u \le 360^{\mathrm{o}}$ and $-90^{\mathrm{o}} < v < 90^{\mathrm{o}}$. The surface unit normal vector can be obtained using the equation:

$$\mathbf{n}(u,v) = \cos v\left(\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y}\right) + \left(\sin v\right)\mathbf{z}. \tag{30}$$

The analytic form of the spherical surface is defined as:

$$S(x,y,z) \equiv x^2 + y^2 + z^2 - R_{\mathrm{s}}{}^2 = 0. \tag{31}$$

A toroidal surface is created when a circle is revolved around a line in its plane. The radius of the revolved circle is called the minor radius, and the distance from the circle's centre to the axis of revolution is called the major radius. Therefore, the major and minor radius, position, and orientation describe the toroidal surface. The toroidal surface has the following parameterisation:

$$.\boldsymbol{\sigma}(u,v) = \mathbf{C} + \left(R_{\mathrm{t}} + r_{\mathrm{t}} \cos v\right)\left(\left(\cos u\right)\mathbf{x} + \left(\sin u\right)\mathbf{y}\right) + r_{\mathrm{t}}\left(\sin v\right)\mathbf{z}, \tag{32}$$

where **z** is the axis of the toroidal surface, while R_{t} and r_{t} are the major and minor radii. The parametrisation ranges from $0 < u, v < 360^{\mathrm{o}}$. The implicit form of the toroidal surface is defined as:

$$S(x,y,z) \equiv x^2 + y^2 + z^2 - 2R_{\mathrm{t}}\sqrt{\left(x^2 + y^2\right)} - r_{\mathrm{t}}^2 + R_{\mathrm{t}}^2 = 0. \tag{33}$$

Next, the numeric surfaces are discussed, starting with spline surfaces. Several types of spline surfaces exist, such as Bezier, B-spline, NURBS, etc. Regardless of the name, a spline surface consists of several patches pieced together with some form of continuity [31]. Non-uniform rational B-splines (NURBS) have become the de facto industry standard for the representation, design, and data exchange of geometric information [112,113]. NURBS are generalisations of nonrational B-splines and rational and nonrational Bezier curves and surfaces. B-spline surfaces are a general form of the rational or polynomial parametric surface represented by control points, basis functions,

and optional weights. The common subtypes are B-spline surface with knots, uniform surface, quasi-uniform surface, and Bezier surface [109]. The B-spline surface has the following parametrisation in the polynomial case:

$$\mathbf{\sigma}(u,v) = \sum_{i=0}^{K_1} \sum_{j=0}^{K_2} P_{ij} N_i^{d_1}(u) N_j^{d_2}(v), \tag{34}$$

In the rational case, the B-spline surface has the following parametrisation:

$$\mathbf{\sigma}(u,v) = \frac{\sum_{i=0}^{K_1} \sum_{j=0}^{K_2} w_{ij} P_{ij} N_i^{d_1}(u) N_j^{d_2}(v)}{\sum_{i=0}^{K_1} \sum_{j=0}^{K_2} w_{ij} N_i^{d_1}(u) N_j^{d_2}(v)}, \tag{35}$$

where K_1 is the upper index on u control points, K_2 the upper index on v control points, P_{ij} represents the control points, w_{ij} the weights, d_1 the u degree, d_2 the v degree, and $N_i^{d_1}(u)$ and $N_i^{d_2}(v)$ the B-spline basis functions. A B-spline surface with knots is a type of B-spline surface in which the knot values are explicitly given and is used to represent non-uniform B-spline surfaces but may also be used for other knot types. A uniform surface is a type of B-spline surface in which the knots are evenly spaced (only if all knots are of multiplicity and they differ by a positive constant from the preceding knot). A quasi-uniform surface is a type of B-spline surface in which the knots are evenly spaced and, except for the first and last, have multiplicity 1. Bezier surface is a type of B-spline surface in which the knots are evenly spaced and have high multiplicities. A rational B-spline surface is a type of B-spline surface that is a piecewise parametric rational surface described in terms of control points, associated weight values, and basis functions. It is instantiated with any other subtypes of the B-spline surface, which provide explicit or implicit knot values from which the basis functions are defined.

Next, the swept surface is constructed by sweeping a curve along another curve [109]. The common subtypes are surface of linear extrusion, surface of revolution, surface curve swept surface, and fixed reference swept surface. A surface of linear extrusion is created when a curve is swept in some direction. The parametrisation is as follows:

$$\mathbf{\sigma}(u,v) = \boldsymbol{\lambda}(u) + v\mathbf{V}, \tag{36}$$

where $\boldsymbol{\lambda}(u)$ is the curve's parametrisation, and $\mathbf{V}$ is the extrusion axis. The parametrisation range for v is $-\infty < v < \infty$, while for u it is defined by the curve parametrisation.

Further, a surface of revolution is generated when a curve is rotated one complete revolution about an axis. The parametrisation is defined as:

$$\begin{aligned}\boldsymbol{\sigma}(u,v) = \mathbf{C} + \left(\boldsymbol{\lambda}(v) - \mathbf{C}\right)\cos u + \left(\left(\boldsymbol{\lambda}(v) - \mathbf{C}\right)\mathbf{V}\right)\mathbf{V}\left(1 - \cos u\right) + \\ + \mathbf{V} \times \left(\boldsymbol{\lambda}(v) - \mathbf{C}\right)\sin u,\end{aligned} \tag{37}$$

where $\boldsymbol{\lambda}(v)$ is the curve's parametrisation, $\mathbf{C}$ is the position location, and $\mathbf{V}$ is the axis of revolution.

A surface curve swept surface and a fixed reference swept surface are types of swept surfaces created when a curve is swept along a directrix curve that lies on the reference surface [109]. Both surface types have the following parametrisation:

$$\boldsymbol{\sigma}(u,v) = \boldsymbol{\mu}(u) + \mathbf{T}(u)\boldsymbol{\lambda}(v), \tag{38}$$

where $\boldsymbol{\mu}(u)$ is the directrix parametrisation, $\boldsymbol{\lambda}(v)$ the swept curve parametrisation, and $\mathbf{T}(u)$ transformation matrix at parameter u. The u parameter range depends on the directrix curve, while the v parameter range depends on the referenced swept curve. The difference between the surface curve swept surface and fixed reference swept surface is that the first has a local X axis in the direction of the normal to the reference surface. In contrast, the second has a local X axis in the direction of the projection of fixed reference onto the normal plane to the directrix at this point [109].

Surface blending, usually through fillet or round features, is widely used in CAD for functional or aesthetic reasons and implies the smooth joining of two or more given surfaces [114]. As a result of surface blending, different types of surfaces may be generated (e.g., cylindrical, spherical, toroidal, blend, etc.), depending on the type of edges or faces to which the blending is applied. During blending, the specific type of surface that may be created is the blend surface (see Figure 11, b). Blend surfaces may be defined procedurally [106,115] or parametrically [116]. Procedurally defined blend surfaces can only be passed into another CAD system with approximation [31]. For instance, CAD systems relying on the Parasolid geometric modelling kernel may exchange blend surfaces via the kernel file format without approximation. However, during the export of blend surfaces to any other neutral file format, they will be

approximated by B-spline surfaces. The parametrically defined blend surface is usually represented by a Bezier or B-Spline surface [117]. The construction of a parametrically defied blend is typically based on surface intersection [115], interpolation [114], and trimming [117]. A blend surface may have the following parametrisation [111]:

$$\boldsymbol{\sigma}(u,v) = \mathbf{C}(u) + R_{\mathrm{b}} \cos\left(v\, a\left(u\right)\right)\mathbf{x}\left(u\right) + R_{\mathrm{b}} \sin\left(v\, a\left(u\right)\right)\mathbf{y}\left(u\right), \tag{39}$$

where $\mathbf{C}(u)$ is the spine curve, R_{b} is the blend radius, $\mathbf{x}(u)$ and $\mathbf{y}(u)$ are unit vectors such, and $a(u)$ is the angle subtended by points on the boundary curves at the spine. The spine is the blend's centre line or the path along which the centre of the ball moves.

The next numeric surface type is the offset surface. An offset surface is mathematically complex and can rarely be represented precisely in NURBS form [118]. Hence, the offset surface is another example of a procedurally defined surface. It has the following parameterisation:

$$\boldsymbol{\sigma}\left(u,v\right) = \mathbf{S}\left(u,v\right) + d\mathbf{n}, \tag{40}$$

where $\mathbf{S}(u,v)$ represents the basis surface, $\mathbf{n}$ is the unit normal vector to the basis surface, and d is the distance, i.e., a constant scalar, which can be positive, negative, or zero to indicate the preferred side of the surface [118]. The introduced equations are used to compute the surface properties, such as the normal vector, centroid, area, etc., for the proposed CASD method (see Chapter 3). Besides surfaces, another vital topological entity is curves, which describe the shape of the associated edges in the B-rep CAD model.

2.3.2 Curves

The common curves in B-Rep are the line, conics (circle, ellipse, hyperbola, and parabola), Pcurve, B-spline curve, etc. A line is defined by a point and a direction, parametrised by the following equation:

$$\boldsymbol{\lambda}\left(u\right) = \mathbf{P} + u\mathbf{V}, \tag{41}$$

where $\mathbf{P}$ is a point and $\mathbf{V}$ is the direction vector. The u parameter ranges from $-\infty < u < \infty$. Conics are a group of curves produced by intersecting a plane with a cone. The first type of conic is the circle, which is defined by a radius R and the location and orientation of the circle. It is parametrised by the equation:

$$\boldsymbol{\lambda}(u) = \mathbf{C} + R\left(\cos(u)\mathbf{x} + \sin(u)\mathbf{y}\right). \tag{42}$$

The implicit form of the circle is defined as:

$$S(x, y) \equiv x^2 + y^2 - R^2 = 0. \tag{43}$$

Another conic, ellipse, is defined by a half-major R_1 and half-minor R_2 radius, the position and orientation of the curve. It is parametrised by the next equation:

$$\boldsymbol{\lambda}(u) = \mathbf{C} + R_1\left(\cos(u)\mathbf{x}\right) + R_2\left(\sin(u)\mathbf{y}\right). \tag{44}$$

The parameter u ranges from $0 \leq u \leq 360^{\circ}$. The implicit form of the ellipse is defined as follows:

$$S(x, y) \equiv \left(\frac{x}{R_1}\right)^2 + \left(\frac{y}{R_2}\right)^2 - 1 = 0\,. \tag{45}$$

The hyperbola is a conic determined by the lengths of the major R_1 and minor R_2 radius, the curve's position and orientation:

$$\boldsymbol{\lambda}(u) = \mathbf{C} + R_1\left(\cosh(u)\mathbf{x}\right) + R_2\left(\sinh(u)\mathbf{y}\right). \tag{46}$$

The parameter u ranges $-\infty < u < \infty$. The hyperbola has the following implicit form:

$$S(x, y) \equiv \left(\frac{x}{R_1}\right)^2 - \left(\frac{y}{R_2}\right)^2 - 1 = 0\,. \tag{47}$$

The last conic, parabola, is described by its focal distance F, position, and orientation, with the following parametrisation:

$$\boldsymbol{\lambda}(u) = \mathbf{C} + F\left(u^2\mathbf{x} + 2u\mathbf{y}\right). \tag{48}$$

The parameter u ranges $-\infty < u < \infty$. The parabola's implicit form is:

$$S(x, y) \equiv 4Fx - y^2 = 0 \tag{49}$$

A B-spline curve is a piecewise parametric polynomial or rational curve described in terms of control points and basis functions. The B-spline curve has been selected as the most stable format to represent all types of polynomial or rational parametric curves. With appropriate attribute values, it can represent single-span or spline curves of explicit polynomial, rational, Bezier or B-spline curve types. The B-spline curve has three special

subtypes where the knots and knot multiplicities can be derived to provide simple default capabilities. The B-spline curve has the following parametrisation in the polynomial case:

$$\boldsymbol{\lambda}(u) = \sum_{i=0}^{k} P_i N_i^d(u) \,. \tag{50}$$

The B-spline curve has the following parametrisation in the rational case:

$$\boldsymbol{\sigma}(u,v) = \frac{\sum_{i=0}^{k} w_i P_i N_i^d(u)}{\sum_{i=0}^{k} w_i N_i^d(u)} \,, \tag{51}$$

where k+1 is the number of control points, P_i are the control points, w_i are the weights, and d is the degree. The introduced equations for several analytic and numeric curves are important for computing the curve properties, such as the length, midpoint, and so on, and are used in the proposed symmetry detection method (see Chapter 3). The type of surfaces and curves available depends on the observed CAD system (geometric modelling kernel) and CAD model formats. Thus, the next subsection gives an overview of CAD model formats applicable as input for CASD, alongside their type of surfaces and curves.

2.4 CAD models formats for CASD

The 3D CAD model file formats can generally be divided into native, kernel, and neutral (see Table 5). The native 3D CAD file formats are strictly related to the corresponding CAD systems (e.g., Solidworks *.sldprt, CATIA V5 *.CATpart, NX *.prt, etc.). Most CAD systems also provide the export of kernel exchange file formats (e.g., Parasolid and ACIS). Neutral file formats enable interoperability between different CAD systems and sharing 3D CAD models among different contributors (CAD, CAE, CAPP, and CAM). The most common neutral file formats are STEP, IGES, JT[8], QIF[9], etc. The 3D CAD model formats can be divided based on whether they use the B-rep technique (see Table 5). This is important as prior CASD research often exploited the B-rep. The past CASD studies used as input neutral STEP [16,63], kernel ACIS file [23,42], and native files CATIA V5 [25,70] and NX [66].

[8] Jupiter Tessellation
[9] Quality Information Framework

Table 5. An overview of different 3D CAD model formats

Format	Extension	Managed by	Standard	Native	Kernel	Neutral	B-rep	Applicable for CASD
STEP	.step, .stp	ISO	ISO 10303-242:2020	×	×	✓	✓	✓
IGES	.iges, .igs	ANSI	NBSIR 80-1978	×	×	✓	✓	✓
Parasolid	.x_t, .x_b	Siemens	-	×	✓	×	✓	✓
ACIS	.sat, .sab .asat, .asab	Dassault Systemes	-	×	✓	×	✓	✓
JT	.jt, .j_t	Siemens	ISO 14306:2017	×	×	✓	✓	✓
QIF	.qif	DMSC	ISO 23952:2020	×	×	✓	✓	×
STL	.stl	3D Systems	-	×	×	✓	×	×
3D PDF	.pdf	3D PDF Consortium	ISO 14739-1:2014	×	×	✓	×	×
eDrawings	.eprt, .easm	Dassault Systemes	-	×	×	✓	×	×
Solidworks	.sldprt .sldasm	Dassault Systemes	-	✓	×	×	✓	✓
Catia V5	.CATpart .CATproduct	Dassault Systemes	-	✓	×	×	✓	✓
NX	.prt	Siemens	-	✓	×	×	✓	✓
Creo Parametric	.prt	PTC	-	✓	×	×	✓	✓
Autodesk Inventor	.ipt, .iam	Autodesk	-	✓	×	×	✓	✓
Open CASCADE Technology	.brep	Open CASCADE	-	✓	×	×	✓	✓

✓ – yes × – no

The smooth exchange of 3D CAD models between different CAD systems is important to ensure reliable product definitions and avoid potential repairing of the geometry. The interpretation of the CAD model is also important factor to consider from the perspective of CASD. As already mentioned, there are three commonly used ways 3D CAD models may be exchanged. The first way is through native file formats, which implies that the

exchange is conducted between two identical CAD systems. This type of exchange is generally without any difficulties; the B-rep data structure and all other relevant CAD information (modelling history, features, parameters, and constraints) can be fully maintained. The only issue that may arise is when exchanging different CAD system versions. For instance, importing a native file created in a higher version into a lower version of the CAD system is usually impossible. The second way of exchanging is by employing kernel file formats such as Parasolid and ACIS. The B-rep data structure remains unchanged if the exchange is being conducted between two CAD systems that share the same geometric modelling kernel. However, the relevant CAD information will be lost. The second scenario is when the kernel file format is imported by a computer-aided x (CAx) system that uses another geometric modelling kernel. This may change the B-rep data structure in terms of geometry, while the CAD information will also be lost. The third way of exchanging CAD models between CAD systems, probably the most common, is through neutral file formats (STEP and IGS). Here, two aspects need to be considered. The first aspect is how one CAD system converts the 3D CAD model into a neutral file format. For instance, the CAx system may support some geometry that is not supported by the neutral file format (e.g., blend surfaces). In this case, the geometry might be approximated. The second aspect is how the other CAD system interprets the generated neutral file format. Mainly, there is a risk of inconsistency between geometry and topology of the CAD model when exchanging from a CAD system with loose numerical tolerances to a system with more stringent criteria [119,120]. Consequently, gaps may arise between vertices and edges, leading to errors and invalid topology and geometry. The neutral file formats can also not share other relevant CAD information such as features, reference geometry (planes, axes, and points), etc.

When exchanging a curve or surface between two CAD systems which represent geometry differently, there are three possible scenarios [31]: a) a mathematically precise conversion, b) a geometrically precise conversion with altered parameterisation, and c) no mathematically precise conversion between the two systems. A mathematically precise conversion means that the transformation is geometrically and parametrically equivalent between the two CAD systems. The only error that might occur is the floating point roundoff error. If two CAD systems support representations of analytic geometry, then the exchange of analytic geometry between them is precise. Surfaces and curves that

are defined procedurally are usually complex (e.g., offset curve and surface, blend surface, etc.) and can rarely be exchanged between CAD systems without approximation in NURBS form. The conversion of a particular NURBS representation into an equivalent type of spline curve or surface, and vice versa, requires the consideration of three aspects: degree, rationality, and continuity. When it comes to the degree, it cannot be decreased, but it can be increased. Non-rational geometry is a sum of polynomials, while rational geometry is a ratio of sums of polynomials. Hence, rational geometry cannot be transferred to CAD systems supporting only non-rational geometry, while vice versa is possible. As an example, two CAD systems are observed: the first allows non-rational geometry and restricts the degree of the curve or surface, and the second allows rational geometry without restricting the degree of the curve or surface. If geometry is passed from the first to the second CAD system, the corresponding curves or surfaces must be approximated. On the other hand, the conversion in the reverse direction puts no restrictions on the geometry. The conversion of type (parametric C or geometric G) and order (C0, C1, C2, G0, G1, G2, etc.) of continuity between the curve segments or surface patches is a more complex topic, and the following points need to be considered [31]:

- If a CAD system allows arbitrary knot spacing and multiplicity, it can represent precisely (geometrically and parametrically) virtually all polynomial and rational spline curves, which are at least C^0 continuous.
- Multiple knots are necessary for *G* continuous curves, but if desired, re-parameterisation can be used to achieve *C* continuity and lower the multiplicity of internal knots.
- The use of re-parametrised curves in certain types of surface constructions (e.g., ruled surfaces) can change surface geometry.

As many existing CASD studies, among them the CASD method proposed in this doctoral thesis, rely on the classification of topological entities of the B-rep, in the next subsections, an overview of different types of geometrical entities is given for native, kernel and neutral file formats.

2.4.1 Native formats

The native file formats are observed in the context of CATIA V5, which uses the Convergence Geometric Modeler (CGM) geometric modelling kernel, and Autodesk

Inventor, which relies on the ShapeManager geometric modelling kernel. Table 6 shows the comparison of geometry in CGM and ShapeManager. In both cases, the structure of the geometry is quite simple. Apart from the standard analytic surfaces (plane, cylinder, cone, sphere, and torus), numerical surfaces are represented as NURBS or B-spline surfaces. The additional geometric entities in the ShapeManager modelling kernel are the elliptical cone, elliptical cylinder surface, and the curve elliptical arc. On the other hand, the additional geometric entity in the CGM modelling kernel is the PCurve. A PCurves is used to define curves in the parameter space of a surface. For instance, a PLine is a curve where the mathematical representation in the space of the surface is a line. Hence, in the 3D space, a PCurve can be a line, circle, or more complex curve on a NURBS surface.

Table 6. Overview of geometry in CGM and ShapeManager

		CGM	ShapeManager
GEOMETRY	**VERTEX**	POINT	POINT
	CURVE	LINE CIRCLE ELLIPSE SPLINE CURVE NURBS CURVE PCURVE[1]	LINE CIRCLE ELLIPSE ELLIPTICAL ARC B-SPLINE CURVE
	SURFACE	PLANE CYLINDER CONE SPHERE TORUS NURBS	PLANE CYLINDER CONE ELLIPTICAL CONE ELLIPTICAL CYLINDER SPHERE TORUS B-SPLINE SURFACE

[1] PLINE, PCIRCLE, PELLIPSE, PSPLINE, PNURBS

The past CASD studies used as input native CAD models from CATIA V5 [25,70] and NX [66]. The native NX format and those from Solidworks, Solid Edge, etc. use the Parasolid geometric modelling kernel. Hence, the geometry found in such native files is discussed from the perspective of kernel formats in the next subsection.

2.4.2 Kernel formats

The two kernel formats are Parasolid and ACIS. The ACIS format was used directly as input in CASD [23,42], while the Parasolid format was indirectly used through the native

NX format [66]. Parasolid is a non-standard kernel file format that supports wireframe, surface, solid, and general non-manifold models [111]. The Parasolid geometric modelling kernel is an integral part of many CAD (NX, Solidworks, and Solid Edge), CAM (e.g., SolidCAM), and CAE systems (e.g., Abaqus and Ansys). Hence, the main advantage of The Parasolid is that it is supported by many CAx systems. The Parasolid format may be represented by two versions: textual or binary. The extension *.x_t refers to the textual version, while the extension *.x_b refers to the binary version. For the same 3D CAD model, both versions will contain the same data but in different forms. The Parasolid format can maintain the part-assembly hierarchy. The advantage is that the format enables interoperability between CAD systems running on the Parasolid geometric modelling kernel. However, since it is a non-standard file format, specific data may be inaccessible or incomplete after being imported into non-Parasolid CAD systems. A blend surface is exchangeable between CAD systems with a Parasolid geometric kernel but will be transformed into a spline surface when exchanged via other formats such as STEP. The second kernel file format is ACIS is employed in many CAD and CAE systems (e.g., SpaceClaim, BricsCAD, IronCAD KeyKreator). Like Parasolid, the ACIS file format may be represented by textual (*.sat) or binary (*.sab) version. A textual representation provides better readability to humans than a binary representation, but it is more sensitive to numerical round-off errors and has a larger storage size. The ACIS file format cannot maintain the part-assembly hierarchy and export mesh models.

The comparison of the available types of geometrical entities in Parasolid and ACIS is given in Table 7. Both provide the same type of analytic surfaces, while numeric surfaces are in ACIS represented by spline surface and in Parasolid with blend, bsurface (i.e., B-spline surface), surface of revolution, extruded surface, offset surface, trimmed and foreign surface. A foreign surface is a particular surface type for representing the user's data. Additionally, in the ACIS, there are the interpolated curve, degenerate curve, and undefined curve. An interpolated curve represents an intersection between two surfaces or the projection of a curve onto a surface. A Degenerate curve is used to build skin or loft surfaces that come to a point at either end. An undefined curve denotes a curve defined only on its endpoints, for which there are explicit positions, directions, and

curvatures. The additional curves in Parasolid are bcurve (B-spline), spcurve, intersection, constparam, and trimmed. An intersection curve is generated at the intersection of two surfaces. A trimmed curve is a basis curve of another bounded region curve. A spcurve is a 3D curve generated as a 2D curve in the parameter space of a surface. The prior CASD studies addressed analytic and spline surfaces using NX (Parasolid) and ACIS format as input [23,42,66]. The type of geometry in neutral formats is discussed in following subsection.

Table 7. Overview of geometry in Parasolid and ACIS

		Parasolid	ACIS
GEOMETRY	**VERTEX**	POINT	POINT
	CURVE	LINE CIRCLE ELLIPSE PARABOLAS[1] HYPERBOLAS[1] BCURVE[2] SPCURVE INTERSECTION CONSTPARAM TRIMMED	STRAIGHT LINE ELLIPSE[3] INTERPOLATED CURVE[4] DEGENERATE CURVE UNDEFINED CURVE
	SURFACE	PLANE CYLINDER CONE SPHERE TORUS BLEND BSURFACE[2] SURFACE OF REVOLUTION EXTRUDED SURFACE OFFSET SURFACE FOREIGN SURFACE	PLANE CONE[5] SPHERE TORUS SPLINE SURFACE

[1] Applies only to Siemens NX
[2] B-spline curve/surface
[3] Includes the circle as a special type of ellipse.
[4] Includes B-splines.
[5] Includes as well cylindrical surfaces.

2.4.3 Neutral formats

The neutral formats applicable in the context of CASD are STEP, IGES and JT. Until now, only the STEP format was exploited as CASD input [16,63]. Most other neutral CAD exchange formats (e.g., JF, QIF, eDrawings, 3D PDF, and STL) are unsuitable for CASD, mainly because they do not rely on the B-rep technique. STEP is the most frequently used neutral exchange 3D CAD file format [121,122]. Most commercial CAx systems support the import and export of STEP file formats, enabling interoperability

between different CAD systems. The Open CASCADE CAD system with the Open CASCADE Technology modelling kernel relies on the STEP file data structure [123]. Usually, the STEP format is represented as a text file using the EXPRESS language [124]. It can contain various B-Rep data such as part-assembly hierarchy, solid, sheet, wireframe bodies, and topological information. The major STEP file format standards related to mechanical engineering are defined by several Application Protocols (AP), including STEP AP203, STEP AP214, and STEP AP242. STEP AP203 defines geometry, topology, and configuration management data for solid models (mechanical parts and assemblies). The STEP AP214, along with the features from STEP AP203, additionally includes colours, layers, and geometric dimensioning and tolerancing (GD&T). STEP AP242 combines both STEP 203 and STEP 214 to support MBD. Exporting a STEP file may be time-consuming for CAD models with complex geometry.

IGES was the first neutral exchange CAD file format introduced [125]. It can be used to represent both B-Rep and CSG geometries. IGES can contain different types of information, such as surface, solid, and circuit diagrams. Many CAD systems support this format as it is one of the oldest CAD formats. However, the disadvantage of this file format is its sensitivity to geometric errors (e.g., gaps between surfaces, missing faces, and wrong surface orientation), which can require intensive repair in extreme cases. IGES describes a body primarily via disconnected surfaces instead of topological graphs. Hence, IGES is more suitable for surface geometry and less suitable for solid models.

The JT could also be applicable in the context of CASD because it contains the B-rep information. JT is a standardised file format in the form of a binary representation, which enables defining different levels of detail of the 3D CAD model and specifying the degree of precision (less precision indicates lower data storage size, higher precision means an increased data storage size) [126]. The JT format supports part-assembly hierarchy, mesh, PMI, and visual attributes (colouring and textures). Compared to the STEP file, the JT file enables the exporting of 3D models with less data storage size of the same geometry. Although the JT file format is used in the automotive and aerospace industry [128], it is limited in practical application because it is supported only by several CAD systems (e.g., NX, PTC Creo, and Autodesk Inventor).

An overview of the type of geometric entities in STEP and IGES is given in Table 8. Analytic surfaces with defined parametric representation are referred to as elementary surfaces in STEP. Analytic surfaces are also supported in IGES; only the circular and conical surfaces are denoted as right circular cylindrical and conical surfaces. The numeric surfaces in STEP are represented in the swept, bounded, and offset surface groups. The swept and offset surfaces were already covered in subsection 2.3.1. Bounded surfaces are surrounded by identifiable boundaries and have a finite area. Those surfaces are the B-spline, rectangular trimmed, curve bounded, rectangular composite, locally refined spline, and Bezier surface (only in Open CASCADE). The B-spline surface and its subtypes were discussed in subsection 2.3.1. A rectangular trimmed surface is defined by a basis surface and bounded by constant parametric lines. A curve bounded surface is a parametric surface with curved boundaries defined by one or more pcurves or boundary curves. A rectangular composite surface is formed by a rectangular array of segments or patches. A locally refined spline surface is a piecewise parametric polynomial or rational surface described with control points and local B-spline functions. Apart from the numeric surfaces already mentioned in native and kernel formats, the additional surfaces in IGES are the ruled and parametric spline surface. A ruled surface is created when sweeping over an area between defined curves, while a parametric spline surface is described by a series of parametric surfaces split into a grid. The curves in STEP are divided into four groups: conics, bounded curves, curves on surface, and offset curves. In IGES, additionally to the mentioned STEP curves, there is the parametric spline curve, composed of a series of parametric polynomials.

Table 8. Overview of geometry in STEP and IGES

		STEP		IGES
GEOMETRY	**VERTEX**	CARTESIAN POINT POINT ON CURVE[1] POINT ON SURF.[1]		POINT
	CURVE	LINE		LINE CIRCULAR ARC CONIC ARC PARAMETRIC SPLINE CURVE DIRECTION RATIONAL B-SPLINE CURVE OFFSET CURVE CURVE ON A PARAMETRIC SURF. COMPOSITE CURVE
		CONICS	CIRCLE ELLIPSE PARABOLA HYPERBOLA	
		BOUNDED CURVES	POLY LINE B-SPLINE CURVE TRIMMED CURVE COMPOSITE CURVE	
		CURVES ON SURFACE	PCURVE SURFACE CURVE INTERSECTION CURVE COMPOSITE CURVE ON SURF. BEZIER CURVE[2]	
		OFFSET CURVES	2D OFFSET CURVE 3D OFFSET CURVE	
	SURFACE	ELEMENTARY SURFACES	PLANE CYLINDRICAL SURFACE CONICAL SURFACE SPHERICAL SURFACE TOROIDAL SURFACE	PLANE SURFACE SPHERICAL SURFACE TOROIDAL SURFACE BOUNDED SURFACE TRIMMED SURFACE OFFSET SURFACE PARAMETRIC SPLINE SURF. RULED SURFACE SURFACE OF REVOLUTION RATIONAL B-SPLINE SURFACE RIGHT CIRCULAR CYLIND. SURF. RIGHT CIRCULAR CONICAL SURF.
		SWEPT SURFACES	SURF. OF LINEAR EXTRUSION SURF. OF REVOLUTION SURF. CURVE SWEPT SURF.[1] FIXED REFERENCE SWEPT SURF.[1]	
		BOUNDED SURFACES	B-SPLINE SURFACE RECTANGULAR TRIMMED SURF. CURVE BOUNDED SURFACE RECTANGULAR COMPOSITE SURF. LOCALLY REFINED SPLINE SURF. BEZIER SURFACE[2]	
		OFFSET SURF.		
	AXIS PLACE-MENT	AXIS PLACEMENT 1 AXIS PLACEMENT 2		
	VECTOR	DIRETCTION VECTOR WITH MAGNITITUDE		

[1] Applies only to STEP
[2] Applies only to Open CASCADE

The types of geometrical entities, i.e., surfaces and curves, used across different 3D CAD model formats are similar. All formats support analytic surfaces (plane, cylindrical, conical, spherical, and toroidal surfaces) and curves (line, circle, and ellipse, while additionally parabola and hyperbola in some formats there are the). The numeric surfaces and edges are also quite similar. For instance, a B-spline surface or curve may have a representation in the form of Bezier, NURBS, parametric spline surface or curve, etc. Other numeric surfaces include the surface of revolution, extruded surface (or surface of

linear extrusion), offset surface, swept surface, etc. The typical numeric curves are trimmed curve, pcurve, intersection curve, offset curve, etc. During the exchange of CAD models, if the face's underlying surface type (e.g., surface of revolution, extruded surface, blend, etc.) is not supported by the CAD format, it needs to be represented by some other numeric surface (usually B-spline) and with certain approximation. The same principle applies to the exchange of edges with different underlying curve types. Classifying topological entities based on the associated geometric entity type represents a common approach in most CASD studies that used the B-rep CAD model as input. Classification of topological entities is also exploited in this research. Based on the theoretical foundations highlighted, the following chapter presents a CASD method for detecting exact global and partial reflectional and axisymmetry in B-rep CAD models.

3 A METHOD FOR COMPUTER-AIDED SYMMETRY DETECTION

This chapter synthesises the relevant insights from the theoretical background into a method for computer-aided symmetry detection in 3D CAD models with B-rep. The introduced method consists of six steps: (1) 3D CAD model interpretation, (2) B-rep analysis, (3) generation, (4) trimming, and (5) evaluation of the POSCs and AOSCs, (6) and visualisation of detected APOS(s) or AAOS. The sections of the chapter are organised in such a way that each represents one step of the proposed symmetry detection method. Finally, a data model for the proposed CASD method is given.

Based on the presented research background, in this chapter, the main research objective of the doctoral thesis is addressed by proposing a CASD method for CAD models. The input for the CASD method is the B-rep of a 3D CAD model, while the possible output of the symmetry detection is exact global or partial symmetry and reflectional symmetry or axisymmetry. By using B-rep as input, the proposed method offers a general approach that can be applied and extended to various CAD systems and formats. The method comprises six steps (the flowchart is shown in Figure 12): (1) interpretation of the 3D CAD model, (2) analysis of B-rep, (3) generation, (4) trimming, and (5) evaluation of POSCs and AOSCs, and (6) visualisation of detected APOS(s) or AAOS.

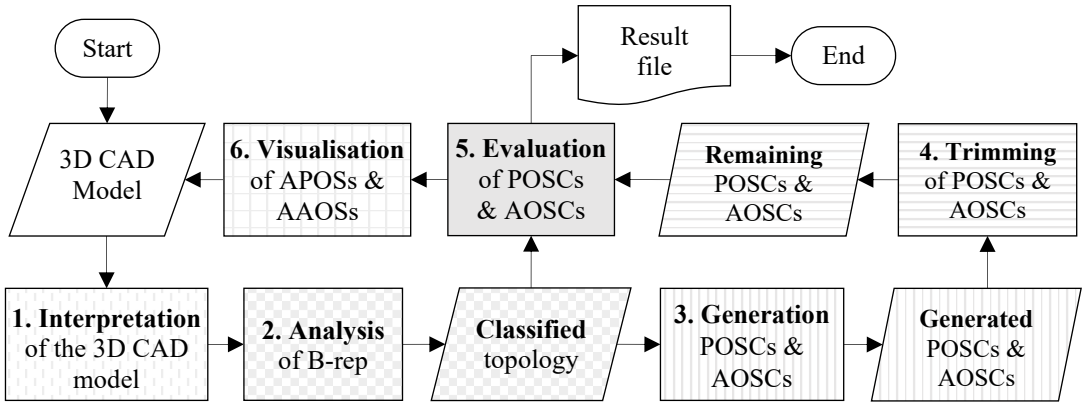


Figure 12. Basic flowchart of the proposed symmetry detection method

First, the 3D CAD model must be interpreted using a suitable CAD system. After that, the 3D CAD model's B-rep is subjected to analysis, which includes classifying

topological entities (faces and edges) according to their underlying geometric entities (types of surfaces and edges). The foundation of the proposed symmetry detection method is faces, which are not limited only to analytic surfaces, but the method can also process numeric surface types. The B-rep analysis step consists of classifying and calculating specific face and edge properties. The proposed CASD method uses a geometry-based approach, where each face is represented by a unique point (centroid or its projection onto the face) and the respective unit normal vector or unit axis vector at this point. Further, the CASD method relies on the implicit symmetry detection approach, meaning that the POSCs and AOSCs are generated from principal axes of inertia (PAOI), pairs of similar faces, and single faces. Then, the generated candidates are subjected to trimming to remove duplicate and unsuitable candidates. Each remaining POSC and AOSC undergoes an evaluation process with respect to the faces and edges using vector calculus and specific face and edge properties obtained from the B-rep analysis. If the evaluation shows that the corresponding POSC or AOSC also represents the APOS or AAOS, it is then visualised within the 3D CAD model.

3.1 Interpretation of the 3D CAD model

The symmetry detection method starts with interpreting the 3D CAD model input. This input can come in various file formats, such as native, kernel, or neutral. Previous studies on CASD have used different file formats, including neutral STEP format [63], kernel formats Parasolid [14,66] and ACIS [23,42]. In addition, some studies have utilised a native file format (CATIA V5) to detect symmetry based on design features [25,70]. However, within this study, to maintain the generality of the proposed CASD method, symmetry detection is performed from the aspect of the B-rep. That makes the method independent and applicable to various file formats and CAD systems. The B-rep technique has been successfully utilised in previous CASD studies [16,14,23,42,63,66] due to its ability to represent continuous data defined as an infinite point set, making it ideal for detecting exact symmetry. The advantage of the B-rep over other input 3D digital objects (e.g., point clouds and mesh models) is that it represents continuous data defined as an infinite point set, which makes it suitable for detecting exact symmetry. As shown in Table 5, many CAD file formats rely on the B-rep technique. For instance, the neutral file formats STEP and IGES or the kernel file formats Parasolid and ACIS can be used

for that purpose. Depending on the CAD system used to interpret the CAD model, a native file format may also be appropriate as input.

The interpretation of the input 3D CAD model is conducted by utilising a CAD system (commercial or open-source). As a result of the interpretation process, the topological and geometrical information of the B-rep becomes available. For instance, in previous CASD studies, a STEP file was used as input for symmetry detection, and two CAD systems (CATIA V5 and Open CASCADE) were used for interpretation [16,63]. To avoid the exchanges between two CAD systems, NX can be used in CASD for interpreting input models in Parasolid format [14,66]. Another solution for interpreting the CAD model is directly at the ACIS geometric modelling kernel level [23,42], providing complete access to the B-rep's topology and geometry.

An important factor to consider when interpreting the 3D CAD model in the context of CASD is accuracy, which is needed to achieve exact symmetry detection, especially when exchanging CAD file formats between CAD or other CAx systems. For instance, the accuracy of geometry data exchange between different CAD systems using STEP AP242 is for numeric geometry below 10^{-6} m, while even higher accuracy can be obtained for analytic geometry [127]. An additional requirement for symmetry detection is that the interpreted 3D CAD model must be free of errors such as invalid topology and its associated geometry. For instance, the Euler equation (15) can used to verify if the topology of the B-rep CAD model is valid. Moreover, the CAD model can be checked for errors automatically by most CAD systems when interpreting the CAD model. The interpreted CAD model needs to be a solid body. Otherwise, the 3D CAD model's mass properties, which are relevant to the proposed CASD method (see Section 3.3), cannot be computed.

Finally, it can be concluded that interpreting the input 3D CAD model by employing a suitable CAD system is necessary to gather the B-rep model's topological and geometrical definition for the following steps. To maintain the generality of the proposed CASD method, it is not yet limited to an input 3D CAD file format or a CAD system (until the proposed CASD method is implemented in Chapter 4). Generally, the only requirement for the input 3D CAD model is that it supports the B-rep technique.

3.2 Analysis of the B-rep

After interpreting the CAD model, the next step of the CASD method is to analyse the B-rep, which is conducted generally through a CAD system. First, the CAD model must be error-free (as highlighted in the previous section). In addition, it needs to be checked whether the input CAD model is a single part with one body (according to the research limitations specified in Section 1.2). The analysis of B-rep consists of three sub-steps (as shown in Figure 13): (1) merging and (2) classification of topology, and (3) calculation of face and edge properties.

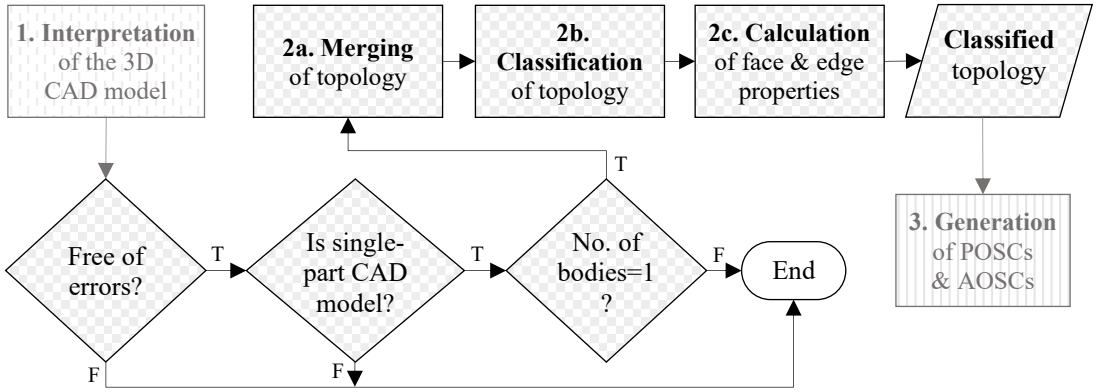


Figure 13. Flowchart of the B-rep analysis step

The default CAD system functionalities generally do not provide access to the B-rep data structure. However, the B-rep data is often accessible through the CAD system's API. In some studies [16,63], two separate CAD systems (CATIA V5 and Open CASCADE) were required to access the B-rep data structure. This is necessary when the first CAD system enables access only to geometrical properties, while the topological information needs to be retrieved through a secondary CAD system. Therefore, a more enhanced approach is to use a CAD system that provides complete access to the CAD model's B-rep data structure, such as NX [14,66]. This enables the development of a custom application directly built on the ACIS geometric modelling kernel [23,42]. In the context of the CASD method within this study, the analysis of the 3D CAD model's B-rep includes the classification of topology and the computation of specific properties from the associated geometry. The previous CASD studies relied on loops [23,42] and faces [16,14,63,66]. A negative aspect of using loops can be the computation accuracy of their properties, which may be imprecise and not supported by the CAD system's functionalities. This approach depends on how the geometric modelling kernel manages

loops, and there may be variations in how loops are handled in other CAD systems [23,42]. Instead of loops, the use of faces represents an inherent approach as they are the basic building unit of the B-rep with demonstrated success in symmetry detection [16,14,66]. Thus, in this thesis, the foundation of the CASD method is also the face. Additionally, edges are exploited for secondary calculations of method-specific face properties such as Cosine similarity (Subsection 3.3.2) and evaluation of POSC and AOSC (Section 3.5). Although the proposed CASD method is geometry-based, it exploits the benefits of the B-rep technique as it takes advantage of the topological information.

The topology of the CAD model needs to be pre-processed before its classification. Theoretically, the input CAD model's topology can be a compound of numerous combinations of faces and edges without changing its shape [129]. Therefore, in the first sub-step of the B-rep analysis, the input CAD model's topology and associated geometry must be checked if merging operations are required. A hypergraph data structure representing adjacency relationships between faces, edges, and vertices in the CAD model can be exploited to identify topological entities suitable for merging [16,63]. In manifold CAD models, to which this research is limited, the primary consideration in terms of topology merging is partitioned periodical faces (e.g., cylindrical, conical, toroidal, spherical surfaces, etc.), which are usually a result of the modelling process in the CAD system. For instance, specific CAD systems (such as CATIA V5) partition periodical faces by default during the modelling process, while other CAD systems (such as Solidworks) do not partition periodical faces at all [27] (Figure 14).

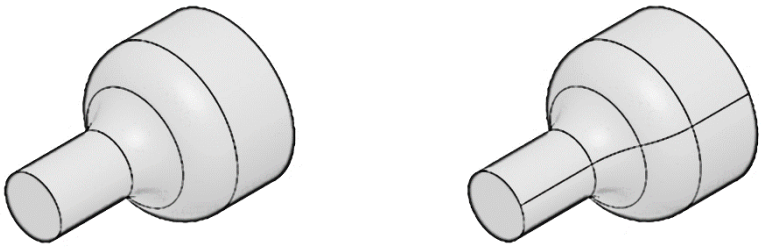


Figure 14. Example of unpartitioned (left) and partitioned periodical faces (right)

Sometimes, the partitioning of periodical faces can result from exchanging CAD models through different file formats, even though the CAD system does not partition faces by default during the modelling process [130]. In the context of CASD, the identification of partitioned faces and edges should be considered by merging identical faces adjacent to each other that share the same surface type and parameters [16,63,66]. The merging of topology and associated geometry can be conducted employing topology merging

operators [131] or even controlled during the import of CAD models into the CAD system [132]. To conclude, the proposed CASD method considers the merging of topological entities, but existing approaches can be used to pre-process the topology of the CAD model. In implementing the CASD method, the merging of periodical topological entities is controlled during the import of CAD models into the CAD system [131], which is further discussed in Chapter 4.

In the next sub-step, classification of topology, the topological entities (faces and edges) in the CAD model are looped and grouped into different classes based on their underlying surface and curve type (Figure 15 and Figure 16), e.g., plane, cylindrical, conical, spherical, toroidal, surface of revolution, spline surface, line, circle, spline curve, etc. In the context of past CASD studies, faces and loops were also classified based on their underlying surface type [14,23,42,66]. Classifying faces is necessary because only faces from the same class require pairwise comparison while evaluating reflectional symmetry. Generally, the types of surfaces available depend on the CAD system used to interpret the input 3D CAD model. Certain surface types are so common that most CAD systems have an equivalent representation (e.g., plane, cylindrical surface, etc.). If the CAD system does not support a specific surface type, it may be replaced by another suitable surface type during the interpretation process. For instance, a blend surface can be exchanged between Parasolid CAD systems. However, specific CAD systems (e.g., CATIA V5 and Autodesk Inventor) or exchange CAD file formats (e.g., STEP and IGES) do not support blend surfaces and may interpret them as spline surfaces.

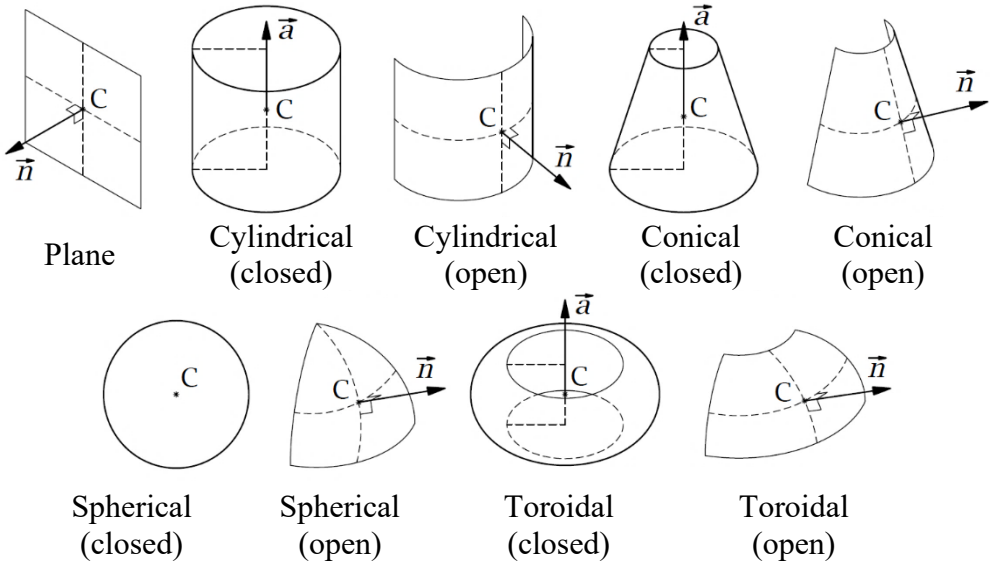
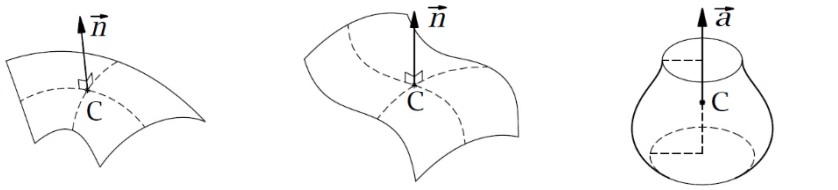


Figure 15. Examples of different types of analytical surfaces



Blend surface spline surface Surface of revolution

Figure 16. Examples of different types of numeric surfaces

An additional concern during classification is the labelling of topological entities within the CAD model to perform different operations on them later, such as tracking, selecting, referring, etc. A CAD system may support the labelling of topological entities as unique IDs at the API level. The CAD system can have default IDs, or they can be assigned by the user, which must ensure that each is unique. The user-defined labelling can be represented by the type of surface or edge and a digit counting to the total number of the corresponding surface or edge type in the CAD model. The labelling of topological entities is further discussed during the method's implementation (see Chapter 4).

Another significant step in prior CASD studies is the retrieval of characteristic parameters of geometric entities [16,23,42,63,66] and calculating characteristic geometric properties of topologic entities such as the loop area and loop centroid [23,42]. Hence, after the classification, the next step of the proposed CAS method is the computation of specific geometric properties of faces and edges, which are used in the subsequent steps to generate, trim, and evaluate the POSCs and AOSCs. For each face, the following geometrical properties are computed: periodicity angle γ, face area, face centroid or its projection onto the face, and face unit normal or axis vector (Table 9). The geometrical properties of a face depend on the type of associated surface and whether the face is open or closed, which is queried with the periodicity angle γ.

Table 9. Characteristic face properties for various surface types

Surface types			Vector	Unique point
Analytic	Plane	–	Normal vector	Face centroid (FC)
	Cylindrical surface	Open	Normal vector	FC projected onto the face
		Closed	Axis vector	Face centroid (FC)
	Conical surface	Open	Normal vector	FC projected onto the face
		Closed	Axis vector	Face centroid (FC)
	Spherical surface	Open	Normal vector	FC projected onto the face
		Closed	Axis vectors	Face centroid (FC)
	Toroidal surface	Open	Normal vector	FC projected onto the face
		Closed	Axis vector	Face centroid (FC)
Numeric	Surface of revolution	Open	Normal vector	FC projected onto the face
		Closed	Axis vector	Face centroid (FC)
	Spline surface	–	Normal vector	FC projected onto the face
	Blend surface	Open	Axis vector	Face centroid (FC)
		Closed	Normal vector	FC projected onto the face
	Extruded surface	–	Normal vector	FC projected onto the face
	Offset surface	–	Normal vector	FC projected onto the face

The periodicity angle γ_u is only computed for periodical surfaces (cylindrical, conical, toroidal, spherical, surface of revolution, and blend) to evaluate if a face is open or closed (Figure 15 and Table 9). The angle γ_u can be computed from the parametrisation range of the face. For surfaces periodical in one direction, u is usually the periodical direction [133]. A spherical surface is periodical in two directions (u and v), so a second periodicity angle γ_v needs to be computed. If $\gamma_u < 2\pi$ (360º) or $\gamma_v < 2\pi$, the face is open, while $\gamma_u = 2\pi$ or $\gamma_v = 2\pi$ indicates that the face is closed. The next characteristic property is either the unit normal vector or the unit axis vector of the face. Which of them is computed depends on the underlying surface type and whether the face is closed or open (Table 9). Closed periodical faces with the underlying surface types cylindrical, conical, toroidal, and surface of revolution are defined by their unit axis vector, which can be retrieved from the parameters that define the corresponding surfaces in the CAD system (Table 10). The functionalities for retrieval of face properties are most often provided by the CAD systems at the API level. However, the axis vector for a closed face of spherical surface type can be computed from its origin and centroid. A sphere or a "ball" (a fully closed face of the spherical surface type with $\gamma_u = 2\pi$ and $\gamma_v = 2\pi$) is not likely to occur in single-body part CAD models unless the part is a bearing ball exhibiting spherical symmetry with an infinite number of APOS and AAOS. Therefore, such faces can be excluded from the

CASD method without negative influence on detecting reflectional symmetry and axisymmetry.

Table 10. Characteristic surface parameters retrievable from a CAD system

Surface types		Point	Vector	Parameter 1	Parameter 2
Analytic	Plane	Root point	Normal vector*	–	–
	Cylindrical surface	Origin	Axis vector*	Radius	–
	Conical surface	Origin	Axis vector*	Radius	Half angle
	Spherical surface	Origin	–	Radius	–
	Toroidal surface	Origin	Axis vector*	Major radius	Minor radius
Numeric	Surface of revolution[1]	Point on axis	Axis vector*	–	–
	Spline surface	–	–	Parameterisation data[3]	–
	Blend surface	–	–	Radius	–
	Extrude surface[1]	–	Direction vector	–	–
	Offset surface[2]	–	–	Offset distance	–

1 – defined by a profile curve
2 – defined by a base surface
3 – knot vectors in u & v direction, control points, order of the surface in u & v direction, etc.
* – used by the proposed CASD method in this research.

Analogical to faces, for each edge, a set of topological and geometrical properties is retrieved from the CAD system or computed, such as the edge length, midpoint, and whether the edge belongs to an outer or inner loop (Table 11). The edge properties are used in the next step to calculate similar face pairs and their midpoints (Subsection 3.3.2).

Table 11. Characteristic edge properties for various curve types

Curve types			Loop type	Unique point
Analytic	Line	–	outer or inner	Midpoint
	Circle	closed	outer or inner	Centre
		open	outer or inner	Midpoint
	Ellipse	Closed	outer or inner	Centre
		open	outer or inner	Midpoint
Numeric	Spline curve	–	outer or inner	Midpoint
	P curve	–	outer or inner	Midpoint
	Intersection curve	–	outer or inner	Midpoint
	Trimmed curve	–	outer or inner	Midpoint
	Offset curve	–	outer or inner	Midpoint

The unit normal vector $\mathbf{n}$ at any point on the surface can be computed from the cross product of the tangent vectors $\mathbf{R}_u$ and $\mathbf{R}_v$:

$$\mathbf{n} = \frac{\mathbf{N}}{\|\mathbf{N}\|} = \frac{\mathbf{R}_u \times \mathbf{R}_v}{\|\mathbf{R}_u \times \mathbf{R}_v\|}, \tag{52}$$

where $\mathbf{R}_u = \frac{\partial \boldsymbol{\sigma}}{\partial u}$ and $\mathbf{R}_v = \frac{\partial \boldsymbol{\sigma}}{\partial v}$. The face's normal orientation points always away from the solid CAD model. The next property, face area, can be computed by integrating the length of the normal vector to the surface over the appropriate domain *D* in the parametric *uv* plane:

$$A = \iint_D \left\| \mathbf{R}_u \times \mathbf{R}_v \right\| \, du dv = \iint_D \left\| \mathbf{N} \right\| \, du dv \,. \tag{53}$$

As already mentioned, the proposed method relies on representing each face by a unique point. The face centroid C can be calculated as follows:

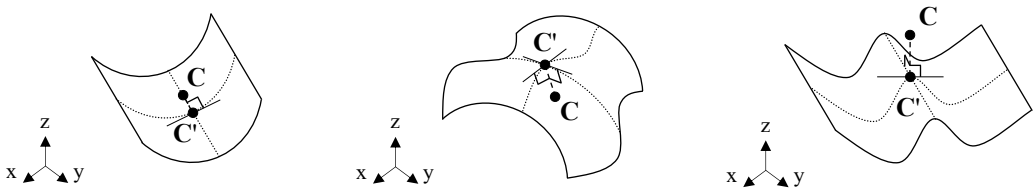
$$\mathrm{C}(x_\mathrm{C}, y_\mathrm{C}, z_\mathrm{C}) = \left(\frac{A_x}{A}, \frac{A_y}{A}, \frac{A_z}{A} \right) = \left(\frac{\iint_D x dS}{A}, \frac{\iint_D y dS}{A}, \frac{\iint_D z dS}{A} \right). \tag{54}$$

The centroid lies either on the face or outside of it. For instance, the face's centroid of a plane surface type lies typically on the face, while those of other surface types are located outside the face. Since the unit normal vector **n** can only be computed for points lying on the face, the centroid is not suitable for open faces (except for planes). Hence, another unique point for open faces needs to be acquired. Theoretically, the unique point of an open face could be obtained from the *uv* midpoint in the parameter domain $\boldsymbol{\sigma}(u_\mathrm{mid}, v_\mathrm{mid})$ to compute the corresponding centre point on the surface domain[10]. However, the parametric representation of a surface is not unique [31], and it was demonstrated that the uniform sampling in the parameter domain may lead to a non-uniform sampling of the surface domain [134]. For those reasons, the unique point representation of faces using the *uv* midpoint and centre point may not assure sufficient repeatability. Instead of the face centre, the CASD method uses the orthogonal projection of the face centroid onto the face, i.e., the closest projection point onto the face. Mathematically, the orthogonal projection of a point implies finding a projected point on a face so that the vector connecting the point in space and the point projected on the face becomes perpendicular to the face. The set of orthogonal projections of a point **C** onto a parametric surface is equal to [135] (Figure 17):

$$\Gamma_{\mathrm{orth}} = \left\{ \mathbf{C}' \middle| \mathbf{C} - \mathbf{C}' \middle| / \nabla f\left(\mathbf{C}'\right), \mathbf{C} \in \mathbf{R}^n \right\}, \tag{55}$$

[10] The *uv* midpoint is computed from the face's *uv* bounds: $u_\mathrm{mid} = (u_\mathrm{min} + u_\mathrm{max})/2$ and $v_\mathrm{mid} = (v_\mathrm{min} + v_\mathrm{max})/2$

where **C** is the centroid point, **C'** is the projected point. Orthogonal projection requires the existence of a tangent plane at the projected point on the surface.



Cylindrical surface Surface of revolution Extruded surface

Figure 17. Examples of orthogonal projections **C'** of the face centroids **C** onto faces

This research does not intend to develop a novel approach for computing the closest projection point on the surface. Existing approaches can be exploited for that purpose (a review is given in the study [135]). Moreover, many CAD systems often provide their own methods and functionalities related to the computation of a projection point onto the face [136]. To conclude, depending on the type (closed or open), a face is represented by two possible types of points: the face centroid or the orthogonal projection of the face centroid onto the face. Besides the face properties, the CASD method exploits specific edge properties (edge length and midpoint) for partial symmetry detection. The edge length can be computed using the equation:

$$L = \int_a^b \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}\, du. \tag{56}$$

The computation of the curve midpoint is accomplished by an iterative process of finding b_0 such that the lengths of the curve on the left and right sides of the midpoint are equal:

$$\int_a^{b_0} \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}\, du = \int_{b_0}^b \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2 + \left(\frac{dz}{du}\right)^2}\, du. \tag{57}$$

Finally, the classified faces and their computed properties are used in the next step of the CASD method to generate the POSCs and AOSCs.

3.3 Generation of planes and axes of symmetry candidates

As already stated, the proposed CASD method relies on an implicit (indirect) symmetry detection approach, meaning a set of POSCs and AOSCs is generated. The classified faces and their specific properties from the previous B-rep analysis step are utilised for that purpose. The POSCs and AOSCs are generated in three sub-steps:

1) the principal axes of inertia,

2) pairs of similar faces and
3) single faces.

Three POSCs and three AOSCs are always generated from the 3D CAD model's PAOI passing through the COG to cover the possible existence of exact global reflectional and axisymmetry. Further, to cover the detection of possible exact global symmetries that are misaligned with the PAOI or partial symmetries within the 3D CAD model, additional POSCs are generated from pairs of similar faces of either the plane, cylindrical or spline surface type. Finally, single faces of the cylindrical, conical, toroidal, and surface of revolution type are employed to generate the AOSCs for detecting partial axisymmetry within the 3D CAD model. The flowchart of the candidate generation step is given in Figure 18 and further discussed in the following subsections.

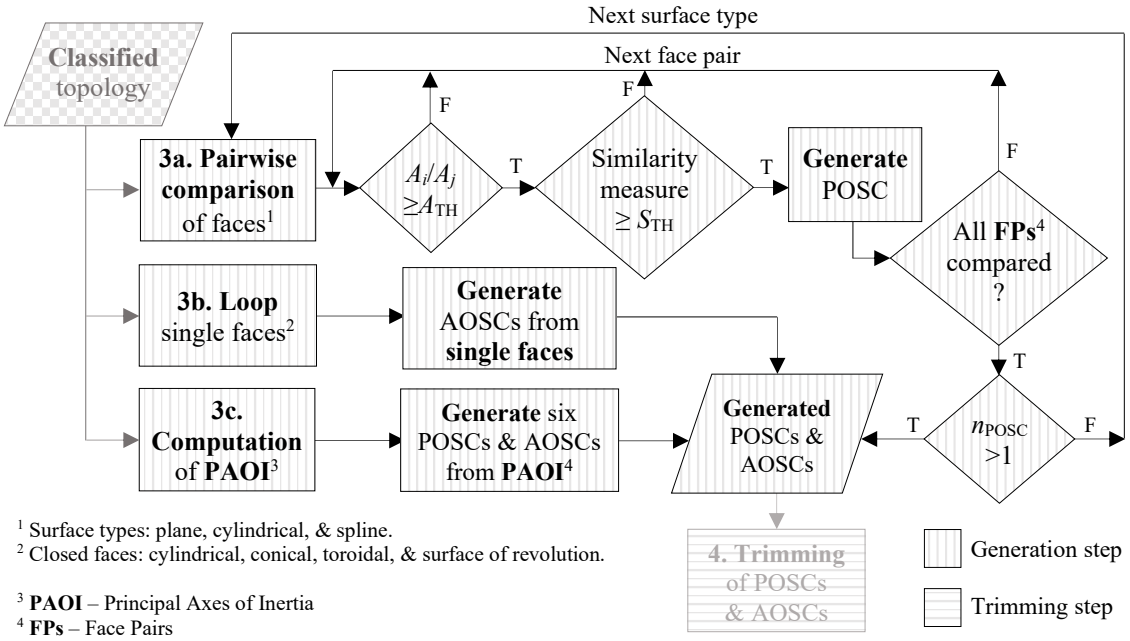


Figure 18. Flowchart of the candidate generation step

3.3.1 Principal axes of inertia

First, three POSCs and three AOSCs are always generated from the 3D model's principal axes of inertia to cover the possible existence of exact global symmetry within the 3D CAD model. The justification for that is reflected in the following facts [137]:

1) If an object, i.e., a 3D CAD model, exhibits exact reflectional or axisymmetry, then corresponding APOS(s) or AAOS must pass through its centre of gravity (COG).

2) If an object, i.e., a 3D CAD model, exhibits exact reflectional symmetry, then the direction normal to the APOS is a principal axis.
3) If an object, i.e., a 3D CAD model, exhibits exact axisymmetry, then the AAOS is a principal axis.

So far, PCA has been used in existing CASD studies [29,30,57,60] for point clouds and mesh models to identify the POSCs and AOSCs. It was also used for detecting approximate symmetries (reflectional and rotational) in 3D CAD models represented by octrees [137]. However, until now, PCA has not been used in the context of B-rep CAD models. Hence, the proposed CASD method, exploits the PAOI to identify three POSCs and three AOSCs. The first step is to calculate the COG. For uniform density throughout the object, the centre of mass and centre of gravity correspond to the volume centroid. For a volume of arbitrary shape, the coordinates of the centroid (x_c, y_c, z_c) are defined by the following equations:

$$x_C = \frac{\int_V x \cdot dV}{\int_V dV}; y_C = \frac{\int_V y \cdot dV}{\int_V dV}; z_C = \frac{\int_V z \cdot dV}{\int_V dV} \tag{58}$$

where the x, y, and z terms inside the integrals denote the distances measured from the reference axes to the centroid of the differential volume. Next, the PAOI are computed. For that purpose, first, the moments of inertia and products of inertia are calculated using the following equations:

$$\begin{aligned} I_{xx} &= \int \left(y^2 + z^2\right) dm & I_{xy} &= \int (xy)\, dm \\ \mathrm{I}\, I_{yy} &= \int \left(z^2 + x^2\right) dm & I_{yz} &= \int (yz)\, dm \\ I_{zz} &= \int \left(x^2 + y^2\right) dm & I_{zx} &= \int (zx)\, dm \end{aligned} \tag{59}$$

Then, the inertia tensor is defined as:

$$\mathbf{I}_P = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix}. \tag{60}$$

The angular momentum vector $\mathbf{L}$ is proportional to the inertia tensor and angular velocity vector $\boldsymbol{\omega}$:

$$\mathbf{L} = \mathbf{I}_P \cdot \boldsymbol{\omega}. \tag{61}$$

To find the axis of rotation where **L** and **ω** are parallel, the equation above can further be written:

$$\mathbf{I}_{\mathrm{P}} \cdot \boldsymbol{\omega} = \lambda \cdot \boldsymbol{\omega}. \tag{62}$$

Finally, the task of finding the principal axis of inertia becomes an eigenvalue problem:

$$\left|\mathbf{I}_{\mathrm{P}} - \lambda \mathbf{I}\right| = 0. \tag{63}$$

From the equation above, the three eigenvalues are the principal moments of inertia, and the three eigenvectors are the PAOI. Most often, the generated POSCs will be parallel with the *XY*, *YZ*, and *ZX* planes, while the AAOSs will be parallel with the coordinate system axes. This is because CAD systems provide default built-in planes (e.g., front aligned with *XY* plane, right aligned with *YZ* plane, and top aligned *ZX* plane) exploited for modelling purposes, and most 3D CAD models have a regular shape. However, the 3D model may also be rotated with respect to the coordinate axes or have an irregular shape, resulting in misalignment of the PAOI with respect to the coordinate system. Finally, a POSC is defined by the plane equation:

$$a\left(x - x_{\mathrm{C}}\right) + b\left(y - y_{\mathrm{C}}\right) + c\left(z - z_{\mathrm{C}}\right) = 0\ , \tag{64}$$

while the line equation defines an AOSC:

$$\frac{\left(x - x_{\mathrm{C}}\right)}{a} = \frac{\left(y - y_{\mathrm{C}}\right)}{b} = \frac{\left(z - z_{\mathrm{C}}\right)}{c}\ , \tag{65}$$

where the variables a, b, and c represent the components of the PAOI, while x_{C}, y_{C}, and z_{C} are the coordinates of the COG. The candidates can be stored in a 2D array of the size 6×6, where the first three rows represent the POSCs, which are stored in the form of (a, b, c, x_{C}, y_{C}, z_{C}), while the last three rows define the AOSCs and are of the following form (o, p, q, x_{C}, y_{C}, z_{C}). The functionalities of CAD systems usually provide mass properties such as the COG (or centre of mass as referred to in some CAD systems) and the PAOI. The six candidates generated from PAOI may not be sufficient to detect exact global reflectional symmetries that are misaligned with the axes or partial reflectional and axisymmetry within the 3D CAD model. Therefore, additional candidates need to be generated by pairing similar faces.

3.3.2 Pairs of similar faces

As already stated, the POSCs are generated from pairs of similar faces and exploited for detecting exact global and partial reflection symmetry in the 3D CAD model. Previous

studies generated the POSCs by pairing identical loops of the plane, cylindrical, conical, toroidal, and spherical surface type [23,42] or from the intersection edges of two adjacent faces (considering different combinations of 5 analytic surface types) [16,63]. In the context of the proposed CASD method, only faces from the same class are compared, and each face pair can generate only one POSC. Ideally, to cover all possible POSCs, the face pairs should be generated considering all face classes. However, this may result in an unnecessarily large number of POSCs with many duplicates. Hence, the strategy for the generation of POSCs is as follows:

1) First, pairs are generated from similar faces of the underlying plane surface type. If at least one POSC is generated, the further generation process stops at this point. This generation process is most often sufficient for a variety of CAD models.
2) If the attempt to generate any POSC failed because no pairs of similar faces are identified or no faces of the underlying plane surface type are present within the 3D CAD model, then the pairs are generated from identical faces of the cylindrical surface type. If at least one POSC is generated, the further generation process stops at this point.
3) Again, if the attempt to generate the POSCs failed for the previously mentioned reasons, then the pairs are generated from identical faces of the underlying spline surface type.

Past studies reported in [23,42] evaluated the similarity between pairs of loops by using their properties (loop type, loop area, and number of edges). However, this may not be an adequate measure since two non-identical loops can have the same properties. Hence, in the context of this study, two criteria are used to identify similar face pairs: the area ratio and a similarity measure. The reason for having two criteria is that the first is used as a pre-filter for the second, which is computationally demanding and needs to be avoided whenever possible. The area ratio (AR) of a face pair represents the surface area of the first face divided by the surface area of the second face:

$$\mathrm{AR} = \frac{A_1}{A_2} \geq \mathrm{AR}_{\mathrm{TH}} \tag{66}$$

where $\mathrm{AR}_{\mathrm{TH}}$ represents the threshold value of the area ratio. The AR is computed so that $A_1 \leq A_2$, so its score ranges between 0 and 1. The significance of AR is that it enables quick filtering of dissimilar face pairs. Only face pairs with $\mathrm{AR} \geq \mathrm{AR}_{\mathrm{TH}}$ are further processed to the second similarity criterion. The similarity between two faces is assessed via a similarity measure. For the selection of an appropriate similarity measure,

preliminary research was conducted. The similarity measures from Equations (9) to (13) were considered, i.e., the Jaccard index (JI), Cosine similarity (CS), Sørensen-Dice coefficient (SDC), Szymkiewicz-Simpson coefficient (SSC), and Braun-Blanquet coefficient (BBC). The preliminary research aimed to investigate the applicability of the mentioned similarity measures for pairing similar faces. Generally, similarity measures are used in statistics to compute the similarity between two finite datasets [75,82]. In the context of B-rep, the basic idea is to observe the faces within the 3D CAD model as finite sets of edges. For that purpose, faces are first decomposed into their edges and labelled using a string code (e.g., "oLI10"). The string code represents a unique label for each edge considering the following properties: loop type, curve type, and curve length. The first letter of the string code indicates to which loop type the edge belongs ("i" stands for an inner loop while "o" for an outer loop). The following two string code letters describe the type of the underlying curve. For instance, lines are lablled with "LI", circles with "CI", spline curves with "SC", and so on. The last part of the string code denotes the length of the curve in millimetres rounded off to two decimal places. Examples of string codes are given in Figure 19.

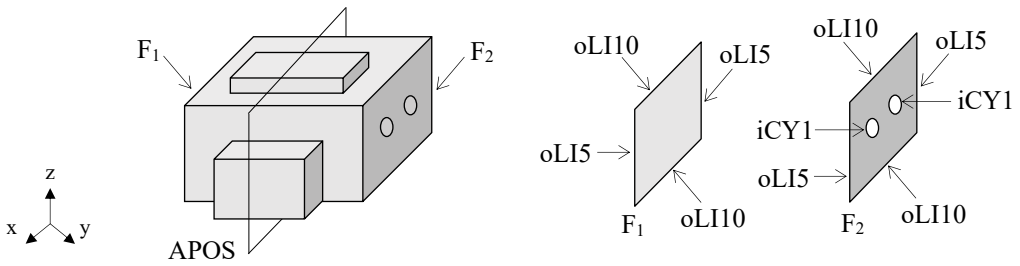
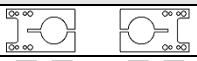
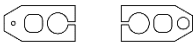
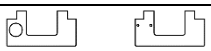
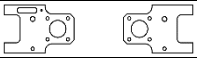
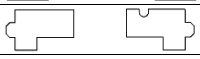


Figure 19. Example of the string code designation for a similar face pair.

The preliminary research selecting an appropriate similarity measure included ten randomly selected test cases of face pairs with varied similarities (Figure 12). The face pairs are classified as identical (Figure 12, 1), similar (Figure 12, 2 – 9), and non-similar at all (Figure 12, 10).

Table 12. Comparison of similarity measure scores for face pairs examples

No.	Face pairs	AR	CS	JI	SDC	SSC	BBC
1		1	1	1	1	1	1
2		0.96	0.80	0.67	0.80	0.80	0.80
3		0.97	0.95	0.90	0.95	0.95	0.95
4		0.96	0.92	0.84	0.91	0.94	0.89
5		0.98	0.91	0.83	0.91	1	0.83
6		0.99	0.77	0.61	0.76	0.92	0.65
7		0.98	0.69	0.52	0.67	0.80	0.60
8		1	0.75	0.60	0.75	0.75	0.75
9		0.94	0.76	0.61	0.76	0.81	0.71
10		1	0	0	0	0	0

For instance, some face pairs have the same outer shape and different inner shapes (Figure 12, 2 – 5). Other face pairs have slightly different outer shapes (Figure 12, 6 – 8) or different outer and inner shapes (Figure 12, 9). The computed scores of the similarity measures are given in Figure 12 and Figure 20. The results show that all similarity measures correctly recognise identical (Figure 12, 1) and non-similar face pairs (Figure 12, 10). The main difference between the computed scores is manifested for similar faces (Figure 12, 2 – 9).

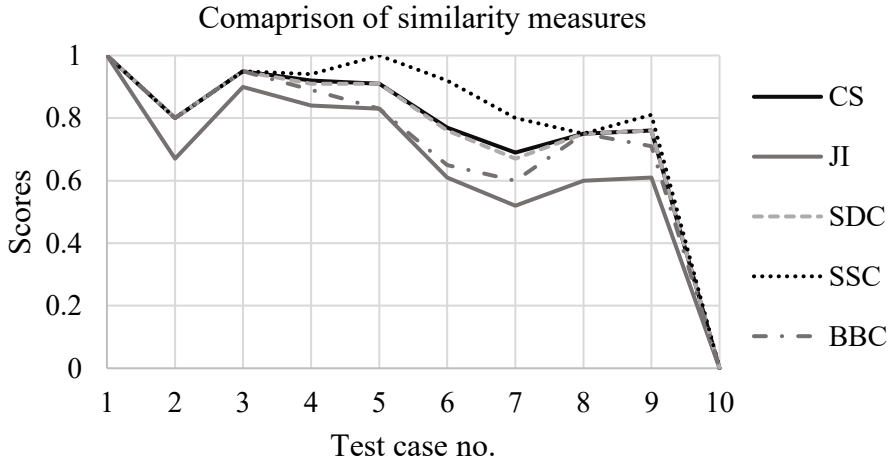


Figure 20. The plot of similarity measures scores for test cases from Figure 12

The basic definitions of similarity measures were investigated to understand their differences. All similarity measures from Equations (9) to (13) have the same numerators, i.e., the number of common features in both datasets, while the main difference between them derives from the denominators. For the given test cases, the JI is the most

conservative similarity measure compared to others because the computed scores are always the lowest. By its definition, the JI penalises differences between two faces more strongly, even when the set of edges of the first face is a proper subset of the set of edges of the second face (Figure 12, 5). This is because the JI considers the total membership of both sets of edges in the denominator. The BBC penalises differences in the size of the sets slightly lower than the JI, as the number of common edges is divided by the larger size between the two sets of edges. The SDC doubles (i.e., "weights") the intersection in the numerator and divides it with the sum of the cardinalities from both sets of edges. Consequently, this produces less penalisation and higher similarity than in the case of the JI and BBC. The difference between the CS and SDC is negligible because their denominators, $\sqrt{(|X|\cdot|Y|)}$ against $0.5(|X|+|Y|)$, will result in nearly identical scores if the number of edges in both faces does not differ considerably (i.e., the difference is not greater than one order of magnitude). The SSC is the least conservative similarity measure, which may result in a false positive fully similar face pair in some instances. This happens if all edges in the "smaller" face are also found in the "larger" face (Figure 12, 8). In that case, the score is SSC=1, regardless of how many additional edges are in the "larger" face. Based on the computed scores for the given test cases, it can be concluded that the JI and BBC are inadequate to measure the similarity between two faces due to the considerable penalisation leading to an underestimation of similarity. In contrast, the SSC may overestimate the similarity in some cases. Hence, the CS and SDC, resulting in nearly identical scores, appear to be the most convenient similarity measure between two faces. A face pair is considered similar if the condition $CS \geq S_{TH}$ or $SDC \geq S_{TH}$ is fulfilled, whereby the threshold value can be set, for instance, to $S_{TH}=0.75$ (based on the computed examples in Figure 12). Also, the threshold value for the area ratio can be set to $AR_{TH}=0.90$. In the context of the proposed method, the Cosine similarity has been selected, which is defined as:

$$\mathrm{CS}(F_1, F_2) = \frac{\mathbf{F}_1 \cdot \mathbf{F}_2}{\|\mathbf{F}_1\| \|\mathbf{F}_2\|} \geq \mathrm{S}_{\mathrm{TH}} \tag{67}$$

where $\mathbf{F}_1$ and $\mathbf{F}_2$ are the binary feature vectors of the first and second face. An example of two binary feature vectors used as input for calculating the Cosine similarity is given in Table 13. The size of both binary feature vectors equals and corresponds to the number of unique edges across both faces. Each edge is assigned with "1" if present in the corresponding feature vector, and with "0" if absent.

Table 13. An example of the binary feature vectors for the face pair in Figure 19

Edges	oLI5	oLI5	oLI10	oLI10	iCY1	iCY1
Face 1 – feature vector $\mathbf{F}_1$	1	1	1	1	0	0
Face 2 – feature vector $\mathbf{F}_2$	1	1	1	1	1	1

The peak number of the planes of symmetry candidates n_{POSC} depends on the number of face pairs. As face pairs are computed either from faces of the plane, cylindrical, or spline surface type, the peak value of n_{POSC} can be calculated using one of the following equations:

$$n_{\text{POSC}} = \frac{n_{\text{PL}}\left(n_{\text{PL}}-1\right)}{2} \text{ or } n_{\text{POSC}} = \frac{n_{\text{BS}}\left(n_{\text{BS}}-1\right)}{2} \text{ or } n_{\text{POSC}} = \frac{n_{\text{BS}}\left(n_{\text{BS}}-1\right)}{2} \tag{68}$$

Where n_{PL}, n_{CY}, and n_{BS} are the number of faces of the planar, cylindrical, and spline surface type in the CAD model. After the identification of similar face pairs, the corresponding planes of symmetry candidates are generated using the plane equation, which is defined by the midpoint M(x_{M}, y_{M}, z_{M}) and the unit normal vector to the plane $\mathbf{n}_{\text{P}}$=(a,b,c):

$$a\left(x-x_{\text{M}}\right)+b\left(y-y_{\text{M}}\right)+c\left(z-z_{\text{M}}\right)=0. \tag{69}$$

Two approaches are used to calculate the midpoint. The first is when the paired faces are considered identical, i.e., CS (F_1,F_2)=1, then the corresponding midpoint M between two faces is calculated from their centroids:

$$\text{M}(x_{\text{M}},y_{\text{M}},z_{\text{M}})=\left(\frac{x_{\text{C}_1}+x_{\text{C}_2}}{2},\frac{y_{\text{C}_1}+y_{\text{C}_2}}{2},\frac{z_{\text{C}_1}+z_{\text{C}_2}}{2}\right), \tag{70}$$

The second approach is used if the paired faces are considered similar, i.e., $S_{\text{TH}} \leq$ CS(F_1,F_2) < 1, then the corresponding midpoint M′ between the two faces is computed from their centre points $\text{E}_1(x_{\text{E}_1},y_{\text{E}_1},z_{\text{E}_1})$ and $\text{E}_2(x_{\text{E}_2},y_{\text{E}_2},z_{\text{E}_2})$ as illustrated in Figure 21:

$$\text{M}'(x_{\text{M}},y_{\text{M}},z_{\text{M}})=\left(\frac{x_{\text{E}_1}+x_{\text{E}_2}}{2},\frac{y_{\text{E}_1}+y_{\text{E}_2}}{2},\frac{z_{\text{E}_1}+z_{\text{E}_2}}{2}\right). \tag{71}$$

The centre points E_1 and E_2 are the average position of the midpoints of edges that are identical and shared by both faces:

$$x_{\text{E}_1}=\frac{1}{n_{\text{E}_1}}\sum_{j=1}^{n_{\text{E}_1}}x_{E_{1j,\text{m}}} \qquad y_{\text{E}_1}=\frac{1}{n_{\text{E}_1}}\sum_{j=1}^{n_{\text{E}_1}}y_{E_{1j,\text{m}}} \qquad z_{\text{E}_1}=\frac{1}{n_{\text{E}_1}}\sum_{j=1}^{n_{\text{E}_1}}z_{E_{1j,\text{m}}}, \tag{72}$$

and

$$x_{\mathrm{E}_2} = \frac{1}{n_{\mathrm{E}_2}} \sum_{j=1}^{n_{\mathrm{E}_2}} x_{E_{2\,j,\mathrm{m}}} \qquad y_{\mathrm{E}_2} = \frac{1}{n_{\mathrm{E}_2}} \sum_{j=1}^{n_{\mathrm{E}_2}} y_{E_{2\,j,\mathrm{m}}} \qquad z_{\mathrm{E}_2} = \frac{1}{n_{\mathrm{E}_2}} \sum_{j=1}^{n_{\mathrm{E}_2}} z_{E_{2\,j,\mathrm{m}}} \tag{73}$$

where n_{E_1} and n_{E_2} represent the number of identical edges shared between the two observed faces ($n_{\mathrm{E}_1} = n_{\mathrm{E}_2}$).

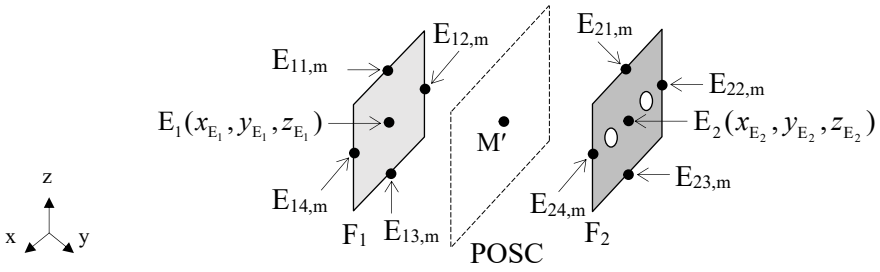
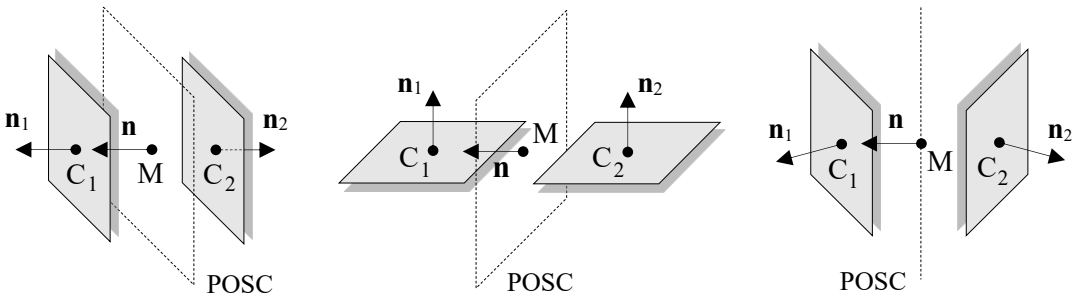


Figure 21. An illustrative example of the faces centre points

The unit normal to the plane of symmetry candidate is computed from the normal of the face pair, considering three possible arrangements between the two faces (Figure 22): parallel, coplanar, or arbitrarily oriented. The arrangement between two faces is calculated from their unit normal at the face centroids (for plane surface) or the projection points of the face centroids onto the faces (for cylindrical and spline surfaces).



a) b) c)

Figure 22. Arrangements between two faces: (a) parallel, (b) coplanar, and (c) arbitrarily oriented.

For parallel arrangement between the unit normal vectors to the faces (Figure 22, a), the unit normal **n** to the POSC corresponds to either the face normal of the first or the second face:

$$\mathbf{n} = (a,b,c) = \mathbf{n}_1 \text{ or } \mathbf{n} = (a,b,c) = \mathbf{n}_2 \tag{74}$$

When the unit normal vectors to the faces are coplanar (Figure 22, b), then the unit normal **n** to the POSC is calculated from the centroid points of the faces (for plane surfaces):

$$\mathbf{n}=(a,b,c)=\frac{\mathbf{C}}{\|\mathbf{C}\|},\ \mathbf{C}=\left(x_{C_2}-x_{C_1},y_{C_2}-y_{C_1},z_{C_2}-z_{C_1}\right), \tag{75}$$

or from the projection points of the face centroids onto the faces (for cylindrical and spline surfaces):

$$\mathbf{n}=(a,b,c)=\frac{\mathbf{C'}}{\|\mathbf{C'}\|},\ \mathbf{C'}=\left(x_{C_2'}-x_{C_1'},y_{C_2'}-y_{C_1'},z_{C_2'}-z_{C_1'}\right). \tag{76}$$

The last arrangement, if unit normal vectors to the faces are arbitrarily oriented (neither parallel nor coplanar), implies that the unit normal **n** to the plane of symmetry candidate is computed as the resultant vector of the face normal vectors (Figure 22, c):

$$\mathbf{n}=(a,b,c)=\frac{\mathbf{R}}{\|\mathbf{R}\|},\ \mathbf{R}=\mathbf{n}_1-\mathbf{n}_2\,. \tag{77}$$

The POSCs can be stored in an array of the size $n_{POSC}\times 6$, where each row has the following form (a, b, c, x_M, y_M, z_M). The described procedure for generating POSCs from similar face pairs covers detecting exact global and partial reflectional symmetry. The AOSCs are generated from single faces to extend the procedure to axisymmetry.

3.3.3 Single faces

Alongside the POSCs and AOSCs generated from PAOI and similar face pairs, additional APOSs are obtained from single closed faces of the following surface types: cylindrical, conical, toroidal, and surface of revolution. The justification for this is that axisymmetric CAD models (exact or partially symmetric) dominantly consist of faces of the mentioned surface types (Figure 23). The past CASD studies generated AOSCs by pairing identical coaxial loops or from single loops of the cylindrical, spherical, and toroidal surface type [23,42], or from the intersection edges between two adjacent faces (considering different combinations of 5 analytic surface types) [16,63].

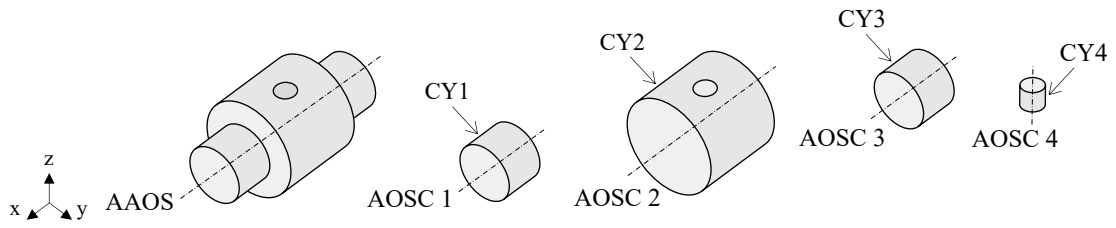


Figure 23. A partially axisymmetric part compound of cylindrical surfaces

The peak number of axes of symmetry candidates n_{AOSC} in the 3D CAD model is equal to the sum of the number of single closed faces of the mentioned surface types:

$$n_{AOSC}=n_{CY}+n_{CO}+n_{TO}+n_{SR}. \tag{78}$$

where n_{CY} is the number of faces of the cylindrical surface type, n_{CO} is the number of faces of the conical surface type, n_{TO} is the number of faces of the toroidal surface type, n_{SR} is the number of faces of the surface of revolution type. The line equation defines the axes of symmetry candidates:

$$\frac{(x-x_A)}{o}=\frac{(y-y_A)}{p}=\frac{(z-z_A)}{q}. \tag{79}$$

where o, p, and q represent the components of the unit direction vector $\mathbf{a}=(o,p,q)$ of the cylinder axis, and $A(x_A, y_A, z_A)$ is any point on the axis. The AOSCs can be stored in an array of the size $n_{AOSC}\times 6$, where each row has the following form (o, p, q, x_A, y_A, z_A). Apart from the AOSCs generated from single faces, additional candidates are always generated from the 3D CAD model's PAOI (see Subsection 3.3.1). To summarise, the candidates generated from single faces, alongside those generated out of similar face pairs and PAOI, are used as input in the following step, where they are subjected to trimming.

3.4 Trimming of planes and axes of symmetry candidates

Once the POSCs and AOSCs have been generated, they are subjected to the trimming step to eliminate duplicates or unsuitable candidates. A duplicate candidate is considered any POSC with at least one coplanar POSC or any AOSC with at least one coaxial AOSC. The coplanarity and coaxiality are queried with the plane and line equations (69) and (79) together with the length of the cross-product length between the POSC's normal vector or the AOSC's direction vector, which needs to be zero. A similar approach of eliminating POSCs and AOSCs duplicates was also used in [23,42]. On the other hand, an unsuitable candidate is any POSC or AOSC that is considerably distanced from the 3D CAD model's COG and is unlikely to become an APOS or AAOS. The reason for trimming unsuitable candidates arises from the following facts:

1) The APOS or AAOS of an exact symmetric object (i.e., a 3D CAD model) must pass through its COG [42] and
2) The APOS and AAOS of a partially symmetric object (i.e., a 3D CAD model) will be close to its centroid [138].

Based on that, unsuitable candidates are queried by the point-to-plane or point-to-line distance. The point-to-plane distance d_{POSC} is used to compute the distance between the COG and the POSC:

$$d_{\mathrm{POSC}} = \frac{\left|ax_{\mathrm{C}} + by_{\mathrm{C}} + cz_{\mathrm{C}} + d\right|}{\sqrt{a^2 + b^2 + c^2}}, \tag{80}$$

where $d = -ax_{\mathrm{M}} - by_{\mathrm{M}} - cz_{\mathrm{M}}$. Analogically, the point-to-line distance d_{AOSC} between the COG and the axis of symmetry candidate is calculated as follows:

$$d_{\mathrm{AOSC}} = \frac{\left\|\mathbf{b} \times \mathbf{a}\right\|}{\left\|\mathbf{a}\right\|}, \tag{81}$$

where **b** is the vector between the COG denoted as C (x_{C}, y_{C}, z_{C}) and any point A on the axis. Finally, to trim unsuitable candidates, d_{POSC} or d_{AOSC} need to be less than the maximum allowed distance $\delta_{\max}$, i.e., $d_{\mathrm{POSC}} \leq \delta_{\max}$ and $d_{\mathrm{AOSC}} \leq \delta_{\max}$. The $\delta_{\max}$ value must be carefully chosen. A too-high value of $\delta_{\max}$ can lead to many untrimmed POSCs or AOSCs, which are not likely to become APOSs or AAOS. Conversely, a too-low value of $\delta_{\max}$ can result in many trimmed POSCs or AOSCs, among which there might be APOSs or AAOS. Hence, the $\delta_{\max}$ was defined empirically, based on the minimum bounding box, i.e., the smallest box in which the 3D CAD model fits:

$$\delta_{\max} = 0.05 \cdot \sqrt{\left(\Delta L_x\right)^2 + \left(\Delta L_y\right)^2 + \left(\Delta L_z\right)^2}, \tag{82}$$

where ΔL_x, ΔL_y, and ΔL_z represent the lengths of the minimum bounding box in the respective directions of coordinate axes (x, y, and z). The interpretation of Equation (82) is that the $\delta_{\max}$ is within 5% of the CAD model's minimum bounding box diagonal length. Hence, the APOS(s) or AAOS of a partial symmetric CAD model is maximumly distanced from the COG by $\delta_{\max}$. Many CAD systems provide the functionalities for calculating the minimum bounding box. Alternatively, the minimum bounding box of a solid CAD model can be computed by exploiting existing methods. For instance, the method reported in [139] uses an iterative approach and shifts the problem of finding the minimum bounding box of a solid into the 2D domain by using projected contours of the solid onto the three principal planes.

Finally, Figure 24 illustrates the corresponding flowchart of the trimming step. The output of this step is a set of remaining POSCs and AOSCs that are used as input in the evaluation step.

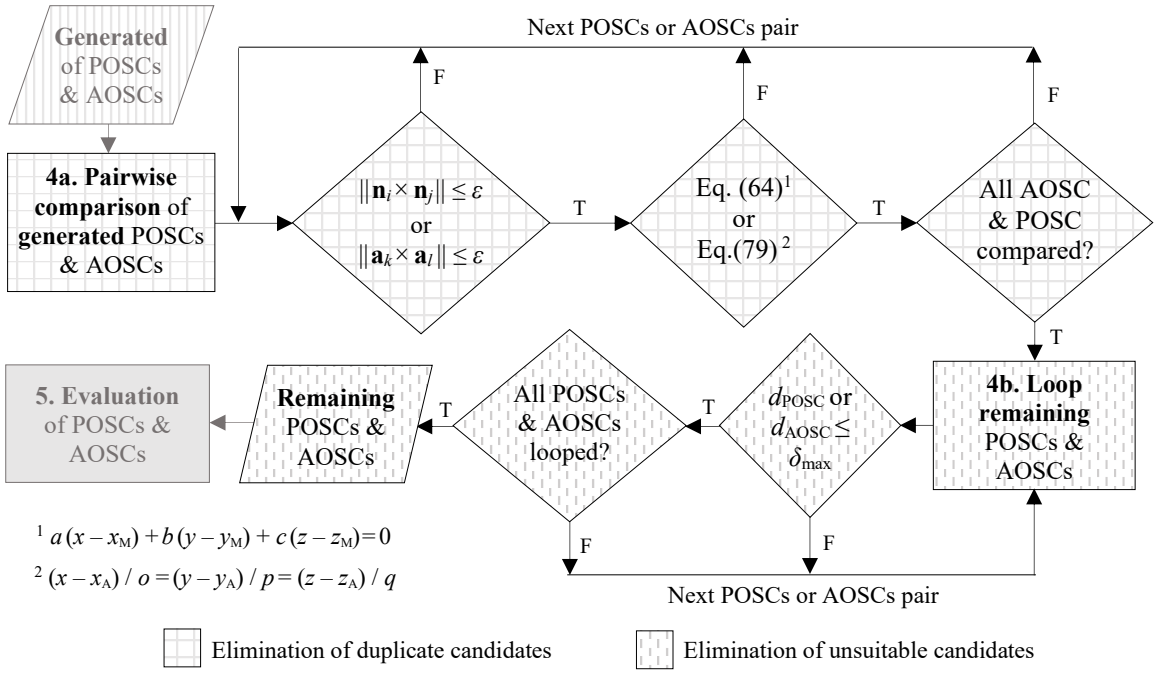


Figure 24. Flowchart of the candidate trimming step

3.5 Evaluation of planes and axes of symmetry candidates

All remaining POSCs and AOSCs from the trimming step are evaluated individually with respect to the 3D CAD model's B-rep. The evaluation relies on vector calculus using classified faces and edges with their specific properties from the B-rep analysis step as input. The previous CASD studies relied their evaluation on the mutual comparison of POSCs and AOSCs and propagation of local symmetry properties associated with topological entities (loops or faces) to the global level. In the past, CASD studies achieved this by ranking rationalised candidates according to the number of loops and total loop area associated with a rationalised candidate [23,42]. Another way was to extend the POSCs' or AOSCs' validity over the largest possible area of the B-rep to detect the 3D CAD model's global symmetry properties [16,63]. The CASD method within this study, however, evaluates each POSC and AOSC individually by searching for symmetric face pairs and self-symmetric stand-alone faces that match a set of criteria discussed in this subsection.

The evaluation step consists of two sub-steps. The first sub-step, illustrated through the flowchart in Figure 25, evaluates the existence of global symmetry. The second sub-step evaluates partial symmetry, and its flowchart is described in Figure 26. Both sub-steps consist of two parts, i.e., the evaluation of POSCs for detecting reflectional symmetry

and the evaluation of AOSCs for detecting axisymmetry. During the implementation of the CASD method, the two parts of the sub-steps can be conducted simultaneously. However, they are described separately in the following subsections for better understanding.

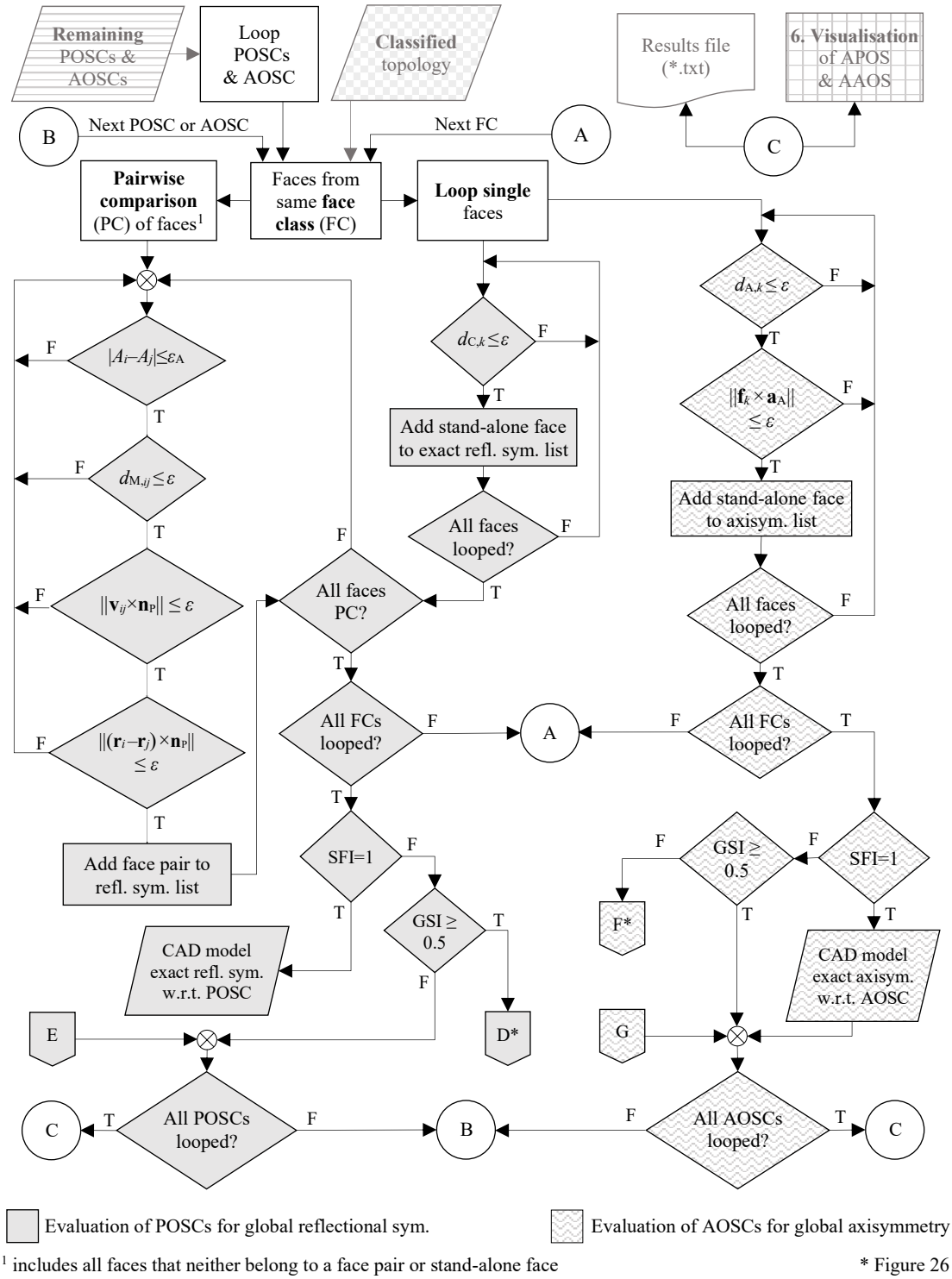
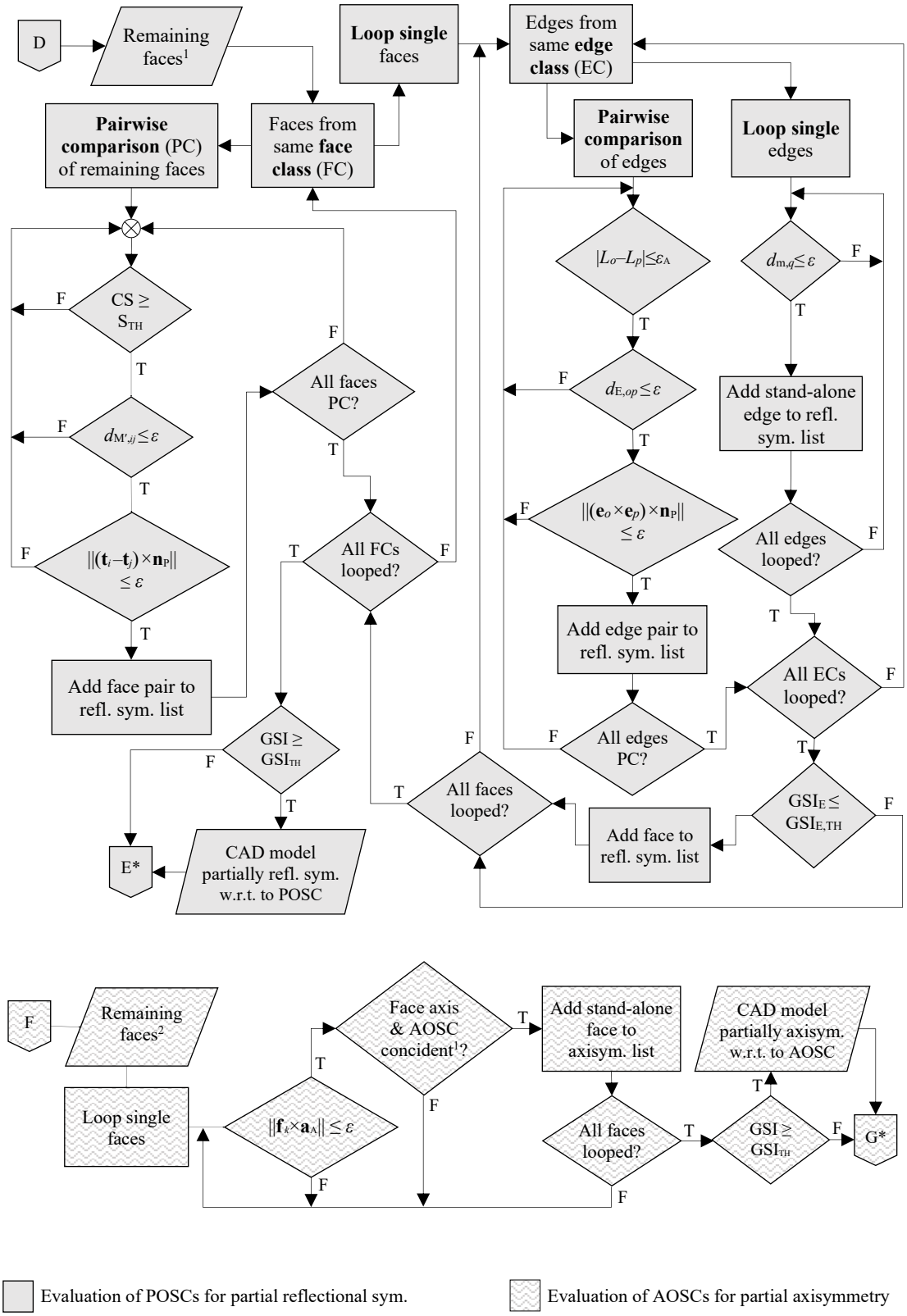


Figure 25. Flowchart of the POSCs and AOSCs evaluation step (part 1)



[1] includes all faces that neither belong to a symmetric face pair or stand-alone face
[2] includes all faces not recognised as self-symmetric stand-alone faces
[3] Eq. (79)
[*] Figure 25

Figure 26. Flowchart of the POSCs and AOSCs evaluation step (part 2)

3.5.1 Global symmetry

The first evaluation sub-step aims to detect exact global symmetry in the 3D CAD model (Figure 25). In this sub-step, the primary inputs for evaluating the POSCs or AOSCs are the B-rep's faces. As a result of this evaluation sub-step, the CASD method detects either global reflectional symmetry or axisymmetry.

3.5.1.1 Reflectional symmetry

The faces of reflectional symmetric 3D CAD models can be divided into three categories:

1) symmetric face pairs,
2) symmetric stand-alone faces, and
3) non-symmetric (i.e., asymmetric) faces.

A symmetric face pair consists of two faces which are reflectional symmetric with respect to the observed POSC. A stand-alone face is a reflectional self-symmetric single face intersected by the observed plane of symmetry candidate and divided into two equal reflectional symmetric parts. All faces with no symmetric pair or that are not reflectional self-symmetric are categorised as non-symmetric. Those faces are declared as remaining faces and used as input in the second evaluation process to verify if they are partially symmetric (Subsection 3.5.3). Symmetric face pairs and stand-alone faces can be found in exact and partial reflectional symmetric 3D CAD models, while non-symmetric faces occur only in partially reflectional symmetric 3D CAD models.

Symmetric face pairs are identified using a pairwise comparison process between all faces of the same class. Three criteria need to be fulfilled for judging whether a face pair is symmetrical with respect to the POSC: equality, coincidence, and orientation. The equality criterion is satisfied if the difference between the surface areas of i-th and j-th face ($i \neq j$) is below the computation error ε_A:

$$\left| A_i - A_j \right| \leq \varepsilon_A . \tag{83}$$

At this point, it can be assumed that $\varepsilon_A = \varepsilon = 10^{-6}$ m. If Equation (83) is satisfied, next, the coincidence criterion is assessed through the point-to-plane distance $d_{M,ij}$ (analogical to Equation (80)) between the ij-th ($i \neq j$) face pair midpoint of the face pair and the corresponding POSC. The face pairs count from 1 to the number of faces in the observed face class. To meet the coincidence criterion, the point-to-plane distance $d_{M,ij}$ needs to be within the specified computation error:

$$d_{\mathrm{M},ij} \leq \varepsilon. \tag{84}$$

The last criterion that must be fulfilled is the orientation of the face pair's normal and position vectors. The normal vectors are queried for each face pair with the cross-product vector $\mathbf{p}_{ij}$, computed between the orientation vector $\mathbf{v}_{ij}$ and the normal vector to the POSC $\mathbf{n}_{\mathrm{P}}$:

$$\mathbf{p}_{ij} = \mathbf{v}_{ij} \times \mathbf{n}_{\mathrm{P}}. \tag{85}$$

The orientation vector $\mathbf{v}_{ij}$ is calculated based on the arrangement between two faces, which can be parallel, coplanar, or arbitrarily oriented (analogical to the cases in Figure 22). The orientation vector $\mathbf{v}_{ij}$ for a face pair is calculated from the normal vectors at the projected point of the face centroid onto the face (for an open face) or their axis vectors at the face centroid (for a closed face). If a face pair is parallel, the orientation vector is obtained from either face's normal vector using Equation (74). For a coplanar face pair, then the orientation vector is obtained from Equation (75). The orientation vector of an arbitrarily oriented face pair is calculated from Equation (76). The first part of the orientation criterion is satisfied if the length of the cross-product vector $\mathbf{p}_{ij}$ is below ε:

$$\left\| \mathbf{p}_{ij} \right\| \leq \varepsilon. \tag{86}$$

The second part of the orientation criterion is related to the position vectors. For each face, the position vector is computed between the point describing the POSC (i.e., the midpoint M(x_{M}, y_{M}, z_{M})) and the characteristic face point (centroid or its projection). The position vectors are exploited to verify if the face pair is invariant to reflectional transformation with respect to the POSC. First, the position vectors are subtracted for each face pair, denoted as the vector $\mathbf{s}_{ij}$:

$$\mathbf{s}_{ij} = \mathbf{r}_i - \mathbf{r}_j, \tag{87}$$

where $\mathbf{r}_i$ and $\mathbf{r}_j$ are the position vectors of the i-th and j-th face ($i \neq j$). Next, to meet the second part of the orientation criterion, the cross-product length between $\mathbf{s}_{ij}$ and $\mathbf{n}_{\mathrm{P}}$ needs to be below the computation error ε:

$$\left\| \mathbf{s}_{ij} \times \mathbf{n}_{\mathrm{P}} \right\| \leq \varepsilon. \tag{88}$$

The second category of faces in reflectional symmetric 3D CAD models are the stand-alone faces, i.e., single faces that do not belong to any symmetrical face pair. A stand-alone face is reflectional self-symmetric and needs to satisfy only the coincidence criterion. In other words, the centroid (for a closed face) or projection of the centroid

onto the face (for an open face) must coincide with the POSC. Again, this is queried with point-to-plane distance $d_{C,k}$ (like in Equation (80)), where k represents the k-th face in the observed face class. To consider a stand-alone face reflectional self-symmetric, the condition $d_{C,k} \leq \varepsilon$ needs to be fulfilled.

Once the mentioned criteria were checked for all faces, the 3D CAD model is considered exact symmetrical if the number of symmetric face pairs n_{FP} and symmetric stand-alone faces n_{SF} is equal the total number of faces n_F in the 3D CAD model:

$$n_{FP} + n_{SF} = n_F. \tag{89}$$

Otherwise, if the equation above is not in equilibrium, the 3D CAD model is potentially partially symmetric. By further rearrangement of the equation above, the symmetrical faces index (SFI) can be introduced:

$$\mathrm{SFI} = \frac{n_{FP} + n_{SF}}{n_F}. \tag{90}$$

The SFI is the ratio between the number of symmetrical faces (face pairs and stand-alone faces) and the total number of faces in the 3D CAD model. The SFI aims to measure the symmetry of the topology in the B-rep CAD model. It is computed for each POSC and has a range of [0, 1], where SFI=0 indicates non-symmetry and SFI=1 exact global reflectional symmetry. SFI < 1 indicates that the 3D CAD model might be partially symmetric. Although the SFI provides a measure for detecting exact global symmetry, it is unsuitable for measuring partial symmetry due to its sensitivity to a lower number of topological entities (faces) in the 3D CAD model. Therefore, another measure appropriate for partial symmetry is proposed, discussed in Subsection 3.5.2. The evaluation of axisymmetry is introduced in the following subsection.

3.5.1.2 Axisymmetry

The category of faces present in exact axisymmetric 3D CAD models are only stand-alone faces, while partially axisymmetric 3D CAD models have also non-symmetric faces. Symmetrical stand-alone faces must fulfil two criteria: coincidence and orientation. A face is axisymmetric with respect to the AOSC if its centroid (applies for a closed face) or projection of the centroid onto the face (applies for an open face) coincides with the AOSC. This is queried with the point-to-line distance from Equation (81), in this case, designated as $d_{A,k}$. The coincidence criterion is fulfilled if $d_{A,k} \leq \varepsilon$. On

the other hand, the orientation criterion is queried with the cross-product vector $\mathbf{q}_k$ of the k-th face, which is computed between the face vector $\mathbf{f}_k$ (normal or axis vector) and the AOSC orientation vector $\mathbf{a}_A$:

$$\mathbf{q}_k = \mathbf{f}_k \times \mathbf{a}_A. \tag{91}$$

The orientation criterion is met if the length of the cross-product vector is below ε:

$$\|\mathbf{q}_k\| \leq \varepsilon. \tag{92}$$

The 3D CAD model exhibits exact axisymmetry if the number of all stand-alone faces n_{SA} is in equilibrium with the total number of faces n in the 3D CAD model:

$$n_{SF} = n_F. \tag{93}$$

In the case of axisymmetry, the number of symmetrical face pairs is zero and can be eliminated from the Equation (90), so the SFI turns into:

$$\mathrm{SFI} = \frac{n_{SF}}{n_F}. \tag{94}$$

Hence, in this case, the SFI is defined as the ratio between the number of stand-alone faces and the total number of faces in the 3D CAD model. Analogical to reflectional symmetry, the SFI is also calculated for each AOSC, where SFI=0 represents non-symmetry and SFI=1 exact global axisymmetry. If SFI<1, the 3D CAD model is further investigated for partial axisymmetry. For that purpose, the global symmetry index is proposed, which also applies to reflectional symmetry.

3.5.2 Global symmetry index

The SFI intends to measure exact symmetry and applies to reflectional and axisymmetry. However, it cannot measure partial symmetry within the 3D CAD model. This is because the SFI score may be sensitive to the number of faces in the 3D CAD, especially if the number of faces is low. Therefore, another symmetry measure is introduced, the global symmetry index (GSI), which represents the ratio between the surface area of symmetrical faces A_{sym} and the total surface area A of the 3D CAD model:

$$\mathrm{GSI} = \frac{\int_{S_{sym}} dA_{sym}}{\int_S dA}. \tag{95}$$

The equation above can be further expressed in terms of the surface area of symmetric face pairs and stand-alone faces:

$$\mathrm{GSI} = \frac{\sum_{i=1}^{n_{\mathrm{FP}}} A_{\mathrm{FP},i} + \sum_{j=1}^{n_{\mathrm{SA}}} A_{\mathrm{SA},j}}{\sum_{k=1}^{n} A_k}, \tag{96}$$

where $A_{\mathrm{FP},i}$ represents the total surface area of the i-th face pair, $A_{\mathrm{SA},j}$ the surface area of j-th stand-alone face, and A_k the surface area of the k-th face in the 3D CAD model. The GSI applies to both reflectional and axisymmetry and is computed for each POSC and AOSC. It has a range of [0, 1], where GSI=0 represents non-symmetry and GSI=1 exact global symmetry. In the case of partial symmetry, the GSI will be close to 1, while the threshold value $\mathrm{GSI}_{\mathrm{TH}}$, below which the 3D CAD model is not considered partially symmetric anymore, needs to be determined by testing. If the 3D CAD model exhibits exact global symmetry, then SFI=1 and GSI=1. The SFI measures the symmetry of the topology of the B-rep CAD model, while the GSI measures the symmetry of the underlying geometry.

In addition to the proposed symmetry measures, there are also certain combinations and restrictions regarding the types of symmetries a 3D CAD model may or may not exhibit, which can be described with the symmetry correlation matrix (Table 14). The correlation between the types of symmetries (globally reflectional, globally axisymmetry, partially reflectional, and partially axisymmetry) must be implemented into the CASD method. Specific correlations are already, by default, maintained by the CASD method through its definition. For instance, a 3D CAD model cannot be simultaneously globally axisymmetric and partially reflectional symmetric. Another example is that a 3D CAD model cannot be simultaneously globally and partially axisymmetric. However, an additional check is required to see if the 3D CAD model is simultaneously globally axisymmetric and reflectionally symmetric. In that case, reflectional symmetry can only exist perpendicular to axisymmetry. Therefore, the detected APOS(s) should be checked for coincidence with the AAOS and eliminated in case of confirmation.

Table 14. Symmetry correlation matrix

Type of symmetry in th 3D CAD model	Globally reflectional symmetry	Globally axisym.	Partially reflectional symmetry	Partially axisym.
Globally reflectional symmetry	–	✓[1]	✓	✓
Globally axisymmetry	✓[1]	–	✗	✗
Partially reflectional symmetry	✓	✗	–	✓
Partially axisymmetry	✓	✗	✓	–

✓ Combination of symmetries possible ✗ Combination of symmetries not possible

[1] The detected APOS and AAOS must be perpendicular to each other

3.5.3 Partial symmetry

If global symmetry is not detected (SFI<1), a second evaluation sub-step is performed to detect partial symmetry in the 3D CAD model (Figure 26). The criterion to start the second evaluation sub-step is that GSI>0.5, i.e., 50% of the total surface area is symmetric. In the second evaluation sub-step, the inputs are all remaining non-symmetric faces (i.e., do not belong to a face pair or stand-alone face) from the first sub-step. Apart from faces, the edges of the B-rep are also exploited as input for evaluating the POSCs or AOSCs. This second evaluation sub-step addresses the detection of partial reflectional and axisymmetry in the 3D CAD model.

3.5.3.1 Reflectional symmetry

The procedure for detecting partially reflectional symmetric face pairs and stand-alone faces is analogous to the previously described global symmetry detection procedure. The main difference is that lower topological entities, i.e., edges, are exploited to evaluate partial symmetry. The remaining undetected faces from the first sub-step are pairwise compared (only from the same class) to detect partially reflectional symmetric face pairs. In contrast, single faces are looped to detect partially reflectional self-symmetric stand-alone faces. Three criteria need to be fulfilled for judging whether a face pair is partially reflectional symmetric with respect to the POSC: similarity, coincidence, and orientation. The similarity criterion implies identifying if a face pair is similar using the Cosine similarity from Equation (67) in the same manner as described in subsection 3.3.2. If the condition $CS \leq S_{TH}$ is satisfied, then the midpoint M' for the similar face pair is calculated from the face centre points. A face centre point is the average position of the edge midpoints that are identical and shared by both faces. The coincidence criterion is assessed using the point-to-plane distance $d_{M'ij}$ (comparable to Equation (80)) between the

midpoint M′ and the POSC. If the coincidence criterion is met, i.e., $d_{M'ij} \leq \varepsilon$, then the orientation criterion is queried. For that purpose, the position vectors of the faces $\mathbf{t}_i$ and $\mathbf{t}_j$ are computed and subtracted. A position vector points from the relevant POSC point to the face centre point E_i. A similar face pair meets the orientation criterion if the cross-product length between subtracted faces' position vectors and the normal to the POSC is below the specified computation error ε:

$$\left\|(\mathbf{t}_i - \mathbf{t}_j) \times \mathbf{n}_P\right\| \leq \varepsilon. \tag{97}$$

Finally, if all three criteria are satisfied (similarity, coincidence, and orientation), then observed face pair is considered partially reflectional symmetric. A stand-alone face is assessed for partial reflectional self-symmetry through its edges. Three edge types are distinguished: reflectional symmetric edge pairs, self-symmetric stand-alone edges, and non-symmetric edges. The assessment of reflectional symmetric edge pairs consists of the following criteria: equality, coincidence, and orientation. The equality criterion implies that two edges belonging to the same class are of equal length, i.e.:

$$\left|L_o - L_p\right| \leq \varepsilon_A. \tag{98}$$

To examine the coincidence criterion, the edge pair midpoint between the o-th and p-th edge is used to assess distance from the POSC. The point-to-plane distance, denoted as $d_{E,op}$, is computed via the modified Equation (80). The last criterion to be met is the orientation of the edges' position vectors $\mathbf{e}_o$ and $\mathbf{e}_p$, which must satisfy the equation:

$$\left\|(\mathbf{e}_o - \mathbf{e}_p) \times \mathbf{n}_P\right\| \leq \varepsilon. \tag{99}$$

An edge position vector points from the relevant POSC point to the edge midpoint. Finally, if all three criteria are satisfied (equality, coincidence, and orientation), the observed edge pair is partially reflectional symmetric. A stand-alone edge is reflectional self-symmetric if it fulfils the coincidence criterion, which is assessed with the point-to-plane distance $d_{m,q}$ computed between the edge midpoint and POSC. To estimate if a stand-alone face is partially reflectional self-symmetric, the GSI needs to be slightly modified by using the edge length instead of the face area:

$$\mathrm{GSI}_E = \frac{\sum_{i=1}^{n_{LP}} L_{EP,i} + \sum_{j=1}^{n_{SL}} L_{SE,j}}{\sum_{k=1}^{n_{LI}} L_k} \leq \mathrm{GSI}_{E,\,TH}, \tag{100}$$

where $L_{EP,i}$ represents the length of the i-th edge pair, $L_{SE,j}$ the length of the j-th stand-alone edge, and L_k the length of the k-th edge. Like the GSI, the GSI_E also ranges between

[0, 1]. Now, it is just a matter of finding the threshold $GSI_{E,TH}$ above which the stand-alone face is partially reflectional self-symmetric.
After examining the existence of partially reflectional symmetric face pairs and self-symmetric stand-alone faces from the remaining faces, the GSI is updated and needs to be $GSI \geq GSI_{TH}$ to confirm that the 3D CAD model is partially reflectional symmetric.

3.5.3.2 Axisymmetry

As already emphasised, the remaining faces from the first sub-step are exploited in the second sub-step for evaluating partial axisymmetry. The categories of faces in partially axisymmetric 3D CAD models are axisymmetric stand-alone, partially axisymmetric stand-alone, and non-symmetric faces. Axisymmetric stand-alone faces are detected through the procedure described in Subsection 3.5.1. Two criteria are assessed to detect a partially axisymmetric stand-alone face: orientation and coincidence. The orientation criterion assesses if the cross-product length between the face's normal or axis vector and the AOSC orientation vector is below ε (Equations (91) and (92)). If this criterion is satisfied, the next criterion is to query if the face axis coincides with the AOSC, which is accomplished using the line equation (79). Finally, a face is considered partially axisymmetric if both criteria are satisfied (orientation and coincidence). After evaluating partially axisymmetric stand-alone faces, the GSI is updated and needs to be $GSI \geq GSI_{TH}$ to confirm that the 3D CAD model is partially axisymmetric with respect to the AOSC, which is then declared as the AAOS. At the end of the evaluation step, each POSC or AOSC that has been evaluated to be an APOS or AAOS is then visualised within the 3D CAD model.

3.6 Visualisation of actual planes and axes of symmetry

The symmetry information is provided by visualising the detected APOS(s) and AAOS within the 3D CAD model. An APOS is defined by its unit normal vector and any point on the plane, i.e., the midpoint between the two similar faces. On the other hand, an AAOS is defined by its unit axis vector and any point on the axis, i.e., the origin of the cylindrical surface. There are several ways planes can be created in CAD systems [140,141]: through point and line, through three points, through two lines, plane equation and reference point, etc. Analogically, there are multiple ways to create axes in CAD systems [140,141]: through two points, point and direction, normal to surface, etc. When it comes to storing the symmetry information, there are two approaches (Figure 27): internal or external. The internal approach implies that the APOS(s) and AAOS are

visualised and permanently stored within the 3D CAD model. On the other hand, in the external approach, the APOS(s) and AAOS can be retrieved from a results file associated with the 3D CAD model and visualised within the 3D CAD model using the computational environment. The drawback of the internal approach is that the APOS(s) and AAOS may be lost during an exchange of the CAD model in a neutral format (STEP, IGES, etc.), while the external approach requires the use of the computational environment to retrieve the APOS(s) and AAOS. Apart from the visualisation of APOS and AAOS, additional visualisation of symmetric and non-symmetric faces which are related to the corresponding APOS or AAOS should be provided. The visualisation of reference geometry (planes and axes) can be accomplished through the CAD system functionalities at the API level. Hence, further details regarding visualisation are discussed during the implementation of the proposed CASD method (Subsection 4.1.5).

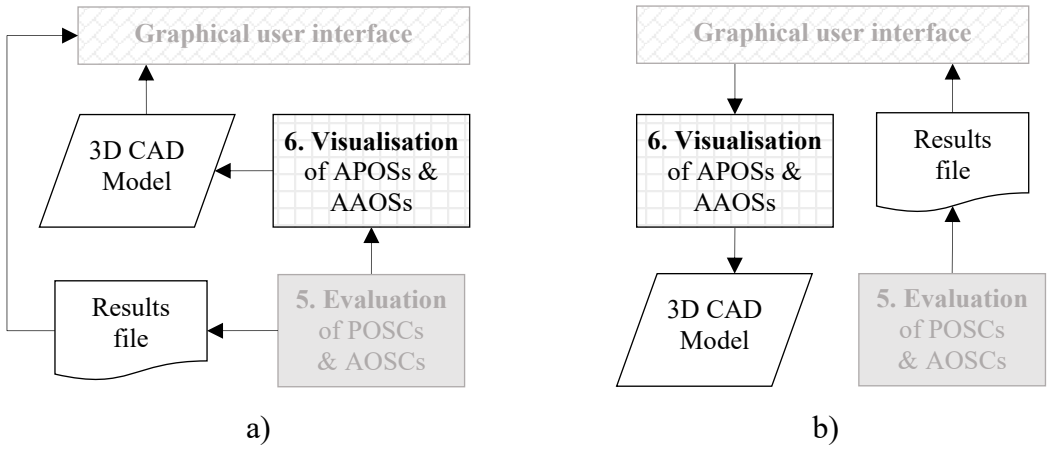


Figure 27. Internal (a) and external (b) approach to store the visualisation of APOSs and AAOSs

Since all steps of the CASD method were presented, a data model for the CASD method can be built. The data model is important because it visually describes the CASD method's data requirements and enables its easier implementation among different CAD systems.

3.7 Data model

The data model is constructed using the unified modelling language (UML), expressed as class diagrams. These diagrams are widely used in object-oriented system modelling [142]. A class diagram typically includes a group of classes and their interrelationships. Each class defines a set of objects that share common attributes (properties), methods (operations), and relationships. The CASD method employs two class diagrams, one

outlining data relevant to the B-rep analysis step and another summarising relevant data for the remaining steps. Figure 28 displays the class diagram for the B-rep analysis step. The importance of the CASD method's data model is its use in the implementation into a CAD system (Chapter 4).

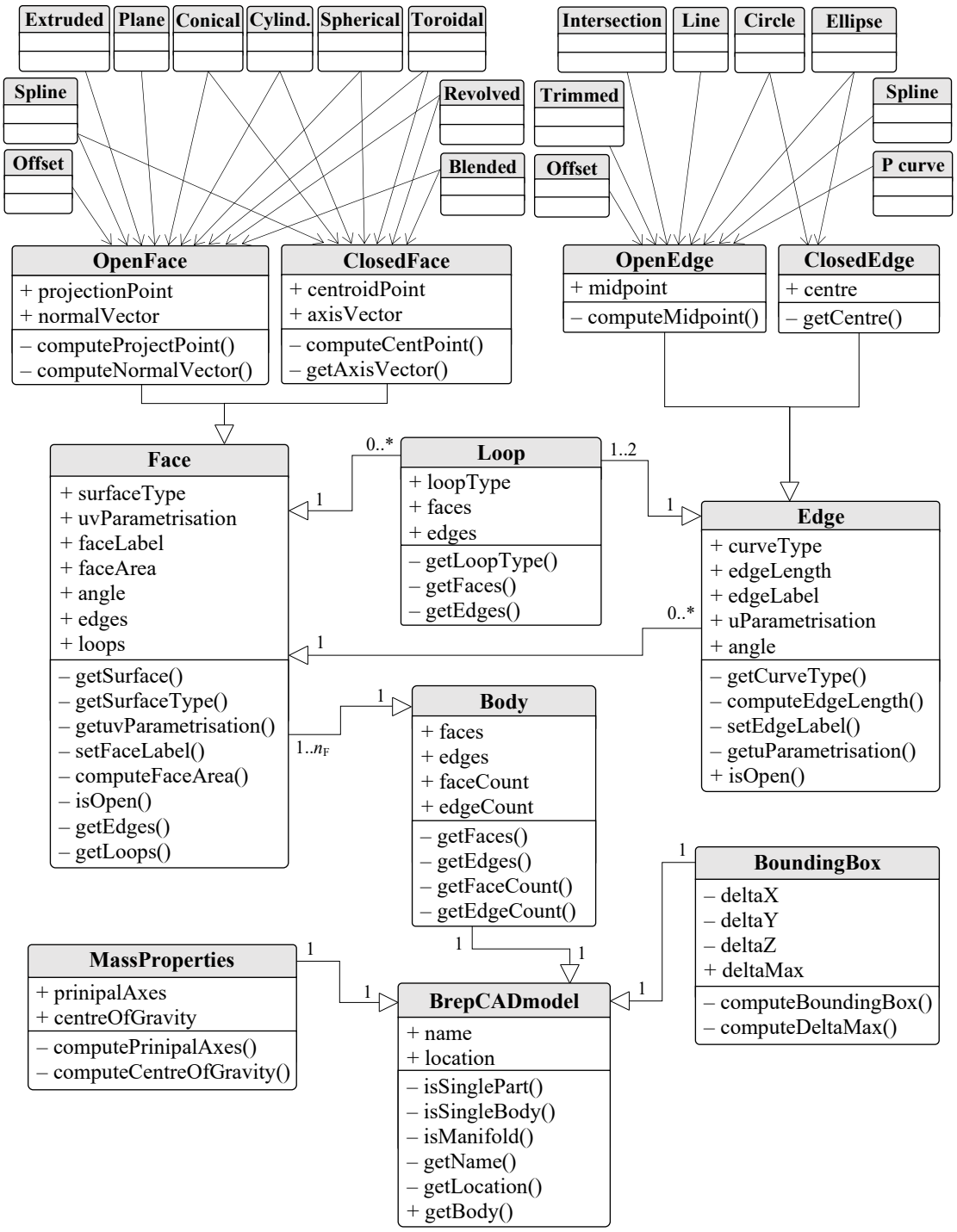


Figure 28. Class diagram for the step analysis of B-rep

The class diagram is structured into distinct classes, each with its own attributes and methods (Figure 28):

- *BRepCADmodel* – this class encompasses general information pertaining to the B-Rep CAD model. Among its attributes are the *name* and *location* of the input CAD model. It also includes methods such as *getName()* to retrieve the name of the input CAD model, *getLocation()* to obtain the location of the input 3D CAD model, *isSinglePart()* to check if the input 3D CAD model is a single part, *isSingleBody()* to verify if the input 3D CAD model has one body, and *isManifold()* to determine if the input 3D CAD model is manifold.
- *MassProperties* – defines the mass properties of the input 3D CAD model. Its attributes are the *prinipalAxes* representing the PAOI, and the *centerOfGravity* designates the centre of gravity, while its methods are related to the computation of the mentioned attributes. The mass properties belong only to one input B-rep CAD model.
- *BoundingBox* – defines the maximum bounding box of the 3D CAD model and consists of the attributes *deltaX*, *deltaY*, and *deltaZ* which represent the lengths of the 3D CAD model's minimum bounding box in the respective directions of the coordinate axes (*x*, *y*, and *z*). The attribute *deltaMax* represents the maximum allowed distance of the candidate from the COG used in the trimming step. The *BRepCADmodel* and the *BoundingBox* relationship is one-to-one, as the 3D CAD model has one minimum bounding box.
- *Face* – this class defines a face in the B-Rep and contains attributes and methods relevant to the CASD method. The attributes are the underlying *surfaceType*, the *faceArea*, the underlying surface *uvParametrisation,* the periodicity *angle*, which describes if the face is open or closed, the *faceLabel*, the face's *edges* and *loops*. A solid B-rep CAD model can have one to n_F number of faces (1..n_F). In the context of the CASD method, a *Face* can be open or closed, so the corresponding sub-classes are the *OpenFace* or *ClosedFace*.
- *OpenFace* – this class defines the relevant information of an open face. Its attributes are the *normalVector_n* and *projectionPoint*.
- *ClosedFace* – this class defines the relevant information of an open face. Its attributes are the *axisVector_a* and *centroidPoint*.
- *Loop* – this class defines the relevant information of a loop. The corresponding attribute is the *loopType*, which can be either outer or inner, and its associated

method is the *getLoopType()*. A *face* can have zero to many loops (0..*), while an *Edge* belongs to one or two loops (1..2).

- *Edge* – this class defines the edge and contains relevant attributes and methods for the B-Rep analysis. The attributes include the underlying *curveType*, *edgeLength*, and *edgeLabel*. A *face* has zero to many edges (0..*). An *edge* can be open or closed. Hence, its corresponding sub-classes are the *OpenEdge* or *ClosedEdge*.
- *OpenEdge* – this class defines the relevant data of an open edge. The attribute of an open edge is its *midpoint*, computed through the corresponding method *computeMidpoint()*.
- *ClosedEdge* – this class defines the relevant data of an open edge, the attribute edge *centre*, and the associated method *getCentre()* retrieves the edge's centre from the associated curve parameters.

The necessary information for generating, trimming, evaluating, and visualising candidates can be found in two class diagrams: reflectional symmetry (Figure 29) and axisymmetry (Figure 30). The reflectional symmetry class diagram is comprised of several classes, each with its attributes and methods (Figure 29):

- *GeneratePOSC* – this class defines the data relevant for generating a POSC from the corresponding *SimilarFacePair* and PAOI. It includes the attributes *planePoint*, which determines the location on the plane, and the *normalVector_n*, which determines the normal vector of the plane. The corresponding methods are *computPlanePoint()*, *computeNormalVector_n()*, and *addPlane()*, which adds the candidate to a list.
- *SimilarFacePair* – is a sub-class of *GeneratePOSC* to generate the candidate from a similar face pair, consisting of the attributes *areaRatio*, *cosineSimilarity*, *featureVector1*, *featureVector2*, and the equivalent methods for computing them *computeAreaRatio()*, *computeCosineSimilarity()*, *computeFeatureVector1()*, and *computeFeatureVector2()*. The method *isSimilar()* determines whether the face pair is similar.
- *TrimDuplicatePOSC* – this class defines the trimming of a duplicate POSC. It consists of the attribute *lgthCross_n1_n2*, which defines the cross-product length between two POSC's normal vectors. The methods are *computePlaneEquation()*, which calculates the plane equation to query if two POSCs are coincident, *computeLgthCross_n1_n2()*, which computes the cross-product length between two candidates to check if they are parallel, *isDuplicate()* provides the information if a candidate is duplicate, and *deletePlane()* which removes the duplicate POSC.

All generated POSCs are exploited in the trimming. Hence, the total number of generated POSCs n_{POSC} from the class *GeneratePlaneCandidate* is used in the class *TrimDuplicatePlane* (n_{POSC} to n_{POSC}).

- *TrimUnsuitablePOSC* – this class defines the trimming of an unsuitable POSC. Its attribute is the *pointPlaneDist*, which defines the point-to-plane distance between the POSC and COG to be compared with δ_{max}, while the methods are *computePointPlaneDist()*, *isUnsuitable()*, which checks if the POSC is unsuitable, and *deletePlane()*, which removes the unsuitable POSC from the list.
- *EvaluatePOSC* – this class defines the evaluation of a POSC. The class consists of the attributes *scoreGSI* and *scoreSFI*, providing the GSI and SFI values for the POSC. Its methods are *computeScoreGSI()*, *computeScoreSFI()*, and *isAPOS()* to determine if the POSC is an APOS. *EvaluatePOSC* consists of four sub-classes: *StandAloneFace*, *FacePair*, *RemainingStandAloneFace*, and *RemainingFacePair*. The first two are related to the evaluation of exact symmetrical faces, while the last two are related to the second evaluation process for partially symmetrical faces.
- *StandAloneFace* – this class defines the data for evaluating a stand-alone face. The attribute is *poinPlaneDist()*, representing the point-to-plane distance between the characteristic face point and the POSC. The methods are *computePoinPlaneDist()* and *isSymmetrical()* for determining whether the stand-alone face is symmetrical.
- *FacePair* – this class defines the data for evaluating a face pair. The relevant attributes are *areaDifference* (difference between the areas of two faces), *midpoint* (the face pair midpoint), *pointPlaneDist* (the point-to-plane distance between midpoint and POSC), *positionVect_r1* (position vector of 1st face), *positionVect_r2* (position vector of 2nd face), *subtractVect_s* (subtracted vector between vectors *positionVect_r1* and *positionVect_r2*), *orientVect_v* (orientation vector of the face pair depending on the arrangement between the 1st and 2nd face), *s_x_n_lgth* (cross-product length between *subtractVect_s* and POSC's normal vector), and *v_x_n_lgth* (cross-product length between orientation vector *orientVect_v* and POSC's normal vector). The methods are related to the computation of the corresponding attributes, and the method *isSymmetrical()*, which describes if the face pair is symmetrical.
- *RemainingStandAloneFace* – this class defines the data to assess a stand-alone face in the second evaluation process. It consists of the attribute *scoreGSI_E*,

which defines the global symmetry index for edges. The methods are *computeScoreGSI_E* and *isPartiallySymmetrical()* (which provides information on whether the face is partially symmetrical). This class consists of two sub-classes, *StandAloneEdge* and *EdgePair*, defining the relevant data for evaluating edges.

- *StandAloneEdge* – this class defines the data to assess a stand-alone edge for reflectional self-symmetry in the second evaluation process. It consists of the attribute *PointPlaneDist()*, which defines the point-to-plane distance between the edge midpoint and the POSC, and the methods *computePointPlaneDist()* and *isSymmetrical()*.
- *EdgePair* – this class consists of data to assess an edge pair reflectional symmetry in the second evaluation process. The attributes are *lengthDifference* (which denotes the length difference between the 1st and the 2nd edge), *midpoint* (which is the edge pair midpoint), *pointPlaneDist* (which defines the point-to-plane distance between the *midpoint* and POSC), *vector_s1* (is the position vector of 1st edge), *vector_s2* (is the position vector of 2nd edge), *subtractVect_e* (is the subtracted vector between *vector_e1* and *vector_e2*), *e_x_n_lgth* and (is the cross-product length between subtracted position vectors *subtractVect_e* and the POSC's normal vector). The methods are related to the computation of the corresponding attributes, including additionally the method *isSymmetrical()*, describing if the edge pair is symmetrical.
- *RemainingFacePair* – this class defines the data to assess a remaining face pair in the second evaluation process. It includes the attributes *cosineSimilarity* (which defines the cosine similarity of the face pair), *faceCentre_E1* (which is the centre of the 1st face), *faceCentre_E2* (which is the centre of the 2nd face), *midpoint* (midpoint between the *faceCentre_E1* and *faceCentre_E2*), *pointPlaneDist* (point-to-plane distance between the *midpoint* and POSC's normal vector), *positionVector_t1* (position vector of 1st face), *positionVector_t2* (position vector of 2nd face), *subtractVect_t* (is the subtracted vector between *positionVector_t1* and *positionVector_t2*) , *t_x_n_lgth* (cross-product length between vector *subtractVect_t* and POSC's normal vector). The methods are related to the computation of the corresponding attributes, including the method that checks if the remaining face pair *isSymmetrical()*.
- *VisualiseAPOS* – this class defines the data for the visualisation of an APOS. The *createPlane()* method is intended to visualise the APOS in the 3D CAD model.

- *ResultsFile* – this class defines the data for generating the results file. It includes the attributes *nameResultFile* and *locationResultFile*, and the methods *createTxtFile()* (creates the results file associated to the 3D CAD model), *addAPOS()* (writes the detected APOS into the results file), and *addSymFaces()* (writes the detected symmetric faces for the APOS into the results file).

Like the reflectional symmetric class diagram, the axisymmetry class diagram is also comprised of several classes, each with its attributes and methods (Figure 30):

- *GenerateAOSC* – this class defines the data relevant for generating an AOSC from the corresponding *SingleFace* and PAOI. It includes the attributes *axisPoint*, which determines the location on the axis, and the *axisVector_a*, which determines the orientation vector of the axis. The corresponding methods are *getAxisPoint()*, *getAxisVector_a()*, and *addAxis()*, which adds the candidate to a list.
- *SingleFace* – is a sub-class of *GenerateAPOSC* to generate the candidate from a single face, consisting of the methods *isClosed()*, *isCylindricalSufrace()*, *isConicalSurface()*, *isToroidalSurface()*, *isRevolvedSurface()*, used to assess if the face is closed and of the specific surface type.
- *TrimDuplicateAOSC* – this class defines the trimming of a duplicate AOSC. It consists of the attribute *lgthCross_a1_a2*, which defines the cross-product length between two AOSC's axis vectors. The methods are *computeLineEquation()*, which calculates the line equation to query if two AOSCs are coincident, *computeLgthCross_a1_a2()*, which computes the cross-product length between two candidates to check if they are parallel, *isDuplicate()* provides the information if a candidate is duplicate, and *deleteAxis()* which removes the duplicate AOSC. All generated AOSCs are exploited in the trimming. Hence, the total number of generated AOSCs n_{AOSC} from the class *GenerateAxisCandidate* is used in the class *TrimDuplicateAxis* (n_{AOSC} to n_{AOSC}).
- *TrimUnsuitableAOSC* – this class defines the trimming of an unsuitable AOSC. Its attribute is the *pointLineDist*, which defines the point-to-line distance between the AOSC and COG to be compared with $\delta_{\max}$, while the methods are *computePointLineDist()*, *isUnsuitable()*, which checks if the AOSC is unsuitable, and *deleteAxis()*, which removes the unsuitable POSC from the list.
- *EvaluateAOSC* – this class defines the evaluation of an AOSC. The class consists of the attributes *scoreGSI* and *scoreSFI*, providing the GSI and SFI values for the

AOSC. Its methods are *computeScoreGSI()*, *computeScoreSFI()*, and *isAAOS()* to determine if the AOSC is an AAOS. *EvaluateAWOSC* consists of two sub-classes: *StandAloneFace* and *RemainingStandAloneFace*. The first is related to the evaluation of exact symmetrical faces, while the second is related to the second evaluation process for partially symmetrical faces.

- *StandAloneFace* – this class defines the data for evaluating a stand-alone face. The attributes are *pointLineDist*, representing the point-to-plane distance between the characteristic face point and the POSC, and *f_x_a_lgth*, which describes the cross-product length between the characteristic face vector and AOSC's orientation vector check for parallelism. The methods are *computePoinPlaneDist()*, *compute_f_x_a_lgth()*, and *isSymmetrical()* for determining whether the stand-alone face is symmetrical.
- *RemainingStandAloneFace* – this class defines the data to assess a stand-alone face in the second evaluation process. It includes the attributes *f_x_a_lgth*, which describes the cross-product length between the characteristic face vector and AOSC's orientation vector to check for parallelism, and *LineEquation*, which calculates the line equation to check if the face axis and AOSC are coincident. The corresponding methods are *compute_f_x_a_lgth()*, *computeLineEquation()*, *isCoincident()*, and *isPartiallySymmetrical()*, which provides information on whether the face is partially symmetrical.
- *VisualiseAAOS* – this class defines the data for the visualisation of an AAOS, and consists of the *createPlane()* method, intended to visualise the AAOS in the 3D CAD model.
- *ResultsFile* – this class defines the data for generating the results file. It includes the attributes *nameResultFile* and *locationResultFile*, and the methods *createTxtFile()* (creates the results file associated to the 3D CAD model), *addAAOS()* (writes the detected AAOS into the results file), and *addSymFaces()* (writes the detected symmetric faces for the AAOS into the results file).

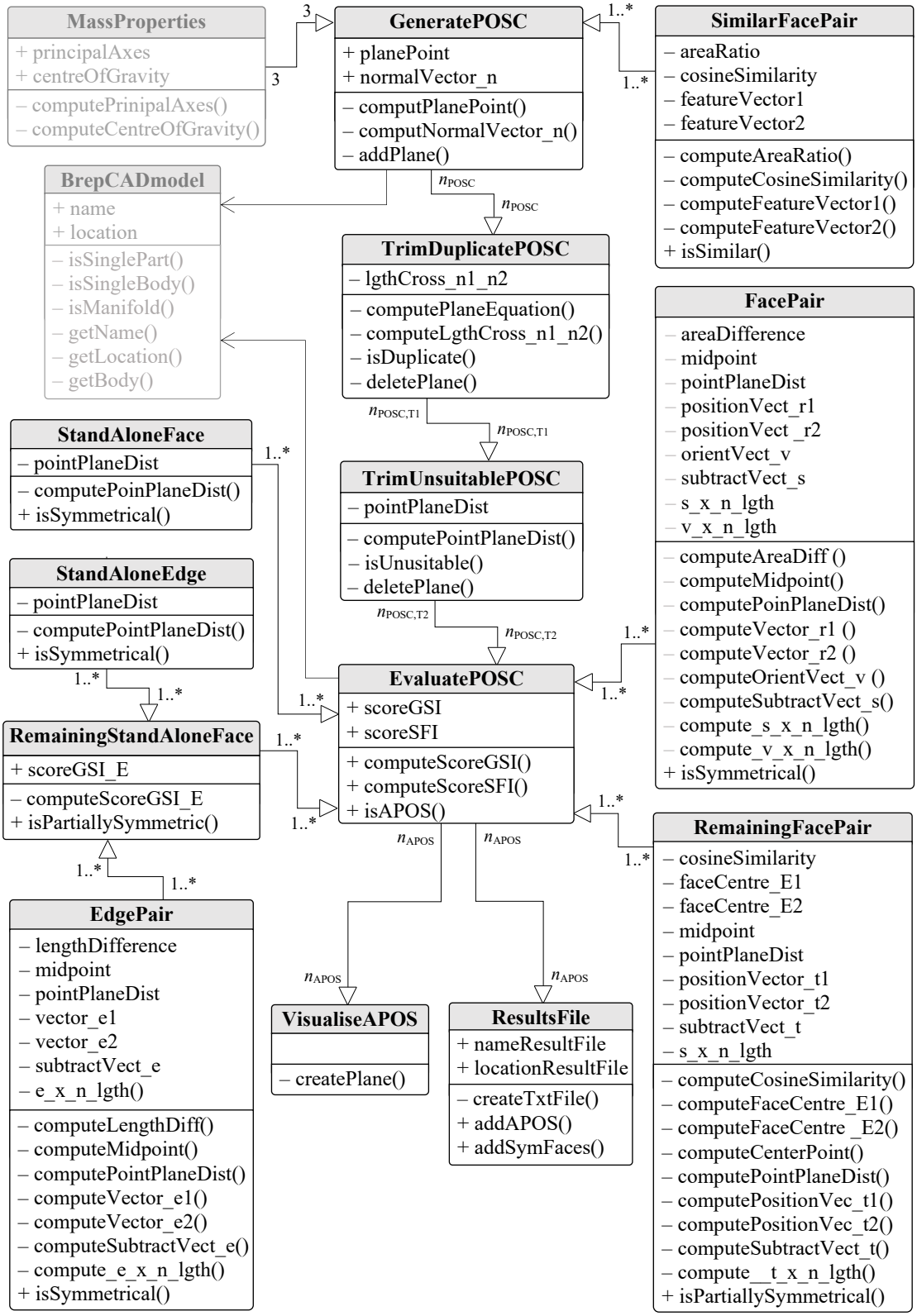


Figure 29. Class diagram for the steps generation, trimming, evaluation, and visualisation of candidates for reflectional symmetry

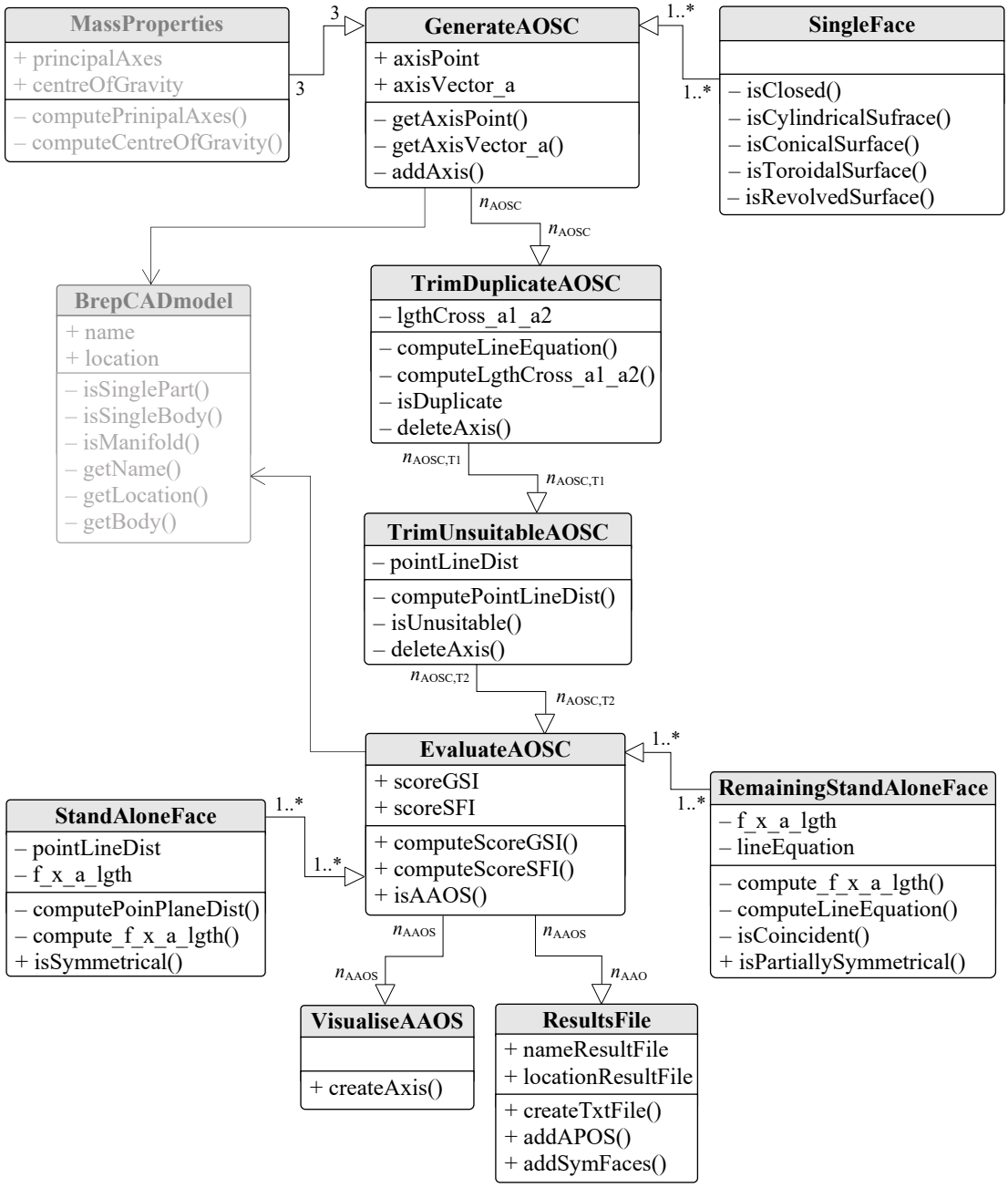


Figure 30. Class diagram for the steps generation, trimming, evaluation, and visualisation of candidates for axisymmetry

4 COMPUTATIONAL ENVIRONMENT

This chapter describes the implementation of the proposed symmetry detection method. As a result of the implementation, a computational environment has been developed by means of the commercial CAD system Solidworks and its Application Programming Interface. In addition, the feasibility of implementing the method into two other CAD systems has been investigated. The additional considered CAD systems were the commercial CAD system NX and the open-source CAD system FreeCAD.

The proposed CASD method described in the previous chapter has been implemented into the commercial CAD system Solidworks 2020 using its Application Programming Interface. The implementation within the CAD system aims to investigate if the method is suitable for practical application. Apart from the implementation in Solidworks, an investigation regarding the feasibility of implementing the method in alternative CAD systems such as NX and FreeCAD has been conducted. Implementing the CASD method in different CAD systems and geometric modelling kernels is useful to examine its generality. On one side, the commercial CAD system NX uses the same geometric modelling kernel with a similar B-rep data structure as in Solidworks. Conversely, the open-source CAD system FreeCAD uses a different modelling kernel and B-rep data structure. For the implementation of the proposed CASD method, the data model presented in Section 3.7 is exploited. As a result of the implementation, a computational environment has been proposed, as shown in (Figure 31).

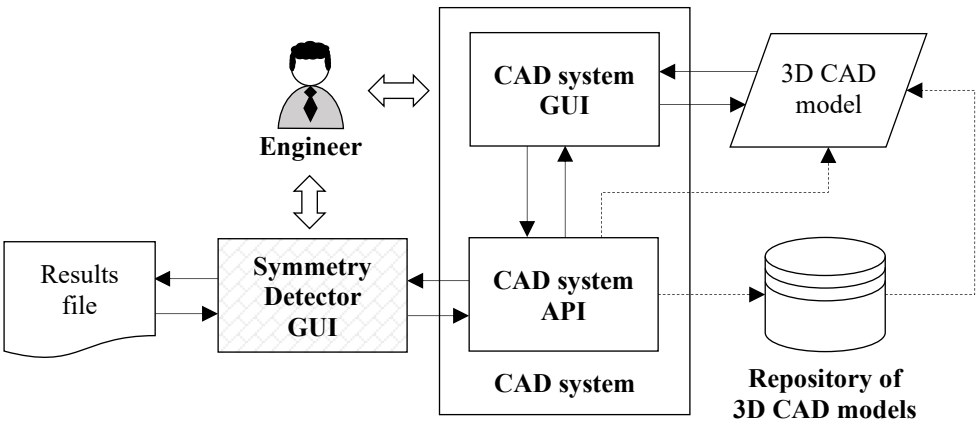


Figure 31. Schematic representation of the computational environment

The computational environment consists of the Symmetry Detector GUI, the CAD system GUI and API, an input 3D CAD model currently open or from a repository of 3D CAD models (optional). The core of the computational environment is the Symmetry Detector GUI, which has been incorporated into the CAD system. It contains a source code that can interact with the CAD system via its API. The Symmetry Detector enables recognising symmetry and provides postprocessing capabilities, i.e., storing and visualising the symmetry detection results within the 3D CAD system. The symmetry detection results are retrievable from a text results file, generated at the end of the symmetry detection, and associated with a 3D CAD model. The Symmetry Detector can process a single 3D CAD model currently open within the CAD system or loop through multiple 3D CAD models from a repository stored at some location (Figure 31).

While implementing the proposed CASD method, the approach was to leverage the capabilities of the CAD system API. However, it is important to note that the accessibility and functionality of the API object hierarchy differ depending on the CAD system. For instance, some CAD systems (like Solidworks) allow full access to the API object hierarchy, while others (like CATIA V5) offer limited access that may require special licensing. Additionally, the programming language used to write the source code may also impact the accessibility and functionality of the API object hierarchy in some CAD systems. Solidworks API was chosen primarily due to its ability to access the entire API object hierarchy without any limitations on licensing. Moreover, there are nearly no limitations to the API object hierarchy accessibility in the available programming languages (VBA, C#, or C++). Consequently, the Symmetry Detector GUI was created at the macro level employing the VBA programming language, as the development of macros is relatively quick and straightforward.

The Solidworks API provides all the necessary functionalities to retrieve the topological and geometrical information from the 3D CAD model's B-rep. Most properties and parameters can be accessed directly through the corresponding API objects and methods. For instance, retrieving the COG and axes of inertia of a 3D CAD model is a straightforward process. However, to retrieve specific properties of the CASD method, a workaround solution may be required. For example, the face centroid cannot be computed directly, but it can be obtained by creating a reference point on the face. Additionally, there may be multiple ways to retrieve a property value. For instance, the face area and

edge length may be obtained by selecting the topological entity and using the measurement tool API object or directly from the related face or edge API object. The following sections give insights into the implementation of the method steps.

4.1 Implementation of the CASD method into a CAD System

The CAD system Solidworks is the interpreter of the input 3D CAD model, while its API represents a "tool" for interacting with the 3D CAD model and computing mathematical expressions from the steps of the symmetry detection method. At this point, it is important to emphasise that any other CAD system and its API with the same functionalities could also have been used for that purpose. Hence, the feasibility of implementing the method into alternative CAD systems NX and FreeCAD is demonstrated in Section 4.2. Solidworks API relies on Microsoft's Component Object Model (COM) that uses interfaces (objects), interface inheritance, and factory methods to return interfaces on existing and new objects. Interfaces provide public properties and methods. The highest-level object in the Solidworks API structure is the *SldWorks* interface (Figure 33).

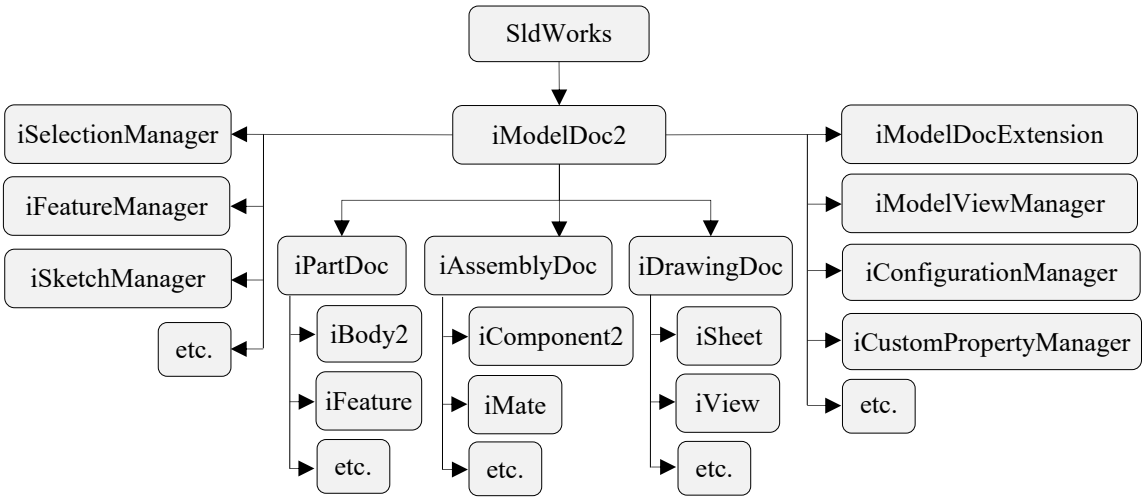


Figure 32. Solidworks API model object hierarchy

It enables direct and indirect access to all interfaces exposed in the Solidworks API object hierarchy [133]. Directly below the SldWorks interface is the *iModelDoc2* interface, which enables access to the three main Solidworks document types: parts, assemblies, and drawings. Each document type has its object (*iPartDoc*, *iAssemblyDoc*, and *iDrawingDoc*) with its related functions [133]. If an object can only be accessed indirectly, it must be referenced from another higher object in the hierarchy. Most of the

Solidworks API objects correspond to user-interface functionality However, certain objects provide functionality only accessible through the Solidworks API (for example, the Attribute object).

Design automation in CAD systems can take various forms, such as stand-alone applications, add-ins, or macros. There are two ways that applications can run: in-process and out-of-process. In-process means the application (e.g., macro) runs within the CAD system process and uses its computer memory. This is usually slower and can cause the CAD system to freeze if the application is demanding. On the other hand, an out-of-process application (e.g., stand-alone) runs in its process in the computer memory, making it much faster as it is independent of the CAD system. The Symmetry Detector GUI has been developed at the macro level. The is modelled as a User Form, while the steps of the proposed CASD method are split into four VBA Sub procedures (Figure 33): Sub AnalysisBrep, Sub GenerateTrimCandidates, Sub EvaluateCandidates, Sub VisualisePlanesAxes.

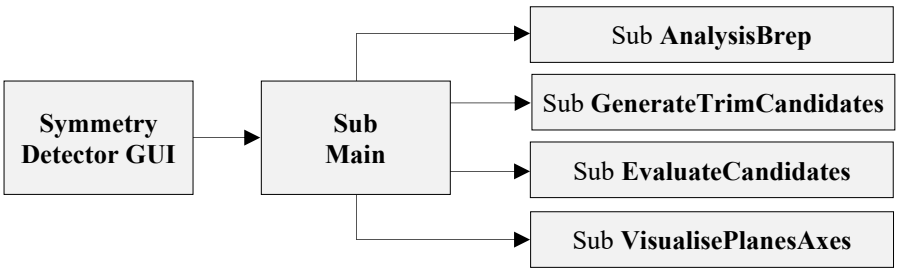


Figure 33. Structure of Sub procedures within the Symmetry Detector

4.1.1 Input 3D CAD models

As already stated, the proposed CASD method intends to be general, and its implementation into a CAD system (and later its validation in Chapter 5) shall confirm the generality. For that purpose, three types of input 3D CAD models were included in the implementation:

1) Native file format (Solidworks),
2) Kernel file format (Parasolid) and
3) Neutral file formats (STEP).

Limiting the CASD to native CAD models designed only in Solidworks would be a significant implementation constraint. Kernel formats facilitate one-to-one interpretation

between CAD systems sharing the same geometric modelling kernel. Neutral file formats are commonly used to exchange CAD models between different systems with varying geometric modelling kernels. Thus, they are a logical input. Until now, none of the previous CASD studies [16,23,42,63,66] had considered all three types of CAD model file formats as inputs for symmetry detection, which represents a novelty. The requirement for symmetry detection is that the 3D CAD model is interpreted free of topology or geometry errors. In Solidworks, any existing error in the 3D CAD model can be checked with *iModelDocExtension.GetWhatsWrong()*. Alternatively, if such analysis tools are unavailable through the CAD system API functionalities, the topology can be checked with the Euler equation (15) or visually by the engineer. After the interpretation of the input 3D CAD model, the next step is the analysis of the B-rep.

4.1.2 Analysis of B-rep

In the Sub procedure AnalysisBrep, the B-rep is analysed. This step of the CASD method is divided into several sub-steps: merging and classification of topology and calculation of face and edge properties. When implementing the method into a CAD system, the topological data structure of the B-rep needs to be considered first. Figure 34 provides a flowchart of the relationship between topological entities and their accessibility via the Solidworks API. The flowchart applies to manifold CAD models. The relationship between the topological entities is essential for classifying faces and edges and determining how they are connected to loops. Loops are utilised for the string code designation of edges when computing the Cosine similarity.

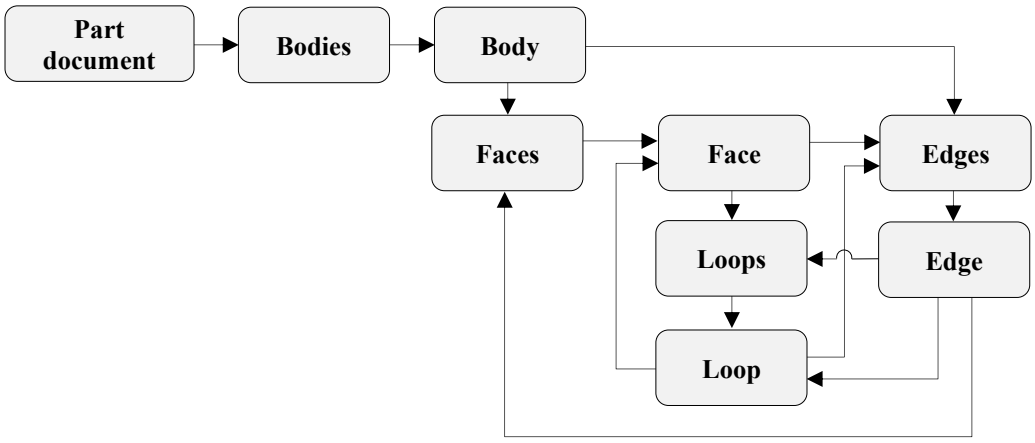


Figure 34. Relationship between topological entities

The topological entities are accessed through a top-to-down approach, with the part document on the roof and one or more bodies below (this research is limited to parts with a single body. The next level below is the body object from which the corresponding faces and edges can be retrieved. Underneath the face are the edges that bound the face and belong to loops. Each edge is associated with two adjacent faces and belongs to one or two loops, both of which can be obtained from the edge object. Each loop points to an associated face and two edges. When implementing the method, it is important to consider the types of surfaces and edges supported by the CAD system. These can be retrieved from the associated faces and edges. Table 15 provides the types of analytic and numeric surfaces and curves included in the implementation in Solidworks.

Table 15. Types of surfaces and edges and their labels.

Types of surfaces and curves in Solidworks			**Label abbreviation**
Faces	**Analytic**	Plane	PL
		Cylinder	CY
		Cone	CO
		Sphere	SP
		Torus	TO
	Numeric	Blend	BL
		B-spline surface	BS
		Extruded	EX
		Offset	OD
		Surface of revolution	SR
Edges	**Analytic**	Line	LI
		Circle	CI
		Ellipse	EL
	Numeric	B-spline curve	BC
		Constant parameter curve	CP
		Intersection	IN
		Trimmed	TR
		SP curve	SC

The naming of faces and edges is taken over from Solidworks. For instance, the cylinder represents a cylindrical surface, the cone a conical surface, the sphere a spherical surface, and so on. These surface types and curves are common in CAD systems that use the Parasolid geometric modelling kernel, including Solidworks, NX, and Solid Edge (Table 7). Solidworks can automatically merge periodic faces during the 3D CAD model import

process, eliminating the need for the sub-step topology merging. However, this is not the case in certain other CAD systems like CATIA V5 and FreeCAD. The B-rep analysis begins by labelling topological entities, including faces and edges, each labelled with two letters (as shown in Table 15) and a digit counting to the total number of faces for a given surface type. For example, planes are designated with PL1, PL2, PL3, PL4, etc., cylinders with CY1, CY2, CY3, CY4, etc., cones with CO1, CO2, CO3, CO4, etc., B-spline surface with BS1, BS2, BS3, BS4, etc., lines with LI1, LI2, LI3, LI4, etc., circles with CI1, CI2, CI3, CI4, etc., B-spline curves with BC1, BC2, BC3, BC4, and so on). The labelling of topological entities enables their selection for retrieving certain properties, for example, the face centroid or the edge midpoint, or their highlighting, colouring, etc. All faces and edges in the 3D CAD model are retrieved as arrays with the *Body2.GetFaces()* and *Body2.GetEdges()* methods. The labelling of topological entities implies that all faces and edges are looped and labelled using the *iModelDoc2.SetEntityName()* or *iPartDoc.SetEntityName()* method, which is valid for the labelling of faces, edges, and vertices. After labelling, all faces and edges are classified according to their associated surface and curve type using the *iSurface.Identity()* and *iCurve.Identity()* methods. Finally, the specific properties of faces and edges can be obtained using the commands outlined in Table 16. The outputs of the B-rep analysis are a set of arrays representing the classes of faces and edges. Additional classes not explicitly given in Table 15 are those generated from open periodical surfaces (open cylinder, open cone, open blend, open sphere, and open torus). Those surface classes are distinguished based on the periodicity angle (as described is Subsection 4.1.2).

The row structures of the arrays depend on the surface type. For planes, the array rows have the following form: (*Face label*; *Area*; x_{FC}; y_{FC}; z_{FC}; $n_{FC,\,x}$; $n_{FC,\,y}$; $n_{FC,\,z}$). All array rows of open faces have the following form: (*Face label*; *Area*; $x_{FC'}$; $y_{FC'}$; $z_{FC'}$; $n_{FC',\,x}$; $n_{FC',\,y}$; $n_{FC',\,z}$). All other closed faces have the following form of the array rows: (*Face label*; *Area*; $x_{FC'}$; $y_{FC'}$; z_{FC}; $a_{FC,\,x}$; $a_{FC,\,y}$; $a_{FC,\,z}$). The coordinates x_{FC}, y_{FC}, and z_{FC} represent the face centroid, while $x_{FC'}$, $y_{FC'}$, and $z_{FC'}$ the projection of the face centroid onto the face. The variables $n_{FC,\,x}$, $n_{FC,\,y}$, and $n_{FC,\,z}$ are the components of the face normal vector at the centroid point, while $n_{FC',\,x}$, $n_{FC',\,y}$, and $n_{FC',\,z}$ are the components of the face normal vector at the projection point of the centroid onto the face. The variables $a_{FC,\,x}$, $a_{FC,\,y}$, and $a_{FC,\,z}$ represents the components of the face's axis vector at the centroid point.

Table 16. Solidworks API objects and methods for obtaining specific properties of edges and faces.

Parameter or property	SOLIDWORKS
Get all faces in 3D CAD model	*iBody2.GetFaces()*
Face area	*iFace2.GetArea()*
Get the loops of a face	*iFace2.GetLoops()*
Get loop type	*iLoop2.IsOuter()*
Get faces in 3D CAD model	*iBody2.GetFaces()*
Get surface type of a face	*iSurface.Identity()*
Get edges in 3D CAD model	*iLoop2.GetEdges()*
Get curve type of an edge	*iCurve.Identity()*
Surface parameters	*iSurface* ↓ *.ConeParams2()* *.CylinderParams()* *.PlaneParams()* *.SphereParams()* *.TorusParams()* *. GetRevsurfParams()* *.GetExtrusionsurfParams()* *.GetOffsetSurfParams2()* *.Parameterization2()*
Face centroid (FC)	*iFeatureManager.InsertReferencePoint()*
FC projection onto the face	*iSurface.GetClosestPointOn()*
Face normal	*iSurface.EvaluateAtPoint()*
Edge length	*iCurve.GetLength2()*
Edge midpoint	*iFeatureManager.InsertReferencePoint()*

After analysing the B-rep, the next steps involve generating, trimming and evaluating POSCs and AOSCs. The part of the source code for conducting the B-rep analysis in the computational environment is provided in Appendix A1.

4.1.3 Generation and trimming of planes and axes of symmetry candidates

The two steps of the CASD method, generation and trimming of POSCs and AOSC, are covered in one Sub procedure, namely GenerateTrimCandidates. This has been done for practical reasons, as those two steps are naturally correlated. As already stated, the proposed method relies on the implicit symmetry detection approach, which means that a set of POSCs and AOSCs are generated from the PAOI, single faces, and pairs of similar faces (see Section 3.3). Usually, the PAOI may be retrieved as a standard mass property in most CAD systems. In Solidworks, this property can be gathered by the

iModelDocExt.GetMassProperties2() command (Table 17). The part of the source code for the retrieval of PAOI within the computational environment is given in Appendix A2.

Table 17. Solidworks API objects and methods for generating and trimming of POSCs and AOSCs

Parameter or property	SOLIDWORKS Command
Get 3D CAD model's COG and PAOI	*iModelDocExt.GetMassProperties2()*
Get 3D CAD model's minimum bounding box	*iPartDoc.GetPartBox()*
Get curve type of an edge	*iFeatureManager.InsertReferencePoint()*

The AOSCs are generated by looping all faces of the closed cylindrical surface type through the CAD model and retrieving their relevant parameters, the axis vector, and the point on the axis. For POSCs, a pairwise comparison is conducted between all faces (plane, cylindrical or B-spline surface type), and the cosine similarity measure is between each pair (as described in Subsection 3.3.2). The generated POSCs and AOSCs are stored in two temporary arrays and are used as input for the trimming step, whose details are presented in Section 3.4. Coincident duplicate candidates are removed by a pairwise comparison process of POSCs first, and AOSCs second. The distance of each remaining candidate is checked from the COG with the point-to-plane distance for POSCs and the point-to-line distance for AOSCs. The command *PartDoc.GetPartBox()* can retrieve the 3D CAD model's minimum bounding box for calculating the candidate's maximum allowed distance from the COG according to Equation (82). A part of the source code for the generation and trimming of candidates within the computational environment is provided in Appendix A3. The remaining candidates are subjected to evaluation relying on vector calculus (details are provided in Section 3.5.).

4.1.4 Evaluation of planes and axes of symmetry candidates

The evaluation of POSCs and AOSCs is performed in the Sub procedure EvaluateCandidates. Generally, the computation error in the implementation was set to $\varepsilon=10^{-6}$ m, which is in the range of manufacturing accuracy in mechanical engineering. After trimming, all remaining POSCs and AOSCs are used as input for the evaluation step, where each candidate is evaluated with respect to the classified faces of the B-rep using vector calculus and the specific face properties. The evaluation of candidates relies on vector calculus, i.e., subtracting or adding two vectors, dot and cross products of two

vectors, vector length, and normalisation of a vector. The Solidworks API provides corresponding objects and methods for vector calculus (see Table 18).

Table 18. Solidworks API objects and methods for evaluation of POSCs and AOSCs

Parameter or property	SOLIDWORKS Command
Create a vector	*iMathUtility.CreateVector()*
Subtract two vectors	*Set VectorOperation = Vector1.Subtract(Vector2)*
Add two vectors	*Set VectorOperation = Vector1.Add(Vector2)*
Get length of a vector	*iMathVector.GetLength()*
Get unit vector	*iMathVector.Normalise()*
Get dot product between two vectors	*iMathVector.Dot()*
Get cross product between two vectors	*iMathVector.Cross()*

Section 3.5 discusses the evaluation process of detecting symmetric face pairs and self-symmetric stand-alone faces. For each POSC, symmetric face pairs are detected by pairwise comparison of faces from the same class, while symmetric stand-alone faces are detected by looping all faces in the 3D CAD model. Similar, for each AOSC, self-symmetric stand-alone faces are evaluated by looping individually all faces in the 3D CAD model. This individual looping of faces can be done simultaneously with the pairwise comparison. The theoretical background of this step is provided in Section 3.5. A part of the source code for the evaluation of candidates within the computational environment is provided in Appendix A4. If the evaluation shows that the corresponding POSC or AOSC also represents the APOS or AAOS, it is visualised within the 3D CAD model.

4.1.5 Visualisation of actual planes and axes of symmetry

The visualisation of APOSs and AAOSs is conducted in the Sub procedure VisualisePlanesAxes. Once the symmetry detection has been finished, the APOSs and AAOS are visualised within the 3D CAD model to provide the symmetry information to the user. The visualisation of the APOS and AAOS implies that they are created within the 3D CAD model as a reference plane or reference axis. In most CAD systems, a reference plane or an axis cannot be created directly but indirectly through other reference geometry. For instance, a reference point indicating the position and a reference line indicating the orientation are required to create a plane. The same applies to a reference axis. The strategy for creating the plane or axis in Solidworks is to create a point and a

line in a 3D sketch, where the point represents the position while the line represents the orientation of the APOS or AAOS. The API objects and methods for that are provided in Table 19. The created plane or axis is uniquely named “POS” or “AOS” including a digit that starts from 1 and counts to the total number of detected APOS or AAOS.

Table 19. Solidworks API objects and methods for creating a plane or axis.

Parameter or property	SOLIDWORKS
Create a 3D sketch	*iSketchManager.Insert3DSketch()*
Create a point within the 3D sketch	*iSketchManager.CreatePoint()*
Create a line within the 3D sketch	*iSketchManager.CreateLine()*
Create a plane within the 3D CAD model	*iFeatureManager.InsertRefPlane()*
Create an axis within the 3D CAD model	*iFeatureManager.InsertRefAxis()*

The visualisation of APOSs and AAOS represents only one part of the symmetry information. Another important aspect is the visualisation of symmetric and asymmetric faces in the 3D CAD model associated with the corresponding APOS or AAOS. This is achieved through a Symmetry Detector GUI. In addition, a results file has been proposed to permanently store the symmetry detection results for each CAD model, which can be loaded into the GUI anytime for visualisation purposes.

4.1.6 Graphical user interface and results file

The developed Symmetry Detector GUI, which is illustrated in Figure 35, provides a set of commands that enables the engineer to conduct the symmetry detection process and postprocessing of the symmetry detection results. Symmetry Detector can process the currently open 3D CAD model within the CAD system or repository of 3D CAD models from some location. Postprocessing includes the retrieval and visualisation of the symmetry detection results.

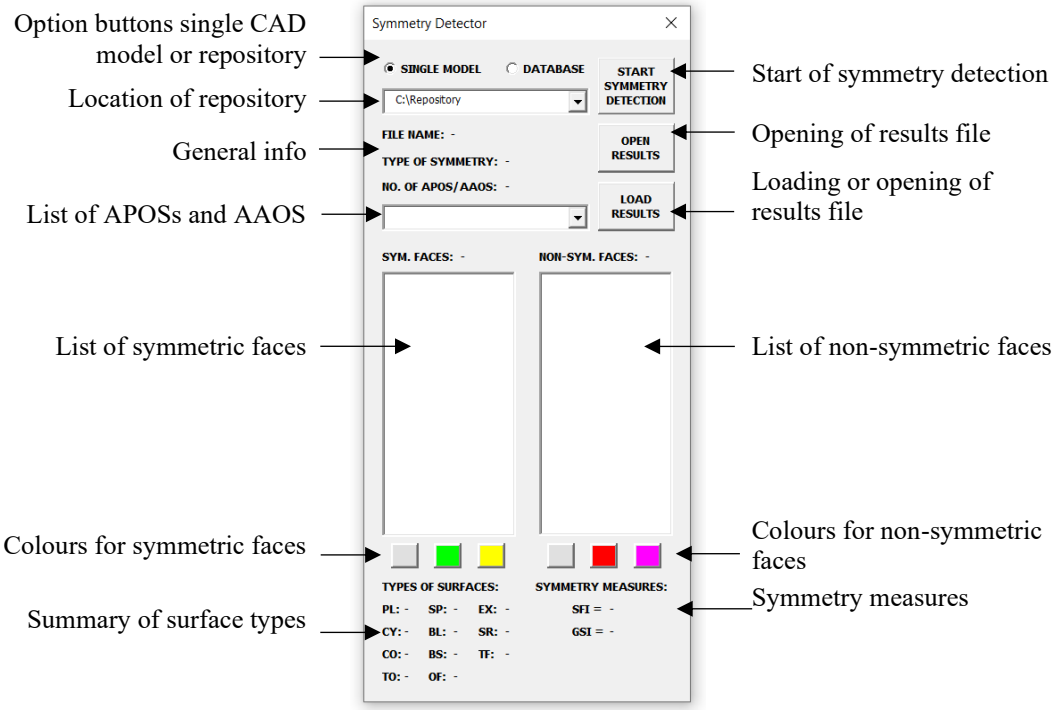


Figure 35. The Symmetry Detector GUI

Symmetry Detector is a tool that offers helpful information about the symmetry of 3D CAD models. It can detect the type of symmetry and provide details about the number of detected APOS and AAOS. The tool has three command buttons that allow users to start the symmetry detection process, load files, and open results. The list of symmetric faces contains face pairs (e.g., PL1-PL2) and stand-alone faces (e.g., CY2). Different colouring of faces can also provide better visualisation of the results. Additionally, Symmetry Detector provides information about the number of faces in a 3D CAD model, as well as corresponding symmetry measure scores (SFI and GSI) for the selected APOS or AAOS. Each time a 3D CAD model is subjected to symmetry detection, results are saved in a *.txt file with the same name and location as the 3D CAD model. The file includes general information, a list of faces and their properties, COG and axes of inertia, an overview and count of surfaces, a list of planes and axes of symmetry and their symmetric faces, as well as the total count of detected symmetry planes and axes (Figure 36).

Figure 36. An example of the results file

4.2 Implementation of the CASD method into alternative CAD systems

The previous section describes the implementation of the method into the CAD system Solidworks using its API functionalities. However, it is important to note that the method could also be implemented in other CAD systems through their APIs. To demonstrate this, the implementation of the method in alternative CAD systems such as NX and FreeCAD was investigated. As previously mentioned, NX uses the Parasolid modelling kernel and offers almost the same geometrical entities as Solidworks, with the addition of two types of curves parabola and hyperbola (Table 7). FreeCAD, on the other hand, is an open-source CAD system that employs the Open CASCADE geometric modelling kernel [143]. The B-rep data structure in Open CASCADE is based on the STEP file data structure (Table 8). Considering the differences between the B-rep data structure in Solidworks/NX and FreeCAD, some adjustments may be necessary in terms of face and edge classes to implement the CASD method. Table 20 compares the existing Solidworks API commands with those available in NX and FreeCAD that provide the same functionalities. None of the listed commands require any special licensing. The commands in Table 20 are mainly related to the B-rep analysis step. Other steps of the CASD method were not considered because they could be replaced by custom-written functions or libraries that provide the required functionalities. Alternatively, other steps can be performed in non-CAD software, such as MATLAB, using the data obtained from the B-rep analysis step.

Table 20. Comparison of API commands in Solidworks, NX, and FreeCAD

Description	Solidworks	NX	FreeCAD
Get all faces in the 3D CAD model	*iBody2* ↓ *.GetFaces()*	** UF.Modeling* ↓ *.AskBodyFaces()* **Body* ↓ *.GetFaces()*	*Part.Shape* ↓ *.Faces()*
Face's surface type	*iSurface* ↓ *.Identity()*	** UF.Modeling* ↓ *.AskFaceType()* **Face* ↓ *.FaceType()*	*Part.Shape.Face* ↓ *.Surface()*
Get edges in 3D CAD model	*iFace2 or iBody or iLoop2* ↓ *.GetEdges()*	** UF.Modeling* ↓ *.AskFaceEdges()* **Body* ↓ *.GetEdges()*	*Shape* ↓ *.Edges()* *Shape.Face* ↓ *.Edges()*
Edge's curve type	*iCurve* ↓ *.Identity()*	** UF.Modeling* ↓ *.AskEdgeType*	*Part.Shape.Edge* ↓ *.Curve()*
Labelling of topology	*iPartDoc.* ↓ *SetEntityName()*	**Face or *Edge* ↓ *.SetName() or .Tag()*	N/A
Face area	*iFace2* ↓ *.GetArea()*	**MeasureFaces* ↓ *.Area()*	*Part.Shape.Face* ↓ *.Area()*
Surface parameters	*iSurface* ↓ *.ConeParams2()* *.CylinderParams()* *.PlaneParams()* *.SphereParams()* *.TorusParams()* *. GetRevsurfParams()* *.GetExtrusionsurfParams()* *.GetOffsetSurfParams2()* *.Parameterization2()*	**UF.Modeling* ↓ *.AskFaceData()* *.AskBsurf()* *.AskFacePeriodicity*	*Part.Shape.Face* ↓ *.ParameterAt()* *.ParameterRange()* *.Surface.Axis()*
Face centroid	*iFeatureManager* ↓ *.InsertReferencePoint()*	**GenericMeasure* ↓ *.MassProperties()*	*Part.Shape.Face* ↓ *.CenterOfMass()*
Face centroid projection onto the face	*iSurface* ↓ *.GetClosestPointOn()*	**UF.Modeling* ↓ *.AskFaceParm()*	*Part.Shape.Face* ↓ *.MakeParallelProjection()*
Face normal	*iSurface* ↓ *.EvaluateAtPoint()*	**UF.Modeling* ↓ *.AskFaceProps()*	*Part.Shape.Face* ↓ *.NormalAt()*
Edge length	*iCurve* ↓ *.GetLength2()*	**Edge* ↓ *.GetEdges()*	*Part.Shape.Edge* ↓ *.Length()*
Edge midpoint	*iFeatureManager* ↓ *.InsertReferencePoint()*	**GenericMeasure* ↓ *.MassProperties()*	*Part.Shape.Edge* ↓ *.CenterOfMass()*
Get loops	*iFace2* ↓ *.GetLoops()*	***Face* ↓ *.Loops()*	*Shape.Face* ↓ *.Wires()*
Loop type	*iLoop2* ↓ *.IsOuter()*	***Loop* ↓ *.OuterLoop()*	*Part.Shape.Face* ↓ *.OuterWire()*
Mass properties (COG or COM and PAOI)	*iModelDocExt* ↓ *.GetMassProperties2()*	**UF.Modeling* ↓ *.AskMassProps3d()*	*Part.Shape* ↓ *.CenterOfMass()* *.PrincipalProperties()*
Minimum bounding box	*iPartDoc* ↓ *.GetPartBox()*	**ModlGeneral* ↓ *.AskBoundingBox()*	*Part.Shape* ↓ *.BoundBox()*

* – NXOpen. ** – SNAP.NX.

It is interesting to note that both alternative CAD systems do not require workaround solutions like in Solidworks API to compute the face centroid or edge midpoint, which can be computed directly through existing API commands. FreeCAD lacks commands for the custom labelling of topological entities. FreeCAD does not have specific commands for the custom labelling of topological entities. However, this is not a significant drawback, as a default labelling of topological entities is provided and can be used for implementation purposes. By default, faces are designated with “Face” plus an enumeration that starts from 1 to the total number of faces in the CAD model (e.g., Face1, Face2, Face3, etc.). The same applies to edges and vertices, except that they are instead of “Face” designated with “Edge” or “Vertex”. In addition, when implementing the method in FreeCAD, it is noticeable that wires are equivalent to loops in NX or Solidworks. Based on the investigation, it is feasible to implement the proposed CASD method into the considered alternative CAD systems using their APIs. However, additional programming effort and possibly changes in the strategy for retrieving method-relevant data are required. The CASD method and computational environment are validated in the following chapter.

5 VALIDATION & DISCUSSION

This chapter focuses on the validation of the proposed method for detecting symmetry, as well as its implementation in a computational environment. The Validation Square, which consists of Structural and Performance validation is employed for that purpose. Additionally, a discussion is conducted alongside the validation process. Finally, the chapter concludes with a discussion of the research implications for both research and practical application.

The proposed CASD method and its implementation into a CAD system are validated using the Validation Square [33]. The Validation square was initially introduced to validate engineering design methods but can also be used for validating research in general because it represents a systematic and comprehensive validation approach. The Validation Square builds confidence in the method's usefulness by evaluating its effectiveness and efficiency through qualitative and quantitative measures. Hence, the Validation Square is compound of two parts [33]:

1) Structural validation, which is a qualitative assessment that aims to evaluate the method’s effectiveness, i.e., whether the symmetry detection is correctly, and
2) Performance validation, which is a quantitative assessment that aims to evaluate method’s efficiency, i.e., whether the symmetry detection is correct.

The process of structural validation involves acknowledging the validity of each individual step that comprises the method, confirming the consistency of the method by examining how these steps are integrated, and verifying the appropriateness of the example problems used in the performance validation. Conversely, the performance validation requires determining the effectiveness of the method in achieving its initial purpose for specific example problems, acknowledging that effectiveness is linked to the application of the method, and assessing the method's overall usefulness beyond the example problems. In this context, structural validation pertains to the validation of the CASD method, while performance validation pertains to the validation of the computational environment in which the CASD method is implemented. For the

performance validation, two datasets were gathered, which are discussed in the following section.

5.1 Data Collection

The CAD models included in both datasets are exclusively solid B-rep CAD models with manifold geometry and single parts with one body, as outlined in Section 1.2. The objective of the first dataset is to validate the sensitivity of the computational environment against various input CAD model formats including native Solidworks, Parasolid, and STEP, created in different CAD systems such as Solidworks, CATIA V5, and FreeCAD. and their interpretation of B-rep within the implemented CAD system. These models are then interpreted as B-rep within the implemented CAD system. The second dataset is intended to validate the accuracy and time complexity of symmetry detection. The CAD models from both datasets were carefully selected to represent relevant examples from practice.

The first dataset comprises 20 representative 3D CAD models each created in three different CAD systems: Solidworks, CATIA V5, and FreeCAD. These CAD models were exported in STEP format and utilised for performance validation. All models within this dataset were either exact reflectional or axisymmetric, without any non-symmetric parts. Each 3D CAD model from this first dataset was modelled in three different CAD system but in the same manner, using features in the same sequence. To confirm that the models were truly exact global reflectional symmetric, the final shape was mirrored with respect to the plane of symmetry. Similarly, the axisymmetric CAD models were obtained by revolving a profile around an axis to ensure that the parts were exact global symmetric. In addition to the native CAD models, 20 Parasolid formats were also included in this dataset, exported from the native Solidworks files. Therefore, the total number of CAD models within this initial dataset is 100 (20 native Solidworks CAD models, 20 Parasolid CAD models, 20 STEP Solidworks CAD models, 20 STEP CATIA V5 CAD models, and 20 STEP FreeCAD CAD models). The 3D CAD models were carefully selected to ensure a balanced representation of both analytic and numeric geometry. Specifically, the first dataset comprises 73% analytic surfaces and 27% numeric surfaces, as depicted in Figure 37. A sample of the first dataset is shown in Figure 38.

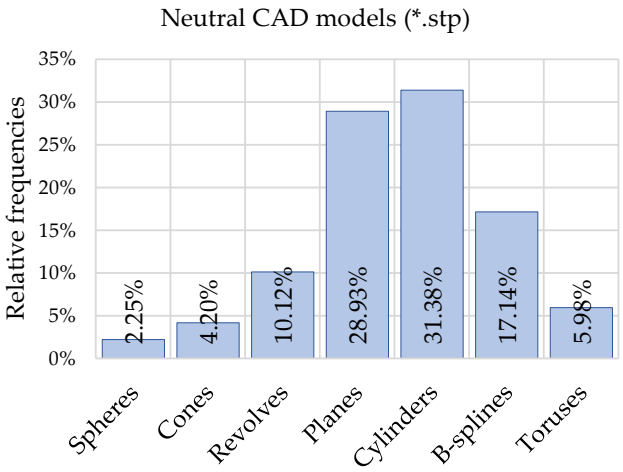
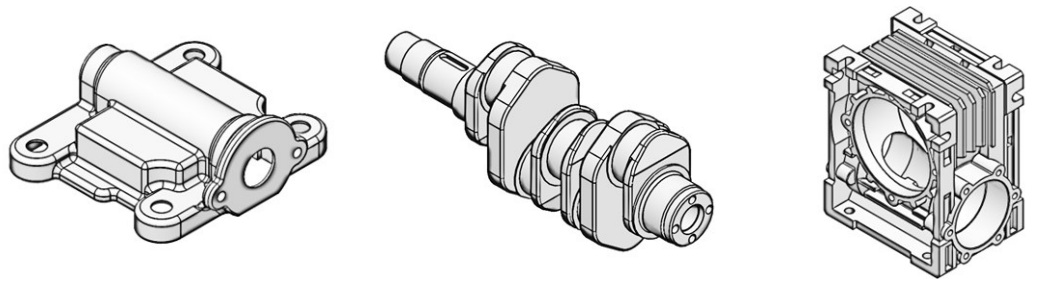
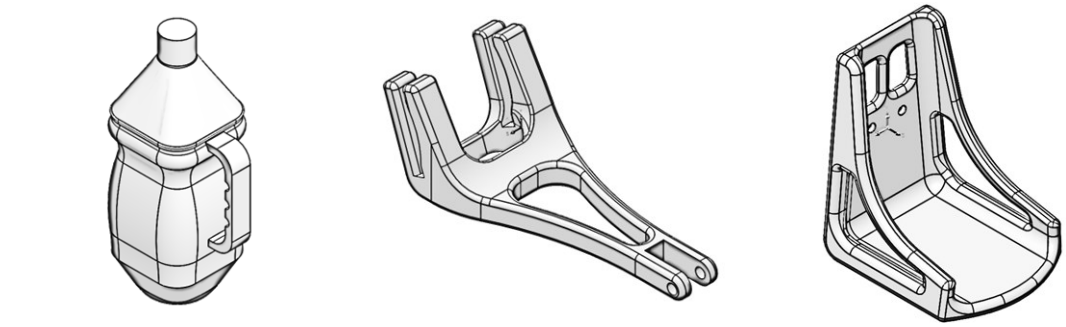


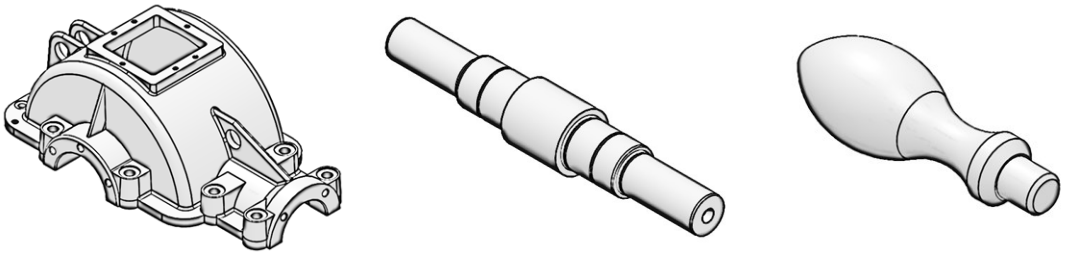
Figure 37. A bar chart of relative frequencies of surfaces within the first dataset



a) Cover b) Crankshaft c) Housing



d) Bottle e) Fork f) Chair



g) Gear housing h) Shaft i) Handle

Figure 38. A sample of the first dataset

The second dataset comprises 1000 CAD models in STEP format and is significantly larger than the first. Figure 40 shows a sample of the collected data. Its purpose is to validate the time complexity and accuracy of the implemented CASD method. The CAD models of this second dataset were collected from GrabCAD and PARTCommunity [144,145], and represent various mechanical parts manufactured through casting, moulding, forming, and machining. The selected 3D CAD models ensure a fair representation of analytic and numeric geometry. Analytic surfaces, such as planes, cylinders, cones, toruses, and spheres, account for roughly 75% of all faces, while numeric surfaces, such as B-splines and surfaces of revolution, account for the remaining 25% (Figure 39). The dataset also includes CAD models in arbitrary orientations and positions in the 3D model space to test the sensitivity of the CASD method. The models in this second dataset are categorised as:

1) exact global reflectional symmetric parts (Figure 40, a – l),
2) exact global axisymmetric parts (Figure 40, m – p),
3) partial reflectional symmetric and axisymmetric parts (Figure 40, q – t), and
4) non-symmetric (Figure 40, u – x).

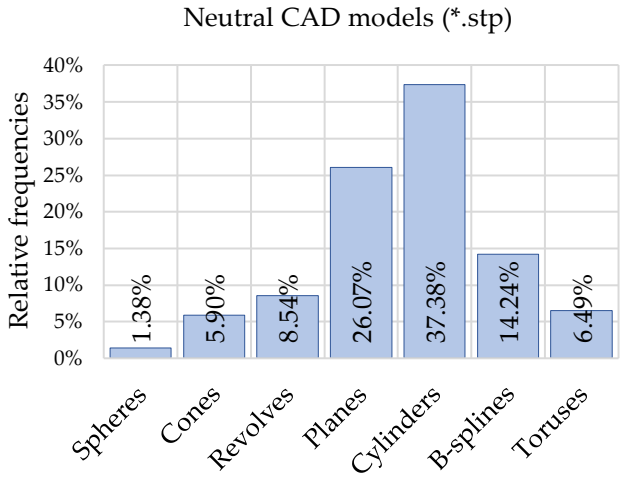


Figure 39. A bar chart of relative frequencies of surfaces within the second dataset.

In Figure 40, there are global reflectional symmetric parts that can have one (a – d), two (e – h), or multiple APOS (i – l). The models vary in complexity, ranging from very simple with only a few faces to highly intricate with several hundred faces.

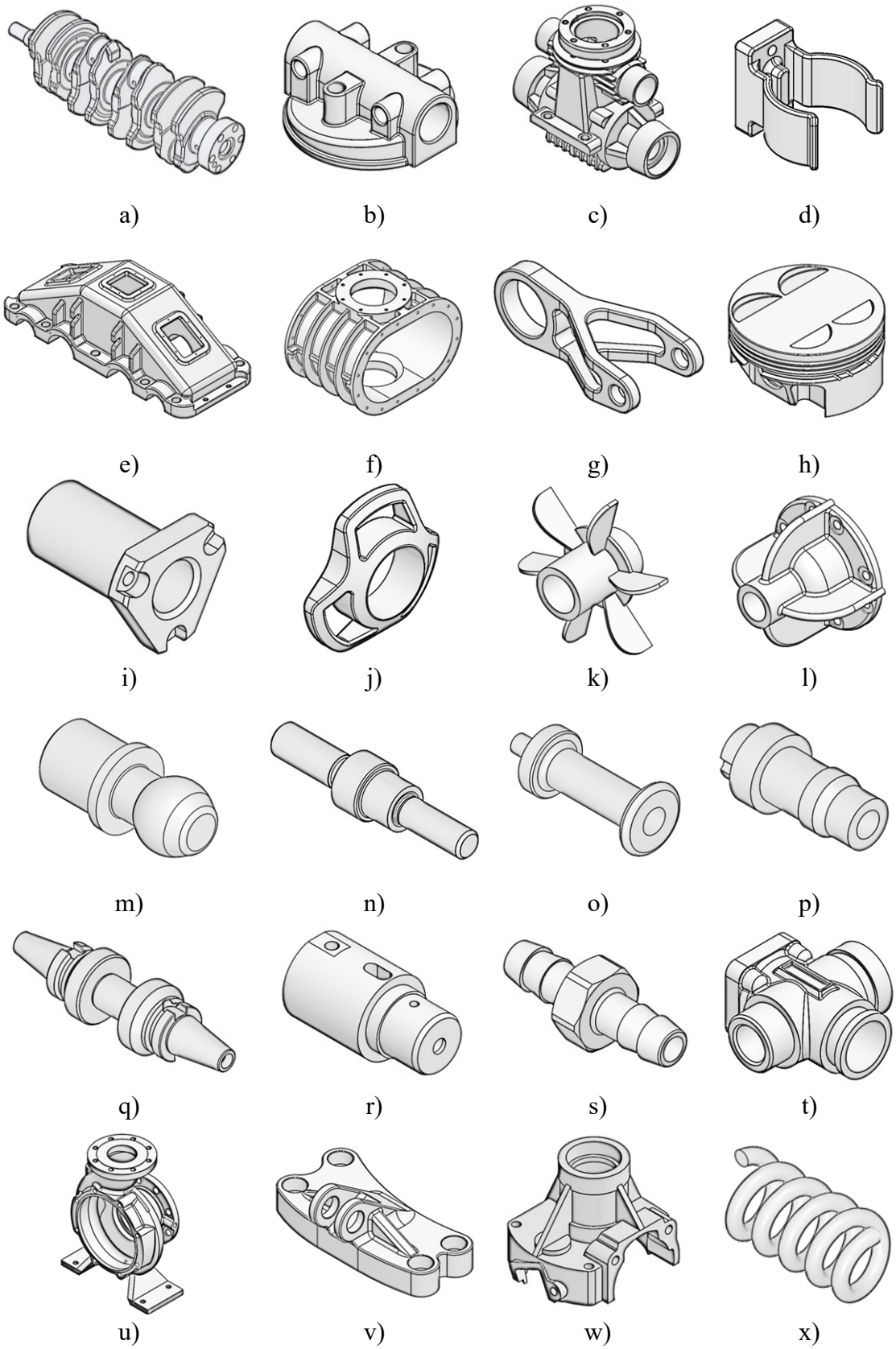


Figure 40. As sample of the second dataset

The total sizes of the datasets in this research are considerably larger compared with prior CASD studies. For example, the studies referenced in [16,63] utilised 45 CAD models in STEP format, while others employed 10 CAD models in ACIS format and 32 CAD models in native CATIA V5 format [23,42,25]. In conclusion, both datasets collected in this study are used in performance validation. However, before that, it is important to examine the appropriateness of the gathered data in the structural validation. The following section focuses on structural validation.

5.2 Structural Validation

As outlined at the beginning of this chapter, the method's structural validation phase comprises three key phases. In the first phase, the method's individual steps, as described in Chapter 3, are validated with the help of literature to build confidence in their validity. Moving on to the second phase, the focus shifts to building confidence in the method's internal consistency by examining the information flow of its steps using flowchart representations. Lastly, the third phase builds confidence in the example problems selected for the performance validation. The following subsections delved deeper into each of these three structural validation phases.

5.2.1 Validity of individual steps of the CASD method

The first step of the symmetry detection method is interpreting the input 3D CAD model. Prior studies [16,14,23,42,63,66] utilised a CAD system as an interpreter to read the B-rep model and retrieve its topological and geometrical information for the subsequent steps. The typical interpreters used in previous studies were commercial CAD systems, such as CATIA V5 and NX, or open-source CAD systems, such as Open CASCADE. The next step, which is the analysis of B-Rep, comprises three sub-steps: merging of topology, classification of topology, and computation of properties of faces and edges. Merging of topology involves combining split faces and edges to obtain the B-rep in terms of maximal faces and edges [20]. Usually, the split faces are adjacent and share the same edge(s), underlying surface type, and properties. Split faces are often adjacent and share the same edge(s), underlying surface type, and properties, and are typically an issue related to periodical surfaces that are partitioned into two halves during the CAD modelling process. Combining split faces is essential to prevent local asymmetry in the

CAD model. For example, if two faces of the cylindrical surface type are reflectional symmetric and both are partitioned into two halves, the symmetry in the CAD model is disturbed unless the partitioned face halves are merged. Previous CASD studies [16,14,63,66] have also emphasised the importance of this sub-step.

During the second sub-step, topology classification, groups topological entities according to their associated geometric entities. topological entities are grouped based on their associated geometric entities. There are two primary reasons for conducting this classification. The first reason is that symmetric properties are shared among faces of the same class, which can then be used to generate POSCs and AOSCs, as established by various studies [16,14,23,42,63,66]. The second reason is that only faces from the same class can be reflectional symmetric. In the final sub-step, face and edge properties are calculated to assign specific geometrical parameters and properties to topological entities, which are then used for symmetry detection. Previous CASD studies have extracted surface and curve parameters for faces and edges [16,14,63,66] and computed loop properties [23,42].

The following two steps of the method, generation and trimming of candidates, are also common in CASD studies [16,14,23,42,63,66]. Candidates are generated by pairing [23,42] and from single topological entities [16,23,42,63] but not from the PAOI, which appears to be novel in the context of symmetry detection in B-rep CAD models. Moreover, the presented method introduces a novel criterion for matching pairs of topological entities. The Cosine similarity measure is proposed instead of comparing properties such as the area and number of edges [23,42]. Unlike previous studies [16,23,42,63], candidate trimming in this research encompasses the removal of not only duplicates but also unsuitable candidates that are significantly distanced from the 3D CAD model's COG.

In implicit CASD approaches [40,55], evaluating candidates is a crucial step that applies not only in the B-rep context. However, there is a key difference between the evaluation methods proposed in this study and those in other B-rep CASD studies [16,23,42,63]. While other studies mainly evaluate candidates to each other, this study evaluates each candidate individually, considering all faces in the 3D CAD model. The evaluation procedure employs vector calculus to create and exploit face position vectors and normal

vectors, which help detect symmetric face pairs, while the face centroid or its projection onto the face helps detect symmetric stand-alone faces. Additionally, our evaluation criteria for symmetry within the B-rep CAD model rely on two proposed measures - the SFI and GSI. The SFI measures the topology's symmetry, while the GSI measures the geometry's symmetry within the B-rep CAD model. The SFI is a function of symmetric face pairs and self-symmetric stand-alone faces, while the GSI is a function of the surface area of symmetric face pairs and self-symmetric stand-alone faces. SFI and GSI aim to provide a measure of exact global, partial, and non-symmetry, which represents a novel approach compared to related past CASD studies. These studies evaluated candidates by ranking them according to loop area and the number of edges [23,42] and propagated candidates over the entire B-rep until all surfaces were covered without the occurrence of asymmetry.

In CASD, it is common practice to visualise the detected planes and axes of symmetry within a 3D object. This step provides engineers with direct information about the detected symmetry and has been utilised in several studies [16,14,23,42,63,66]. Additionally, the method entails visualising symmetric and non-symmetric faces within the 3D CAD model in relation to the corresponding planes and axes of symmetry. Furthermore, the symmetry detection results are permanently stored in an external results file. This represents a novel approach to the practical application of symmetry detection in B-rep CAD models, which is noteworthy.

To summarise, this study proposes a novel method for detecting symmetry using a geometric-based CASD approach. Unlike previous studies that viewed the B-rep as an infinite point set model, this method represents each face by a unique point. The face centroid, or its projection onto the face, ensures that identical-shaped faces are always represented by equivalent points. By transforming the B-rep CAD model into a set of points, the number of points equals the total number of faces in the CAD model. Apart from the position information, each point has additional characteristics, the normal or axis vector, that provide surface orientation information. This unique point approach reduces the size of the analysis model without losing symmetry recognition capabilities. It also changes the subsequent steps compared to prior CASD studies, as each POSC and AOSC is evaluated individually based on the unique points of the CAD model rather than through mutual comparison [16,42].

5.2.2 Consistency of the CASD method

During the second phase of structural validation, emphasis is placed on ensuring the consistency of the method by utilising an information flow-chart representation. This approach instils confidence in the way the method steps are put together. In the context of the proposed symmetry detection method, a comprehensive information flow chart has already been presented for each step within Chapter 3, detailing the inputs and outputs of each step. However, further explanation of each step's input and output is provided in the following paragraphs.

The first input in the information flowchart that needs to be considered is the 3D CAD model employed for symmetry detection. The presented CASD method essentially puts no restrictions on the input 3D CAD model, if it is a B-rep model. This is because the method relies on the unique points approach that could be adjusted to any input 3D CAD model format with B-rep. For implementation purposed the input 3D CAD models were limited to native Solidworks, kernel Parasolid, and neutral STEP formats. The datasets used for validation are described in Section 5.1. The inclusion of other CAD formats, such as native from other CAD systems, may be limited to the interpretation possibilities of the CAD system where the computational environment has been developed. However, including three different types of CAD model formats in this study represents an enhancement compared to past studies [16,23,66]. Also, the introduced symmetry detection method can process analytic and numeric surfaces. At the same time, the past CASD studies were mainly restricted to analytic surfaces due to the difficulties in detecting the symmetry properties of numeric surfaces [63]. The input 3D CAD model for CASD is used in the interpretation step, which is conducted utilising a CAD system.

The way in which the input CAD model is interpreted may vary depending on the file format. When using a native CAD model, interpretation issues are usually not a concern. A CAD model in kernel format can also be interpreted without issue if the underlying modelling kernel is the same as that of the CAD system. However, interpreting neutral files can be more challenging as it may result in invalid topology or geometry, as well as an approximation of geometry (possible scenarios have been discussed in Section 2.4). Despite this, it has been demonstrated that CAD systems are able to interpret neutral STEP files with a high degree of accuracy, with an accuracy of at least below 10^{-6} m for

both analytic and numeric surfaces [127]. A detailed discussion of the interpretation process can be found in Section 5.3.

The interpreted CAD model is used as input for the B-rep analysis step to generate classified topology as output. The output of the B-rep analysis step is the classified topology, including their specific properties (Figure 13). The classified topology is used as input in two steps. The first is the candidate generation step to generate POSCs and AOSCs (Figure 18), and the second is the candidate evaluation step (Figure 18). The candidate generation step produces a set of POSCs and AOSCs as output, which is employed as input in the candidate trimming step (Figure 24). The initial set of generated POSCs and AOSCs is reduced in the trimming step. Hence, the output of this step is the remaining POSCs and AOSCs. These remaining POSCs and AOSCs, along with the classified geometry, are utilised as input to evaluated for symmetry. If the evaluation step confirms the existence of symmetry through the proposed symmetry measures, the output of this step is the detected APOS or AAOS.

The interpreted CAD model is used as input for the B-rep analysis step to generate classified topology as output (Figure 13). This classified topology is then used as input in the candidate generation (Figure 18) and candidate evaluation steps (Figure 25 and Figure 26). The candidate generation step produces a set of POSCs and AOSCs, which is further reduced in the candidate trimming step to obtain a final set of POSCs and AOSCs (Figure 24). These remaining POSCs and AOSCs, along with the classified geometry, are subjected to evaluation for reflection symmetry or axisymmetry (Figure 25 and Figure 26). If the evaluation confirms the presence of symmetry, the final output of the CASD method is either an APOS or AAOS visualised in the 3D CAD model.

5.2.3 Appropriateness of the example problems

The final phase of structural validation is accepting chosen example problems for performance evaluation by building confidence in their appropriateness. As proposed in the Validation Square [33], these examples are case studies that support a claim of generality. However, in this study, two datasets consisting of representative 3D CAD models were utilised for performance validation instead of case studies. The appropriateness of these datasets can be observed from several aspects, including their

size, origin, CAD format types, symmetry types, surface types, and types of mechanical parts.

The first aspect of the appropriateness of example problems is the size of the dataset. A considerable dataset size utilised for validation suggests a broad range of CAD models comprising diverse combinations of topological entities and geometric shapes. In sum, there are 1100 CAD models undergoing symmetry detection, which instils trust that the CASD method is exposed to various example problem scenarios that may not arise in a considerably smaller dataset.

Moving on to the next aspect, the origin of the datasets, it is worth noting that the CAD models in the first smaller set were created using three distinct CAD systems, each utilising a different geometric modelling kernel. On the other hand, the second larger dataset was meticulously curated from online libraries of CAD models [144,145], featuring various relevant examples from real-world practice, modelled in diverse CAD systems by a multitude of users with their own unique CAD modelling techniques. This builds confidence that the sample problems are abundant in diversity.

The following aspect to consider is the variety of CAD formats present in the datasets. The first dataset includes native Solidworks, kernel Parasolid, and neutral STEP formats. This dataset is comprised of 20 unique CAD models designed in distinct CAD systems and represented by various formats. The STEP formats in this dataset originate from three CAD systems (Solidworks, CATIA V5, and FreeCAD), each utilising a unique geometric modelling kernel. The native and Parasolid files were created using Solidworks software. The CAD models were meticulously designed to ensure that their geometric shapes are exactly reflectional symmetric and axisymmetric. As a result, this dataset serves as an etalon for validating the CASD method's ability to detect symmetry. Therefore, this aspect instils confidence that the CAD models used for symmetry detection are appropriate to test the CASD method's generality.

Another aspect is to ensure representative types of symmetries within the datasets are appropriate, particularly in the second larger dataset. This dataset is utilised to validate the accuracy of the symmetry detection. As previously noted in Section 5.1, the highlighted types of symmetries include exact global and partial reflectional symmetric

and axisymmetric models, as well as non-symmetric CAD models. This aspect builds confidence for validating the accuracy of the symmetry detection.

The last aspect to consider when evaluating the suitability of example problems is including a wide range of mechanical part types in the datasets. To achieve this, various CAD models were gathered based on common manufacturing processes such as casting, moulding, forming, and machining. This diverse collection of CAD models ensures that the datasets adequately represent analytical and numerical surface types. A visual depiction of the corresponding bar charts of surface types in the datasets is given in Figure 37 and Figure 39. The datasets consist of parts where symmetry represents a functional requirement. For example, for rotational parts (gears, shafts, etc.), the detection of symmetry is essential to verify if they meet the functional requirements. Further, the datasets consist of parts where symmetry represents a technological requirement. For instance, detecting reflectional symmetry in plastic injection mould parts may provide information about its partitioning line. Symmetry recognition within mechanical parts is generally meaningful from the perspective of CAD, CAE, and CAM. From the standpoint of CAD, engineers rely on this information to verify if the 3D CAD model's geometrical shape meets the design intent. Even if it appears visually symmetrical, local missing features like fillets or chamfers can disrupt the symmetry. In CAE, identifying symmetry can help reduce the size of the analysis model and its computational demand. In CAM, recognising axisymmetry in 3D CAD models manufactured by turning represents valuable manufacturing information. To summarise, considering the outlined arguments for the mentioned aspects (data size and origin, CAD format types, symmetry types, surface types, and mechanical part types), one can confidently conclude that the collected dataset is appropriate for performance validation.

5.3 Performance Validation

As already stated, performance validation is a quantitative process with three phases. During the first phase, representative 3D CAD models are utilised to assess the effectiveness of the CASD method in detecting symmetry within a model. This should be accurate and insensitive to input CAD model format and modelling origin while maintaining an acceptable time complexity. In the second phase, the confidence of the usefulness of the tested CAD models is built by individually evaluating each step of the

CASD method. Finally, the third phase involves validating the usefulness of the CASD method beyond example problems. The three phases of performance validation are described in more detail in the following subsections.

5.3.1 Usefulness of the CASD method with respect to example problems

The first phase involves accepting the usefulness of the initial purpose of the implemented CASD method for example problems. These example problems are represented by the 3D CAD models from the datasets, and their appropriateness for symmetry detection has been demonstrated in Subsection 5.2.3. The validation process is divided into two parts, with the description of the datasets provided in Section 5.1. The first validation part is carried out on the first dataset, and it aims to evaluate the sensitivity of the CASD method against different input CAD model formats created using different CAD systems. The primary objective is to confirm the consistency of the proposed CASD method and detect any possible numerical errors. Additionally, it investigates whether CAD model interpretation may affect the symmetry detection results and identifies any possible differences in the detection of symmetric face pairs and stand-alone faces among input models generated using different CAD systems. The purpose of the second dataset is to validate the accuracy [147] and time complexity [148] of the CASD method via the computational environment. The accuracy gauges the correctness of symmetry detection, while the time complexity reflects the speed at which symmetry detection can be performed. Additionally, this validation part aims to expose any unforeseen computational or technical challenges. To accomplish this, a test-debug-test [32] cycle was executed to ensure adherence to the intended function, with necessary modifications made if such functionality was not achieved.

The first part of the validation is illustrated through several representative examples in Figure 41 to Figure 42. Overall, it was found that the corresponding APOS or AAOS were accurately detected in all 3D CAD models from this dataset, regardless of CAD system used to design them (Figures 41 – 43). This held true also for models represented in both native Solidworks and Parasolid formats. However, it was identified that in some rare cases local false negative symmetry detection may occur. Specifically, a small number of face pairs or stand-alone faces in the 3D CAD models may not be detected as symmetrical (as shown in Figures 42 – 43). Importantly, this does not impact the overall symmetry

detection results at the global level, only at the local level. The primary cause of false negative symmetry detection is that the specified computation error for exact symmetry (ε=10^{-6} m) cannot always be achieved. In these cases, the computation error typically falls within the range of ε=10^{-5} m.

False negative symmetry detection cases include when two reflectional symmetric faces fail to meet the equality criterion, meaning the difference between their surface areas exceeds ε_A (as per Equation (83)). This issue is common for numeric surfaces and may also occur in analytic surfaces with complex numeric intersection curves due to inaccurate computation of face area in Solidworks, resulting in an approximate value [146]. Additionally, if the coincidence criterion is not met within the specified ε for reflectional symmetric or axisymmetric faces (Equations (80) and (81)), the computed face centroid point or its point projected onto the face may not coincide with a POSC or AOSC, or the calculated midpoint of a face pair may not coincide with a POSC. Lastly, the orientation criterion may not be fulfilled when the lengths of cross-product vectors (Equations (86), (88), and (92)) exceed the computation error ε during querying for reflectional symmetric face pairs or axisymmetry stand-alone face.

The detection of global symmetry in 3D CAD models may be affected to a small extent by variations in CAD systems, but this impact is limited to local detection. It is worth noting that certain 3D CAD models, particularly those designed in CATIA V5, can have partitioned B-spline faces that could cause local symmetry misdetection. However, it is important to highlight that no false positive APOSs or AAOSs were identified. Likewise, the first dataset contained no false positive face pairs or standalone faces.

Based on this first validation part, it can be concluded that the implemented CASD method remains consistent and insensitive in detecting corresponding APOS or AAOS, even if the input 3D CAD models were generated in different CAD systems or are in native, Parasolid or STEP format.

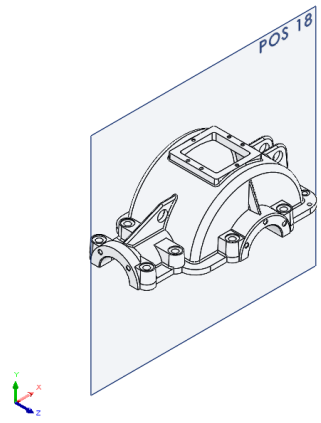
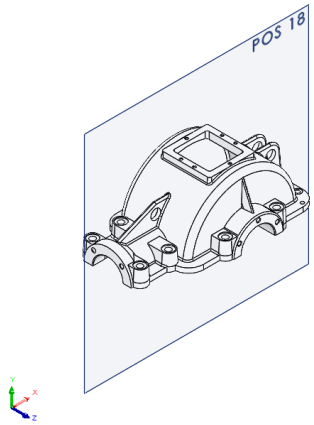
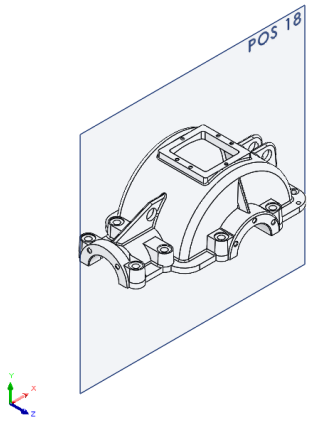
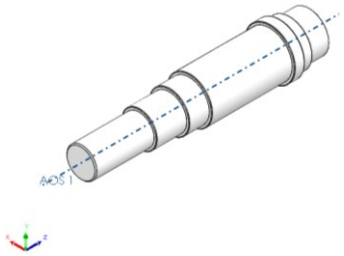
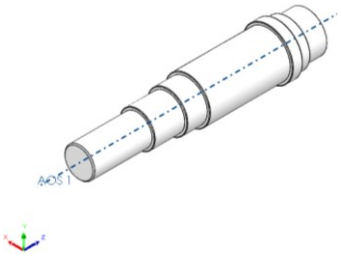
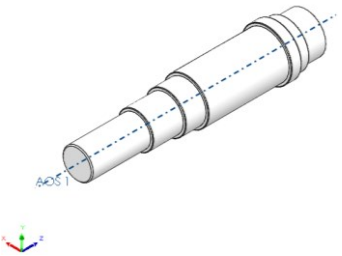
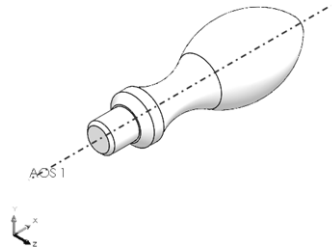
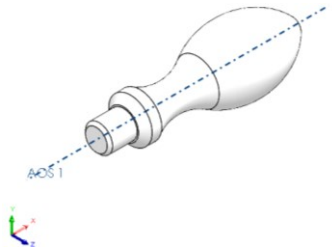
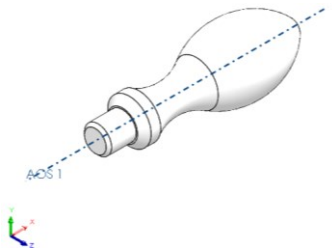
Solidworks	CATIA V5	FreeCAD
		
SFI=1 GSI=1	SFI=1 GSI=1	SFI=1 GSI=1
		
SFI=1 GSI=1	SFI=1 GSI=1	SFI=1 GSI=1
		
SFI=1 GSI=1	SFI=1 GSI=1	SFI=1 GSI=1

Figure 41. Examples of CAD models from the first dataset with the detected APOS and AAOS (Part 1)

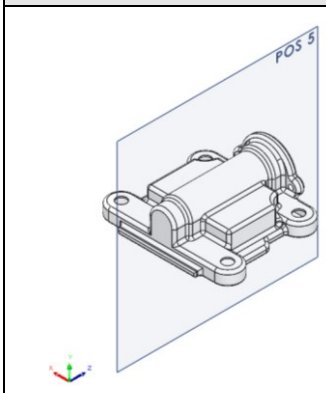
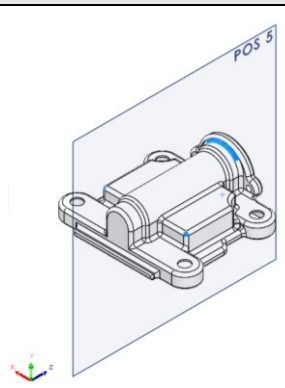
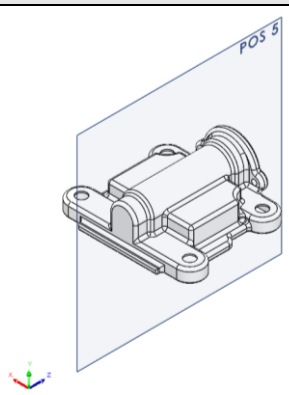
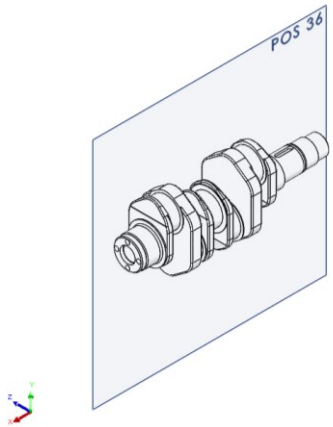
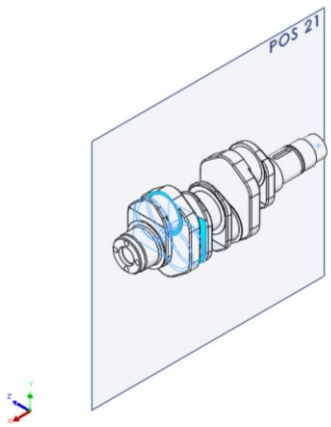
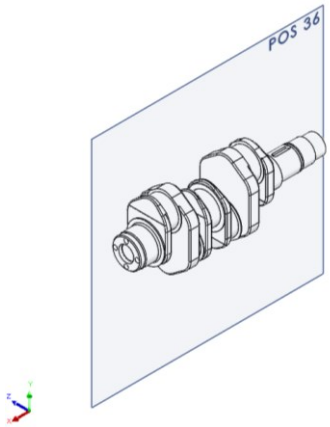
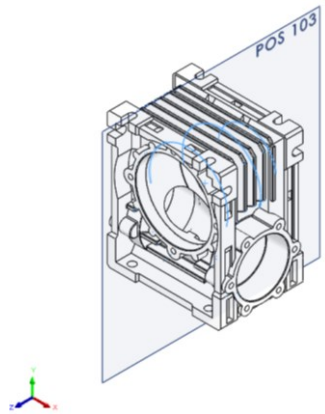
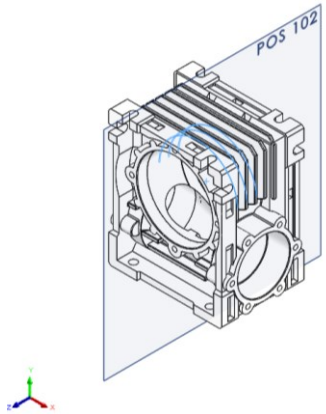
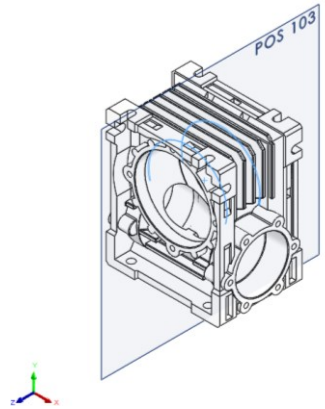
Solidworks	CATIA V5	FreeCAD
		
SFI=1 GSI=1	SFI=0.97 GSI=0.99	SFI=1 GSI=1
		
SFI=1 GSI=1	SFI=0.97 GSI=0.94	SFI=1 GSI=1
		
SFI=0.99 GSI=0.99	SFI=0.99 GSI=0.99	SFI=0.97 GSI=0.99

Figure 42. Examples of CAD models from the first dataset with the detected APOS and AAOS (Part 2)

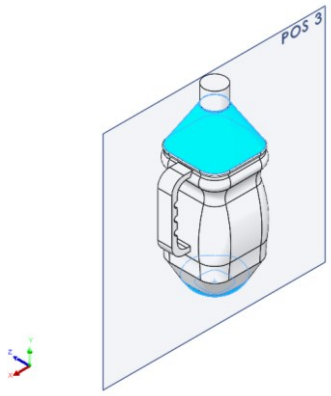
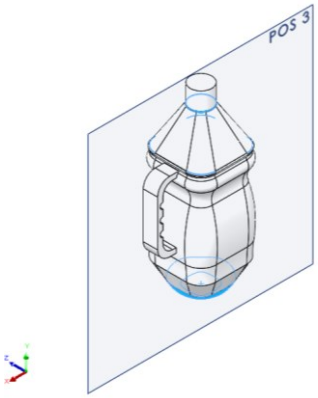
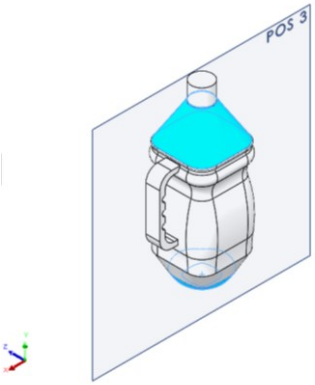
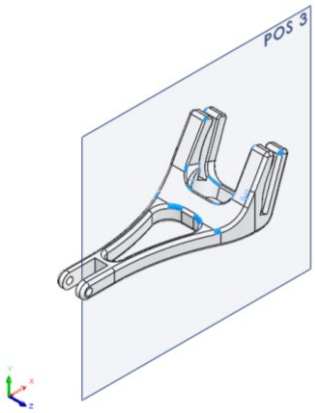
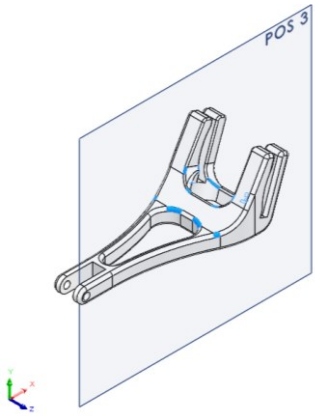
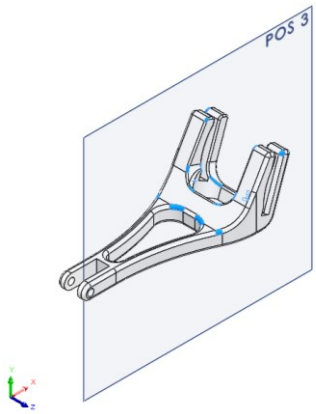
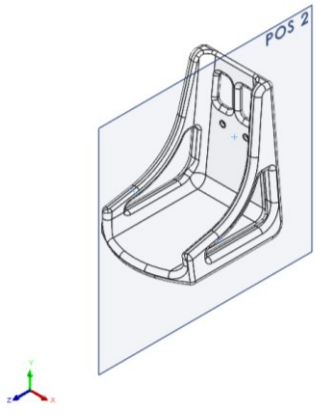
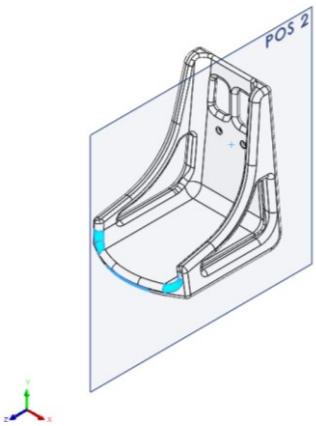
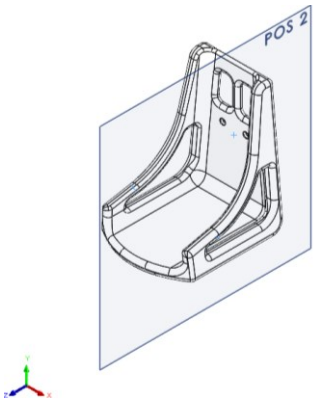
Solidworks	CATIA V5	FreeCAD
		
SFI=0.98 GSI=0.99	SFI=0.98 GSI=0.99	SFI=0.93 GSI=0.98
		
SFI=0.91 GSI=0.99	SFI=0.89 GSI=0.99	SFI=0.91 GSI=0.99
		
SFI=0.99 GSI=0.99	SFI=0.97 GSI=0.98	SFI=0.99 GSI=0.99

Figure 43. Examples of CAD models from the first dataset with the detected APOS and AAOS (Part 3)

The second part of the validation assesses the accuracy and time complexity of the CASD method using the second dataset, described in Section 5.1. The CASD method attempts to identify the corresponding planes and axes of symmetry in the 3D CAD model and identify whether the detected symmetry is exact global reflectional symmetry, partial reflectional symmetry, exact global axisymmetry, partial axisymmetry, or non-symmetry. Hence, evaluating the method's accuracy can be observed as a classification problem. For that purpose, the accuracy [147] can be calculated as follows:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP +TN + FP + FN}}, \tag{101}$$

where TP stands for true positive, TN for true negative, FP for false positive, and FN for false negative symmetry detection result. A true positive result denotes the detection of symmetry in the 3D CAD model when it does exist. A true negative result represents that symmetry is not detected and does not exist in the 3D CAD model. A false positive result indicates the detection of symmetry when it does not exist in the 3D CAD model. A false negative result implies that symmetry is not detected in the 3D CAD model when it exists.

The time complexity has been evaluated both theoretically and experimentally using the Big-O complexity chart [148]. This chart expresses the time complexity as a function of the input size, which is the number of faces n_F in the 3D CAD model. The time complexity or running time is important for assessing if the CASD method can detect symmetry in a reasonable time for practical applications. Moreover, the validation also included the assessment of the global symmetry index threshold GSI_{TH} value, which defines the border between partially symmetric and non-symmetric CAD models.

To ensure consistent results, the second part of the validation was conducted on the same hardware for all 3D CAD models in the second dataset, on a Dell Precision Working station with Intel i7-1165G7 up to 4.7 GHz processor and 16 GB RAM. As depicted in Figure 44, the results of the symmetry detection, i.e., the detected APOS and AAOS, were applied to a range of 3D CAD models that varied in symmetry type, including global reflectional symmetric (GRS), partial reflectional symmetric (PRS), global axisymmetric (GAS), partial axisymmetric (PAS), and non-symmetric (NS).

Figure 44. A sample of CAD models with the detected APOS and AAOS

The assessment of TP, TN, FP, and FN symmetry detection results involved evaluating exact and partial symmetry together. The overall accuracy of the symmetry detection was calculated based on these results. To ensure a conservative approach, a TP result was only

counted if all possible planes and axes of symmetry were detected in the CAD model. At the same time, a TN was accounted for only if all planes and axes of symmetry were not detected. Conversely, an FP was recorded if even one plane or axis of symmetry was incorrectly identified, and an FN was noted if even one plane or axis of symmetry in the CAD model was missed. Table 21 presents the results of this evaluation, and based on these findings, the symmetry detection accuracy for the 3D CAD model from the second dataset is calculated to be 0.87.

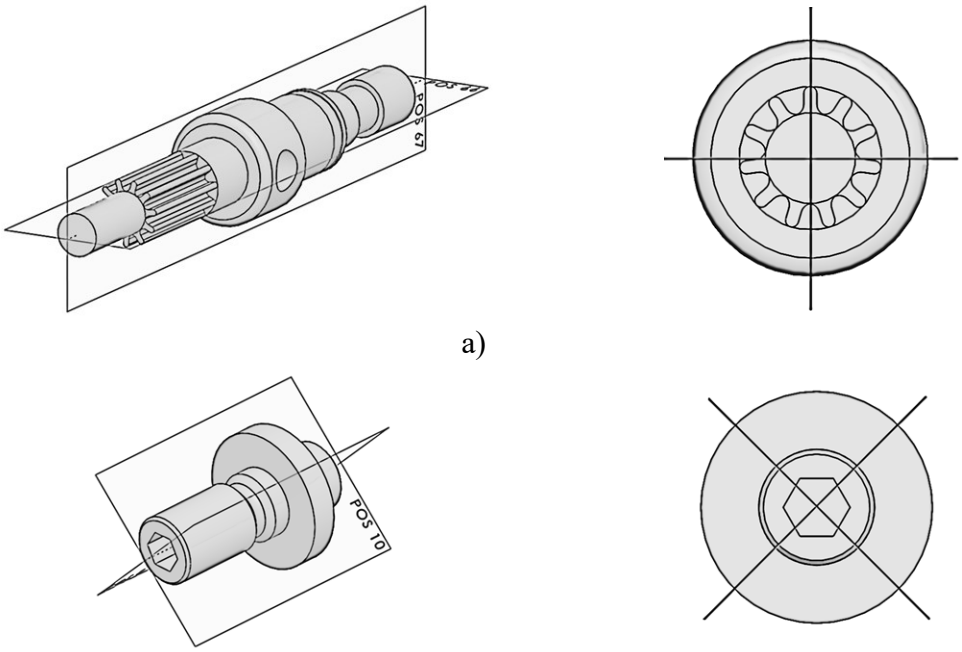
Table 21. Accuracy score of the symmetry detection

Result	No. of CAD Models
TP	779
TN	86
FP	59
FN	76
Accuracy	0.87

The CASD method can detect symmetry at the minimum manufacturing accuracy in mechanical engineering (ε=10^{-6} m), which can be achieved for both analytical and numeric surfaces. However, in specific cases, the method may fail to detect symmetry. The typical types of computational errors occurring during symmetry detection have already been outlined during the first part of the validation. While these errors are at the local level and may cause a negligible reduction of the SFI and GSI scores (Figure 44, CAD models (1), (11) and (12)), they will not result in an overall miss-detection of symmetry in the 3D CAD model. Another issue that may arise pertains to creating POSCs and AOSCs from the PAOI. This problem is revealed in the alignment of the PAOI with the 3D CAD model. Sometimes, a round-off error may occur at the fifth or sixth decimal place, leading to a computational error during candidate evaluation further down the line. The root of this error can be traced back to the CAD system's functionality for computing the PAOI, although this is a rare occurrence. To resolve this issue, it may be helpful to stringent the computation error ε for trimming candidates (from the current ε=10^{-6} m to ε=10^{-7} m) to avoid eliminating duplicate candidates from similar face pairs that are coincident with the POAI exhibiting alignment issues.

The detection of FP symmetry can occur at either the global or local scale. When detected globally, it means that the identified APOS is incorrect. On the other hand, when detected

locally, it indicates that the identified face pair or stand-alone face is incorrect (as previously discussed for the first dataset in this subsection). Figure 45 provides two examples of FP results at the global scale, which can happen in parts that exhibit both multiple reflectional symmetry and partial axisymmetry. False positive APOSs may be detected from the PAOI POSCs. In the first example, only one APOS is FP (the horizontal plane in Figure 45 – a), whereas in the second example, two APOSs are FP (Figure 45 – b). Note that only the APOSs from the PAOI are displayed in Figure 45, as other APOSs were detected but not plotted to avoid confusion. Although the detected APOSs are not incorrect because they reveal partial reflectional symmetry in the CAD model, they are meaningless since an infinite number of such APOSs exist. Therefore, these detected APOSs can be considered as FP.



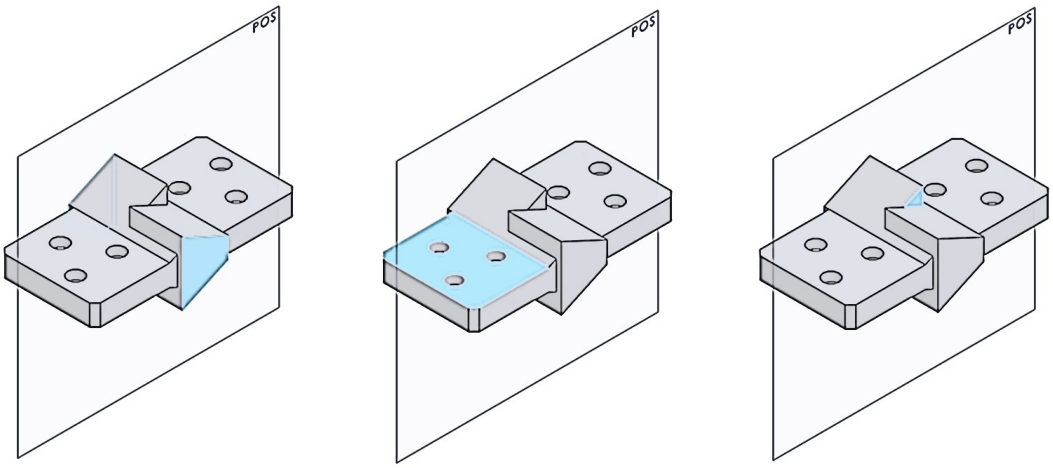
a)

b)

Figure 45. Examples of global FP symmetry detection

Figure 46 and Figure 47 show several examples of FP symmetry detection at the local scale. Figure 46 shows an artificial object created to demonstrate local FP reflectional symmetry. Within this example, a face pairs that is not reflectional symmetric (Figure 46 – a) and two stand-alone face that are not self-symmetric (Figure 46 – b and c), were

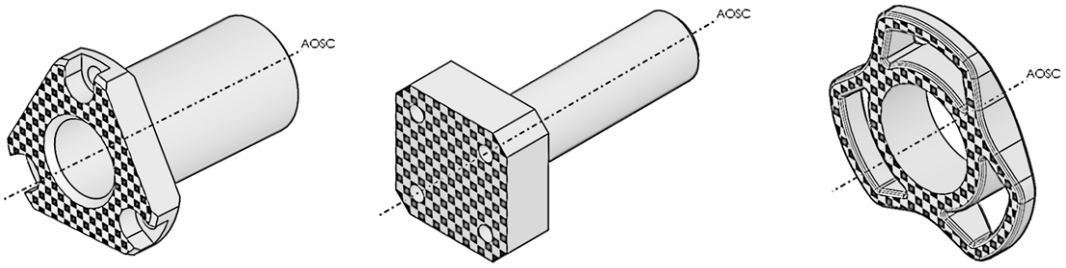
identified by the CASD method based on the corresponding criteria (equality, coincidence, or orientation criteria).



a) b) c)

Figure 46. Examples of FP stand-alone faces and face pairs

The CASD method may also detect FP stand-alone faces in axisymmetric 3D CAD models (Figure 47). This is because the face centroid coincides with the AOSC, even though the face itself is not axisymmetric. However, such FPs are not typical in practice. In case of occurrence, they usually do not lead to global symmetry misdetection, as other adjacent faces in the 3D CAD model are not recognised as FP.



a) b) c)

FP axisymmetric stand-alone face

Figure 47. Examples of local FP symmetry detection

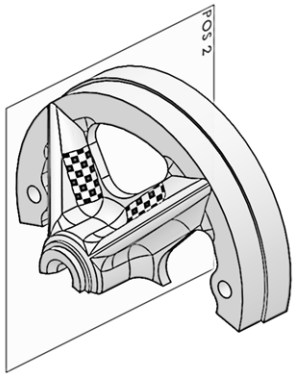
Global and local FN symmetry detection has also been identified in 3D CAD models from the dataset. An FN at the global scale implies that the APOS of the 3D CAD model has not been detected at all, while at the local scale, it may occur as a misdetection of a face pair or stand-alone face. Typical examples of FN symmetry detection at the global level are partially reflectional symmetric 3D CAD models (Figure 48), which have no symmetric face pairs (planar, cylindrical, and B-spline surface type) that can be exploited

for the generation of the POSC(s). However, exact reflectional symmetric 3D CAD models are less likely to experience FN symmetry detection since the POSCs can be detected from the PAOI, even without symmetric face pairs.

a) b)

Figure 48. Examples of global scale FN symmetry detection

Figure 49 illustrates an instance of FN symmetry detection at the local level caused by symmetric face pairs. The reason for FN is that symmetric face pairs belong to different face classes or surface types. This is rare in practice, often resulting from modelling or interpretation. When design features are split into multiple operations, different surface types may be created instead of one, or the CAD system may interpret two symmetric faces as different surface types during 3D CAD model import. Consequently, the proposed CASD method fails to identify such face pairs since it only evaluates faces within the same class.



FN symmetrical face pair

Figure 49. An example of a FN symmetric face pair

The proposed CASD method is insensitive to specific positions or orientations of a 3D CAD model within the 3D modelling space. That is an important aspect as the 3D CAD

model COG does not necessarily need to be coincident with the origin, or its PAOI may not be aligned with the coordinate axes of the modelling space. In some instances, the proposed method indirectly detects cyclic symmetry by detecting multiple reflectional symmetries in the CAD model (Figure 50). However, identifying this type of symmetry is still up to the user since there is no adequate symmetry measure yet. Therefore, in addition to exact global and partial axisymmetry, the proposed CASD method can indirectly detect exact global rotational symmetries in some instances.

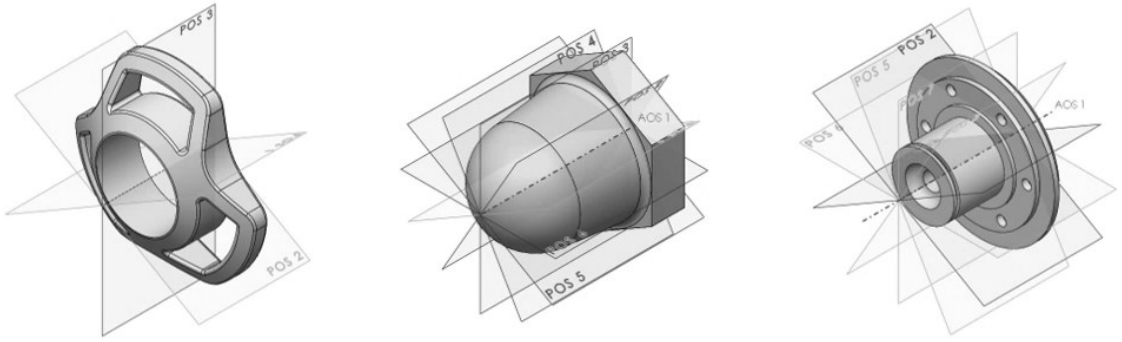


Figure 50. Examples of 3D CAD models that are multiple reflectional symmetric as well as cyclic symmetric.

The time complexity has been estimated theoretically and experimentally as in previous studies [16,42]. The theoretically estimated time complexity represents the worst-case with the longest running time, where only the highest-order term is considered for each step of the method because it dominates over the other terms on large inputs [148]. During the B-rep Analysis step, each face in the CAD model is labelled and analysed for its properties, which takes linear time $O(n_F)$. In the candidate generation and trimming step, both the POSCs and AOSCs must be pairwise compared for trimming. The time complexity in those steps depends on the number of faces of the plane (n_{PL}), cylindrical (n_{CY}) and B-spline (n_{BS}) surface type in the CAD model. However, if the CAD model consists entirely of one surface type ($n_F=n_{PL}$), the time complexity can be expressed as $O(n_F^2)$ in the worst-case scenario. This also applies to evaluating candidates, where pairwise face comparisons require $O(n_F^2)$ time. The visualisation step involves looping through all candidates, with a time complexity of $O(n_F)$ in the most conservative case ($n=n_{PL}$). The theoretical time complexities of each step are summarised in Table 22.

Table 22. Theoretical time complexity of the proposed CASD method

CASD method's step	Time Complexity
B-rep analysis	$O(n_F)$
Generation and trimming of POSCs and AOSCs	$O(n_F^2)$
Evaluation of POSCs and AOSCs	$O(n_F^2)$
Visualisation of APOSs and AAOS	$O(n_F)$
All steps	$O(n_F^2)$

As mentioned, the experimental time complexity has been obtained through testing on the collected 3D CAD models, and the results for each particular step are illustrated in Figures 51 – 54, as well as the overall time complexity of the symmetry detection method in Figure 55.

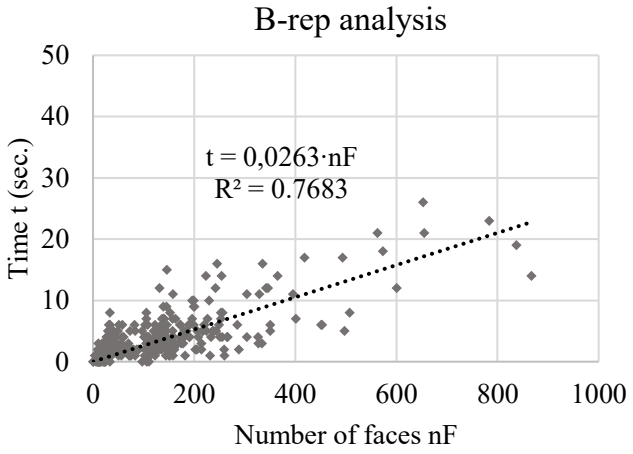


Figure 51. Big-O charts for B-rep analysis step

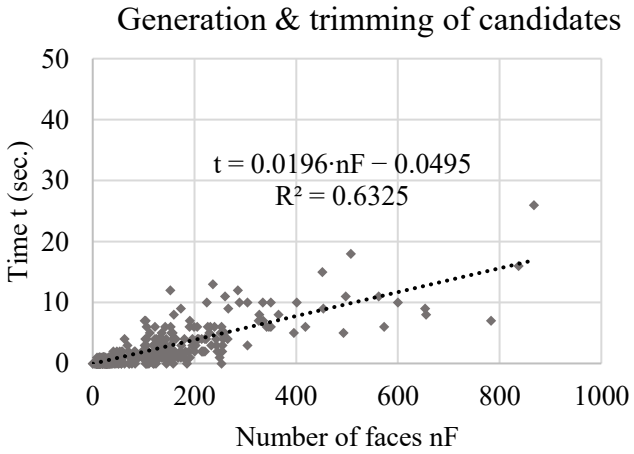


Figure 52. Big-O charts for generation and trimming of candidates' steps

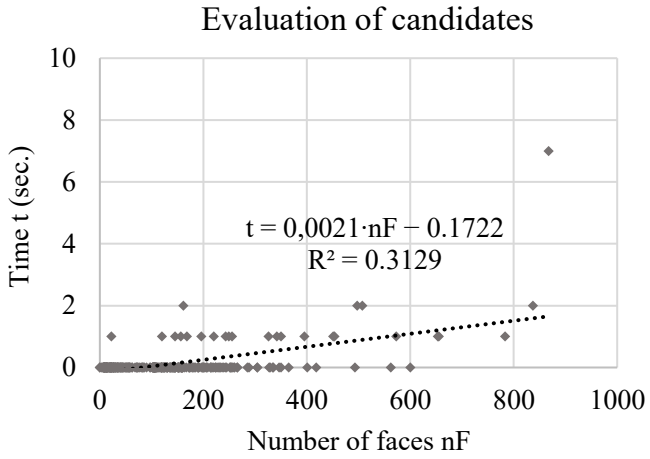


Figure 53. Big-O charts for evaluation of candidates' step

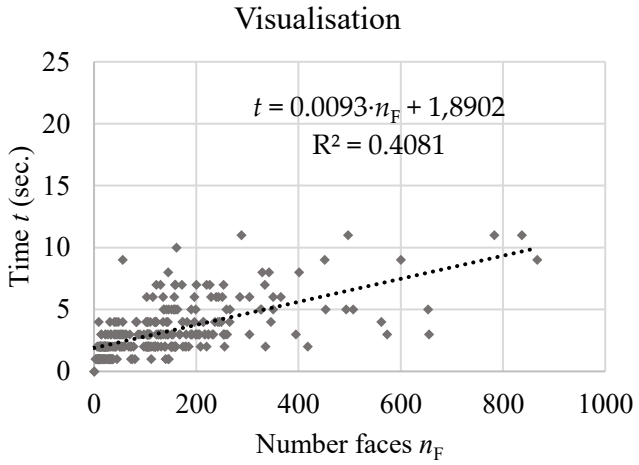


Figure 54. Big-O charts for visualisation step

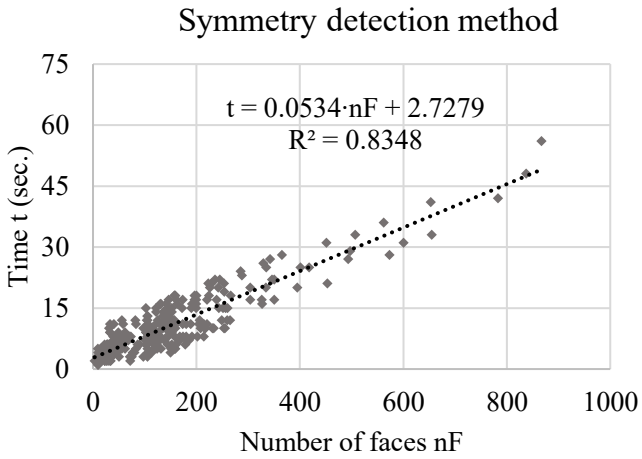


Figure 55. Experimental time complexity of the symmetry detection method

The comparison of theoretical and experimental time complexity reveals that there is a difference between the candidate generation, trimming and evaluation steps, as well as the overall time complexity of the symmetry detection method. Theoretically, these steps have been estimated to have a time complexity of $O(n_F^2)$ time, while experimentally, the time complexity is estimated to be $O(n_F)$ time. This difference is because the theoretical time complexity assumes a worst-case scenario, where all faces in the 3D CAD model are of only one surface type, such as planes. However, the CAD models from the dataset are comprised of various surface types, and only faces of the same type require pairwise comparison, not all faces in the 3D CAD model. Therefore, the actual number of inputs is less than the total number of faces in the CAD model, corresponding to the highest number of faces of the corresponding surface type. Consequently, the experimental time complexity represents the average running time. Additionally, the linear trend $O(n_F)$ of the experimental time complexities could be attributed to the fact that the number of faces needs to be significantly larger to produce the quadratic trend. Nonetheless, the structural validation in Subsection 5.2.3 has confirmed the appropriateness of the datasets, and there is no need to artificially increase the number of faces. Finally, the experimentally evaluated $O(n)$ time complexity of the CASD method is improved compared to the prior CASD studies, which estimated their algorithms to $O(n^4)$ [23] and close to $O(n^2)$ [63] time.

Finally, to estimate the threshold value for the global symmetry index GSI_{TH}, a plot of the GSI and SFI has been done for the 3D CAD model of the second dataset (Figure 56). The categories of the 3D CAD models can be divided into exact global, partial, and non-symmetric. As already stated, the SFI measures the symmetry of the topology, while the GSI measures the symmetry of the geometry. Exact global symmetry implies that SFI=1 and GSI=1. In the case of partial symmetry, SFI<1 and $GSI \geq GSI_{TH}$. Non-symmetry implies SFI<1 and $GSI < GSI_{TH}$. The analysis of the CAD models in Figure 56 suggests that a value of approximately $GSI_{TH}=0.70$ would be an appropriate threshold between partial symmetry and non-symmetry. However, this is just a proposal, and the engineer can select another value of GSI_{TH}.

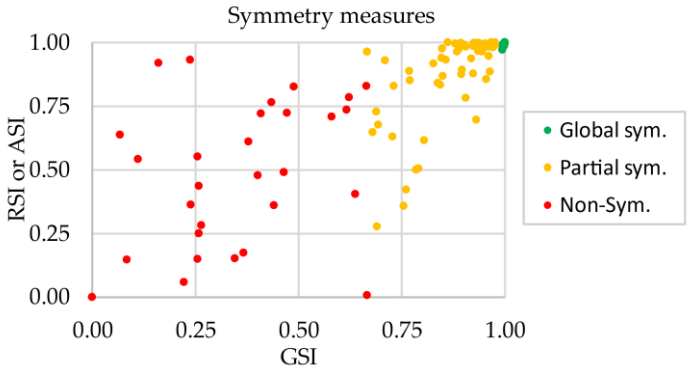


Figure 56. GSI vs. SFI plot

5.3.2 Usefulness linked to applying the CASD method

The second phase of performance validation is to build confidence that the usefulness of the symmetry detection results for the 3D CAD models is linked to applying the CASD method. This is achieved through a quantitative assessment of the usefulness of each step individually by comparing the symmetry detection results with and without the step. Furthermore, comparisons are made between the symmetry detection outcomes of certain steps and those from previous CASD studies.

In the context of this validation phase, the interpretation of the 3D CAD model represents a mandatory step without which symmetry detection cannot be conducted. Similarly, B-rep analysis is also an irreplaceable step; symmetry detection can only be conducted with it. However, since B-rep analysis involves multiple sub-steps, including topology merging and classification, as well as face and edge property calculation, the validation phase can be conducted within each sub-step. Theoretically, the symmetry detection could be conducted without the first sub-step topology merging. It was already mentioned in Subsection 5.3.1 that the false negative symmetry detection results may occur in 3D CAD models that have partitioned faces (this is common when designing in CATIA V5). Figure 57 shows the symmetry detection results with and without this sub-step for several example cases from the second dataset.

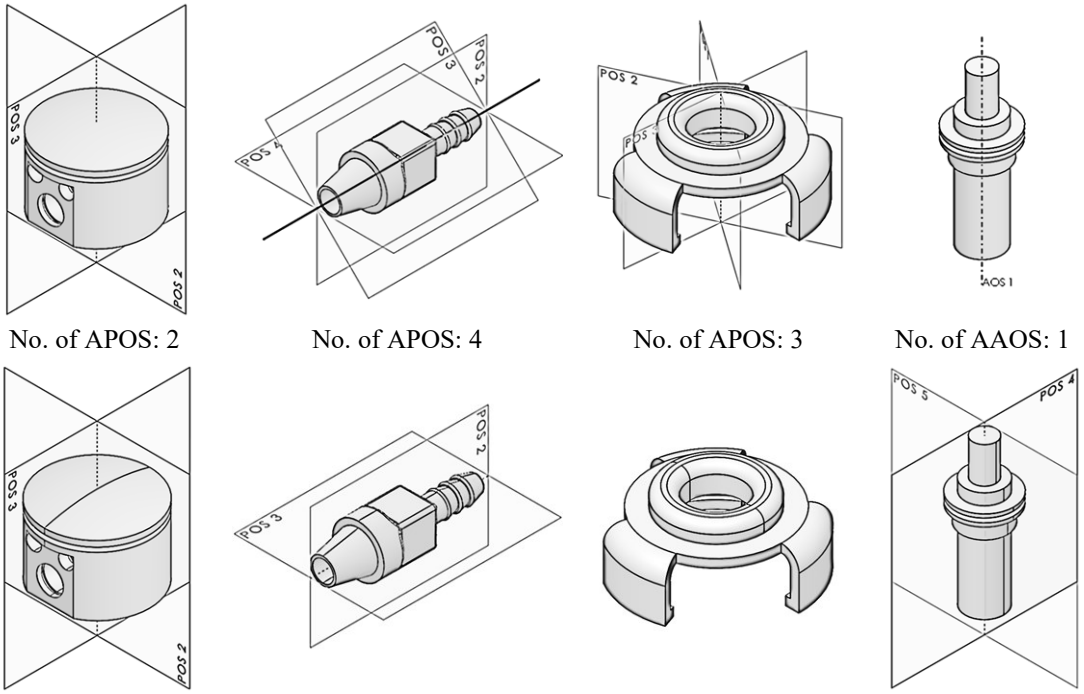


Figure 57. Symmetry detection results with (upper row) and without (lower row) the sub-step classification of topology

Without the topology merging sub-step, a closed periodical face (such as cylindrical, toroidal, conical, or spherical surfaces) may be partitioned into two face halves (Figure 57, lower row). Consequently, this may lead to an increased number of false negative (Figure 57, b and c) or false positive (Figure 57, d) symmetry detection results. This is because the partitioned face halves may disturb the symmetry of the 3D CAD model. The partitioned face halves can disrupt the symmetry of the 3D CAD model, except in cases where they are symmetrically arranged with respect to a candidate (Figure 57, a). The comparison of the symmetry detection results with and without the merging of topology proves the usefulness of this sub-step.

The second sub-step involves classifying topological entities based on their geometric type. Topological entities from the same class are then pairwise compared in the generation and evaluation steps. Omitting this sub-step would require a pairwise comparison of all topological entities, resulting in higher time complexity. The usefulness of this sub-step is confirmed by the fact that the time complexity for pairwise comparison of all faces in the 3D CAD model is higher than the sum of the time complexities for pairwise comparison of faces in different classes $O(n^2) > O(n_{\mathrm{PL}}^2) + O(n_{\mathrm{CY}}^2) + O(n_{\mathrm{CO}}^2) +$

$O(n_{SP}^2) + O(n_{TO}^2) + O(n_{BS}^2) + \ldots$. Finally, the sub-step face and edge properties calculation is essential for the subsequent steps of the CASD method, although it is not quantitatively measurable in the validation phase. Without this sub-step, the symmetry detection process would not succeed. Thus, the usefulness of this sub-step is indisputable.

The next step of the CASD method deals with generating POSCs and AOSCs. As the proposed CASD method relies on the implicit (indirect) symmetry detection approach, it is impossible to conduct the symmetry detection without this step. Thus, the usefulness of this step will be observed from the perspective of its sub-steps. The candidates are generated from the PAOI, pairs of similar faces, and single faces. The candidates generated from the PAOI are used to detect exact global reflectional or axisymmetry in the 3D CAD model. Without the mentioned sub-step, the risk of false negative symmetry detection increases. To prove this statement, several CAD models from the second dataset were subjected to symmetry detection without this sub-step (Figure 58).

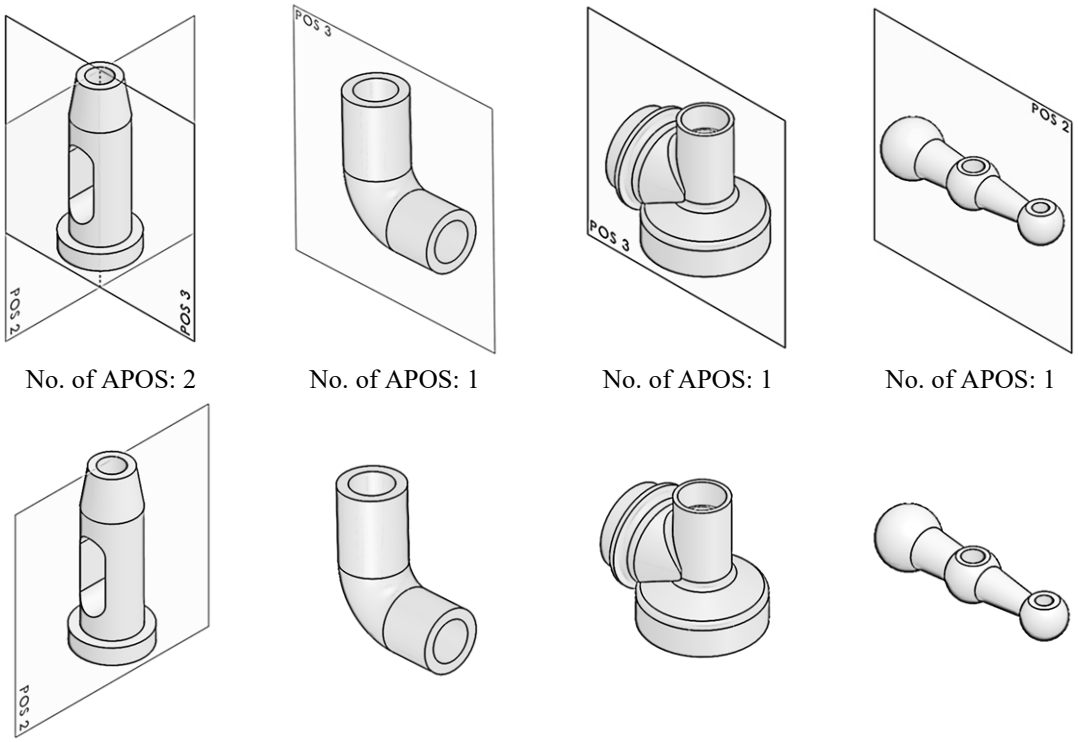


Figure 58. Symmetry detection results with (upper row) and without (lower row) the sub-step generation of candidates from the PAOI

The results demonstrate that the CASD method may fail to detect exact reflectional symmetric 3D CAD models (Figure 58, lower row). Typically, this happens when there

are no reflectional symmetric face pairs within the 3D CAD model to replace the missing PAOI candidates in the following sub-step. The symmetry detection results with and without the generation of candidates from the PAOI prove this sub-step's usefulness, especially regarding reflectional symmetry. However, the lack of this sub-step does not influence the detection of axisymmetry, as the missing PAOI candidates are replaceable through the generation of candidates from single faces.

Next, the symmetry detection results with and without the sub-step generation of candidates from similar face pairs are observed (Figure 59). Removing this sub-step from the CASD method may reduce the number of POSCs and increase the risk of false negative symmetry detection. In particular, it negatively influences the detection of exact symmetries that are not aligned with the PAOI (Figure 59, a – c) and partial reflectional symmetry (Figure 59, a – c). Thus, the symmetry detection results with and without this sub-step provide compelling evidence for its usefulness.

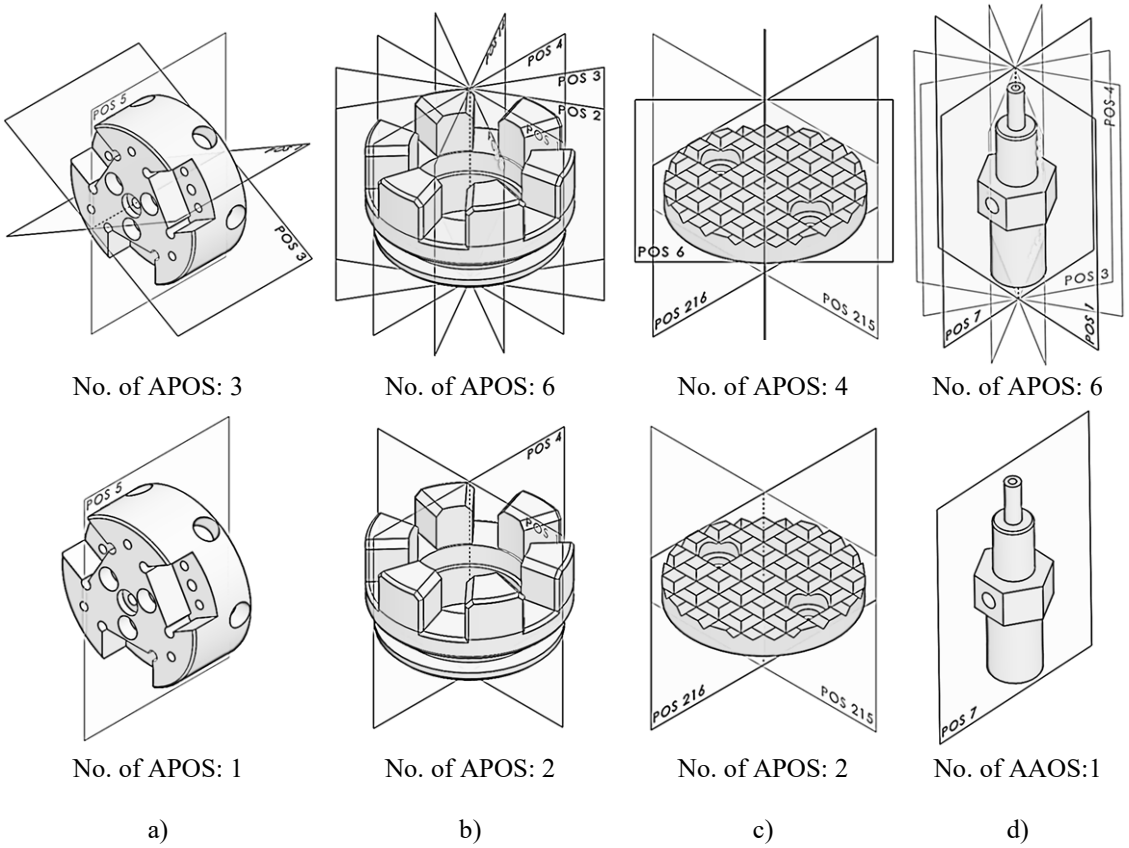


Figure 59. Symmetry detection results with (upper row) and without (lower row) the sub-step generation of candidates from similar face pairs

Finally, the last sub-step consists of the generation of AOSCs out of single faces. The impact of removing this sub-step is illustrated in Figure 59 through several test cases from

the second dataset (the upper row represents the results with the sub-step, while the lower row without the sub-step). Apart from AOSCs generated from the PAOI, there are no additional AOSCs generated without this sub-step. Thus, the CASD method fails to detect partial axisymmetry within the 3D CAD model (Figure 59, lower row), while there is no impact on the detection of reflectional symmetry. Based on the results, the usefulness of this sub-step can be confirmed.

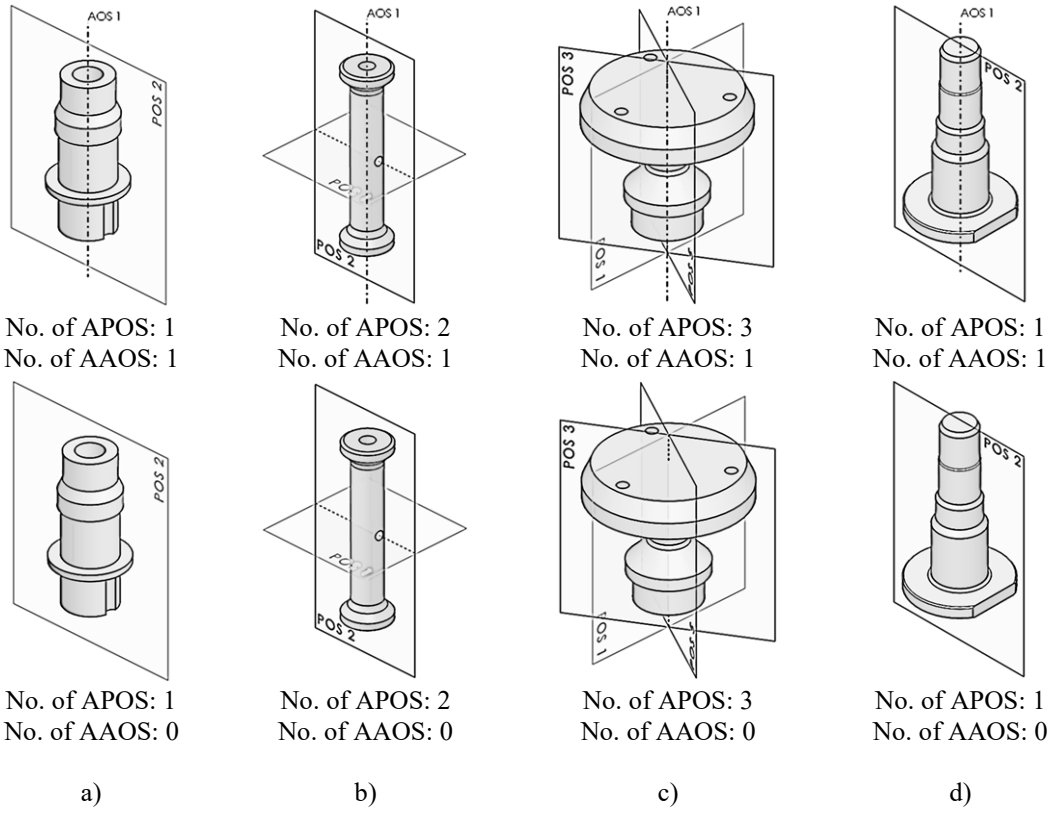


Figure 60. Symmetry detection results with (upper row) and without (lower row) the sub-step generation of candidates from single faces

Next in line is the step trimming of candidates. Theoretically, the CASD method can work without this step because it does not produce additional candidates and thus has no effect on the accuracy of symmetry detection. Instead, it attempts to reduce the number of initially generated candidates to avoid their unnecessary evaluation. In this way, the time complexity of the CASD method can be reduced. The step consists of two sub-steps – eliminating duplicate and unsuitable candidates. The usefulness of this step is validated on the second dataset by experimentally investigating the number of candidates before and after trimming, which is also compared with prior CASD studies [16,23]. Identifying too many POSCs and AOSCs without their practical need for detecting symmetries can

lead to a computationally demanding CASD method. Conversely, identifying of too few POSCs and AOSCs increases the risk of symmetry detection failure. The proposed CASD method generates, on average, 1141 candidates (POSCs and AOSCs) per 3D CAD model, while after trimming, the average number of candidates per 3D CAD model is reduced to 15. This proves the usefulness of the trimming step, as the number of candidates can be reduced by ≈41 times. Table 23 compares the number of generated candidates with prior CASD studies [23,63] regarding the average number of candidates per CAD model. The number of candidates from prior studies has been estimated theoretically from the equations. Compared to the study in [23], the present CASD method produces ≈72 times fewer candidates and ≈57 times fewer candidates than in the study [63].

Table 23. Comparison of the number of candidates with prior studies

CASD study	Equation	$(n_{POSC} + n_{AOSC})$ per CAD model	$(n_{POSC} + n_{AOSC})$ per CAD model used as input for CASD
Study [23]	(1)	2108	2108[a]
Study [63]	(2)	26776	2678[b]
CASD method	-	1514	37[c]

[a] The initially generated candidates are also used as input for symmetry detection.

[b] After the rationalisation of candidates. Assumed 10% of the initially generated candidates, the exact number cannot be estimated.

[c] After trimming of candidates.

In the CASD method, the final two steps involve assessing candidates in relation to the B-rep of the 3D CAD model to identify the corresponding APOS or AAOS and then displaying it in the 3D CAD model. These steps are crucial, and the CASD method cannot deliver complete symmetry detection results without them. As such, within the current performance validation, the importance of these steps is self-evident.

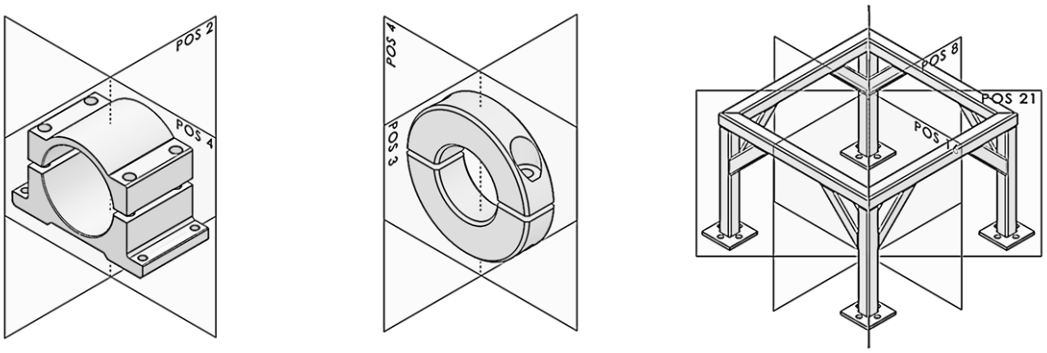
In conclusion, validating each step of the CASD method individually by comparing symmetry detection results with and without the step provides compelling evidence of its usefulness. It is worth noting that certain steps, such as interpreting the CAD model, B-rep analysis, candidate generation, and evaluation, are indispensable to the success of symmetry detection. Moreover, it was demonstrated that omitting their sub-steps from the steps can have an adverse impact on the accuracy of symmetry detection. While trimming candidates is not a necessary step and theoretically can be excluded from the CASD method, doing so could negatively impact the time complexity of symmetry detection.

5.3.3 Usefulness of the CASD method beyond example problems

The final performance validation phase aims to investigate the method's usefulness beyond the tested example problems to produce belief in its generality. In contrast to the Validation Square's suggestion of using only a few case studies [33], this study analysed two datasets comprising 1100 representative 3D CAD models. While this is a substantial number, the datasets were restricted to single parts with one body and manifold geometry, as outlined in Section 1.2. Furthermore, the tested CAD models consisted of a combination of analytic and numeric surfaces. Despite these limitations, the CASD method's applicability beyond the tested models can be evaluated from various aspects.

One important aspect is the usefulness of the CASD method for single parts with multiple bodies. Usually, a part is represented by only one body, but it can have multiple bodies in some instances, such as welded structures or injection moulded plastic parts with metal inserts. Therefore, it is preferable to also consider such cases in the context of symmetry detection. The second aspect is the CASD method’s usefulness for CAD models with non-manifold geometry (examples are illustrated in Figure 5). This is particularly relevant for symmetry detection in FEA models [14,66], which often use non-manifold geometry (examples of manifold geometry are illustrated in Figure 5). The third aspect is the usefulness of the CASD method for assembly CAD models, which are also frequently exploited in CAD. The fourth aspect is the usefulness of the CASD method for CAD models with a file format different from the example cases, such as ACIS and IGES. The last aspect is the CASD method’s usefulness for CAD models, which are dominantly or entirely made of numeric surfaces, usually generated with surface modelling tools. This aspect is particularly important in industrial design, automotive, and aerospace industries.

For the evaluation of the CASD method's usefulness on single parts with multiple bodies, several 3D CAD models were subjected to symmetry detection via the computational environment (Figure 61). These models included a clamping ring and housing with two bodies (Figure 61 – a and b), as well as a welded table with 20 bodies (Figure 61 – c).



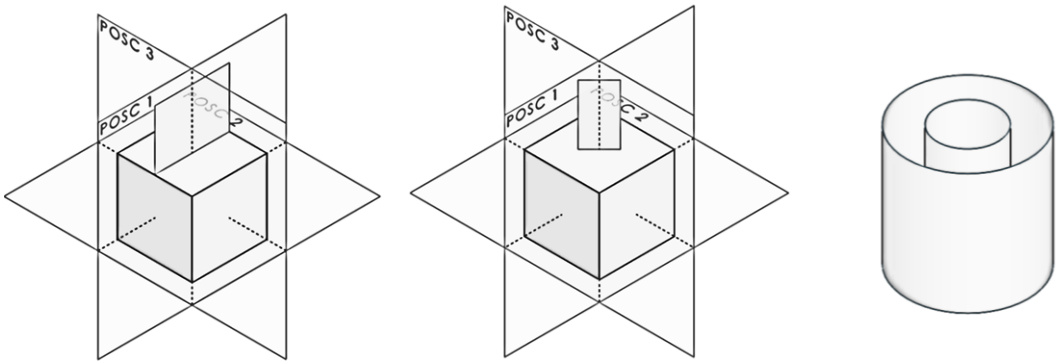
a) b) c)

Figure 61. Examples of single part 3D CAD models with multiple bodies

Based on the symmetry detection results illustrated in Figure 61, it can be concluded that the CASD method is useful for detecting global APOS and AAOS of parts with multiple bodies. Moreover, with specific modifications to the source code, the implemented CASD method could be configured to detect APOS or AAOS for each individual body, thereby enabling symmetry detection at the local level. To achieve this, the GSI and SFI should be calculated by incorporating only those faces that pertain to a given body.

The second aspect is the usefulness of the CASD method for non-manifold CAD models. In particular, the focus is on non-manifold geometry common in FEA, where symmetry detection is advantageous in reducing the overall computational effort of the analysis. CAD models for FEA are often idealised to make them more mesh-friendly, such as thin-wall structures or sheet metal parts. These CAD models are initially designed as solids but can be idealised by a mid-surface model or represented as a combination of bodies and mid-surfaces. This idealisation can result in non-manifold geometry, as shown in Figure 62. Since Solidworks is a non-manifold CAD system, performing symmetry detection of non-manifold CAD models via the computational environment is impossible. Therefore, the usefulness of the CASD method for non-manifold CAD models is discussed theoretically. In manifold CAD models, an edge can only be shared by two adjacent faces, while in non-manifold CAD models, it can be shared by more than two faces (e.g., by three faces, as shown in Figure 62). While this is generally not an issue fir the CASD method, when combining solids and surfaces in non-manifold 3D CAD models, it can affect the generation of POSCs and AOSCs from the PAOI. In such models, surfaces lack thickness and mass properties, and the PAOI are only computed from solid bodies. For example, two different manifold CAD models have the same POSCs (POSC

1 to POSC 3) generated by the CASD method because they share the exact solid shape, the cube in Figure 62 – a and b. In both CAD models, symmetry would be detected from the candidates generated from similar face pairs. In manifold 3D CAD models entirely composed of surfaces (Figure 62 – c), there is a risk of misdetection of exact reflectional symmetry because the PAOI cannot be computed. It can be concluded that the CASD method is generally applicable to non-manifold CAD models. However, generating POSCs and AOSCs requires improvement to overcome the highlighted risks.



a) b) c)

Figure 62. Examples of non-manifold 3D CAD models

The next aspect is the usefulness of the CASD method in assembly CAD models. Most often, assembly CAD models consist of multiple-part CAD models (Figure 63). These part CAD models are separate files associated with the assembly CAD model and can have one or multiple bodies. Alternatively, an assembly CAD model can be transformed into a part CAD model to decrease file size and the total number of files. Consequently, within such an assembly CAD model, each part CAD model is turned into a body (Figure 64). While it has been demonstrated that the CASD method can manage multi-body part CAD models, further exploration is necessary to assess its applicability to assembly CAD models.

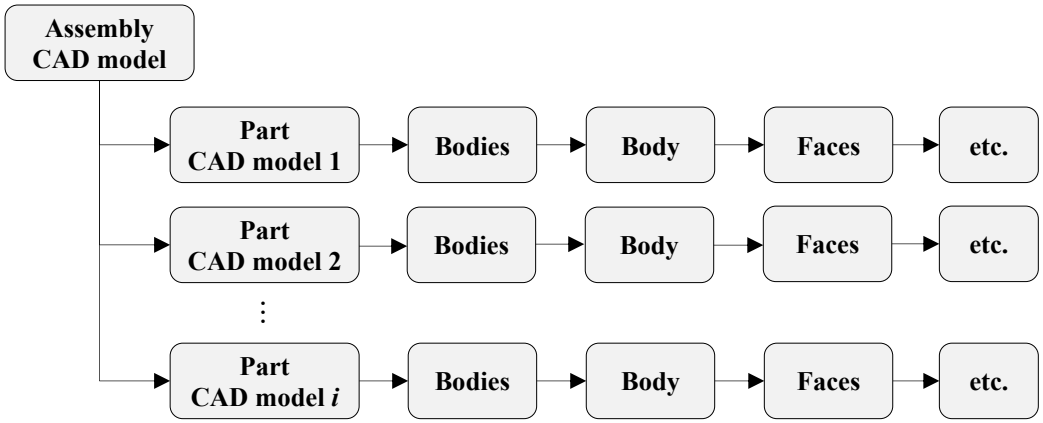


Figure 63. Assembly CAD model data structure compound of multiple-part CAD models

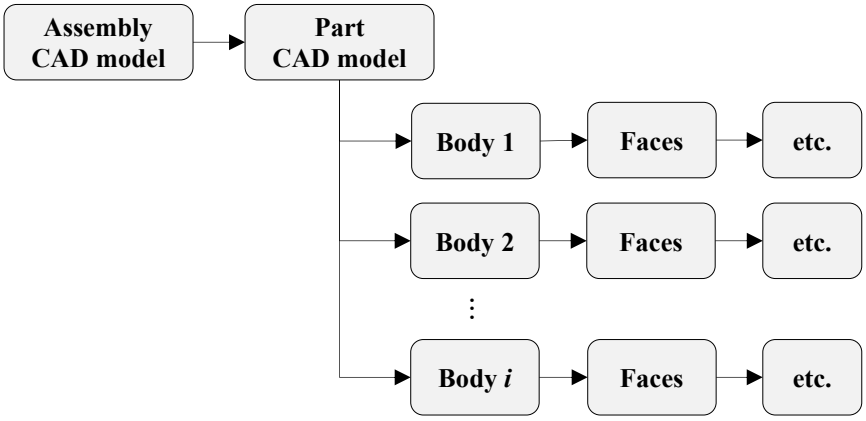


Figure 64. An assembly CAD model represented as multi-body part CAD model

The CASD method is designed to identify symmetry in single-part CAD models, enabling symmetry detection in each part of the CAD models that make up the assembly. This produces local symmetry information within the assembly CAD model. To detect global symmetry in the assembly CAD model, all faces and edges of the part CAD models need to be implemented in the B-rep analysis. There are two ways to generate candidates. The first way, covered by the initially proposed CASD method, is to generate candidates separately from each part CAD model. The second way of generating candidates is directly from the assembly CAD model, which requires modifying the proposed CASD method to incorporate all faces for the candidate generation. Analogically, all faces from the assembly CAD model must be included also during the candidate evaluation step. Two potential failures not covered by the proposed CASD method may occur during the evaluation. The first is the detection of false positive symmetric face pairs, which can happen if mating faces between parts are coincident and identical or similar. The second

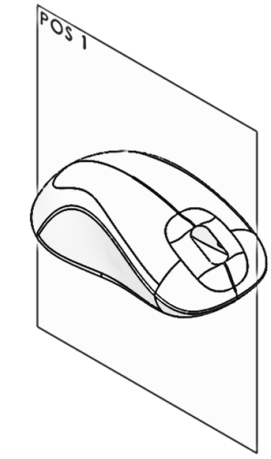
failure is related to the orientation of parts in the assembly. For example, bolts can have a symmetrical or non-symmetrical arrangement depending on their rotational positions in the assembly (Figure 65).

reflectional symmetric bolts non-symmetric bolts

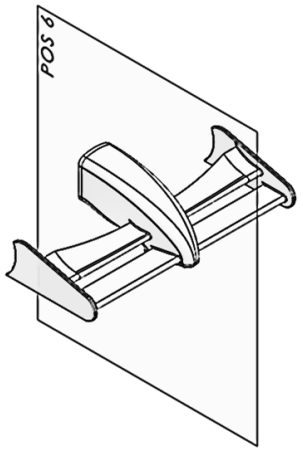
Figure 65. Examples of orientation of bolts in an assembly CAD model

While the proposed CASD method is suitable for CAD assemblies represented as multi-body part CAD models, further improvement is necessary for assembly CAD models consisting of part CAD models to overcome the mentioned failures and ensure the detection of the APOS and AAOS.

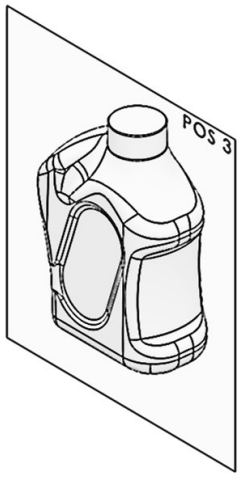
The next aspect considered is the usefulness of the CASD method for CAD models compound predominantly of numeric surfaces. The CASD method has already been validated (see Section 5.3.1) on the second dataset with a share of analytic and numeric surfaces of approximately 75% against 25%, considering the total number of faces in the dataset. To expand beyond the these dataset, additional CAD models with a share of numeric surfaces over 50% up to 90% were considered (Figure 66). The symmetry detection process was applied to example cases using the computational environment, and the corresponding APOS were detected accurately, as illustrated in Figure 66. Even when the CAD models were primarily composed of numeric surfaces, all relevant POSCs in the example cases were detected correctly.



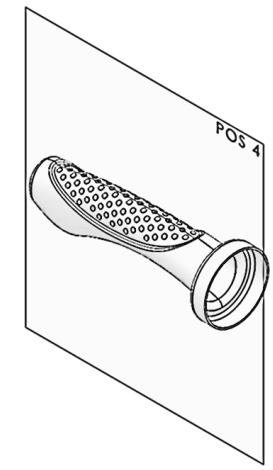
a) Mouse housing
79.3% numeric surfaces
SFI=0.98
GSI=0.91



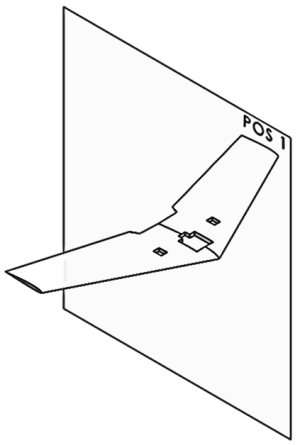
b) Race car front wing
66.2% numeric surfaces
SFI=0.99
GSI=0.98



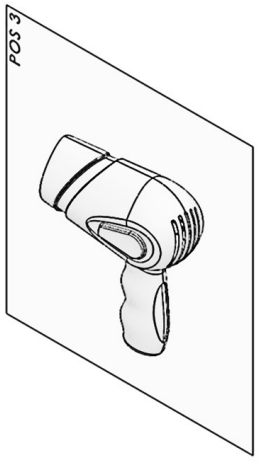
c) Plastic bottel
54.5% numeric surfaces
SFI=1
GSI=1



d) handle cover
82.2% numeric surfaces
SFI=0.97
GSI=0.97



e) Airplane wing
94.7% numeric surfaces
SFI=1
GSI=1



f) Hairdryer
94.3% numeric surfaces
SFI=0.91
GSI=0.94

Figure 66. Examples of CAD models with predominantly numeric surfaces

Another aspect is the usefulness of the CASD method beyond the tested CAD model formats (native Solidworks, Parasolid, and STEP). For this purpose, several CAD models in ACIS and IGES file format taken from the first dataset have been additionally subjected to symmetry detection through the computational environment (Figure 67). The analysed CAD models are reflectional symmetric (Figure 67, a – d), reflectional-axisymmetric, (Figure 67, e), and axisymmetric (Figure 67, f – g). The results reveal that symmetry can be detected in CAD models represented by the ACIS and IGES formats.

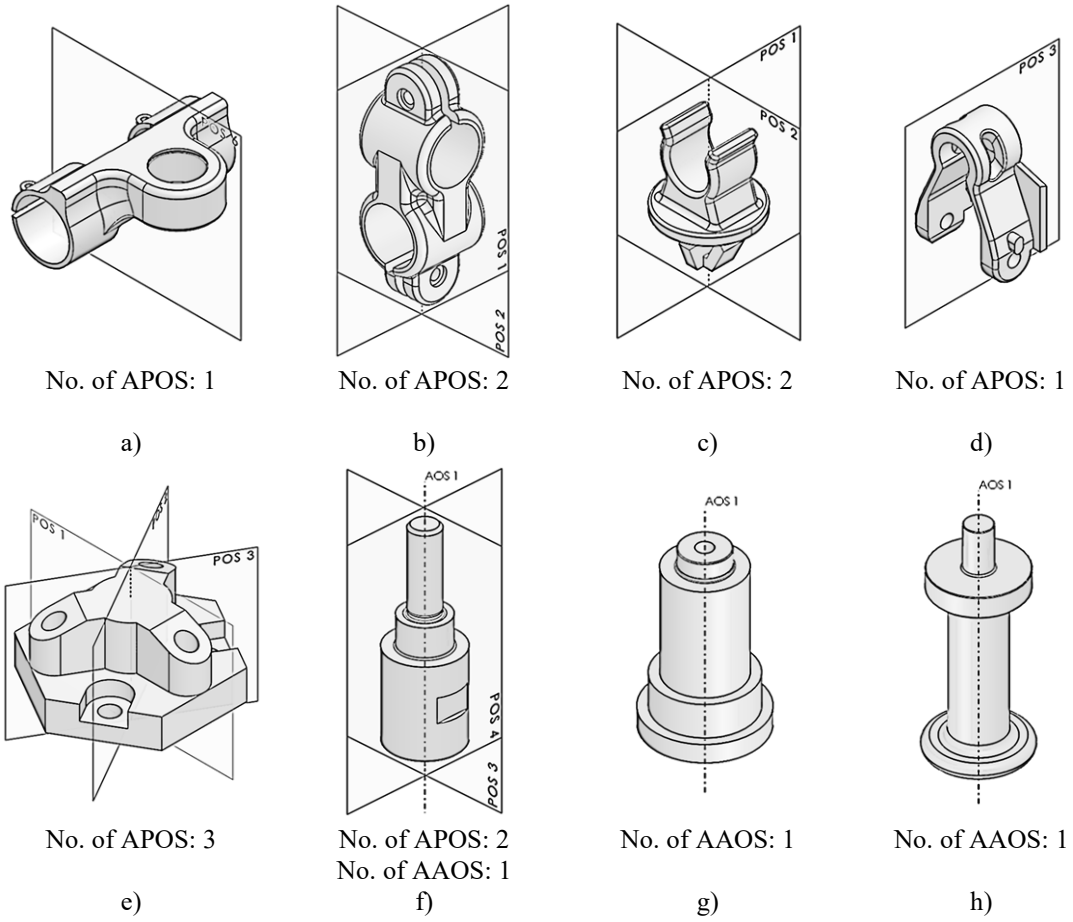
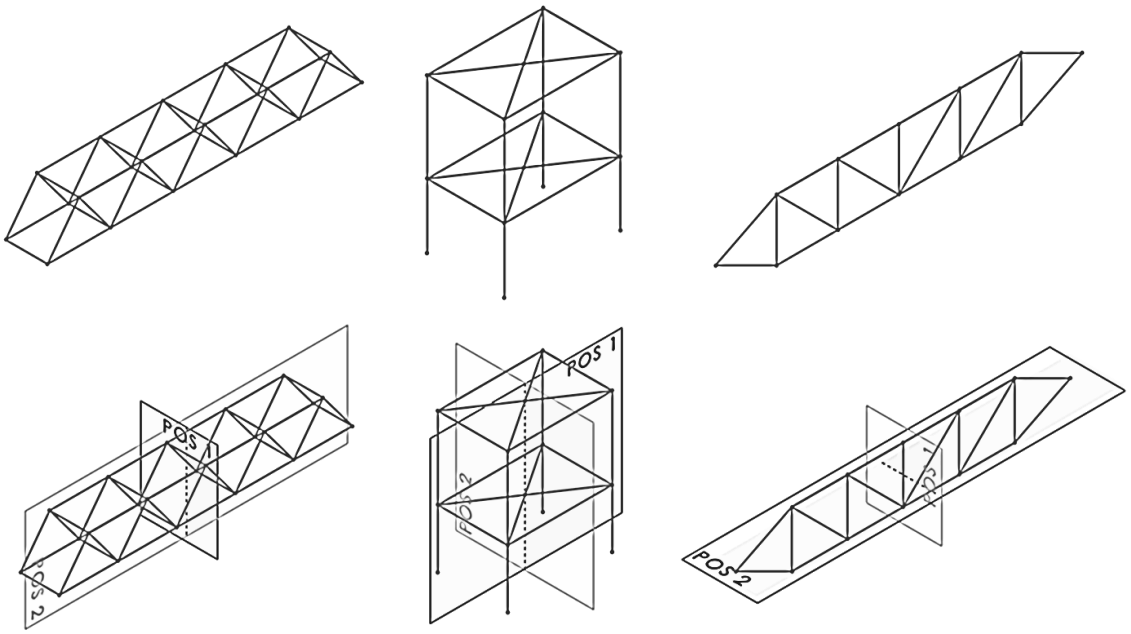


Figure 67. Examples of 3D CAD models in ACIS and IGES file format

Apart from B-rep CAD models, the usefulness of the CASD method can also be observed generally for its application in 3D digital objects. For that purpose, the steps of the proposed CASD method could be modified to work beyond B-rep CAD models. Just like with CAD models, analysing and interpreting the geometry of 3D digital objects is essential. Previous CASD studies have also utilised principal component analysis to generate POSCs and AOSCs [29,30,57,60]. However, generating POSCs from face pairs and AOSCs from single faces presents challenges when working with other 3D digital objects, as these steps rely heavily on the specific definition of B-rep CAD models. Nevertheless, trimming duplicate and unsuitable candidates significantly distanced from the COG can still be useful for other 3D digital objects. Evaluating candidates based on vector calculus can also be adapted to other 3D digital objects containing position and surface information. Mesh models are a prime example of such digital objects. Finally, the visualisation of APOS and AAOS is a common practice in many CASD studies.

To explore the CASD method's usefulness in relation to other 3D digital objects, the extension of the method to grid or cable-strut structures has been investigated. These types of structures are frequently utilised in mechanical engineering for cranes, as well as in civil engineering for bridges. The example cases analysed involved exact global reflectional symmetric grid or cable-strut structures (axisymmetry is not applicable for such structures), as illustrated in Figure 68. Thus, the input for the CASD method is yet a point-line model.



a) b) c)

Figure 68. Examples of other 3D digital objects (cable-strut structures)

In the B-rep analysis step, the inputs are now lines and points instead of faces and edges. This means that specific properties are different, such as replacing the face centroid with the line midpoint, the surface area with the line length, and the normal vector with the line direction vector. POSCs cannot be generated from the PAOI as lines have no mass properties, but they can be generated from pairs of lines of equal length. The position of a POSC is the midpoint computed between midpoints of two identical lines, while the POSC's normal vector is computed based on the arrangement between two equal lines (parallel, coaxial, or arbitrarily oriented). In the trimming step, only duplicate POSCs need to be removed since unsuitable candidates are relevant for partial symmetry, which is not considered. The evaluation step still utilises vector calculus. However, an adjustment is needed for the SFI since faces are no longer the input. The SFI becomes the

symmetrical lines index (SLI) and is used to detect exact global symmetry. It is defined as follows:

$$\mathrm{SLI} = \frac{n_{\mathrm{LP}} + n_{\mathrm{SL}}}{n_{\mathrm{LI}}}, \tag{102}$$

where n_{LP} is the number of symmetrical line pairs, n_{SL} is the number of symmetrical stand-alone lines, and n_{LI} is the total number of lines in the input models. SLI ranges between [0,1]. Figure 68 shows the results of symmetry detection in the example cases. In the first two cases (Figure 68, a and b), all APOSs were accurately detected. However, in the third case (Figure 68, c), one false positive APOS was detected due to the midpoints of diagonal lines being coincident with the plane, while their endpoints were not symmetric. Thus, to improve the CASD method for point-line input models and eliminate false positive results, endpoints of lines should also be included in the evaluation step. The CASD method needs certain modifications to detect symmetry in other 3D digital objects, particularly point-line input models. Additionally, the method could be useful for partially reflectional symmetric cable-strut structures, with partial symmetry measured using the GSIE from Equation (100).

5.4 Improvement of the CASD method

Based on the validation results, this section presents an enhancement to the initial CASD method proposed in Chapter 3. As already outlined, in some instances described through the validation, the proposed CASD method may fail to detect symmetric face pairs or stand-alone faces in the 3D CAD model. While this failure is localised and does not affect symmetry detection at the global level, the CASD method could benefit from an improvement by subjecting the undetected symmetry faces to a third evaluation step. The third evaluation step could be conducted by default for all faces with undetected symmetry or be an additional command in the Symmetry Detector GUI, empowering engineers to determine whether specific faces should be evaluated.

The fundamental principle of this third evaluation step is to represent faces with sampled points, which offer a more comprehensive description of their shape, instead of the unique point representation (centroid or its projection onto the face). The flowchart of the third evaluation step is displayed in Figure 69. First, the remaining faces with undetected symmetry are subjected to point sampling (Figure 70). Its position and normal vector

define each sampled point. The face's sampled points are used as input to estimate if a single face is self-symmetric or if a face pair is symmetric with respect to a POSC. To detect symmetry within $\varepsilon=10^{-6}$ m, it is crucial to obtain uniformly sampling of faces. Uniformity entails dividing the face into rectangular patches with equal-length sides while ensuring that the shape and size of quad patches are as similar as possible. Uniform means that the face is divided into rectangular patches with sides of equal lengths, while the shape and size of quad patches should be the same as much as possible. However, uniform sampling of the parameter domain can result in non-uniform sampling of the face itself since the mapping from the parameter domain to the face can be highly distortive [134]. Hence, to achieve uniform sampling of the face, a non-uniform sampling of the parameter domain needs to be conducted.

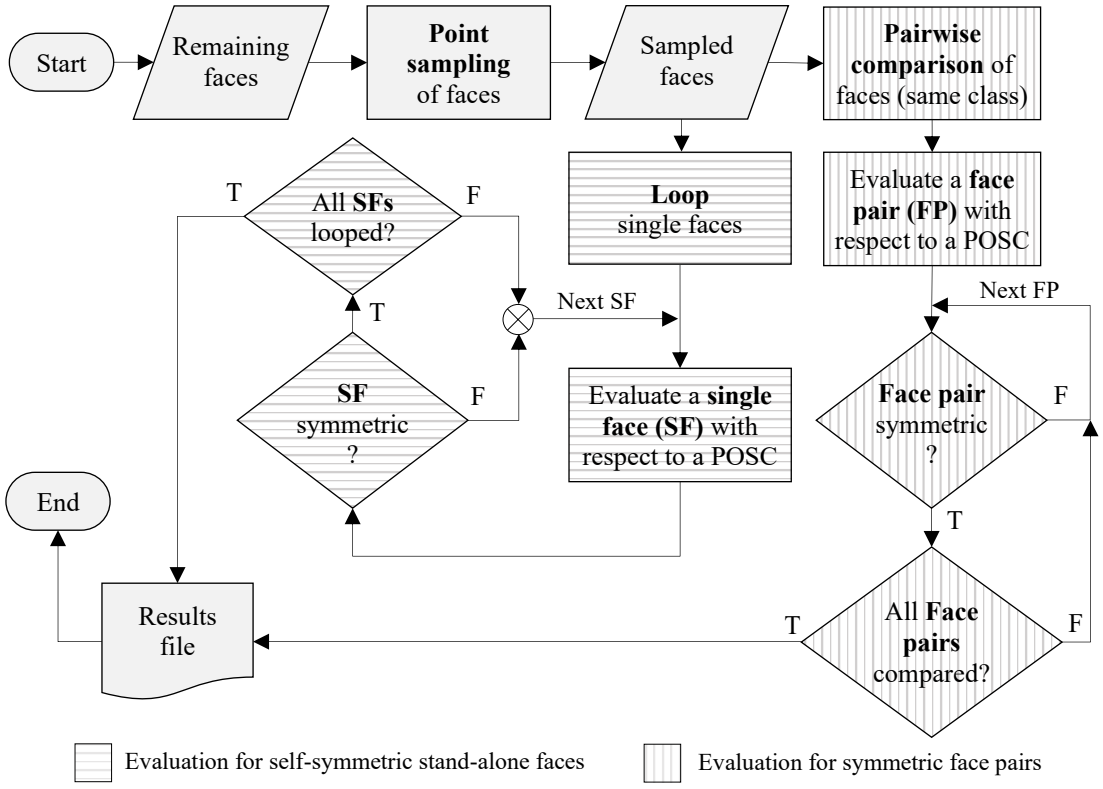


Figure 69. Flowchart of the second evaluation step for remaining faces

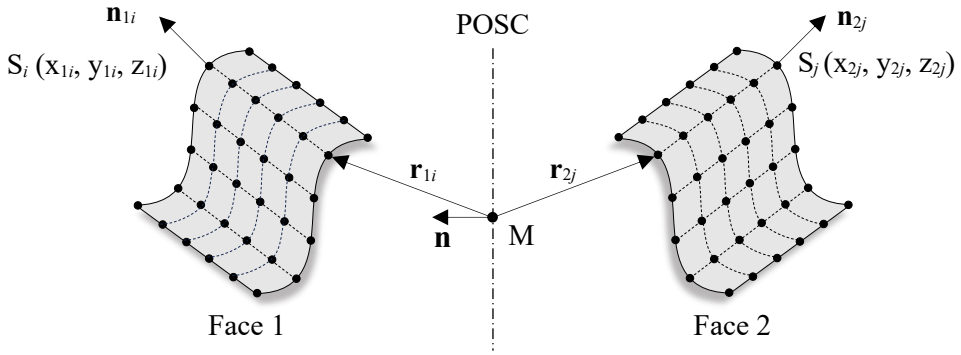
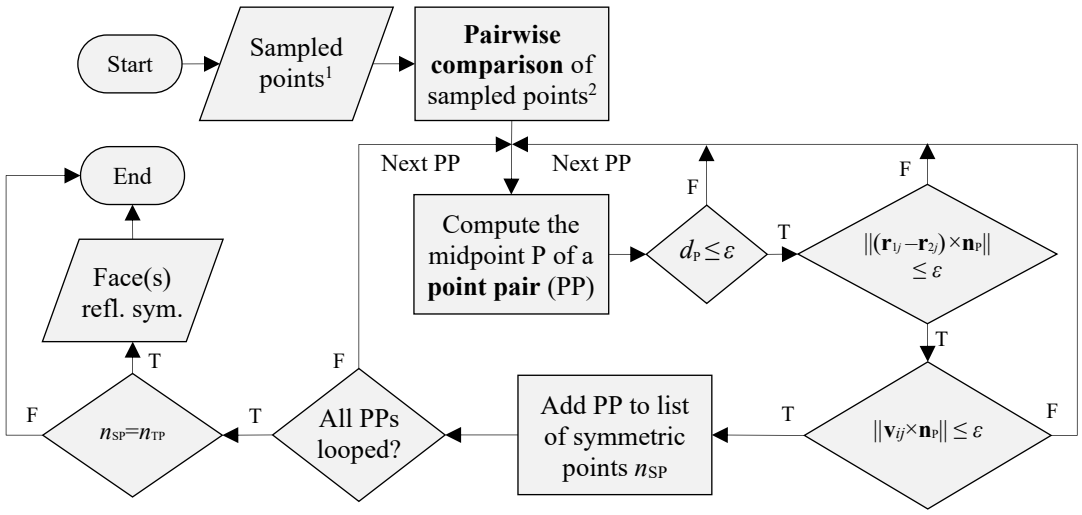


Figure 70. Uniform sampled faces

The existing sampling methods often rely on re-parameterisation and the curvature of the surface [134,149–151]. The sampling method reported in [134] proposed an optimality criterion for re-parameterising Bezier surfaces by measuring their quality using deviation minimisation from an ideal uniform parameterisation. The research in [149] introduced an approach for sampling NURBS based on the mean curvature parametrisation. Another sampling method presented in [150] proposes to create points on NURBS by ranking according to their complexity (determined by the mean Gaussian curvature and the size of the patches), and several points were distributed according to the rank. The study in [151] proposed a similar method, dividing NURBS surfaces into patches based on knot vectors, ranking patches by size, and then sampling points accordingly (patches with higher ranks received more points proportional to their size).

The evaluation step flowchart for reflection symmetry to detect stand-alone faces or face pairs is presented in Figure 71. When evaluating a stand-alone face for reflectional self-symmetry, all its points are pairwise compared. On the other hand, when evaluating a face pair for reflectional symmetry, each point from the first face is compared with those from the second face. In both cases, the points are evaluated with respect to a POSC. To do so, first the midpoint P between a point pair is computed. Then, the point-to-plane distance d_P between the midpoint P and the POSC is computed analogically to Equation (80). If the condition $d_P \leq \varepsilon$ is satisfied, the position vectors $\mathbf{r}_i$ and $\mathbf{r}_j$ of the i-th and j-th ($i \neq j$) point is computed for each point pair (as illustrated in Figure 70). The initial point of each position vector is the same (point M on the POSC), while the terminal point varies depending on the sample point on the face.



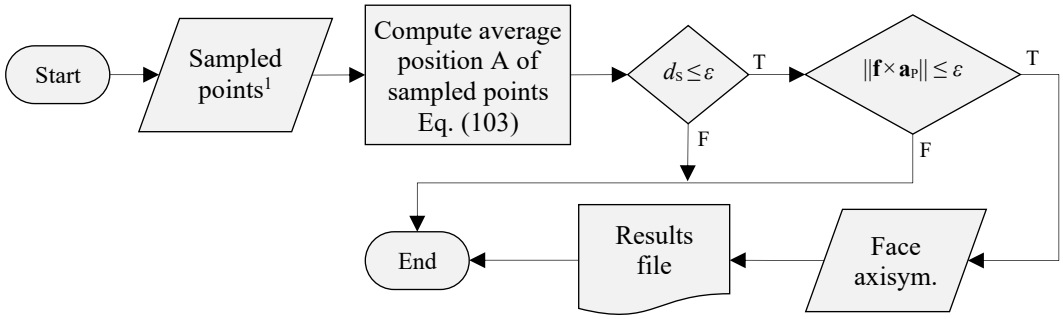
[1] of a stand-alone face or a face pair

[2] for a stand-alone face, its points are compared pairwise; for a face pair, the points of the first face are compared with those of the second face.

Figure 71. Flowchart of the evaluation step of sampled points with respect to a POSC

To evaluate reflectional symmetry, the position vectors are subtracted for each point pair, and the cross-product length is computed between the subtracted position vectors and the normal vector to the POSC. If the length of this cross-product vector is zero, the normal vectors at the sampled points on the face are computed. Depending on the arrangement between two normal vectors (parallel, coplanar, or arbitrarily angled), the resultant vector **v** is computed (as discussed in Section 3.5). If the length of the cross-product between the resultant vector **v** and the POSC normal $\mathbf{n}_P$ is below ε, then the points are considered symmetric, and the procedure is repeated for the next pair of points. Finally, the face is considered symmetric if the number of symmetric points n_{SP} equals number of total sampled points n_{TP}, i.e., $n_{SP}=n_{TP}$.

A stand-alone face is evaluated for axisymmetry utilising the flowchart illustrated in Figure 72. To conduct this assessment, uniformly sampled points on the face (as illustrated in Figure 73) can be employed, using one of the sampling methods previously mentioned [134,149–151].



[1] of a stand-alone face

Figure 72. Flowchart of the evaluation of sampled points with respect to an AOSC

First, the average position $\mathbf{S}(x_S,y_S,z_S)$ of the sampled points on the face are calculated through the following equations:

$$x_S = \frac{1}{n_{TP}} \sum_{i=1}^{n_{TP}} x_{S_i} \qquad y_S = \frac{1}{n_{TP}} \sum_{i=1}^{n_{TP}} y_{S_i} \qquad z_S = \frac{1}{n_{TP}} \sum_{i=1}^{n_{TP}} z_{S_i} \,. \tag{103}$$

To qualify as axisymmetric, the average position **S** of the sampled points needs to be coincident with the APOS. This is determined by computing the point-to-line distance d_S analogical to Equation (81). If the point-to-line distance d_S is below the computational error ε, then the cross-product length between the face's axis vector **f** and the AOSC's orientation vector $\mathbf{a}_P$ is computed and must be below ε.

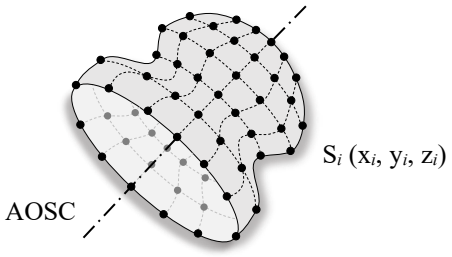


Figure 73. A uniformly sampled axisymmetric face

The third evaluation step offers a potential upgrade to the original CASD method introduced in Chapter 3. However, implementing this upgrade would require an increase in computational effort due to the larger number of input points. Furthermore, it is important to validate the proposed upgrade in future to determine any potential issues.

5.5 Implications of research findings

The proposed CASD method and its implementation in the form of the developed computational environment, alongside with the findings from the validation, have various implications, which are discussed in the context of research and practice.

5.5.1 Implications for research

It is worth noting that the CASD method in this thesis differs from previous research that primarily employed analytical geometry for symmetry detection [16,42]. While previous studies addressed spline surfaces, they could have been more effective in detecting exact symmetry for CAD models with spline surfaces [23,42]. Instead, they proposed approximate solutions for computing symmetry properties of topological entities. The proposed CASD method can also be applied to exact symmetry and numerical geometry, including spline, surface of revolution, extruded surface, offset surface, blended surface, and more. Furthermore, this research presents improvements to the initially introduced CASD method for conducting symmetry detection in CAD models represented predominantly by numeric surfaces (Section 5.4). Therefore, this doctoral thesis may inspire researchers to conduct further CASD studies for even more complex parts consisting predominantly of numeric geometry, such as aeroplane wings, car bodies, turbine blades, and other products with complex aesthetic shapes.

The presented method offers a unique approach to matching pairs of topological entities using similarity measures, such as Cosine similarity. Indeed, this method could be employed in prior symmetry detection studies [23], which utilised a loop-based pairing approach with loop area and the number of edges as pair-matching criteria. In addition, researchers may find value in using similarity measures from this doctoral thesis in other research topics, such as recognising similar 3D CAD models to reuse existing design solutions [78]. This research outlines the importance of merging partitioned topological entities as a pre-step for symmetry detection, albeit in a simplified manner, by leveraging the CAD system's automatic merging of partitioned periodic faces during model import. This could spur researchers to develop researchers to develop novel topology merging methods for B-rep CAD models.

Previous studies in [16,63] detected symmetry by generating candidates based on local symmetry properties of topological entities and propagating them globally over the B-rep. Another study utilised a ranking system based on loop area and number of edges to detect symmetry [23,42]. However, these studies did not include a distance function to evaluate candidates with respect to geometry. In contrast, the proposed CASD method within this doctoral thesis evaluates each candidate with respect to the geometry using a procedure based on vector calculus. Specifically, the inputs for the vector calculus are faces, each represented by a unique point (face centroid or its projection onto the face) and a vector (axis or normal vector) at this point. This approach may also be applicable to other CASD studies utilising mesh models as input, as seen in reference [47].

This study introduces two symmetry measures (SFI and GSI) that can differentiate between exact, partial, and non-symmetric B-rep CAD models. These measures can potentially be applied to existing CASD studies [16,42] to measure partial symmetry. While this research focused solely on single-part manifold CAD models, promising results were obtained during validation (Subsection 5.3.3) for other types of input CAD models as well, including single-part multi-body CAD models, non-manifold CAD models, assembly CAD models, and CAD models primarily composed of numeric surfaces, as well as other input CAD model formats such as IGES and ACIS. Additionally, the proposed method shows promise in being extendable and applicable to other digital objects, such as cable-strut structures.

Finally, it is important to note that the collected dataset of over 1000 different 3D CAD models in STEP format represents a valuable repository that could be used to validate previous CASD studies [16,42,66] or used in other research fields such as feature recognition [90].

5.5.2 Implications for practice

There are several implications of this study related to practice. One of them is that it provides a computational environment for computer-aided symmetry detection. This enables engineers to analyse symmetry within the CAD model during the design process or downstream activities such as CAE, CAPP, CAM, etc. While the computational environment is specific to a particular CAD system, the CASD method is general and can be extended to other CAD systems, as discussed in Section 4.2.

Apart from a single CAD model, the developed computational environment can automatically analyse a set of CAD models located at a specific location, making it applicable for symmetry detection in large CAD repositories. The detected type of symmetry, whether global exact or partial reflectional symmetry, global exact or partial axisymmetry, or even non-symmetry, can be attributed to CAD models and used as a search criterion in a CAD repository or a product data management system. The computational environment also allows engineers to postprocess the results by visualising and colouring the faces that are symmetric or disturb the symmetry related to the corresponding APOS or AAOS.

In addition, the computational environment creates an external .txt results file associated with the 3D CAD model, which provides a solution for permanently storing symmetry detection results. In this way, the symmetry information can be retained for neutral and kernel CAD formats (STEP, IGES, ACIS, and Parasolid) that do not support the storage of reference geometry. An additional advantage is that the detected symmetry information for the CAD models is not lost and can be shared among different contributors in the product development process.

5.6 Limitations

The limitations of this study can be observed from the aspect of the CASD method and its implementation, i.e., the computational environment. The input of the proposed symmetry detection method is restricted to 3D CAD models with B-rep. The 3D CAD models are limited to single parts with only one body and manifold geometry, as stated in the research objectives (Subsection 1.2). Although the method theoretically puts no limitations on the type of numeric geometry, the implementation was limited to the numeric surfaces available in the selected CAD system Solidworks (B-spline surface, surface of revolution, offset, extruded, and blend surface). The types of symmetry that are addressed by the proposed CASD method are reflectional symmetry and axisymmetry, although mechanical components may also exhibit other types of symmetry, such as cyclic and translational. Further, the CASD method is restricted to detecting global and partial symmetry and does not support local symmetry detection within the 3D CAD model. From the aspect of implementation, the computational environment is limited to the CAD system used for development (i.e., Solidworks), which

puts certain restrictions on practical application. The considered input CAD models were native (Solidworks), kernel (Parasolid, ACIS), and neutral (STEP, IGES) file formats. The CASD method is also limited to CAD models that need to be free of topological errors (e.g., missing faces), because otherwise, the mass properties of the 3D CAD model cannot be computed, and the candidates from the PAOI cannot be generated.

6 CONCLUSIONS

This final chapter reflects on the aims and research hypothesis, providing a conclusion of the conducted research. At the end, future research directions are outlined.

Symmetry is an essential geometric property often employed in mechanical engineering. To detect symmetry in 3D CAD models, engineers could utilise CASD. While previous CASD studies have focused on CAD models with analytic surfaces, numeric surfaces have not been adequately considered. Furthermore, there is a need for research to propose a measure for detecting partial symmetry. This doctoral thesis tackles these research gaps by presenting a geometry-based CASD method for recognising exact global and partial reflectional and axisymmetry in CAD models with B-rep. The method involves six steps (Chapter 3), including interpretation of B-rep, analysis, generation, trimming, evaluation, and visualisation of POSCs and AOSCs. Importantly, the method is versatile and can work with input CAD models in various formats like STEP, Parasolid, ACIS, IGES, and more. The research novelty lies in the unique representation of each face with a point and a characteristic vector. The method relies on the implicit symmetry detection approach, which implies generating a set of POSCs and AOSCs.

This study introduces a combined approach to generating POSCs and AOSCs, utilising single faces, pairs of similar faces, and PAOI. A unique aspect of this approach is using similarity measures, such as cosine similarity, to identify pairs of similar faces. This offers the advantage of detecting exact global and partial symmetry. Additionally, using PAOI in the context of B-rep CAD models allows for detecting symmetries that align with these axes. After generating candidates, a trimming process is applied to remove duplicate and unsuitable candidates that are far from the COG. While eliminating duplicates is not new, removing unsuitable candidates has not been utilised in B-rep CAD models. This eliminates candidates who are unlikely to become APOS or AAOS, streamlining the evaluation process.

Following the trimming step, each remaining POSC and AOSC undergoes evaluation with respect to the B-rep using a vector calculus procedure that takes the face's unique

points and characteristic vectors as inputs. This vector calculus procedure has not been employed in previous studies. To determine whether the B-rep CAD model is symmetric with respect to a POSC or AOSC, two novel symmetry measures are proposed: the SFI and GSI. The SFI measures the topology symmetry, while the GSI measures the geometry symmetry of the B-rep CAD model. The GSI, in particular, provides a measure for defining the threshold values between exact global, partial symmetry, and non-symmetry. Finally, if evaluation confirms that the corresponding POSC or AOSC also represents APOS or AAOS, it is stored in a result file and visualised in the 3D CAD model.

By introducing a computational environment in Chapter 4 that utilises the Solidworks CAD system API functionalities, the CASD method has been successfully implemented. Furthermore, the research findings indicate that this method can be extended to include two other CAD systems, namely NX and FreeCAD. However, the implementation process across different CAD systems may vary due to the specific types of surfaces and edges they support.

The proposed CASD method and computational environment underwent extensive structural and performance validation, as outlined in Chapter 5. The results of this validation demonstrate the insensitivity of the CASD method in detecting the corresponding APOS or AAOS, even when input 3D CAD models originate from different CAD systems. In addition, any interpretation differences between 3D CAD models from different systems are unlikely to impact the detection of symmetry, as any false negative face pairs or stand-alone faces would be at the local level. Furthermore, the validation confirms that the CASD method is highly accurate in detecting the corresponding APOS or AAOS, all while maintaining a linear time complexity.

The proposed CASD method utilises the geometrical information of the B-rep CAD models, allowing for the detection of exact global and partial reflectional symmetry as well as axisymmetry. Through validation, it has been confirmed that this geometry-based method can successfully detect symmetry in various CAD models containing both analytic and numeric surfaces. Thus, the hypothesis that a geometry-based approach can be used to detect symmetry in such models has been confirmed. Furthermore, the proposal of the CASD method and its implementation through the computational environment confirm the expected scientific contribution of this research.

6.1 Future research directions

While the proposed CASD method represents an improvement in symmetry detection in B-rep CAD models (partial symmetry detection and models with numeric surfaces), there are still avenues for further research. The CASD method in this research is limited to reflectional symmetry and axisymmetry, but mechanical parts may exhibit other types of symmetry, including cyclic, translational, and dihedral. These types of symmetry still need to be solved in the context of B-rep CAD models, and future research could focus on extending the CASD method's capabilities. Additionally, the proposed CASD method can be enhanced by addressing the detection of local symmetry. While local symmetry was addressed in previous CASD studies regarding analytic surfaces, it remains an open topic from the perspective of numeric surfaces. Although the validation confirmed the usefulness of the CASD method beyond example problems, further research is needed related to symmetry detection in 3D CAD models that are compound predominantly or entirely of numeric geometry. To that end, an improvement of the CASD method has been presented (Subsection 5.5), which requires further research and validation. The CASD method shows promising results when it comes to extending it to non-manifold B-rep CAD models, single-part multi-body CAD models, assembly CAD models, various CAD formats (ACIS and IGES), and other 3D digital objects such as cable-strut structures. However, further research is necessary in these fields. Finally, improving the computational environment by developing a stand-alone application would make the CASD independent of the CAD system, which is a possible research area.

REFERENCES

[1] Beruski O, Vidal LN. Algorithms for computer detection of symmetry elements in molecular systems. J Comput Chem. 2014;35(4):290-9. doi: 10.1002/jcc.23493.

[2] Marcellini S. When Brachyury meets Smad1: the evolution of bilateral symmetry during gastrulation. Bioessays. 2006 Apr;28(4):413-20. doi: 10.1002/bies.20387. PMID: 16547957.

[3] Guo Q, Guo F, Shao J. Irregular shape symmetry analysis: theory and application to quantitative galaxy classification. IEEE Trans Pattern Anal Mach Intell. 2010; 32(10):1730-43. doi: 10.1109/TPAMI.2010.13.

[4] Dolgy DV, Kim DS, Kim T, et al. Identities of symmetry for Carlitz q-Bernoulli polynomials arising from q-Volkenborn integral on Zp under symmetry group S3. Advanced Studies in Theoretical Physics. 2014; 8: 737–744. doi: 10.12988/astp.2014.4682

[5] Yang CN. Conceptual beginnings of various symmetries in twentieth century physics. Chinese J Phys 1994; 32: 1437–1446.

[6] Fernandez E. Symmetry: key to nature and natural philosophy. Metascience. 2004; 13: 329–333.

[7] Nikolić V, Radović L, Marković B. Symmetry of "Twins". Symmetry. 2015; 7(1):164-181. doi: 10.3390/sym7010164.

[8] Zingoni A. Insights on the vibration characteristics of double-layer cable nets of D4h symmetry. International Journal of Solids and Structures. 2018; 135:261–273. doi: 10.1016/j.ijsolstr.2017.11.025.

[9] Cao W, Wang C, Chen W. et al. Fully integrated parity–time-symmetric electronics. Nat. Nanotechnol. 2022; 17:262–268. doi: 10.1038/s41565-021-01038-4

[10] Li WH, Zang AM, Kleeman L. Bilateral Symmetry Detection for Real-time Robotics Applications. Int. J. Robot. Res. 2008; 27, 785–814. doi: 10.1177/0278364908092131

[11] Qiu Q, Chen X, Yang C, Feng P. Classification and Effects of Symmetry of Mechanical Structure and Its Application in Design. Symmetry. 2021; 13(4):683. doi: 10.3390/sym13040683

[12] Simmons CH, Phelps N. Manual of Engineering Drawing Technical Product Specification and Documentation to British and International Standards, Fourth Edition, Butterworth-Heinemann, 2012. ISBN: 978-0-08-096652-6

[13] International Standard Organization, ISO 129-1:2004, Technical drawings – Indication of dimensions and tolerances – Part 1: General principles, 2004

[14] Tierney C, Boussuge F, Robinson T, Nolan D, Armstrong C. Efficient symmetry-based decomposition for meshing quasi-axisymmetric assemblies. Computer-Aided Design and Applications. 2019; 16(3):478-495. doi: 10.14733/cadaps.2019.478-495

[15] Suresh K., Sirpotdar A. Automated symmetry exploitation in engineering analysis. Engineering with Computers. 2006; 21: 304–311. doi: 10.1007/s00366-006-0021-2

[16] Li K, Foucault G, Léon J, Trlin M. Fast global and partial reflective symmetry analyses using boundary surfaces of mechanical. 2014; 53:70-89. doi: 10.1016/j.cad.2014.03.005

[17] Deng X, Wang J. Research on the manufacturing of mechanical parts based on the theory of space symmetry group. Academic Journal of Manufacturing Engineering. 2017; 15(1):64-71.

[18] Ma, Z.; Zhang, T.; Liu, F.; Yang, J. Knowledge discovery in design instances of mechanical structure symmetry. Adv. Mech. Eng. 2015, 7, 1–19. doi: 10.1177/1687814015615044.

[19] Giesecke FE, Lockhart S, Goodman M, Johnson C. Technical Drawing with Engineering Graphics; 15th Edition; Pearson Education: Hoboken, NJ, USA, 2016.

[20] Li C, Li M, GaoS.: Multi-scale symmetry detection of CAD models, Computer-Aided Design and Applications, 2018; 16(1): 50-66. doi: 10.14733/cadaps.2019.50-66

[21] Mitra NJ, Guibas L, Pauly M. Partial and approximate symmetry detection for 3d geometry. ACM Transactions on Graphics (Proc. SIGGRAPH) 2006; 25(3): 560–568. doi: 10.1145/1179352.1141924

[22] Mitra N, Pauly M, Wand M, Ceylan D. Symmetry in 3D Geometry: Extraction and Applications. Comput. Graph. Forum. 2013; 32: 1–23. doi: 10.1111/cgf.12010.

[23] Tate SJ. Symmetry and Shape Analysis for Assembly-Oriented CAD. Doctoral Thesis, Cranfield University, Bedford, UK, May 2000.

[24] Buric M , Brcic M , Bojcetic N, Skec S. Computer-Aided Detection of Exact Reflection and Axisymmetry in B-rep CAD Models. Computer-Aided Design & Applications, 2023; 20(5): 884-897. doi: 10.14733/cadaps.2023.884-897

[25] Jiang J, Chen Z, He K. A feature-based method of rapidly detecting global exact symmetries in CAD models. Computer Aided Design 2013; 45: 1081–1094. doi: 10.1016/j.cad.2013.04.005.

[26] Solidworks 2021 Help - Symmetry Check Utility, https://help.solidworks.com/2021/english/SolidWorks/sldworks/HIDD_SYMCHECK_MAINPAGE.htm (accessed: 20.06.2023)

[27] Stroud I, Nagy H. Solid modelling and CAD systems: How to survive a CAD system, Springer, London, 2011. doi: 10.1007/978-0-85729-259-9

[28] Stroud I, Boundary Representation Modelling Techniques, Springer, London, 2010. doi: 10.1007/978-1-84628-616-2

[29] Li B, Johan H, Ye Y, Lu Y. Efficient view-based 3d reflection symmetry detection. SIGGRAPH Asia 2014 Creative Shape Modeling and Design (SA '14). , New York, NY, USA, 3-6.12.2014, Association for Computing Machinery, pp 1–8. doi: 10.1145/2669043.2669045

[30] Li B, Johan H, Ye Y, Lu Y. Efficient 3D reflection symmetry detection: A view-based approach, Graphical Models, 2016; 83: 2-14. doi: 10.1016/j.gmod.2015.09.003

[31] Piegl L, Tiller W. The NURBS Book, 2nd edition, Springer-Verlag, New York, NY, USA, 1996.

[32] Blessing L, Chakrabarti A. DRM, a Design Research Methodology. Springer-Verlag, London, 2009. doi: 10.1007/978-1-84882-587-1.

[33] Pedersen K, Emblemsvag J, Reid Bailey, Allen JK, Mistree F. Validating Design Methods & Research: The Validation Square. 2000 ASME Design Engineering Technical Conferences. 2000.

[34] Nagar R, Raman S. 3DSymm: Robust and Accurate 3D Reflection Symmetry Detection. Pattern Recognition. 2020; 107:107483. doi: 10.1016/j.patcog.2020.107483.

[35] Xue F, Chen K, Lu W. Architectural Symmetry Detection from 3D Urban Point Clouds: A Derivative-Free Optimization (DFO) Approach. Advances in Informatics and Computing in Civil and Construction Engineering: Proceedings of the 35th CIB W78 2018 Conference: IT in Design, Construction, and Management; Springer: Berlin/Heidelberg, Germany, 2019; pp. 513–519. doi: 10.1007/978-3-030-00220-6_61.

[36] Alcázara JG, Hermosoa C, Muntinghb G. Symmetry Detection of Rational Space Curves from their Curvature and Torsion. Comput. Aided Geom. Des. 2015; 33: 51–65. doi: 10.1016/j.cagd.2015.01.003.

[37] Fotouhi J, Taylor G, Unberath M, Johnson A, Lee S C, Osgood G, Armand M, Navab N. Exploring Partial Intrinsic and Extrinsic Symmetry in 3D Medical Imaging. Medical Image Analysis. 2020; 72: 102127. doi: 10.1016/j.media.2021.102127

[38] Ji P, Liu X. A fast and efficient 3D reflection symmetry detector based on neural networks, Multimedia Tools and Applications, 2019; 78: 35471–35492. doi: 10.1007/s11042-019-08043-9

[39] Thrun S, Wegbreit B. Shape from symmetry. Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2 (ICCV '05). IEEE Computer Society, USA, 1824–1831. doi: 10.1109/ICCV.2005.221

[40] Hruda, L.; Kolingerová, I.; Váša, L.: Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure, The Visual Computer, 38(24), 2022, 555–571. https://doi.org/10.1007/s00371-020-02034-w

[41] Korman S, Litman R, Avidan S, Bronstein A. Probably Approximately Symmetric: Fast Rigid Symmetry Detection with Global Guarantees. Computer Graphics Forum. 2014;34(1):2-13. doi:10.1111/cgf.12454.

[42] Tate S, Jared G. Recognising symmetry in solid models, 2003; 35(7): 673–692. doi: 10.1016/S0010-4485(02)00093-3

[43] Chen Y, Linzi F, Feng J. Automatic and Exact Symmetry Recognition of Structures Exhibiting High-Order Symmetries, Journal of Computing in Civil Engineering. 32(2): 04018002. doi: 10.1061/(ASCE)CP.1943-5487.0000743

[44] Dang Q, Morin G, Mouysset S. Symmetry and Fourier Descriptor: A Hybrid Feature for NURBS based B-Rep Models Retrieval. Conference: Eurographics Workshop on 3D Object Retrieval, 2014; 45-52. doi: 10.2312/3dor.20141049

[45] Dang Q, Mouysset S, Morin G. Symmetry-Based Alignment for 3D Model Retrieval. In Proceedings of the 12th International Workshop on Content-Based Multimedia Indexing (CBMI), Klagenfurt, Austria, 18–20.06.2014. doi:10.1109/CBMI.2014.6849816.

[46] Sipiran I, Gregor R, Schreck T. Approximate Symmetry Detection in Partial 3D Meshes. Computer Graphics Forum. 2014; 33(7): 131-140. doi:10.1111/cgf.12481.

[47] Hruda L, Dvorák J. Estimating Approximate Plane of Symmetry of 3D Triangle Meshes. Proceedings Central European Seminar on Computer Graphics, Smolenice, Slovakia, 2017.

[48] Kazhdan M, Chazelle B, Dobkin D, Funkhouser T, Rusinkiewicz S. A Reflective Symmetry Descriptor for 3D Models. Algorithmica 2004; 38(1): 201–225. doi: 10.1007/s00453-003-1050-5.

[49] Chang M, Park SC. Reverse engineering of a symmetric object. Computers & Industrial Engineering, 2008; 55(2): 311-320. doi: 10.1016/j.cie.2007.12.015

[50] Sun C, Sherrah J. 3D symmetry detection using the extended Gaussian image. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997; 19(2): 164-168. doi: 10.1109/34.574800.

[51] Martinet A, Soler C, Holzschuch N, Sillion FX. Accurate Detection of Symmetries in 3D Shapes. ACM Transactions on Graphics, Association for Computing Machinery, 2006; 25(2): 439 - 464. doi: 10.1145/1138450.1138462

[52] Kakarala R, Kaliamoorthi P, Premachandran V. Three-Dimensional Bilateral Symmetry Plane Estimation in the Phase Domain, IEEE Conference on Computer Vision and Pattern Recognition, 23-28.06.2013 Portland, OR, USA, 2013. p. 249-256, doi: 10.1109/CVPR.2013.39.

[53] Podolak J, Shilane P, Golovinskiy A, Rusinkiewicz S, Funkhouser T. A planar-reflective symmetry transform for 3D shapes. ACM Transactions on Graphics. 2006; 25(3): 549–559. doi: 10.1145/1141911.1141923

[54] Raviv D, Bronstein AM, Bronstein MM., Kimmel R. Full and Partial Symmetries of Non-rigid Shapes. International Journal of Computer Vision. 2010; 89: 18–39. doi:10.1007/s11263-010-0320-3

[55] Zingoni A. Symmetry recognition in group-theoretic computational schemes for complex structural systems, Computers and Structures, 2012; 94-95: 34-44. doi: 10.1016/j.compstruc.2011.12.004

[56] Gothandaraman R, Jha R, Muthuswamy S. Reflectional and rotational symmetry detection of CAD models based on point cloud processing. In Proceedings of the IEEE 4th Conference on Information & Communication Technology (CICT), Chennai, India, 3–5 December 2020. doi:10.1109/CICT51604.2020.9312109

[57] Stephenson M, Clark A, Green R. Novel methods for reflective symmetry detection in scanned 3D models. International Conference on Image and Vision Computing, Auckland (IVCNZ), New Zealand, 23-24.11.2015, IEEE, 2015. p.1-6. doi:10.1109/IVCNZ.2015.7761555.

[58] Lipman Y, Xiaobai C, Daubechies I, Funkhouser T. Symmetry Factored Embedding And Distance. ACM Transactions on Graphics (TOG). 2010; 29(4): 103. doi: 10.1145/1778765.1778840.

[59] Minovic P, Ishikawa S, Kato K. Symmetry identification of a 3-D object represented by octree. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993; 15(5): 507-514. doi: 10.1109/34.211472.

[60] Tedjokusumo J, Leow, WK. Normalization and Alignment of 3D Objects Based on Bilateral Symmetry Planes. Cham, TJ., Cai, J., Dorai, C., Rajan, D., Chua, TS., Chia, LT. (eds) Advances in Multimedia Modeling. MMM 2007. Lecture Notes in Computer Science, vol 4351. Springer, Berlin, Heidelberg, 2006. doi: 10.1007/978-3-540-69423-6_8

[61] Yeh YT, Měch R. Detecting Symmetries and Curvilinear Arrangements in Vector Art. Computer Graphics Forum, 2009; 28: 707-716. doi: 10.1111/j.1467-8659.2009.01411.x

[62] Hruda L. Symmetry Detection in Geometric Models [Master Thesis]. [Plzen]: University of West Bohemia; 2018

[63] Li K. Shape Analysis of B-Rep CAD Models to Extract Partial and Global Symmetries. PhD Thesis, University Grenoble, Saint-Martin-d'Hères, France, November 2011.

[64] Tayangkanon T, Sompagdee P, Li X. 3D Model Compression over ASCII Encoded Using Rotational and Reflective Symmetry, 2018 10th International Conference on Knowledge and Smart Technology (KST), 31.1.-3.2.2018, p.53-58. doi: 10.1109/KST.2018.8426067

[65] Li M, Langbein F, Martin R. Detecting design intent in approximate CAD models using symmetry. Computer-Aided Design, 2010; 42: 183-201. doi: 10.1016/j.cad.2009.10.001

[66] Boussuge F, Tierney CM, Robinson TT, Armstrong CG. Symmetry-based decomposition for meshing quasi-axisymmetric components. Procedia Engineering, 2017; 203: 375-387. doi: 10.1016/j.proeng.2017.09.812.

[67] Li M, Langbein FC, Martin RR. Detecting approximate incomplete symmetries in discrete point sets. Proceedings of the 2007 ACM symposium on Solid and physical modelling (SPM '07). Association for Computing Machinery, New York, NY, USA, 2007, p.335–340. doi: 10.1145/1236246.1236294

[68] Li M, Langbein FC, Martinn RR. Detecting approximate symmetries of discrete point subsets, Computer-Aided Design, 2008; 40(1): 76-93. doi: 10.1016/j.cad.2007.06.007.

[69] Li M, Langbein F, Ralph M. Constructing Regularity Feature Trees for Solid Models. Lecture Notes in Computer Science. 2006; p.4077. 267-286. doi: 10.1007/11802914_19.

[70] Jiang, J, Gong, Q, Chen, Z. He, K.; Ruang, R.: A Feature-based Method of Rapidly Detecting Global Symmetries of Static Assembly CAD Models, Jisuanji Fuzhu Sheji Yu Tuxingxue Xuebao/Journal of Computer-Aided Design and Computer Graphics, 29(5), 2017, 950-957.

[71] Jiang J, Chen Z, He K. A feature-based approach for detecting global symmetries in CAD models with free-form surfaces, 13th International Conference on Computer-Aided Design and Computer Graphics, 2013, doi: 10.1109/CADGraphics.2013.58

[72] Marola, G. On the Detection of the Axes of Symmetry of Symmetric and Almost Symmetric Planar Images. IEEE Trans. Pattern Anal. Mach. Intell. 1989, 11, 104–108. doi: 10.1109/34.23119.

[73] Zabrodsky H, Peleg S. Avnir, D. Symmetry as a continuous feature. IEEE Trans. Pattern Anal. Mach. Intell. 1995; 17: 1154–1166. doi: 10.1109/34.476508.

[74] Kulkarni, P.; Dutta, D.; Saigal, R. An Investigation of Techniques for Asymmetry Rectification, J. Mech. Des. 1995; 117: 620–626. doi: 10.1115/1.2826730.

[75] Fedotova S, Kushnir O, Seredin O. Comparison of Binary Images based on Jaccard Measure using Symmetry Information. 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications. 27-29.2.2020., Valletta, Malta, 2020; p.398-404. doi: 10.5220/0008989803980404.

[76] Chaudhari A, Bilionis I, Panchal J. Similarity in Engineering Design: A Knowledge-Based Approach, Proceedings of the ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 7: 31st International Conference on Design Theory and Methodology. Anaheim, California, USA. 18–21.08.2019. ASME. 2019, doi: 10.1115/DETC2019-98272

[77] Cardone A, Gupta SK. Deshmukh A, Karnik M. Machining feature-based similarity assessment algorithms for prismatic machined parts, Computer-Aided Design, 2006; 38(9): 954-972, doi: 10.1016/j.cad.2006.08.001

[78] Zehtaban L, Elazhary O, Roller R. A framework for similarity recognition of CAD models, Journal of Computational Design and Engineering, 2016; 3(3): 274-285, doi: 10.1016/j.jcde.2016.04.002

[79] Wang J, Yan W, Huang C. Surface shape-based clustering for B-rep models, Multimedia Tools and Applications, 2020; 7: 25747–25761, doi: 10.1007/s11042-020-09252-3

[80] Nandy A, Dong A, Goucher-Lambert K. Evaluating Quantitative Measures for Assessing Functional Similarity in Engineering Design. ASME. Journal of Mechanical Design. 2022; 144(3): 031401. doi: 10.1115/1.4052302

[81] Zehtaban L, Elazhary O, Roller R. A framework for similarity recognition of CAD models. J. Comput. Des. Eng. 2016; 3: 274–285. doi: 10.1016/j.jcde.2016.04.002.

[82] Kim S, Kato I, Zhang X. Comparative Analysis of Binary Similarity Measures for Compound Identification in Mass Spectrometry-Based Metabolomics. Metabolites. 2022; 12(8):694. doi:10.3390/metabo12080694

[83] Seifoddini H, Djassemi M. Merits of the production volume based similarity coefficient in machine cell formation, J. Manuf. Syst. 1995; 14: 35–44. doi: 10.1016/0278-6125(95)98899-H.

[84] Lucchese L. Frequency domain classification of cyclic and dihedral symmetries of finite 2-D patterns, Pattern Recognition, 2004;37(12):2263-2280. doi: 10.1016/j.patcog.2004.04.012.

[85] Zou HL, Lee YT. Skewed rotational symmetry detection from a 2D line drawing of a 3D polyhedral object. Computer-Aided Design. 2006; 38:1224-1232. doi: 10.1016/j.cad.2006.08.003.

[86] Pauly M, Mitra NJ, Wallner J, Pottmann H, Guibas LJ. Discovering structural regularity in 3D geometry. ACM Trans. Graph. 2008;27(3):1–11. doi:10.1145/1360612.1360642

[87] Xu K, Zhang H, Tagliasacchi A, Liu L, Li G, Meng M, Xiong Y. Partial intrinsic reflectional symmetry of 3D shapes. ACM Trans. Graph. 2009;28(5):1–10. doi: 10.1145/1618452.1618484

[88] Schiebener D, Schmidt A, Vahrenkamp N, Asfour T. Heuristic 3D object shape completion based on symmetry and scene context,2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea, 2016, p. 74-81, doi:10.1109/IROS.2016.7759037.

[89] Gnutti A, Guerrini F, Leonardi R. Combining Appearance and Gradient Information for Image Symmetry Detection. IEEE Transactions On Image Processing, 2021;30:5708-5723. doi:10.1109/TIP.2021.3085202

[90] Venu B, Komma VR, Srivastava D. STEP-based Feature Recognition System for B-spline Surface Features. Int. J. Autom. Comput. 2018; 15: 500–512 doi: 10.1007/s11633-018-1116-0

[91] Burić M, Marjanović D. A tool for idealisation of CAD models, Proceedings of the DESIGN 2018 15th International Design Conference, 2018, p.205-214 doi: 10.21278/idc.2018.0367

[92] Raffaeli R, Cicconi P, Germani M. Automation of drafting execution by schemes definitions and feature recognition. Computer-Aided Design and Applications. 2015; 13: 1-12. doi: 10.1080/16864360.2015.1131538.

[93] CATIA V5 API help documentation – CAA V5 Encyclopedia, Available: https://www.maruf.ca/files/caadoc/CAACenV5Default.htm (accessed 20.06.2023)

[94] Open CASCADE Technology API help documentation https://dev.opencascade.org/doc/overview/html/ (accessed 20.06.2023)

[95] Slyadnev S, Malyshev A, Turlapov V. CAD model inspection utility and prototyping framework based on OpenCascade, Conference: GraphiCon 2017, Perm, Russia, 2017

[96] Müller JD, Zhang X, Akbarzadeh S, Wang Y. Geometric continuity constraints of automatically derived parametrisations in CAD-based shape optimisation. International Journal of Computational Fluid Dynamics 2019; 33(6-7): 272-288. doi: 10.1080/16864360.2018.1462881

[97] Banović M, Mykhaskiv O, Auriemma S, Walther A, Legrand H, Müller JD. Algorithmic differentiation of the Open CASCADE Technology CAD kernel and its coupling with an adjoint CFD solver. 2018; 33(4-6):813-828. doi: 10.1080/10556788.2018.1431235

[98] NX12 API Programming Tools Help, Available: https://docs.plm.automation.siemens.com/tdoc/nx/12/nx_api/ (accessed 20.6.2023)

[99] NX Getting Started with SNAP, 2014, Available: https://docs.plm.automation.siemens.com/data_services/resources/nx/10/nx_api/en_US/graphics/fileLibrary/nx/snap/SNAP_Getting_Started_V10.pdf (accessed 20.06.2023)

[100] Getting Started with NX Open, 2017, Available: https://docs.plm.automation.siemens.com/data_services/resources/nx/12/nx_api/common/en_US/graphics/fileLibrary/nx/nxopen/nxopen_getting_started_v12.pdf (accessed 20.06.2023)

[101] Song Y Li A, Jia Y, Huang J, Zhao X. Knowledge Fusion: Introduction of Concepts and Techniques. 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), Hangzhou, China, 2019, pp. 112-118, doi: 10.1109/DSC.2019.00025.

[102] Buric M, Brcic M, Škec S. Towards Automated Drafting in CAD Systems. Conference: EEET 2021: 2021 4th International Conference on Electronics and Electrical Engineering Technology, December 2021, doi: 10.1145/3508297.3508335

[103] JSDAI. Java Standard Data Access Interface. Available: http://www.jsdai.net/ (Accessed: 20.06.2023)

[104] Wang J, Yan W, Huang C. Surface shape-based clustering for B-rep models. Multimed Tools Appl, 2020; 79: 25747–25761. doi: 10.1007/s11042-020-09252-3

[105] Wang J. Research on Shape Feature Recognition of B-Rep Model Based on Wavelet Transform. Mathematical Problems in Engineering. 2018. 1-8. doi: 10.1155/2018/6310482.

[106] Goldman R. An Integrated Introduction to Computer Graphics and Geometric Modeling. 1st Edition, CRC Press, Boca Raton-London-New York, 2009

[107] Nasr EA, Kamrani AK. Computer-Based Design and Manufacturing: An Information-Based Approach, Springer, New York, USA, 2007

[108] Tornincasa S, Di, F. (2010). The future and the evolution of CAD. Conference: 14th International Research/Expert Conference. ”Trends in the Development of Machinery and Associated Technology”, TMT 2010 Proceedings, 2010

[109] STEP ISO 10303-42:2021 Available: https://ap238.org/SMRL_v8_final/data/resource_docs/geometric_and_topological_representation/sys/cover.htm (accessed 20.06.2023)

[110] Filip DJ, Ball TW. Procedurally representing lofted surfaces, IEEE Computer Graphics and Applications, 1989; 9(6): 27-33, doi: 10.1109/38.41467

[111] Parasolid XT Format Reference, 2008, Available: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8dc47dbb5fe601bc0b73ba64b6f2a7a50e1eea59 (accessed 20.06.2023)

[112] Ziani M. Towards adaptation of the NURBS weights in shape optimization. MMC. 2022; 9(4): 959–967. Doi: 10.23939/mmc2022.04.959

[113] Wu Z, Seah HS, Lin F. NURBS Volume for Modelling Complex Objects. In: Chen, M., Kaufman, A.E., Yagel, R. (eds) Volume Graphics. Springer, London. 2000. https://doi.org/10.1007/978-1-4471-0737-8_9

[114] Pasadas M, Rodríguez M. Construction of blending surfaces by parametric discrete interpolation PDE splines. Mathematics and Computers in Simulation. 2008; 77: 282-doi290. 10.1016/j.matcom.2007.08.011.

[115] Choi BK. Surface Modeling for CAD/CAM, Elsevier, New York, 1991,

[116] Vida J, Martin RR., Varady T. A survey of blending methods that use parametric surfaces, Computer-Aided Design, 1994, 26(5): 341-365. doi: 10.1016/0010-4485(94)90023-X.

[117] Huang L, Zhu X. Construction of Blending Surfaces. School of Mechanical Engineering & Automation, Beijing University of Aeronautics & Astronautics, Beijing. Available: http://www.linghuang.org/research/paper/blending.htm (accessed 20.6.2023)

[118] Farouki RT. The approximation of non-degenerate offset surfaces, Computer Aided Geometric Design, 1986; 3(1): 15-43, doi: 10.1016/0167-8396(86)90022-1.

[119] Kim J, Pratt MJ, Iyer RG, Sriram RD. Data Exchange of Parametric CAD Models Using ISO 10303-108, 2007. Available: https://www.govinfo.gov/content/pkg/GOVPUB-C13-38c6804180ddd38aa87a68f74ad21bcc/pdf/GOVPUB-C13-38c6804180ddd38aa87a68f74ad21bcc.pdf (accessed: 20.06.2023)

[120] Kim J, Pratt MJ, Iyer RG, Sriram RD, Standardized data exchange of CAD models with design intent. Computer-Aided Design. 2008; 40(7): 760-777, doi: 10.1016/j.cad.2007.06.014.

[121] Pan C, Smith S. Extracting geometrical data from CAD step files. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. ASME. 2003. doi: 10.1115/DETC2003/CIE-48224.

[122] Shahin TM. Feature-Based Design – An Overview. Computer-aided Design and Applications. Computer-Aided Design and Applications, 2008; 5(5): 639-653. doi: 10.3722/cadaps.2008.639-653.

[123] Open CASCADE Technology – Geometry Class Reference, Available: https://dev.opencascade.org/doc/refman/html/class_geom___geometry.html. Accessed: 20.06.2023)

[124] International Standard Organization, ISO 10303-11:2004 Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual

[125] Somiya S. Handbook of Advanced Ceramics: Materials, Applications, Processing, and Properties, 2nd Edition, Elsevier, 2013, doi: 10.1016/C2010-0-66261-4

[126] Katzenbach A, Handschuh S, Vettermann S. JT Format (ISO 14306) and AP 242 (ISO 10303): The Step to the Next Generation Collaborative Product Creation. in: Kovacs, G.L., Kochan, D. (Eds.), Digital Product and Process Development Systems, Springer Berlin Heidelberg, p. 41–52 2013. doi: 10.1007/978-3-642-41329-2_6.

[127] Bijnens J, Kellens K, Cheshire D. Accuracy of geometry data exchange using STEP AP242. Procedia CIRP. 2018; 78: 219-224, doi: 10.1016/j.procir.2018.09.048.

[128] Trinkel T, Anderl R, Zocholl M, Eichhorn H. Long-Term Archiving of Exact 3D CAD Geometry With JT. ASME 2015 International Mechanical Engineering Congress and Exposition American Society of Mechanical Engineers, 2015, doi: 10.1115/IMECE2015-53211

[129] Boussuge F, Léon JC, Hahmann S, Fine L. Idealized Models for FEA Derived from Generative Modeling Processes Based on Extrusion Primitives. Engineering with Computers, 2015; 31: 513-527. doi:10.1007/978-3-319-02335-9_8

[130] Solidworks 2021 Help documentation – STEP Export Options, Available: https://help.solidworks.com/2021/english/SolidWorks/sldworks/hidd_export_options_step.htm (Accessed: 20.06.2023)

[131] Tierney CM, Sun L, Robinson TT, Armstrong CG. Using virtual topology operations to generate analysis topology. Computer-Aided Design, 2017; 85: 154-167. doi: 10.1016/j.cad.2016.07.015.

[132] Solidworks 2021 Help documentation – General Import Options, Available: https://help.solidworks.com/2021/English/SolidWorks/sldworks/hidd_options_import_general.htm (Accessed: 20.06.2023)

[133] Solidworks 2021 API Help – Surface Parameterization Data Interface, Available: https://help.solidworks.com/2015/english/api/sldworksapi/SOLIDWORKS.Interop.sldworks~SOLIDWORKS.Interop.sldworks.ISurfaceParameterizationData.html

[134] Li Y, Wangmn W, Tu C. Optimal Sampling of Parametric Surfaces. Computer Aided Design and Application. 2012; 9: 55–60. doi: 10.3722/cadaps.2012.55-60.

[135] Ko K, Sakkalis T. Orthogonal projection of points in CAD/CAM applications: an overview. Journal of Computational Design and Engineering. 2014; 1(2): 116-127. doi: 10.7315/JCDE.2014.012.

[136] Solidworks 2021 API Help – Get Closest Point On Face Method, Available: https://help.solidworks.com/2020/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.iface2~getclosestpointon.html (Accessed: 20.06.2023)

[137] Minovic P, Ishikawa S, Kato K. Symmetry identification of a 3-D object represented by octree. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993; 15(5): 507-514. doi: 10.1109/34.211472.

[138] Parry-Barwick S, Bowyer A. Symmetry analysis and geometric modelling. ed. KK Fung and A. Ginige, Dicta, 1993; 93: 39-46.

[139] Chan CK, Tan ST. Determination of the minimum bounding box of an arbitrary solid: an iterative approach. Computers & Structures. 2001; 79(15): 1433-1449. doi: 10.1016/S0045-7949(01)00046-3.

[140] CATIA V5 help documentation – Reference elements. Available: http://catiadoc.free.fr/online/prtug_C2/prtugbt1600.htm (Accessed: 20.06.2023)

[141] Solidworks 2021 Help documentation – Reference geometry. Available: https://help.solidworks.com/2021/english/SolidWorks/sldworks/c_Reference_Geometry_Overview.htm (Accessed: 20.06.2023)

[142] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide, 2nd Edition, Addison-Wesley Professional, 2005

[143] FreeCAD Help documentation. Available: https://wiki.freecad.org/OpenCASCADE (Accessed: 20.06.2023)

[144] GrabCAD. Available online: https://grabcad.com/dashboard (Accessed: 01.03.2023)

[145] PARTcommunity. Available online: https://b2b.partcommunity.com/community/ (Accessed: 01.03.2023)

[146] Solidworks 2021 Help documentation – Face2.GetArea. Available: https://help.solidworks.com/2015/english/api/sldworksapi/solidworks.interop.sldworks~solidworks.interop.sldworks.iface2~getarea.html (Accessed: 20.06.2023)

[147] Vakili, M., Ghamsari, M.K., & Rezaei, M. Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification. ArXiv, 2020; abs/2001.09636.

[148] Sipser, M. Introduction to the Theory of Computation, 3rd ed.; Cengage Learning: Boston, MA, USA, 2021.

[149] Pagani L., Scott P.J. Curvature based sampling of curves and surfaces. Computer Aided Geometric Design. 2018; (59): 32-48; doi: 10.1016/j.cagd.2017.11.004.

[150] Elkott D. F., Elmaraghy H. A., Elmaraghy W. H. Automatic sampling for CMM inspection planning of free-form surfaces. International Journal of Production Research. 2002; 40(11): 2653-2676, doi: 10.1080/00207540210133435

[151] Obeidat S., Raman S. An intelligent sampling method for inspecting free-form surfaces. The International Journal of Advanced Manufacturing Technology. 2009; 40:1125-1136. doi: 10.1007/s00170-008-1427-3.

BIOGRAPHY

Mladen Burić was born in 1987 in Odžak (Bosnia and Herzegovina). He attended the elementary schools in Darmstadt (Germany) and Rovinj (Croatia). During his elementary school education, he participated two times and achieved notable results in the state competition of young technicians of the Republic of Croatia. In 2006, he graduated from the mathematical gymnasium in Rovinj and entered the Faculty of Mechanical Engineering and Naval Architecture in Zagreb in the same year. He gained a bachelor's degree in mechanical engineering in 2010 and a master's degree in 2012. He started his professional career while studying and working in Switzerland, a six-month student practice at ABB Turbo Systems. From January 2013 to September 2016, he was employed at Alstom Power (later General Electric) in Karlovac as a mechanical integrity engineer for gas turbines. From October 2016 to December 2019, he worked as a design engineer at Yazaki Europe Limited in Zagreb on developing connectors and electrical chargers for the automotive industry. Since January 2019, he has worked at PIA Automation Croatia as a mechanical engineer designing custom automation machines for the automotive industry. In parallel with work, he is a part-time PhD researcher at the Faculty of Mechanical Engineering and Naval Architecture in Zagreb. He speaks English and German language.

ŽIVOTOPIS

Mladen Burić rođen je 1987. u Odžaku (Bosna i Hercegovina). Osnovne školu pohađao je u Darmstadtu (Njemačka) i u Rovinju (Hrvatska). Tijekom osnovnoškolskog obrazovanja dva put sudjeluje i ostvaruje zapažene rezultate na državnom natjecanju mladih tehničara Republike Hrvatske iz područja strojarstva. Godine 2006. završava matematičku gimnaziju u Rovinju i iste godine upisuje Fakultet strojarstva i brodogradnje u Zagrebu (konstrukcijski smjer). Titulu prvostupnika strojarstva stječe 2010. godine, a 2012. magistra inženjera strojarstva. Svoju profesionalnu karijeru započinje tijekom studija radeći studentsku praksu u Švicarskoj u trajanju od šest mjeseci u tvrtki ABB Turbo Systems. Od siječnja 2013. do listopada 2016. zaposlen je u tvrtki Alstom Power (kasnije General Electric) u Karlovcu kao inženjer za proračun mehaničkog integriteta plinskih turbina. Od listopada 2016. do prosinca 2018. radi kao inženjer konstruktor u tvrtki Yazaki Europe Limited u Zagrebu na razvoju konektora i električnih punjača za automobilsku industriju. Od siječnja 2019. je zaposlen na poziciji inženjer konstruktor u tvrtki PIA Automation Croatia, te se bavi konstruiranjem automatiziranih strojeva specijalne namjene za automobilsku industriju. Paralelno uz posao studira na doktorskom studiji na Fakultetu strojarstva i brodogradnje u Zagrebu. Govori engleskim i njemačkim jezikom.

LIST OF PUBLICATIONS

1. Buric M, Skec S. A Procedure for Identifying Planes and Axes of Symmetry Candidates in B-rep CAD Models. CAD'23, Mexico City, July 10-12, 2023. doi: 10.14733/cadconfP.2023.297-301

2. Buric M, Bosner T, Skec S. A Framework for Detection of Exact Global and Partial Symmetry in 3D CAD Models. Symmetry. 15(5):1058, 2023, doi: 10.3390/sym15051058

3. Buric M, Brcic M, Bojcetic N, Skec S. Computer-Aided Detection of Exact Reflection and Axisymmetry in B-rep CAD Models. Computer-Aided Design and Applications, 20(5):884, 2023, doi: 10.14733/cadaps.2023.884-897

4. Buric M, Brcic M, Bojcetic N, Skec S. Computer-Aided Detection of Exact Reflection and Axisymmetry in B-rep CAD Models. In: CAD'22 Proceedings, Beijing, Chine, July 11-13, 2022, pp. 251-256, https://doi.org/, doi: 10.14733/cadconfP.2022.251-256

5. Buric M, Brcic M, Skec S. Towards Automated Drafting in CAD Systems. In: 4th International Conference on Electronics and Electrical Engineering Technology (EEET 2021) Nanjing: ACM, 2021. pp 233-238. doi:10.1145/3508297.3508335

6. Buric M, Marjanovic D. A tool for Idealisation of CAD Models. In: Proceedings of the 15th International Design Conference (DESIGN 2018), Dubrovnik, Croatia, pp. 205-214 doi:10.21278/idc.2018.0367

7. Buric M, Katana B. Low-Cost Experimental Setup for 2D Digital Image Correlation Method. In: 5th International Conference Vallis Aurea, Požega, Croatia, 2016. pp. 43-53

8. Katana B, Buric M. Polymer Materials in the Food Industry. In: 5th International Conference Vallis Aurea, Požega, Croatia, 2016. pp. 199-206

APPENDIX

A1 – ANALYSIS OF B-REP

```
Sub TopologyClassification()

Dim swApp As SldWorks.SldWorks
Dim swModel As SldWorks.ModelDoc2
Dim swSelMgr As SldWorks.SelectionMgr
Dim swSelData As SldWorks.SelectData
Dim swModelDocExt As SldWorks.ModelDocExtension

Dim swPart As SldWorks.PartDoc
Dim swBody As SldWorks.Body2
Dim swFace As SldWorks.face2
Dim swEdge As SldWorks.Edge
Dim swVertex As SldWorks.Vertex
Dim swCurve As SldWorks.Curve
Dim swLoop As SldWorks.Loop2
Dim swEnt As SldWorks.Entity
Dim swEntity As Entity
Dim swDataselection As SelectData
Dim swSurf As SldWorks.Surface
Dim nStatus As Long
Dim bRet As Boolean
Dim vBodies As Variant
Dim vSectionPropeties1 As Variant

Dim swFeatureManager As FeatureManager
Dim vFeat As Variant
Dim vRefPointFeatures As Variant
Dim swRefPt As RefPoint
Dim swMathPt As MathPoint
Dim swRefPtData As SldWorks.RefPointFeatureData
Dim swMathUntility As MathUtility

Dim CentroidVector As MathVector
Dim CentroidVector2 As MathVector
Dim NormalVector As MathVector
Dim NormalVector2 As MathVector
Dim VectorLength As Double
Dim VectorLength2 As Double
Dim VectorOperation As MathVector
Dim VectorOperation2 As MathVector
Dim NormalizedVector As MathVector
Dim NormalizedVector2 As MathVector
Dim NormalizedVector3 As MathVector
Dim CrossProdCentVect As MathVector
Dim CrossProdNormVect As MathVector
Dim swCoPlanarVector As MathVector

Dim vCylinder As Variant
Dim vCone As Variant
Dim vSphere As Variant
Dim vTorus As Variant
Dim vPoint As Variant
Dim vVertices As Variant
```

```
Dim vCircle As Variant
Dim vEllipse As Variant

Dim CenterPoints(1, 2)
Dim OuterLoopCount As Integer
Dim vPt As Variant
Dim vNorm As Variant
Dim dAngle As Double
Dim dAngle2 As Double
Dim CentroidPoint(2) As Double
Dim CentroidPoint2(2) As Double
Dim NormalPoint(2) As Double
Dim NormalPoint2(2) As Double
Dim vCentroidPoint As Variant
Dim vCentroidPoint2 As Variant
Dim vNormalPoint As Variant
Dim vNormalPoint2 As Variant
Dim vTempPoint As Variant
Dim vTempPoint2 As Variant
Dim vTempPoint3 As Variant
Dim vTempPoint4 As Variant
Dim vArrayVector
Dim vArrayVector2
Dim vArrayCoplanarVector2
Dim vArrayNormVector
Dim vArrayNormVector2
Dim vArrayNormVector3
Dim vPt3(2) As Double
Dim vPt4 As Variant

Set swApp = CreateObject("SldWorks.Application")
Set swModel = swApp.ActiveDoc
Set swFeatureManager = swModel.FeatureManager
Set swPart = swModel
Set swMathUntility = swApp.GetMathUtility
Set swModelDocExt = swModel.Extension
Set swSelMgr = swModel.SelectionManager
Set swSelData = swSelMgr.CreateSelectData
vBodies = swPart.GetBodies2(swAllBodies, True)
Set swBody = vBodies(0)
Set swFace = swBody.GetFirstFace
swModel.ClearSelection2 True

Dim DecimalPlaces2 As Integer
DecimalPlaces = 6
DecimalPlaces2 = 6
DecimalPlaces3 = 6

Dim TextFile As Integer
Dim strText As String
Dim LineArray() As String

iPlanesCount = 0
iCylindersCount = 0
iSemiCylindersCount = 0
iConesCount = 0
iSemiConesCount = 0
iSpheresCount = 0
```

```
iSemiSpheresCount = 0
iTorusesCount = 0
iSemiTorusesCount = 0
iBlendsCount = 0
iExtrusionsCount = 0
iRevolvesCount = 0
iSemiRevolvesCount = 0
TotalFacesCount = 0

SymmetryPlane1Count = 0
SymmetryPlane2Count = 0
SymmetryPlane3Count = 0

Dim LowerRange As Double
Dim UpperRange As Double

LowerRange = 0.99
UpperRange = 1.01

Dim NormalDirection1 As Boolean
Dim NormalDirection2 As Boolean
Dim String1 As String
Dim String2 As String
Dim vPlane1 As Variant

Do While Not swFace Is Nothing
      Set swEnt = swFace
      Set swSurf = swFace.GetSurface
      vPlane1 = swSurf.PlaneParams
      NormalDirection1 = swFace.FaceInSurfaceSense
      If NormalDirection1 <> False Then
      vPlane1(0) = (-1) * vPlane1(0)
      vPlane1(1) = (-1) * vPlane1(1)
      vPlane1(2) = (-1) * vPlane1(2)
      Else
End If

   If swSurf.Identity = 4001 Then
      bRet = swEnt.Select4(False, swSelData)
      vSectionPropeties1 = swModel.Extension.GetSectionProperties2(swFace)
      swModel.ClearSelection2 True
      sPLANES(iPlanesCount, 0) = swModel.GetEntityName(swEnt)
      sPLANES(iPlanesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sPLANES(iPlanesCount, 2) = Round((vSectionPropeties1(2) * 1000), DecimalPlaces)
      sPLANES(iPlanesCount, 3) = Round((vSectionPropeties1(3) * 1000), DecimalPlaces)
      sPLANES(iPlanesCount, 4) = Round((vSectionPropeties1(4) * 1000), DecimalPlaces)
      sPLANES(iPlanesCount, 5) = Round(vPlane1(0), DecimalPlaces)
      sPLANES(iPlanesCount, 6) = Round(vPlane1(1), DecimalPlaces)
      sPLANES(iPlanesCount, 7) = Round(vPlane1(2), DecimalPlaces)
      iPlanesCount = iPlanesCount + 1
   End If

   If swSurf.Identity = 4002 Then
      GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
      If dAngle < 360 Then
      sSEMICYLINDERS(iSemiCylindersCount, 1) = Round((swFace.GetArea * 1000000),
      DecimalPlaces2)
      sSEMICYLINDERS(iSemiCylindersCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
```

```
      sSEMICYLINDERS(iSemiCylindersCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sSEMICYLINDERS(iSemiCylindersCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sSEMICYLINDERS(iSemiCylindersCount, 5) = Round(vNorm(0), DecimalPlaces)
      sSEMICYLINDERS(iSemiCylindersCount, 6) = Round(vNorm(1), DecimalPlaces)
      sSEMICYLINDERS(iSemiCylindersCount, 7) = Round(vNorm(2), DecimalPlaces)
      iSemiCylindersCount = iSemiCylindersCount + 1
   ElseIf dAngle = 360 Then
      vCylinder = swSurf.CylinderParams
      bRet = swEnt.Select4(False, swSelData)
      vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
      For Each vFeat In vRefPointFeatures
      Set swFeat = vFeat
      Set swRefPt = swFeat.GetSpecificFeature2
      Set swRefPtData = swFeat.GetDefinition
      Set swMathPt = swRefPt.GetRefPoint
      boolstatus = swFeat.Select2(False, -1)
      boolstatus = swModelDocExt.DeleteSelection2(1)
      Next
      swModel.ClearSelection2 True
      sCYLINDERS(iCylindersCount, 0) = swModel.GetEntityName(swEnt)
      sCYLINDERS(iCylindersCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces)
      sCYLINDERS(iCylindersCount, 2) = Round((swMathPt.ArrayData(0)) * 1000, DecimalPlaces)
      sCYLINDERS(iCylindersCount, 3) = Round((swMathPt.ArrayData(1)) * 1000, DecimalPlaces)
      sCYLINDERS(iCylindersCount, 4) = Round((swMathPt.ArrayData(2)) * 1000, DecimalPlaces)
      sCYLINDERS(iCylindersCount, 5) = Round(vCylinder(3), DecimalPlaces)
      sCYLINDERS(iCylindersCount, 6) = Round(vCylinder(4), DecimalPlaces)
      sCYLINDERS(iCylindersCount, 7) = Round(vCylinder(5), DecimalPlaces)
      iCylindersCount = iCylindersCount + 1
   End If
End If

If swSurf.Identity = 4003 Then
   GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
   If dAngle < 360 Then
      sSEMICONES(iSemiConesCount, 0) = swModel.GetEntityName(swEnt)
      sSEMICONES(iSemiConesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sSEMICONES(iSemiConesCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
      sSEMICONES(iSemiConesCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sSEMICONES(iSemiConesCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sSEMICONES(iSemiConesCount, 5) = Round(vNorm(0), DecimalPlaces)
      sSEMICONES(iSemiConesCount, 6) = Round(vNorm(1), DecimalPlaces)
      sSEMICONES(iSemiConesCount, 7) = Round(vNorm(2), DecimalPlaces)
      iSemiConesCount = iSemiConesCount + 1
   ElseIf dAngle = 360 Then
      vCone = swSurf.ConeParams2
      bRet = swEnt.Select4(False, swSelData)
      vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
      For Each vFeat In vRefPointFeatures
      Set swFeat = vFeat
      Set swRefPt = swFeat.GetSpecificFeature2
      Set swRefPtData = swFeat.GetDefinition
      Set swMathPt = swRefPt.GetRefPoint
      boolstatus = swFeat.Select2(False, -1)
      boolstatus = swModelDocExt.DeleteSelection2(1)
      Next
      swModel.ClearSelection2 True
      sCONES(iConesCount, 0) = swModel.GetEntityName(swEnt)
      sCONES(iConesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
```

```
      sCONES(iConesCount, 2) = Round((swMathPt.ArrayData(0)) * 1000, DecimalPlaces)
      sCONES(iConesCount, 3) = Round((swMathPt.ArrayData(1)) * 1000, DecimalPlaces)
      sCONES(iConesCount, 4) = Round((swMathPt.ArrayData(2)) * 1000, DecimalPlaces)
      sCONES(iConesCount, 5) = Round(vCone(3), DecimalPlaces)
      sCONES(iConesCount, 6) = Round(vCone(4), DecimalPlaces)
      sCONES(iConesCount, 7) = Round(vCone(5), DecimalPlaces)
      iConesCount = iConesCount + 1
   End If
End If

If swSurf.Identity = 4004 Then
GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
If dAngle < 360 Then
   sSEMISPHERES(iSemiSpheresCount, 0) = swModel.GetEntityName(swEnt)
   sSEMISPHERES(iSemiSpheresCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
   sSEMISPHERES(iSemiSpheresCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
   sSEMISPHERES(iSemiSpheresCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
   sSEMISPHERES(iSemiSpheresCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
   sSEMISPHERES(iSemiSpheresCount, 5) = Round(vNorm(0), DecimalPlaces)
   sSEMISPHERES(iSemiSpheresCount, 6) = Round(vNorm(1), DecimalPlaces)
   sSEMISPHERES(iSemiSpheresCount, 7) = Round(vNorm(2), DecimalPlaces)
   iSemiSpheresCount = iSemiSpheresCount + 1

ElseIf dAngle = 360 Then
   vSphere = swSurf.SphereParams
   bRet = swEnt.Select4(False, swSelData)
   vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
   For Each vFeat In vRefPointFeatures
   Set swFeat = vFeat
   Set swRefPt = swFeat.GetSpecificFeature2
   Set swRefPtData = swFeat.GetDefinition
   Set swMathPt = swRefPt.GetRefPoint
   boolstatus = swFeat.Select2(False, -1)
   boolstatus = swModelDocExt.DeleteSelection2(1)
   Next
   swModel.ClearSelection2 True

   vNorm(0) = swMathPt.ArrayData(0) - vSphere(0)
   vNorm(1) = swMathPt.ArrayData(1) - vSphere(1)
   vNorm(2) = swMathPt.ArrayData(2) - vSphere(2)

   vNorm(0) = vNorm(0) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))
   vNorm(1) = vNorm(1) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))
   vNorm(2) = vNorm(2) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))

   sSPHERES(iSpheresCount, 0) = swModel.GetEntityName(swEnt)
   sSPHERES(iSpheresCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
   sSPHERES(iSpheresCount, 2) = Round((swMathPt.ArrayData(0) * 1000), DecimalPlaces)
   sSPHERES(iSpheresCount, 3) = Round((swMathPt.ArrayData(1) * 1000), DecimalPlaces)
   sSPHERES(iSpheresCount, 4) = Round((swMathPt.ArrayData(2) * 1000), DecimalPlaces)
   sSPHERES(iSpheresCount, 5) = Round(vNorm(0), DecimalPlaces)
   sSPHERES(iSpheresCount, 6) = Round(vNorm(1), DecimalPlaces)
   sSPHERES(iSpheresCount, 7) = Round(vNorm(2), DecimalPlaces)
   iSpheresCount = iSpheresCount + 1
End If
End If

If swSurf.Identity = 4005 Then
```

```
    GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
    If dAngle < 360 Then
       sSEMITORUSES(iSemiTorusesCount, 0) = swModel.GetEntityName(swEnt)
       sSEMITORUSES(iSemiTorusesCount, 1) = Round((swFace.GetArea * 1000000),
       DecimalPlaces2)
       sSEMITORUSES(iSemiTorusesCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
       sSEMITORUSES(iSemiTorusesCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
       sSEMITORUSES(iSemiTorusesCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
       sSEMITORUSES(iSemiTorusesCount, 5) = Round(vNorm(0), DecimalPlaces)
       sSEMITORUSES(iSemiTorusesCount, 6) = Round(vNorm(1), DecimalPlaces)
       sSEMITORUSES(iSemiTorusesCount, 7) = Round(vNorm(2), DecimalPlaces)
       iSemiTorusesCount = iSemiTorusesCount + 1
    ElseIf dAngle = 360 Then
       vTorus = swSurf.TorusParams
       bRet = swEnt.Select4(False, swSelData)
       vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
       For Each vFeat In vRefPointFeatures
       Set swFeat = vFeat
       Set swRefPt = swFeat.GetSpecificFeature2
       Set swRefPtData = swFeat.GetDefinition
       Set swMathPt = swRefPt.GetRefPoint
       boolstatus = swFeat.Select2(False, -1)
       boolstatus = swModelDocExt.DeleteSelection2(1)
       Next
       swModel.ClearSelection2 True
       sTORUSES(iTorusesCount, 0) = swModel.GetEntityName(swEnt)
       sTORUSES(iTorusesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
       sTORUSES(iTorusesCount, 2) = Round((swMathPt.ArrayData(0) * 1000), DecimalPlaces)
       sTORUSES(iTorusesCount, 3) = Round((swMathPt.ArrayData(1) * 1000), DecimalPlaces)
       sTORUSES(iTorusesCount, 4) = Round((swMathPt.ArrayData(2) * 1000), DecimalPlaces)
       sTORUSES(iTorusesCount, 5) = Round(vTorus(3), DecimalPlaces)
       sTORUSES(iTorusesCount, 6) = Round(vTorus(4), DecimalPlaces)
       sTORUSES(iTorusesCount, 7) = Round(vTorus(5), DecimalPlaces)
       iTorusesCount = iTorusesCount + 1
    End If
End If

If swSurf.Identity = 4006 Then
GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
    If dAngle = 360 Or dAngle2 = 360 Then
    bRet = swEnt.Select4(False, swSelData)
    vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
    For Each vFeat In vRefPointFeatures
    Set swFeat = vFeat
    Set swRefPt = swFeat.GetSpecificFeature2
    Set swRefPtData = swFeat.GetDefinition
    Set swMathPt = swRefPt.GetRefPoint
    boolstatus = swFeat.Select2(False, -1)
    boolstatus = swModelDocExt.DeleteSelection2(1)
    Next
    sBSURFACES(iBSurfacesCount, 0) = swModel.GetEntityName(swEnt)
    sBSURFACES(iBSurfacesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
    sBSURFACES(iBSurfacesCount, 2) = Round((swMathPt.ArrayData(0)) * 1000, DecimalPlaces)
    sBSURFACES(iBSurfacesCount, 3) = Round((swMathPt.ArrayData(1)) * 1000, DecimalPlaces)
    sBSURFACES(iBSurfacesCount, 4) = Round((swMathPt.ArrayData(2)) * 1000, DecimalPlaces)
    sBSURFACES(iBSurfacesCount, 5) = Round(vNorm(0), DecimalPlaces)
    sBSURFACES(iBSurfacesCount, 6) = Round(vNorm(1), DecimalPlaces)
```

```
      sBSURFACES(iBSurfacesCount, 7) = Round(vNorm(2), DecimalPlaces)
      Else
      sBSURFACES(iBSurfacesCount, 0) = swModel.GetEntityName(swEnt)
      sBSURFACES(iBSurfacesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sBSURFACES(iBSurfacesCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
      sBSURFACES(iBSurfacesCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sBSURFACES(iBSurfacesCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sBSURFACES(iBSurfacesCount, 5) = Round(vNorm(0), DecimalPlaces)
      sBSURFACES(iBSurfacesCount, 6) = Round(vNorm(1), DecimalPlaces)
      sBSURFACES(iBSurfacesCount, 7) = Round(vNorm(2), DecimalPlaces)
      End If
      iBSurfacesCount = iBSurfacesCount + 1
   End If

   If swSurf.Identity = 4007 Then
      GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
      sBLENDS(iBlendsCount, 0) = swModel.GetEntityName(swEnt)
      sBLENDS(iBlendsCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sBLENDS(iBlendsCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
      sBLENDS(iBlendsCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sBLENDS(iBlendsCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sBLENDS(iBlendsCount, 5) = Round(vNorm(0), DecimalPlaces)
      sBLENDS(iBlendsCount, 6) = Round(vNorm(1), DecimalPlaces)
      sBLENDS(iBlendsCount, 7) = Round(vNorm(2), DecimalPlaces)
      iBlendsCount = iBlendsCount + 1
   End If

If swSurf.Identity = 4008 Then
      GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
      sOFFSETS(iOffsetsCount, 0) = swModel.GetEntityName(swEnt)
      sOFFSETS(iOffsetsCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sOFFSETS(iOffsetsCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
      sOFFSETS(iOffsetsCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sOFFSETS(iOffsetsCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sOFFSETS(iOffsetsCount, 5) = Round(vNorm(0), DecimalPlaces)
      sOFFSETS(iOffsetsCount, 6) = Round(vNorm(1), DecimalPlaces)
      sOFFSETS(iOffsetsCount, 7) = Round(vNorm(2), DecimalPlaces)
      iOffsetsCount = iOffsetsCount + 1
   End If

   If swSurf.Identity = 4009 Then
      GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
      sEXTRUSIONS(iExtrusionsCount, 0) = swModel.GetEntityName(swEnt)
      sEXTRUSIONS(iExtrusionsCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
      sEXTRUSIONS(iExtrusionsCount, 2) = Round((vPt(0)) * 1000, DecimalPlaces)
      sEXTRUSIONS(iExtrusionsCount, 3) = Round((vPt(1)) * 1000, DecimalPlaces)
      sEXTRUSIONS(iExtrusionsCount, 4) = Round((vPt(2)) * 1000, DecimalPlaces)
      sEXTRUSIONS(iExtrusionsCount, 5) = Round(vNorm(0), DecimalPlaces)
      sEXTRUSIONS(iExtrusionsCount, 6) = Round(vNorm(1), DecimalPlaces)
      sEXTRUSIONS(iExtrusionsCount, 7) = Round(vNorm(2), DecimalPlaces)
      iExtrusionsCount = iExtrusionsCount + 1
   End If

   If swSurf.Identity = 4010 Then
      GetFaceCenterParameters swFace, vPt, vNorm, dAngle, dAngle2
      If dAngle2 < 360 Then
         vRevolve = swSurf.GetRevsurfParams
         sSEMIREVOLVES(iSemiRevolvesCount, 0) = swModel.GetEntityName(swEnt)
```

```
        sSEMIREVOLVES(iSemiRevolvesCount, 1) = Round((swFace.GetArea * 1000000),
        DecimalPlaces2)
        sSEMIREVOLVES(iSemiRevolvesCount, 2) = Round((vPt(0) * 1000), DecimalPlaces)
        sSEMIREVOLVES(iSemiRevolvesCount, 3) = Round((vPt(1) * 1000), DecimalPlaces)
        sSEMIREVOLVES(iSemiRevolvesCount, 4) = Round((vPt(2) * 1000), DecimalPlaces)
        sSEMIREVOLVES(iSemiRevolvesCount, 5) = Round(vNorm(0), DecimalPlaces)
        sSEMIREVOLVES(iSemiRevolvesCount, 6) = Round(vNorm(1), DecimalPlaces)
        sSEMIREVOLVES(iSemiRevolvesCount, 7) = Round(vNorm(2), DecimalPlaces)
        iSemiRevolvesCount = iSemiRevolvesCount + 1
      ElseIf dAngle2 = 360 Then
        vRevolve = swSurf.GetRevsurfParams
        bRet = swEnt.Select4(False, swSelData)
        vRefPointFeatures = swFeatureManager.InsertReferencePoint(4, 0, 0, 1)
        For Each vFeat In vRefPointFeatures
        Set swFeat = vFeat
        Set swRefPt = swFeat.GetSpecificFeature2
        Set swRefPtData = swFeat.GetDefinition
        Set swMathPt = swRefPt.GetRefPoint
        boolstatus = swFeat.Select2(False, -1)
        boolstatus = swModelDocExt.DeleteSelection2(1)
        Next
        swModel.ClearSelection2 True

        vNorm(0) = swMathPt.ArrayData(0) - vRevolve(0)
        vNorm(1) = swMathPt.ArrayData(1) - vRevolve(1)
        vNorm(2) = swMathPt.ArrayData(2) - vRevolve(2)

        vNorm(0) = vNorm(0) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))
        vNorm(1) = vNorm(1) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))
        vNorm(2) = vNorm(2) / (Sqrt(vNorm(0) ^ 2 + vNorm(1) ^ 2 + vNorm(2) ^ 2))

        sREVOLVES(iRevolvesCount, 0) = swModel.GetEntityName(swEnt)
        sREVOLVES(iRevolvesCount, 1) = Round((swFace.GetArea * 1000000), DecimalPlaces2)
        sREVOLVES(iRevolvesCount, 2) = Round((swMathPt.ArrayData(0) * 1000), DecimalPlaces)
        sREVOLVES(iRevolvesCount, 3) = Round((swMathPt.ArrayData(1) * 1000), DecimalPlaces)
        sREVOLVES(iRevolvesCount, 4) = Round((swMathPt.ArrayData(2) * 1000), DecimalPlaces)
        sREVOLVES(iRevolvesCount, 5) = Round(vNorm(0), DecimalPlaces)
        sREVOLVES(iRevolvesCount, 6) = Round(vNorm(1), DecimalPlaces)
        sREVOLVES(iRevolvesCount, 7) = Round(vNorm(2), DecimalPlaces)
        iRevolvesCount = iRevolvesCount + 1
      End If
    End If

      Set swFace = swFace.GetNextFace
    Loop

TotalFacesCount = iPlanesCount + iCylindersCount + iSemiCylindersCount + iConesCount +
iSemiConesCount + iSpheresCount + iSemiSpheresCount + iTorusesCount + iSemiTorusesCount +
iBSurfacesCount + iOffsetsCount + iBlendsCount + iExtrusionsCount + iRevolvesCount +
iSemiRevolvesCount

End Sub
```

A2 - COMPUTATION OF MASS PROPERTIES

```
Sub CenterOfMass()

Dim swApp As SldWorks.SldWorks
Dim swModel As SldWorks.ModelDoc2
```

```
Dim swModelDocExt As SldWorks.ModelDocExtension
Dim swMassProperty As MassProperty
Dim nStatus As Long

Dim DecimalPlaces As Integer
DecimalPlaces = 9

Set swApp = CreateObject("SldWorks.Application")
Set swModel = swApp.ActiveDoc

Set swModelDocExt = swModel.Extension
vMassProp = swModelDocExt.GetMassProperties2(2, nStatus, True)
Set swMassProperty = swModelDocExt.CreateMassProperty

vXAxis = swMassProperty.PrincipleAxesOfInertia(0)
vYAxis = swMassProperty.PrincipleAxesOfInertia(1)
vZAxis = swMassProperty.PrincipleAxesOfInertia(2)

dTotalArea = vMassProp(4) * 1000000

End Sub
```

A3 – GENERATION & TRIMMING OF POSCs & AOSCs

```
Sub SymmetryCandidates()

Dim swApp As SldWorks.SldWorks
Dim swModelDoc As SldWorks.ModelDoc2
Dim swPart As PartDoc
Dim swModelDocExt As SldWorks.ModelDocExtension
Dim swSelMgr As SldWorks.SelectionMgr
Dim swMathUntility As MathUtility
Dim boolstatus As Boolean
Dim swMeasure As SldWorks.Measure
Dim swEntity1 As Entity
Dim swEntity2 As Entity
Dim swFeatureManager As FeatureManager
Dim swRefPlane As RefPlane
Dim vSectionPropFace1 As Variant
Dim vSectionPropFace2 As Variant
Dim dMidpoint(2) As Variant

Dim vCOG As Variant
Dim COGStatus As Long
Dim vBoundBox As Variant
Dim dDeltaL(2) As Double
Dim dDeltaTotal As Double

Dim iDecimalPlaces As Integer
iDecimalPlaces = 4

Set swApp = Application.SldWorks
Set swModelDoc = swApp.ActiveDoc
Set swPart = swModelDoc
Set swModelDocExt = swModelDoc.Extension
Set swSelMgr = swModelDoc.SelectionManager
Set swFeatureManager = swModelDoc.FeatureManager
Set swMathUntility = swApp.GetMathUtility
```

```
vCOG = swModelDocExt.GetMassProperties2(2, COGStatus, True)
vBoundBox = swPart.GetPartBox(True)

Dim PointToPlaneDistance As Double
Dim CosineSimlarity As Double
Dim SetA As Integer
Dim SetB As Integer

Dim dArea1 As Double
Dim dArea2 As Double

Dim IntersectionAB As Integer
Dim UnionAB As Integer
Dim IntersectionOuterLoopsAB As Integer
Dim UnionOuterLoopsAB As Integer
Dim IntersectionInnerLoopsAB As Integer
Dim UnionInnerLoopsAB As Integer

Dim FaceName1 As String
Dim face1 As face2
Dim swSurface1 As Surface
Dim vFace1 As Variant
Dim vEdges1 As Variant
Dim swEdge1 As Edge
Dim swLoop1 As Loop2
Dim swCurve1 As Curve
Dim iEdgeType1 As Integer
Dim iBCURVE1 As Integer
Dim iCIRCLE1 As Integer
Dim iCONSTPARAM1 As Integer
Dim iELLIPSE1 As Integer
Dim iINTERSECTION1 As Integer
Dim iLINE1 As Integer
Dim iSPCURVE1 As Integer
Dim iTRIMMED1 As Integer
Dim vCurveParam1 As Variant
Dim vLoops1 'As Variant
'Dim vOuterLoops1 As Variant
'Dim vInnerLoops1 As Variant
Dim iOuterLoopsCount1 As Integer
Dim iInnerLoopsCount1 As Integer
Dim iLoopsCount1 As Integer
Dim vInnerLoopEdges1 'As Variant
Dim vOuterLoopEdges1 'As Variant
Dim iInnerEdgesCount1 As Integer
Dim iOuterEdgesCount1 As Integer
Set vFace1 = face1
Dim FaceCount1 As Integer
Dim StringEdges1 As String
Dim v1 As Variant
Dim i As Integer

Dim FaceName2 As String
Dim face2 As face2
Dim swSurface2 As Surface
Dim vFace2 As Variant
Dim vEdges2 As Variant
```

```
Dim swLoop2 As Loop2
Dim swEdge2 As Edge
Dim swCurve2 As Curve
Dim iEdgeType2 As Integer
Dim iBCURVE2 As Integer
Dim iCIRCLE2 As Integer
Dim iCONSTPARAM2 As Integer
Dim iELLIPSE2 As Integer
Dim iINTERSECTION2 As Integer
Dim iLINE2 As Integer
Dim iSPCURVE2 As Integer
Dim iTRIMMED2 As Integer
Dim vCurveParam2 As Variant
Dim vLoops2 As Variant
Dim iOuterLoopsCount2 As Integer
Dim iInnerLoopsCount2 As Integer
Dim iLoopsCount2 As Integer
Dim vInnerLoopEdges2 As Variant
Dim vOuterLoopEdges2 As Variant
Dim iInnerEdgesCount2 As Integer
Dim iOuterEdgesCount2 As Integer
Set vFace2 = face2
Dim FaceCount2 As Integer
Dim StringEdges2 As String
Dim v2 As Variant
Dim CurveLength As Single

Dim face3 As face2
Dim swEntity3 As Entity
Dim swSurface3 As Surface
Dim vCylinder As Variant
Dim AxisVector As MathVector
Dim APVector As MathVector
Dim CrossProductVector As MathVector
Dim CrossProductVectorNormalized As MathVector
Dim vAxisDir(2) As Double
Dim vAxisPoint(2) As Double
Dim vAPPoint(2) As Double
Dim vAxisDirTemp As Variant
Dim vAxisPointTemp As Variant
Dim vAPPointTemp As Variant
Dim PointToAxisDistance As Double
Dim AxisVector1 As MathVector
Dim AxisVector2 As MathVector
Dim dAxisVector1(2) As Double
Dim dAxisVector2(2) As Double
Dim vAxisVector1 As Variant
Dim vAxisVector2 As Variant

Dim FaceNormalDirection1 As Boolean
Dim FaceNormalDirection2 As Boolean
Dim vPlaneFace1 As Variant
Dim vPlaneFace2 As Variant
Dim vFaceNormal1(2) As Double
Dim vFaceNormal2(2) As Double
Dim vFaceNormal1Temp As Variant
Dim vFaceNormal2Temp As Variant
```

```
Dim iCounter1 As Integer
Dim iCounter2 As Integer

Dim AreaRatio As Double
Dim PerimeterRatio As Double

Dim dAreaFace1 As Double
Dim dAreaFace2 As Double

Dim Point1(2) As Double
Dim Point2(2) As Double

Dim vXAxisVector1 As MathVector
Dim vYAxisVector2 As MathVector
Dim vZAxisVector3 As MathVector
Dim vCrossVectorPlane1 As MathVector
Dim vCrossVectorPlane2 As MathVector
Dim vCrossVectorPlane3 As MathVector
Dim vCrossVectorAxis1 As MathVector
Dim vCrossVectorAxis2 As MathVector
Dim vCrossVectorAxis3 As MathVector

Dim NormalVector1 As MathVector
Dim NormalVector2 As MathVector
Dim CrossProdNormVect As MathVector
Dim NormalizedVector As MathVector
Dim NormalizedVector1 As MathVector
Dim NormalizedVector2 As MathVector
Dim VectorOperation As MathVector
Dim vPtTemp(2) As Double
Dim vPtTemp2 As Variant
Dim vArrayVector
Dim vArrayNormVector

Dim bTempCandi As Boolean
Dim bTempCandi2 As Boolean

Erase POSCandidates()
Erase POSCandidates2()
Erase AOSCandidates()
Erase AOSCandidates2()
iPOSCandidatesCount = 0
iPOSCandidatesCount2 = 0
iAOSCandidatesCount = 0
iAOSCandidatesCount2 = 0

dDeltaL(0) = Abs((vBoundBox(0))) + Abs((vBoundBox(3))) * 1000#
dDeltaL(1) = Abs((vBoundBox(1))) + Abs((vBoundBox(4))) * 1000#
dDeltaL(2) = Abs((vBoundBox(2))) + Abs((vBoundBox(5))) * 1000#

dDeltaTotal = 0.075 * Sqr((dDeltaL(0) ^ 2 + dDeltaL(1) ^ 2 + dDeltaL(2) ^ 2))

Set vXAxisVector1 = swMathUntility.CreateVector(vXAxis)
Set vYAxisVector2 = swMathUntility.CreateVector(vYAxis)
Set vZAxisVector3 = swMathUntility.CreateVector(vZAxis)

For q = 0 To (iPlanesCount - 2)
For r = q + 1 To (iPlanesCount - 1)
```

```
FaceName1 = sPLANES(q, 0)
FaceName2 = sPLANES(r, 0)
Set face1 = swPart.GetEntityByName(FaceName1, 2)
Set face2 = swPart.GetEntityByName(FaceName2, 2)

dArea1 = face1.GetArea
dArea2 = face2.GetArea

If ((dArea1 / dArea2) >= 0.85) And ((dArea1 / dArea2) <= 1.15) Then

Set swEntity1 = face1
FaceCount1 = face1.GetEdgeCount
ReDim aEdges1(FaceCount1) As String
vEdges1 = face1.GetEdges
iLoopsCount1 = face1.GetLoopCount

Set swSurface1 = face1.GetSurface
vPlaneFace1 = swSurface1.PlaneParams
FaceNormalDirection1 = face1.FaceInSurfaceSense
If FaceNormalDirection1 <> False Then
   vFaceNormal1(0) = (-1) * vPlaneFace1(0)
   vFaceNormal1(1) = (-1) * vPlaneFace1(1)
   vFaceNormal1(2) = (-1) * vPlaneFace1(2)
Else
   vFaceNormal1(0) = vPlaneFace1(0)
   vFaceNormal1(1) = vPlaneFace1(1)
   vFaceNormal1(2) = vPlaneFace1(2)
End If

ReDim vInnerLoops1(iLoopsCount1) As Loop2
ReDim vOuterLoops1(iLoopsCount1) As Loop2

vLoops1 = face1.GetLoops

For j = 0 To (iLoopsCount1 - 1)
Set swLoop1 = vLoops1(j)
   If swLoop1.IsOuter <> False Then
   Set vOuterLoops1(iOuterLoopsCount1) = vLoops1(j)
   iOuterLoopsCount1 = iOuterLoopsCount1 + 1
   iOuterEdgesCount1 = iOuterEdgesCount1 + swLoop1.GetEdgeCount
   Else
   Set vInnerLoops1(iInnerLoopsCount1) = vLoops1(j)
   iInnerLoopsCount1 = iInnerLoopsCount1 + 1
   iInnerEdgesCount1 = iInnerEdgesCount1 + swLoop1.GetEdgeCount
   End If
Next

ReDim aEdgesInnerLoops1(iInnerEdgesCount1) As String
ReDim aEdgesOuterLoops1(iOuterEdgesCount1) As String

'LOOPS FACE 1
For o = 0 To (iInnerLoopsCount1 - 1)
Set swLoop1 = vInnerLoops1(o)
vInnerLoopEdges1 = swLoop1.GetEdges
   For p = 0 To (swLoop1.GetEdgeCount - 1)
   Set swEdge1 = vInnerLoopEdges1(p)
   Set swCurve1 = swEdge1.GetCurve
```

```
   iEdgeType = swCurve1.Identity
   vCurveParam1 = swEdge1.GetCurveParams2
   CurveLength = Round(swCurve1.GetLength2(vCurveParam1(6), vCurveParam1(7)) * 1000#, 2)
   Select Case iEdgeType

         Case 3001
         aEdgesInnerLoops1(iCounter1) = "LI" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3002
         aEdgesInnerLoops1(iCounter1) = "CI" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3003
         aEdgesInnerLoops1(iCounter1) = "EL" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3004
         aEdgesInnerLoops1(iCounter1) = "IN" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3005
         aEdgesInnerLoops1(iCounter1) = "BC" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3006
         aEdgesInnerLoops1(iCounter1) = "SC" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3007
         aEdgesInnerLoops1(iCounter1) = "BL" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3008
         aEdgesInnerLoops1(iCounter1) = "CP" & CurveLength
         iCounter1 = iCounter1 + 1

         Case 3009
         aEdgesInnerLoops1(iCounter1) = "TR" & CurveLength
         iCounter1 = iCounter1 + 1

   End Select

   Next p
Next o

For o = 0 To (iOuterLoopsCount1 - 1)
Set swLoop1 = vOuterLoops1(o)
vOuterLoopEdges1 = swLoop1.GetEdges
   For p = 0 To (swLoop1.GetEdgeCount - 1)
   Set swEdge1 = vOuterLoopEdges1(p)
   Set swCurve1 = swEdge1.GetCurve
   iEdgeType = swCurve1.Identity
   vCurveParam1 = swEdge1.GetCurveParams2
   CurveLength = Round(swCurve1.GetLength2(vCurveParam1(6), vCurveParam1(7)) * 1000#, 2)
   Select Case iEdgeType

         Case 3001
```

```
            aEdgesOuterLoops1(p) = "LI" & CurveLength

            Case 3002
            aEdgesOuterLoops1(p) = "CI" & CurveLength

            Case 3003
            aEdgesOuterLoops1(p) = "EL" & CurveLength

            Case 3004
            aEdgesOuterLoops1(p) = "IN" & CurveLength

            Case 3005
            aEdgesOuterLoops1(p) = "BC" & CurveLength

            Case 3006
            aEdgesOuterLoops1(p) = "SC" & CurveLength

            Case 3007
            aEdgesOuterLoops1(p) = "BL" & CurveLength

            Case 3008
            aEdgesOuterLoops1(p) = "CP" & CurveLength

            Case 3009
            aEdgesOuterLoops1(p) = "TR" & CurveLength

        End Select

        Next p
    Next o

    'FACE 1
    For i = 0 To (face1.GetEdgeCount - 1)
        Set swEdge1 = vEdges1(i)
        Set swCurve1 = swEdge1.GetCurve
        iEdgeType = swCurve1.Identity
        vCurveParam1 = swEdge1.GetCurveParams2
        CurveLength = Round(swCurve1.GetLength2(vCurveParam1(6), vCurveParam1(7)) * 1000#, 2)
        Select Case iEdgeType

            Case 3001
            iLINE1 = iLINE1 + 1
            aEdges1(i) = "LI" & CurveLength

            Case 3002
            iCIRCLE1 = iCIRCLE1 + 1
            aEdges1(i) = "CI" & CurveLength

            Case 3003
            iELLIPSE1 = iELLIPSE1 + 1
            aEdges1(i) = "EL" & CurveLength

            Case 3004
            iINTERSECTION1 = iINTERSECTION1 + 1
            aEdges1(i) = "IN" & CurveLength

            Case 3005
            iBCURVE1 = iBCURVE1 + 1
```

```
            aEdges1(i) = "BC" & CurveLength

            Case 3006
            iSPCURVE1 = iSPCURVE1 + 1
            aEdges1(i) = "SC" & CurveLength

            Case 3007
            iBLEND1 = iBLEND1 + 1
            aEdges1(i) = "BL" & CurveLength

            Case 3008
            iCONSTPARAM1 = iCONSTPARAM1 + 1
            aEdges1(i) = "CP" & CurveLength

            Case 3009
            iTRIMMED1 = iTRIMMED1 + 1
            aEdges1(i) = "TR" & CurveLength

    End Select

StringEdges1 = StringEdges1 & "-" & aEdges1(i)

Next i

Set swEntity2 = face2
FaceCount2 = face2.GetEdgeCount
ReDim aEdges2(FaceCount2) As String
vEdges2 = face2.GetEdges
iLoopsCount2 = face2.GetLoopCount

Set swSurface2 = face2.GetSurface
vPlaneFace2 = swSurface2.PlaneParams
FaceNormalDirection2 = face2.FaceInSurfaceSense
If FaceNormalDirection2 <> False Then
    vFaceNormal2(0) = (-1) * vPlaneFace2(0)
    vFaceNormal2(1) = (-1) * vPlaneFace2(1)
    vFaceNormal2(2) = (-1) * vPlaneFace2(2)
Else
    vFaceNormal2(0) = vPlaneFace2(0)
    vFaceNormal2(1) = vPlaneFace2(1)
    vFaceNormal2(2) = vPlaneFace2(2)
End If

ReDim vInnerLoops2(iLoopsCount2) As Loop2
ReDim vOuterLoops2(iLoopsCount2) As Loop2

vLoops2 = face2.GetLoops

For j = 0 To (iLoopsCount2 - 1)
Set swLoop2 = vLoops2(j)
    If swLoop2.IsOuter <> False Then
    Set vOuterLoops2(iOuterLoopsCount2) = swLoop2
    iOuterLoopsCount2 = iOuterLoopsCount2 + 1
    iOuterEdgesCount2 = iOuterEdgesCount2 + swLoop2.GetEdgeCount
    Else
    Set vInnerLoops2(iInnerLoopsCount2) = swLoop2
    iInnerLoopsCount2 = iInnerLoopsCount2 + 1
    iInnerEdgesCount2 = iInnerEdgesCount2 + swLoop2.GetEdgeCount
```

```
   End If
Next

'LOOPS FACE 2
For o = 0 To (iOuterLoopsCount2 - 1)
Set swLoop2 = vOuterLoops2(o)
vOuterLoopEdges2 = swLoop2.GetEdges
   For p = 0 To (swLoop2.GetEdgeCount - 1)
   Set swEdge2 = vOuterLoopEdges2(p)
   Set swCurve2 = swEdge2.GetCurve
   iEdgeType = swCurve2.Identity
   vCurveParam2 = swEdge2.GetCurveParams2
   CurveLength = Round(swCurve2.GetLength2(vCurveParam2(6), vCurveParam2(7)) * 1000#, 2)
   Select Case iEdgeType

         Case 3001
           aEdgesOuterLoops2(p) = "LI" & CurveLength
         Case 3002
           aEdgesOuterLoops2(p) = "CI" & CurveLength
         Case 3003
            aEdgesOuterLoops2(p) = "EL" & CurveLength
         Case 3004
           aEdgesOuterLoops2(p) = "IN" & CurveLength
         Case 3005
           aEdgesOuterLoops2(p) = "BC" & CurveLength
         Case 3006
           aEdgesOuterLoops2(p) = "SC" & CurveLength
         Case 3007
           aEdgesOuterLoops2(p) = "BL" & CurveLength
         Case 3008
           aEdgesOuterLoops2(p) = "CP" & CurveLength
         Case 3009
            aEdgesOuterLoops2(p) = "TR" & CurveLength

   End Select

   Next p
Next o

For o = 0 To (iInnerLoopsCount2 - 1)
Set swLoop2 = vInnerLoops2(o)
vInnerLoopEdges2 = swLoop2.GetEdges
   For p = 0 To (swLoop2.GetEdgeCount - 1)
   Set swEdge2 = vInnerLoopEdges2(p)
   Set swCurve2 = swEdge2.GetCurve
   iEdgeType = swCurve2.Identity
   vCurveParam2 = swEdge2.GetCurveParams2
   CurveLength = Round(swCurve2.GetLength2(vCurveParam2(6), vCurveParam2(7)) * 1000#, 2)
   Select Case iEdgeType
         Case 3001
           aEdgesInnerLoops2(iCounter2) = "LI" & CurveLength
           iCounter2 = iCounter2 + 1
         Case 3002
           aEdgesInnerLoops2(iCounter2) = "CI" & CurveLength
            iCounter2 = iCounter2 + 1
         Case 3003
           aEdgesInnerLoops2(iCounter2) = "EL" & CurveLength
           iCounter2 = iCounter2 + 1
```

```
        Case 3004
          aEdgesInnerLoops2(iCounter2) = "IN" & CurveLength
          iCounter2 = iCounter2 + 1
        Case 3005
          aEdgesInnerLoops2(iCounter2) = "BC" & CurveLength
          iCounter2 = iCounter2 + 1
        Case 3006
          aEdgesInnerLoops2(iCounter2) = "SC" & CurveLength
          iCounter2 = iCounter2 + 1
        Case 3007
          aEdgesInnerLoops2(iCounter2) = "BL" & CurveLength
          iCounter2 = iCounter2 + 1
        Case 3008
          aEdgesInnerLoops2(iCounter2) = "CP" & CurveLength
          iCounter2 = iCounter2 + 1
        Case 3009
          aEdgesInnerLoops2(iCounter2) = "TR" & CurveLength
          iCounter2 = iCounter2 + 1
    End Select

    Next p
Next o

For i = 0 To (face2.GetEdgeCount - 1)
    Set swEdge2 = vEdges2(i)
    Set swCurve2 = swEdge2.GetCurve
    iEdgeType = swCurve2.Identity
    vCurveParam2 = swEdge2.GetCurveParams2
    CurveLength = Round(swCurve2.GetLength2(vCurveParam2(6), vCurveParam2(7)) * 1000#, 2)
    Select Case iEdgeType
        Case 3001
          iLINE2 = iLINE2 + 1
          aEdges2(i) = "LI" & CurveLength
        Case 3002
          iCIRCLE2 = iCIRCLE2 + 1
          aEdges2(i) = "CI" & CurveLength
        Case 3003
          iELLIPSE2 = iELLIPSE2 + 1
          aEdges2(i) = "EL" & CurveLength
        Case 3004
          iINTERSECTION2 = iINTERSECTION2 + 1
          aEdges2(i) = "IN" & CurveLength
        Case 3005
          iBCURVE2 = iBCURVE2 + 1
          aEdges2(i) = "BC" & CurveLength
        Case 3006
          iSPCURVE2 = iSPCURVE2 + 1
          aEdges2(i) = "SC" & CurveLength
        Case 3007
          iBLEND2 = iBLEND2 + 1
          aEdges2(i) = "BL" & CurveLength
        Case 3008
          iCONSTPARAM2 = iCONSTPARAM2 + 1
          aEdges2(i) = "CP" & CurveLength
        Case 3009
          iTRIMMED2 = iTRIMMED2 + 1
          aEdges2(i) = "TR" & CurveLength
    End Select
```

```
StringEdges2 = StringEdges2 & "-" & aEdges2(i)

Next i

swModelDoc.ClearSelection2 True
swEntity1.Select (True)
vSectionPropFace1 = swModelDoc.Extension.GetSectionProperties2(face1)

Set swMeasure = swModelDocExt.CreateMeasure
swMeasure.LengthDecimalPlaces = 14
boolstatus = swMeasure.Calculate(Nothing)
swMeasure.ArcOption = 0
dAreaFace1 = swMeasure.Area * 1000000

q1 = Sqr(swMeasure.Area * 1000000)
q2 = (swMeasure.Perimeter * 1000) '^ 2

' Get the section properties for the selected face

swModelDoc.ClearSelection2 True
swEntity2.Select (True)
vSectionPropFace2 = swModelDoc.Extension.GetSectionProperties2(face1)

Set swMeasure = swModelDocExt.CreateMeasure
swMeasure.LengthDecimalPlaces = 14
boolstatus = swMeasure.Calculate(Nothing)
swMeasure.ArcOption = 0
dAreaFace2 = swMeasure.Area * 1000000

p1 = Sqr(swMeasure.Area * 1000000)
p2 = (swMeasure.Perimeter * 1000) '^ 2

swModelDoc.ClearSelection2 True

vFaceNormal1Temp = vFaceNormal1
vFaceNormal2Temp = vFaceNormal2

Set NormalVector1 = swMathUntility.CreateVector(vFaceNormal1Temp)
Set NormalVector2 = swMathUntility.CreateVector(vFaceNormal2Temp)

Set CrossProdNormVect = NormalVector1.Cross(NormalVector2)

(vSectionPropFace1(4) - vSectionPropFace2(4))
PlaneParameter1 = vFaceNormal1(0) * (vSectionPropFace1(2) - vSectionPropFace2(2)) +
vFaceNormal1(1) * (vSectionPropFace1(3) - vSectionPropFace2(3)) + vFaceNormal1(2) *
(vSectionPropFace1(4) - vSectionPropFace2(4))

If Round(CrossProdNormVect.GetLength(), 6) = 0 Then
   If Round(PlaneParameter1, 12) = 0 Then
      FaceStatus = "COPLANAR"
      vPtTemp(0) = vSectionPropFace1(2) - vSectionPropFace2(2)
      vPtTemp(1) = vSectionPropFace1(3) - vSectionPropFace2(3)
      vPtTemp(2) = vSectionPropFace1(4) - vSectionPropFace2(4)
      vPtTemp2 = vPtTemp
      Set VectorOperation = swMathUntility.CreateVector(vPtTemp2)
   Else
      FaceStatus = "PARALLEL"
```

```
      Set VectorOperation = NormalVector1
   End If
Else
      FaceStatus = "ELSE"
      Set VectorOperation = NormalVector1.Subtract(NormalVector2)
End If

Set NormalizedVector = VectorOperation.Normalise

vArrayVector = VectorOperation.ArrayData

vArrayNormVector = NormalizedVector.ArrayData

EuclideanDistance = Sqr((((vSectionPropFace2(2) + vSectionPropFace1(2)) / 2) - vCOG(0)) ^ 2 +
(((vSectionPropFace2(3) + vSectionPropFace1(3)) / 2) - vCOG(1)) ^ 2 + (((vSectionPropFace2(4) +
vSectionPropFace1(4)) / 2) - vCOG(2)) ^ 2)

dMidpoint(0) = (vSectionPropFace2(2) + vSectionPropFace1(2)) / 2
dMidpoint(1) = (vSectionPropFace2(3) + vSectionPropFace1(3)) / 2
dMidpoint(2) = (vSectionPropFace2(4) + vSectionPropFace1(4)) / 2

dA = vArrayNormVector(0)
dB = vArrayNormVector(1)
dC = vArrayNormVector(2)
dD = -(dA * dMidpoint(0) + dB * dMidpoint(1) + dC * dMidpoint(2))

On Error Resume Next
PointToPlaneDistance = (Abs((dA * vCOG(0)) + (dB * vCOG(1)) + (dC * vCOG(2)) + dD)) / Sqr(dA ^ 2
+ dB ^ 2 + dC ^ 2)

Set vCrossVectorPlane1 = vXAxisVector1.Cross(NormalizedVector)
Set vCrossVectorPlane2 = vYAxisVector2.Cross(NormalizedVector)
Set vCrossVectorPlane3 = vZAxisVector3.Cross(NormalizedVector)

Point1(0) = q1
Point1(1) = g2
Point1(2) = 0

Point2(0) = p1
Point2(1) = p2
Point2(2) = 0

vPoint1 = Point1
vPoint2 = Point2

AB = (q1 * p1) + (q2 * p2)
A = Sqr((q1) ^ 2 + (q2) ^ 2)
b = Sqr((p1) ^ 2 + (p2) ^ 2)

UnionAB = FaceCount1 + FaceCount2

For k = 0 To FaceCount1

For l = 0 To FaceCount2

If aEdges1(k) <> "" Or aEdges2(l) <> "" Then
   If aEdges1(k) = aEdges2(l) Then
   IntersectionAB = IntersectionAB + 1
```

```
   UnionAB = UnionAB - 1
   aEdges1(k) = ""
   aEdges2(l) = ""
   End If
End If

Next l
Next k

UnionInnerLoopsAB = iInnerEdgesCount1 + iInnerEdgesCount2

For k = 0 To iInnerEdgesCount1
For l = 0 To iInnerEdgesCount2
If aEdgesInnerLoops1(k) <> "" Or aEdgesInnerLoops2(l) <> "" Then
   If aEdgesInnerLoops1(k) = aEdgesInnerLoops2(l) Then
   IntersectionInnerLoopsAB = IntersectionInnerLoopsAB + 1
   UnionInnerLoopsAB = UnionInnerLoopsAB - 1
   aEdgesInnerLoops1(k) = ""
   aEdgesInnerLoops2(l) = ""
   End If
End If
Next l
Next k

UnionOuterLoopsAB = iOuterEdgesCount1 + iOuterEdgesCount2

For k = 0 To iOuterEdgesCount1
For l = 0 To iOuterEdgesCount2
If aEdgesOuterLoops1(k) <> "" Or aEdgesOuterLoops2(l) <> "" Then
   If aEdgesOuterLoops1(k) = aEdgesOuterLoops2(l) Then
   IntersectionOuterLoopsAB = IntersectionOuterLoopsAB + 1
   UnionOuterLoopsAB = UnionOuterLoopsAB - 1
   aEdgesOuterLoops1(k) = ""
   aEdgesOuterLoops2(l) = ""
   End If
End If
Next l
Next k

If p1 >= q1 Then
AreaRatio = q1 / p1
Else
AreaRatio = p1 / q1
End If

If p2 >= q2 Then
PerimeterRatio = q2 / p2
Else
PerimeterRatio = p2 / q2
End If

CosineSimlarity = IntersectionAB & "/" & Sqr(SetA * SetB)

If CosineSimlarity >= 0.70 And (Round(PointToPlaneDistance, iDecimalPlaces) * 1000) <=
(dDeltaTotal / 2) Then
If Round(PointToAxisDistance, iDecimalPlaces) * 1000 = 0 And
((Round(vCrossVectorPlane1.GetLength, iDecimalPlaces) = 0) Or
```

```
(Round(vCrossVectorPlane2.GetLength, iDecimalPlaces) = 0) Or
(Round(vCrossVectorPlane3.GetLength, iDecimalPlaces) = 0)) Then
Else
swEntity1.Select (True)
swEntity2.Select (True)

POSCandidates(iPOSCandidatesCount, 0) = FaceName1 & "-" & FaceName2
POSCandidates(iPOSCandidatesCount, 1) = dMidpoint(0)
POSCandidates(iPOSCandidatesCount, 2) = dMidpoint(1)
POSCandidates(iPOSCandidatesCount, 3) = dMidpoint(2)
POSCandidates(iPOSCandidatesCount, 4) = vArrayNormVector(0)
POSCandidates(iPOSCandidatesCount, 5) = vArrayNormVector(1)
POSCandidates(iPOSCandidatesCount, 6) = vArrayNormVector(2)
POSCandidates(iPOSCandidatesCount, 7) = FaceStatus
POSCandidates(iPOSCandidatesCount, 8) = PointToPlaneDistance
iPOSCandidatesCount = iPOSCandidatesCount + 1
End If
End If

swModelDoc.ClearSelection2 True

Erase vOuterLoops1
Erase vInnerLoops1

iOuterLoopsCount1 = 0
iInnerLoopsCount1 = 0
iInnerEdgesCount1 = 0
iOuterEdgesCount1 = 0

iCounter1 = 0

iLINE1 = 0
iCIRCLE1 = 0
iINTERSECTION1 = 0
iBCURVE1 = 0
iSPCURVE1 = 0
iCONSTPARAM1 = 0
iTRIMMED1 = 0

Erase dMidpoint()

dA = 0
dB = 0
dC = 0
dD = 0

PointToPlaneDistance = 0

Erase vOuterLoops2
Erase vInnerLoops2

iOuterLoopsCount2 = 0
iInnerLoopsCount2 = 0
iInnerEdgesCount2 = 0
iOuterEdgesCount2 = 0

iCounter2 = 0
```

```
iLINE2 = 0
iCIRCLE2 = 0
iINTERSECTION2 = 0
iBCURVE2 = 0
iSPCURVE2 = 0
iCONSTPARAM2 = 0
iTRIMMED2 = 0

dAreaFace1 = 0
dAreaFace2 = 0

Erase Point1()
Erase Point2()
Erase vPoint1
Erase vPoint2
Erase aEdges1
Erase aEdges2
Erase vPlaneFace1
Erase vPlaneFace2
Erase vSectionPropFace1
Erase vSectionPropFace2
Erase vPtTemp()
Erase vArrayVector
Erase vArrayNormVector

A = 0
b = 0
AB = 0
p1 = 0
p2 = 0
q1 = 0
q2 = 0

IntersectionAB = 0
UnionAB = 0
UnionOuterLoopsAB = 0
UnionInnerLoopsAB = 0
IntersectionOuterLoopsAB = 0
IntersectionInnerLoopsAB = 0

PlaneParameter1 = 0
iPOSCandidatesCount2 = 0
iPOSCandidatesCount3 = 0

Erase aEdgesOuterLoops1
Erase aEdgesOuterLoops2

End If

   Next r
Next q

'*****************************
'Eliminating POS/AOS duplicates
'*****************************
bTempCandi = False

For q = 0 To (iPOSCandidatesCount - 2)
```

```
For r = q + 1 To (iPOSCandidatesCount - 1)

If POSCandidates(q, 0) <> "" And POSCandidates(r, 0) <> "" Then
PlaneParameter1 = POSCandidates(q, 4) * (POSCandidates(q, 1) - POSCandidates(r, 1)) +
POSCandidates(q, 5) * (POSCandidates(q, 2) - POSCandidates(r, 2)) + POSCandidates(q, 6) *
(POSCandidates(q, 3) - POSCandidates(r, 3))

vFaceNormal1(0) = POSCandidates(q, 4)
vFaceNormal1(1) = POSCandidates(q, 5)
vFaceNormal1(2) = POSCandidates(q, 6)

vFaceNormal2(0) = POSCandidates(r, 4)
vFaceNormal2(1) = POSCandidates(r, 5)
vFaceNormal2(2) = POSCandidates(r, 6)

vFaceNormal1Temp = vFaceNormal1
vFaceNormal2Temp = vFaceNormal2

Set NormalVector1 = swMathUntility.CreateVector(vFaceNormal1Temp)
Set NormalVector2 = swMathUntility.CreateVector(vFaceNormal2Temp)

Set CrossProdNormVect = NormalVector1.Cross(NormalVector2)

If Round(CrossProdNormVect.GetLength(), 6) = 0 Then
   If Round(PlaneParameter1, 6) = 0 Then
      If POSCandidates(q, 7) = "COPLANAR" Then
         If POSCandidates(r, 7) = "COPLANAR" Then
            If r < (iPOSCandidatesCount - 1) Then
               POSCandidates(r, 0) = ""
               iPOSCandidatesCount3 = iPOSCandidatesCount3 + 1
            Else
               If bTempCandi = False Then
               POSCandidates2(iPOSCandidatesCount2, 0) = POSCandidates(q, 0)
               POSCandidates2(iPOSCandidatesCount2, 1) = POSCandidates(q, 1)
               POSCandidates2(iPOSCandidatesCount2, 2) = POSCandidates(q, 2)
               POSCandidates2(iPOSCandidatesCount2, 3) = POSCandidates(q, 3)
               POSCandidates2(iPOSCandidatesCount2, 4) = POSCandidates(q, 4)
               POSCandidates2(iPOSCandidatesCount2, 5) = POSCandidates(q, 5)
               POSCandidates2(iPOSCandidatesCount2, 6) = POSCandidates(q, 6)
               POSCandidates2(iPOSCandidatesCount2, 7) = POSCandidates(q, 7)
               POSCandidates2(iPOSCandidatesCount2, 8) = POSCandidates(q, 8)
               iPOSCandidatesCount2 = iPOSCandidatesCount2 + 1
               bTempCandi = True
               End If
               POSCandidates(r, 0) = ""
               iPOSCandidatesCount3 = iPOSCandidatesCount3 + 1
            End If
         Else
            If bTempCandi = False Then
               POSCandidates2(iPOSCandidatesCount2, 0) = POSCandidates(r, 0)
               POSCandidates2(iPOSCandidatesCount2, 1) = POSCandidates(r, 1)
               POSCandidates2(iPOSCandidatesCount2, 2) = POSCandidates(r, 2)
               POSCandidates2(iPOSCandidatesCount2, 3) = POSCandidates(r, 3)
               POSCandidates2(iPOSCandidatesCount2, 4) = POSCandidates(r, 4)
               POSCandidates2(iPOSCandidatesCount2, 5) = POSCandidates(r, 5)
               POSCandidates2(iPOSCandidatesCount2, 6) = POSCandidates(r, 6)
               POSCandidates2(iPOSCandidatesCount2, 7) = POSCandidates(r, 7)
               POSCandidates2(iPOSCandidatesCount2, 8) = POSCandidates(r, 8)
```

```
            iPOSCandidatesCount2 = iPOSCandidatesCount2 + 1
            bTempCandi = True
          End If
          POSCandidates(r, 0) = ""
          iPOSCandidatesCount3 = iPOSCandidatesCount3 + 1
        End If

      Else
        If bTempCandi = False Then
        POSCandidates2(iPOSCandidatesCount2, 0) = POSCandidates(q, 0)
        POSCandidates2(iPOSCandidatesCount2, 1) = POSCandidates(q, 1)
        POSCandidates2(iPOSCandidatesCount2, 2) = POSCandidates(q, 2)
        POSCandidates2(iPOSCandidatesCount2, 3) = POSCandidates(q, 3)
        POSCandidates2(iPOSCandidatesCount2, 4) = POSCandidates(q, 4)
        POSCandidates2(iPOSCandidatesCount2, 5) = POSCandidates(q, 5)
        POSCandidates2(iPOSCandidatesCount2, 6) = POSCandidates(q, 6)
        POSCandidates2(iPOSCandidatesCount2, 7) = POSCandidates(q, 7)
        POSCandidates2(iPOSCandidatesCount2, 8) = POSCandidates(q, 8)
        iPOSCandidatesCount2 = iPOSCandidatesCount2 + 1
        bTempCandi = True
        End If
        POSCandidates(r, 0) = ""
        iPOSCandidatesCount3 = iPOSCandidatesCount3 + 1
      End If

   End If
Else
End If
End If

PlaneParameter1 = 0

Erase vFaceNormal1Temp
Erase vFaceNormal2Temp
Erase vFaceNormal1
Erase vFaceNormal2

Next r

bTempCandi = False

Next q

bTempCandi = False

If iPOSCandidatesCount2 <> (iPOSCandidatesCount - iPOSCandidatesCount3) Then
For i = 0 To (iPOSCandidatesCount - 1)
   For j = 0 To (iPOSCandidatesCount2 - 1)
      If POSCandidates(i, 0) <> "" And POSCandidates2(j, 0) <> "" Then
         If POSCandidates(i, 0) = POSCandidates2(j, 0) Then
         bTempCandi = True
         End If
      End If

   Next j
   If bTempCandi = False And POSCandidates(i, 0) <> "" Then
      POSCandidates2(iPOSCandidatesCount2, 0) = POSCandidates(i, 0)
      POSCandidates2(iPOSCandidatesCount2, 1) = POSCandidates(i, 1)
```

```
        POSCandidates2(iPOSCandidatesCount2, 2) = POSCandidates(i, 2)
        POSCandidates2(iPOSCandidatesCount2, 3) = POSCandidates(i, 3)
        POSCandidates2(iPOSCandidatesCount2, 4) = POSCandidates(i, 4)
        POSCandidates2(iPOSCandidatesCount2, 5) = POSCandidates(i, 5)
        POSCandidates2(iPOSCandidatesCount2, 6) = POSCandidates(i, 6)
        POSCandidates2(iPOSCandidatesCount2, 7) = POSCandidates(i, 7)
        POSCandidates2(iPOSCandidatesCount2, 8) = POSCandidates(i, 8)
        iPOSCandidatesCount2 = iPOSCandidatesCount2 + 1
    End If
bTempCandi = False
Next i
End If

If iPOSCandidatesCount2 = 0 Then

POSCandidates2(iPOSCandidatesCount2, 0) = ""
POSCandidates2(iPOSCandidatesCount2, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2, 2) = vMassProp(1)
POSCandidates2(iPOSCandidatesCount2, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2, 4) = vXAxis(0)
POSCandidates2(iPOSCandidatesCount2, 5) = vXAxis(1)
POSCandidates2(iPOSCandidatesCount2, 6) = vXAxis(2)
POSCandidates2(iPOSCandidatesCount2, 7) = ""

POSCandidates2(iPOSCandidatesCount2 + 1, 0) = ""
POSCandidates2(iPOSCandidatesCount2 + 1, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2 + 1, 2) = vMassProp(1)
POSCandidates2(iPOSCandidatesCount2 + 1, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2 + 1, 4) = vYAxis(0)
POSCandidates2(iPOSCandidatesCount2 + 1, 5) = vYAxis(1)
POSCandidates2(iPOSCandidatesCount2 + 1, 6) = vYAxis(2)
POSCandidates2(iPOSCandidatesCount2 + 1, 7) = ""

POSCandidates2(iPOSCandidatesCount2 + 2, 0) = ""
POSCandidates2(iPOSCandidatesCount2 + 2, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2 + 2, 2) = vMassProp(1)
POSCandidates2(iPOSCandidatesCount2 + 2, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2 + 2, 4) = vZAxis(0)
POSCandidates2(iPOSCandidatesCount2 + 2, 5) = vZAxis(1)
POSCandidates2(iPOSCandidatesCount2 + 2, 6) = vZAxis(2)
POSCandidates2(iPOSCandidatesCount2 + 2, 7) = ""

iPOSCandidatesCount2 = iPOSCandidatesCount2 + 3

Else

POSCandidates2(iPOSCandidatesCount2 + 0, 0) = ""
POSCandidates2(iPOSCandidatesCount2 + 0, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2 + 0, 2) = vMassProp(1)
POSCandidates2(iPOSCandidatesCount2 + 0, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2 + 0, 4) = vXAxis(0)
POSCandidates2(iPOSCandidatesCount2 + 0, 5) = vXAxis(1)
POSCandidates2(iPOSCandidatesCount2 + 0, 6) = vXAxis(2)
POSCandidates2(iPOSCandidatesCount2 + 0, 7) = ""

POSCandidates2(iPOSCandidatesCount2 + 1, 0) = ""
POSCandidates2(iPOSCandidatesCount2 + 1, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2 + 1, 2) = vMassProp(1)
```

```
POSCandidates2(iPOSCandidatesCount2 + 1, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2 + 1, 4) = vYAxis(0)
POSCandidates2(iPOSCandidatesCount2 + 1, 5) = vYAxis(1)
POSCandidates2(iPOSCandidatesCount2 + 1, 6) = vYAxis(2)
POSCandidates2(iPOSCandidatesCount2 + 1, 7) = ""

POSCandidates2(iPOSCandidatesCount2 + 2, 0) = ""
POSCandidates2(iPOSCandidatesCount2 + 2, 1) = vMassProp(0)
POSCandidates2(iPOSCandidatesCount2 + 2, 2) = vMassProp(1)
POSCandidates2(iPOSCandidatesCount2 + 2, 3) = vMassProp(2)
POSCandidates2(iPOSCandidatesCount2 + 2, 4) = vZAxis(0)
POSCandidates2(iPOSCandidatesCount2 + 2, 5) = vZAxis(1)
POSCandidates2(iPOSCandidatesCount2 + 2, 6) = vZAxis(2)
POSCandidates2(iPOSCandidatesCount2 + 2, 7) = ""

iPOSCandidatesCount2 = iPOSCandidatesCount2 + 3

End If

For t = 0 To (iCylindersCount - 1)

FaceName3 = sCYLINDERS(t, 0)
Set face3 = swPart.GetEntityByName(FaceName3, 2)
Set swEntity3 = face3
Set swSurface3 = face3.GetSurface
vCylinder = swSurface3.CylinderParams
swEntity3.Select (True)

vAxisPoint(0) = vCylinder(0)
vAxisPoint(1) = vCylinder(1)
vAxisPoint(2) = vCylinder(2)

vAxisDir(0) = vCylinder(3) * 2
vAxisDir(1) = vCylinder(4) * 2
vAxisDir(2) = vCylinder(5) * 2

vAPPoint(0) = vCOG(0) - vAxisPoint(0)
vAPPoint(1) = vCOG(1) - vAxisPoint(1)
vAPPoint(2) = vCOG(2) - vAxisPoint(2)

vAPPointTemp = vAPPoint
vAxisDirTemp = vAxisDir

Set APVector = swMathUntility.CreateVector(vAPPointTemp)
Set AxisVector = swMathUntility.CreateVector(vAxisDirTemp)

Set CrossProductVector = APVector.Cross(AxisVector)

PointToAxisDistance = CrossProductVector.GetLength / AxisVector.GetLength

Set vCrossVectorAxis1 = vXAxisVector1.Cross(AxisVector)
Set vCrossVectorAxis2 = vYAxisVector2.Cross(AxisVector)
Set vCrossVectorAxis3 = vZAxisVector3.Cross(AxisVector)

If (Round(PointToAxisDistance, iDecimalPlaces) * 1000) <= (dDeltaTotal / 2) Then
If Round(PointToAxisDistance, iDecimalPlaces) * 1000 = 0 And ((Round(vCrossVectorAxis1.GetLength,
iDecimalPlaces) = 0) Or (Round(vCrossVectorAxis2.GetLength, iDecimalPlaces) = 0) Or
(Round(vCrossVectorAxis3.GetLength, iDecimalPlaces) = 0)) Then
```

```
Else

AOSCandidates(iAOSCandidatesCount, 0) = FaceName3
AOSCandidates(iAOSCandidatesCount, 1) = vCylinder(0)
AOSCandidates(iAOSCandidatesCount, 2) = vCylinder(1)
AOSCandidates(iAOSCandidatesCount, 3) = vCylinder(2)
AOSCandidates(iAOSCandidatesCount, 4) = vCylinder(3)
AOSCandidates(iAOSCandidatesCount, 5) = vCylinder(4)
AOSCandidates(iAOSCandidatesCount, 6) = vCylinder(5)
'AOSCandidates(iAOSCandidatesCount, 7) = FaceStatus
AOSCandidates(iAOSCandidatesCount, 8) = PointToAxisDistance
iAOSCandidatesCount = iAOSCandidatesCount + 1

End If
End If

swModelDoc.ClearSelection2 True

Next t

For t = 0 To (iSemiCylindersCount - 1)

FaceName3 = sSEMICYLINDERS(t, 0)
Set face3 = swPart.GetEntityByName(FaceName3, 2)
Set swEntity3 = face3
Set swSurface3 = face3.GetSurface
vCylinder = swSurface3.CylinderParams
swEntity3.Select (True)
vAxisPoint(0) = vCylinder(0)
vAxisPoint(1) = vCylinder(1)
vAxisPoint(2) = vCylinder(2)

vAxisDir(0) = vCylinder(3) * 2
vAxisDir(1) = vCylinder(4) * 2
vAxisDir(2) = vCylinder(5) * 2

vAPPoint(0) = vCOG(0) - vAxisPoint(0)
vAPPoint(1) = vCOG(1) - vAxisPoint(1)
vAPPoint(2) = vCOG(2) - vAxisPoint(2)

vAPPointTemp = vAPPoint
vAxisDirTemp = vAxisDir

Set APVector = swMathUntility.CreateVector(vAPPointTemp)
Set AxisVector = swMathUntility.CreateVector(vAxisDirTemp)

Set CrossProductVector = APVector.Cross(AxisVector)
'Set CrossProductVectorNormalized = CrossProductVector.Normalise

PointToAxisDistance = CrossProductVector.GetLength / AxisVector.GetLength

Set vCrossVectorAxis1 = vXAxisVector1.Cross(AxisVector)
Set vCrossVectorAxis2 = vYAxisVector2.Cross(AxisVector)
Set vCrossVectorAxis3 = vZAxisVector3.Cross(AxisVector)

If (Round(PointToAxisDistance, iDecimalPlaces) * 1000) <= (dDeltaTotal / 2) Then
```

```
If Round(PointToAxisDistance, iDecimalPlaces) * 1000 = 0 And ((Round(vCrossVectorAxis1.GetLength,
iDecimalPlaces) = 0) Or (Round(vCrossVectorAxis2.GetLength, iDecimalPlaces) = 0) Or
(Round(vCrossVectorAxis3.GetLength, iDecimalPlaces) = 0)) Then
Else
AOSCandidates(iAOSCandidatesCount, 0) = FaceName3
AOSCandidates(iAOSCandidatesCount, 1) = vCylinder(0)
AOSCandidates(iAOSCandidatesCount, 2) = vCylinder(1)
AOSCandidates(iAOSCandidatesCount, 3) = vCylinder(2)
AOSCandidates(iAOSCandidatesCount, 4) = vCylinder(3)
AOSCandidates(iAOSCandidatesCount, 5) = vCylinder(4)
AOSCandidates(iAOSCandidatesCount, 6) = vCylinder(5)
AOSCandidates(iAOSCandidatesCount, 7) = FaceStatus
AOSCandidates(iAOSCandidatesCount, 8) = PointToAxisDistance
iAOSCandidatesCount = iAOSCandidatesCount + 1

End If
End If

swModelDoc.ClearSelection2 True

Next t

bTempCandi = False

For q = 0 To (iAOSCandidatesCount - 2)
For r = q + 1 To (iAOSCandidatesCount - 1)

If AOSCandidates(q, 0) <> "" And AOSCandidates(r, 0) <> "" Then

dAxisVector1(0) = AOSCandidates(q, 4)
dAxisVector1(1) = AOSCandidates(q, 5)
dAxisVector1(2) = AOSCandidates(q, 6)

dAxisVector2(0) = AOSCandidates(r, 4)
dAxisVector2(1) = AOSCandidates(r, 5)
dAxisVector2(2) = AOSCandidates(r, 6)

vAxisVector1 = dAxisVector1
vAxisVector2 = dAxisVector2

Set AxisVector1 = swMathUntility.CreateVector(vAxisVector1)
Set AxisVector2 = swMathUntility.CreateVector(vAxisVector2)

Set CrossProductVector = AxisVector1.Cross(AxisVector2)

If Round(CrossProductVector.GetLength, 6) = 0 Then

'TWO axis vector components = 0
If Abs(Round(dAxisVector1(0), 6)) = 1 And Round(dAxisVector1(1), 6) = 0 And Round(dAxisVector1(2),
6) = 0 Then
   If (Round(AOSCandidates(q, 2), 6) = Round(AOSCandidates(r, 2), 6)) And (Round(AOSCandidates(q,
3), 6) = Round(AOSCandidates(r, 3), 6)) Then
      If bTempCandi = False Then
         AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
         AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
         AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
         AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
         AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
```

```
            AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
            AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
            AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
            AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
            iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
            bTempCandi = True
        End If
        AOSCandidates(r, 0) = ""
        iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
    End If
End If

If Round(dAxisVector1(0), 6) = 0 And Abs(Round(dAxisVector1(1), 6)) = 1 And Round(dAxisVector1(2),
6) = 0 Then
    If (Round(AOSCandidates(q, 1), 6) = Round(AOSCandidates(r, 1), 6)) And (Round(AOSCandidates(q,
3), 6) = Round(AOSCandidates(r, 3), 6)) Then
        If bTempCandi = False Then
            AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
            AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
            AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
            AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
            AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
            AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
            AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
            AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
            AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
            iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
            bTempCandi = True
        End If
        AOSCandidates(r, 0) = ""
        iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
    End If
End If

If Round(dAxisVector1(0), 6) = 0 And Round(dAxisVector1(1), 6) = 0 And Abs(Round(dAxisVector1(2),
6)) = 1 Then
    If (Round(AOSCandidates(q, 1), 6) = Round(AOSCandidates(r, 1), 6)) And (Round(AOSCandidates(q,
2), 6) = Round(AOSCandidates(r, 2), 6)) Then
        If bTempCandi = False Then
            AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
            AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
            AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
            AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
            AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
            AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
            AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
            AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
            AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
            iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
            bTempCandi = True
        End If
        AOSCandidates(r, 0) = ""
        iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
    End If
End If


'ONE axis vector component = 0
```

```
If Round(dAxisVector1(0), 6) = 0 And Round(dAxisVector1(1), 6) <> 0 And Round(dAxisVector1(2), 6)
<> 0 Then
K2 = (AOSCandidates(q, 2) - AOSCandidates(r, 2)) / AOSCandidates(q, 5)
K3 = (AOSCandidates(q, 3) - AOSCandidates(r, 3)) / AOSCandidates(q, 6)
   If Round(K2, 6) = Round(K3, 6) Then
      If bTempCandi = False Then
         AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
         AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
         AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
         AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
         AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
         AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
         AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
         AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
         AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
         iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
         bTempCandi = True
      End If
      AOSCandidates(r, 0) = ""
      iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
   End If
End If

If Round(dAxisVector1(0), 6) <> 0 And Round(dAxisVector1(1), 6) = 0 And Round(dAxisVector1(2), 6)
<> 0 Then
K1 = (AOSCandidates(q, 1) - AOSCandidates(r, 1)) / AOSCandidates(q, 4)
K3 = (AOSCandidates(q, 3) - AOSCandidates(r, 3)) / AOSCandidates(q, 6)
   If Round(K1, 6) = Round(K3, 6) Then
      If bTempCandi = False Then
         AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
         AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
         AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
         AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
         AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
         AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
         AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
         AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
         AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
         iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
         bTempCandi = True
      End If
      AOSCandidates(r, 0) = ""
      iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1

   End If
End If

If Round(dAxisVector1(0), 6) <> 0 And Round(dAxisVector1(1), 6) <> 0 And Round(dAxisVector1(2), 6)
= 0 Then
K1 = (AOSCandidates(q, 1) - AOSCandidates(r, 1)) / AOSCandidates(q, 4)
K2 = (AOSCandidates(q, 2) - AOSCandidates(r, 2)) / AOSCandidates(q, 5)
   If Round(K1, 6) = Round(K2, 6) Then
      If bTempCandi = False Then
         AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
         AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
         AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
         AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
         AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
```

```
            AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
            AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
            AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
            AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
            iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
            bTempCandi = True
        End If
        AOSCandidates(r, 0) = ""
        iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
    End If
End If


'ALL components of axis vector <> 0
If Round(dAxisVector1(0), 6) <> 0 And Round(dAxisVector1(1), 6) <> 0 And Round(dAxisVector1(2), 6)
<> 0 Then
K1 = (AOSCandidates(q, 1) - AOSCandidates(r, 1)) / AOSCandidates(q, 4)
K2 = (AOSCandidates(q, 2) - AOSCandidates(r, 2)) / AOSCandidates(q, 5)
K3 = (AOSCandidates(q, 3) - AOSCandidates(r, 3)) / AOSCandidates(q, 6)
    If Round(K1, 6) = Round(K2, 6) And Round(K2, 6) = Round(K3, 6) And Round(K3, 6) = Round(K1, 6)
Then
        If bTempCandi = False Then
            AOSCandidates2(iAOSCandidatesCount2, 0) = AOSCandidates(q, 0)
            AOSCandidates2(iAOSCandidatesCount2, 1) = AOSCandidates(q, 1)
            AOSCandidates2(iAOSCandidatesCount2, 2) = AOSCandidates(q, 2)
            AOSCandidates2(iAOSCandidatesCount2, 3) = AOSCandidates(q, 3)
            AOSCandidates2(iAOSCandidatesCount2, 4) = AOSCandidates(q, 4)
            AOSCandidates2(iAOSCandidatesCount2, 5) = AOSCandidates(q, 5)
            AOSCandidates2(iAOSCandidatesCount2, 6) = AOSCandidates(q, 6)
            AOSCandidates2(iAOSCandidatesCount2, 7) = AOSCandidates(q, 7)
            AOSCandidates2(iAOSCandidatesCount2, 8) = AOSCandidates(q, 8)
            iAOSCandidatesCount2 = iAOSCandidatesCount2 + 1
            bTempCandi = True
        End If
        AOSCandidates(r, 0) = ""
        iAOSCandidatesCount3 = iAOSCandidatesCount3 + 1
    End If
End If

End If
End If

Next r

bTempCandi = False

Next q

If iAOSCandidatesCount2 = 0 Then

AOSCandidates2(iAOSCandidatesCount2, 0) = ""
AOSCandidates2(iAOSCandidatesCount2, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2, 4) = vXAxis(0)
AOSCandidates2(iAOSCandidatesCount2, 5) = vXAxis(1)
AOSCandidates2(iAOSCandidatesCount2, 6) = vXAxis(2)
AOSCandidates2(iAOSCandidatesCount2, 7) = ""
```

```
AOSCandidates2(iAOSCandidatesCount2 + 1, 0) = ""
AOSCandidates2(iAOSCandidatesCount2 + 1, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2 + 1, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2 + 1, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2 + 1, 4) = vYAxis(0)
AOSCandidates2(iAOSCandidatesCount2 + 1, 5) = vYAxis(1)
AOSCandidates2(iAOSCandidatesCount2 + 1, 6) = vYAxis(2)
AOSCandidates2(iAOSCandidatesCount2 + 1, 7) = ""

AOSCandidates2(iAOSCandidatesCount2 + 2, 0) = ""
AOSCandidates2(iAOSCandidatesCount2 + 2, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2 + 2, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2 + 2, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2 + 2, 4) = vZAxis(0)
AOSCandidates2(iAOSCandidatesCount2 + 2, 5) = vZAxis(1)
AOSCandidates2(iAOSCandidatesCount2 + 2, 6) = vZAxis(2)
AOSCandidates2(iAOSCandidatesCount2 + 2, 7) = ""

iAOSCandidatesCount2 = iAOSCandidatesCount2 + 3

Else

AOSCandidates2(iAOSCandidatesCount2 + 0, 0) = ""
AOSCandidates2(iAOSCandidatesCount2 + 0, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2 + 0, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2 + 0, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2 + 0, 4) = vXAxis(0)
AOSCandidates2(iAOSCandidatesCount2 + 0, 5) = vXAxis(1)
AOSCandidates2(iAOSCandidatesCount2 + 0, 6) = vXAxis(2)
AOSCandidates2(iAOSCandidatesCount2 + 0, 7) = ""

AOSCandidates2(iAOSCandidatesCount2 + 1, 0) = ""
AOSCandidates2(iAOSCandidatesCount2 + 1, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2 + 1, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2 + 1, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2 + 1, 4) = vYAxis(0)
AOSCandidates2(iAOSCandidatesCount2 + 1, 5) = vYAxis(1)
AOSCandidates2(iAOSCandidatesCount2 + 1, 6) = vYAxis(2)
AOSCandidates2(iAOSCandidatesCount2 + 1, 7) = ""

AOSCandidates2(iAOSCandidatesCount2 + 2, 0) = ""
AOSCandidates2(iAOSCandidatesCount2 + 2, 1) = vMassProp(0)
AOSCandidates2(iAOSCandidatesCount2 + 2, 2) = vMassProp(1)
AOSCandidates2(iAOSCandidatesCount2 + 2, 3) = vMassProp(2)
AOSCandidates2(iAOSCandidatesCount2 + 2, 4) = vZAxis(0)
AOSCandidates2(iAOSCandidatesCount2 + 2, 5) = vZAxis(1)
AOSCandidates2(iAOSCandidatesCount2 + 2, 6) = vZAxis(2)
AOSCandidates2(iAOSCandidatesCount2 + 2, 7) = ""

iAOSCandidatesCount2 = iAOSCandidatesCount2 + 3

End If

End Sub
```

A4 – EVALUATION OF POSCs & AOSCs

```
Sub MultipleCandidates(iArrayCounter() As Integer, sSymmetryPlane() As String, dSymmetricArea() As
Double, SymmetryPlaneCount() As Integer, iArrayCounter2() As Integer, sSymmetryAxis() As String,

Dim swApp As SldWorks.SldWorks
Dim swMathUntility As MathUtility
Dim dNormalPoint(2) As Double
Dim dNormalPoint2(2) As Double
Dim dNormalPoint3(2) As Double
Dim dCentroidPoint(2) As Double
Dim dCentroidPoint2(2) As Double
Dim dSurfMidPoint(2) As Double
Dim dPt3(2) As Double
Dim vArrayVector
Dim vArrayNormVector
Dim vNormalPoint
Dim vNormalPoint2
Dim vNormalPoint3
Dim vCentroidPoint
Dim vCentroidPoint2
Dim vPt4
Dim NewPoint1as As Double
Dim NewPoint2 As Double
Dim NormalVector As MathVector
Dim NormalVector2 As MathVector
Dim NormalVector3 As MathVector
Dim CrossProdNormVect As MathVector
Dim CrossProdNormVect2 As MathVector
Dim VectorOperation As MathVector
Dim CrossProdCentVect As MathVector
Dim CrossProdCentVect2 As MathVector
Dim VectorOperation2 As MathVector
Dim dEuclideanDistance1 As Double
Dim dEuclideanDistance2 As Double
Dim sSURFACE() As String
Dim PlaneParameter() As Variant
Dim u As Integer
Dim APVector As MathVector
Dim CrossProductVector As MathVector
Dim vAPPoint(2) As Double
Dim vAPPointTemp As Variant
Dim PointToAxisDistance As Double
Dim AxisVector1 As MathVector
Dim AxisVector2 As MathVector
Dim dAxisVector1(2) As Double
Dim dAxisVector2(2) As Double
Dim vAxisVector1 As Variant
Dim vAxisVector2 As Variant
Dim bSecondCount1 As Boolean
Dim bSecondCount2 As Boolean
Dim Point1(2) As Double
Dim Point2(2) As Double
Dim Point3(2) As Double
Dim Point4(2) As Double
Dim LowerRange As Double
Dim UpperRange As Double

LowerRange = 0.997
```

```
UpperRange = 1.003
bSecondCount1 = False
bSecondCount2 = False
DecimalPlaces = 7
DecimalPlaces2 = 7

ReDim PlaneParameter(iPOSCandidatesCount2)
ReDim sSymmetryPlane(iPOSCandidatesCount2, TotalFacesCount)
ReDim dSymmetricArea(iPOSCandidatesCount2)
ReDim SymmetryPlaneCount(iPOSCandidatesCount2)
ReDim iArrayCounter(iPOSCandidatesCount2)
ReDim sSymmetryAxis(iAOSCandidatesCount2, TotalFacesCount)
ReDim dSymmetricArea2(iAOSCandidatesCount2)
ReDim SymmetryAxisCount(iAOSCandidatesCount2)
ReDim iArrayCounter2(iAOSCandidatesCount2)

Set swApp = CreateObject("SldWorks.Application")
Set swMathUntility = swApp.GetMathUtility

For s = 0 To 14
Select Case s

Case 0
If iPlanesCount > 0 Then
sSURFACE = sPLANES()
iSurfaceCount = iPlanesCount
Else
GoTo Line1
End If

Case 1
If iCylindersCount > 0 Then
sSURFACE = sCYLINDERS()
iSurfaceCount = iCylindersCount
Else
GoTo Line1
End If

Case 2
If iSemiCylindersCount > 0 Then
sSURFACE = sSEMICYLINDERS()
iSurfaceCount = iSemiCylindersCount
Else
GoTo Line1
End If

Case 3
If iConesCount > 0 Then
sSURFACE = sCONES()
iSurfaceCount = iConesCount
Else
GoTo Line1
End If

Case 4
If iSemiConesCount > 0 Then
sSURFACE = sSEMICONES()
iSurfaceCount = iSemiConesCount
```

```
Else
GoTo Line1
End If

Case 5
If iSpheresCount > 0 Then
sSURFACE = sSPHERES()
iSurfaceCount = iSpheresCount
Else
GoTo Line1
End If

Case 6
If iSemiSpheresCount > 0 Then
sSURFACE = sSEMISPHERES()
iSurfaceCount = iSemiSpheresCount
Else
GoTo Line1
End If

Case 7
If iTorusesCount > 0 Then
sSURFACE = sTORUSES()
iSurfaceCount = iTorusesCount
Else
GoTo Line1
End If

Case 8
If iSemiTorusesCount > 0 Then
sSURFACE = sSEMITORUSES()
iSurfaceCount = iSemiTorusesCount
Else
GoTo Line1
End If

Case 9
If iBlendsCount > 0 Then
sSURFACE = sBLENDS()
iSurfaceCount = iBlendsCount
Else
GoTo Line1
End If

Case 10
If iBSurfacesCount > 0 Then
sSURFACE = sBSURFACES()
iSurfaceCount = iBSurfacesCount
Else
GoTo Line1
End If

Case 11
If iExtrusionsCount > 0 Then
sSURFACE = sEXTRUSIONS()
iSurfaceCount = iExtrusionsCount
Else
GoTo Line1
```

```
End If

Case 12
If iRevolvesCount > 0 Then
sSURFACE = sREVOLVES()
iSurfaceCount = iRevolvesCount
Else
GoTo Line1
End If

Case 13
If iSemiRevolvesCount > 0 Then
sSURFACE = sSEMIREVOLVES()
iSurfaceCount = iSemiRevolvesCount
Else
GoTo Line1
End If

Case 14
If iOffsetsCount > 0 Then
sSURFACE = sOFFSETS()
iSurfaceCount = iOffsetsCount
Else
GoTo Line1
End If

End Select

'****************************
'******SINGLE SURFACE******
'****************************

'***Reflection symmetry check***
If iSurfaceCount = 1 Then
For u = 0 To (iPOSCandidatesCount2 - 1)
PlaneParameter(u) = POSCandidates2(u, 4) * (POSCandidates2(u, 1) * 1000# - sSURFACE(0, 2)) +
POSCandidates2(u, 5) * (POSCandidates2(u, 2) * 1000# - sSURFACE(0, 3)) + POSCandidates2(u, 6) *
(POSCandidates2(u, 3) * 1000# - sSURFACE(0, 4))
   If Round(PlaneParameter(u), DecimalPlaces) = 0 Then
      sSymmetryPlane(u, iArrayCounter(u)) = sSURFACE(0, 0)
      dSymmetricArea(u) = dSymmetricArea(u) + sSURFACE(0, 1)
      SymmetryPlaneCount(u) = SymmetryPlaneCount(u) + 1
      iArrayCounter(u) = iArrayCounter(u) + 1
   End If
Next u
'**************************

'***Axisymmetry check***
For v = 0 To (iAOSCandidatesCount2 - 1)

dAxisVector1(0) = AOSCandidates2(v, 4)
dAxisVector1(1) = AOSCandidates2(v, 5)
dAxisVector1(2) = AOSCandidates2(v, 6)
dAxisVector2(0) = sSURFACE(0, 5)
dAxisVector2(1) = sSURFACE(0, 6)
dAxisVector2(2) = sSURFACE(0, 7)

vAxisVector1 = dAxisVector1
```

```
vAxisVector2 = dAxisVector2

Set AxisVector1 = swMathUntility.CreateVector(vAxisVector1)
Set AxisVector2 = swMathUntility.CreateVector(vAxisVector2)
Set CrossProductVector = AxisVector1.Cross(AxisVector2)

If Round(CrossProductVector.GetLength, 6) = 0 Then
vAPPoint(0) = sSURFACE(0, 2) - AOSCandidates2(v, 1) * 1000#
vAPPoint(1) = sSURFACE(0, 3) - AOSCandidates2(v, 2) * 1000#
vAPPoint(2) = sSURFACE(0, 4) - AOSCandidates2(v, 3) * 1000#

vAPPointTemp = vAPPoint
Set APVector = swMathUntility.CreateVector(vAPPointTemp)
Set CrossProductVector = APVector.Cross(AxisVector1)
PointToAxisDistance = CrossProductVector.GetLength / AxisVector1.GetLength

   If Round(PointToAxisDistance, DecimalPlaces2) = 0 Then
      sSymmetryAxis(v, iArrayCounter2(v)) = sSURFACE(0, 0)
      dSymmetricArea2(v) = dSymmetricArea2(v) + sSURFACE(0, 1)
      SymmetryAxisCount(v) = SymmetryAxisCount(v) + 1
      iArrayCounter2(v) = iArrayCounter2(v) + 1
   End If
End If
Next v

End If
'*********************************

'*********************************
'******MULTIPLE SURFACES******
'*********************************
If iSurfaceCount > 1 Then

For i = 0 To (iSurfaceCount - 2) '***Loop faces 1st time for paiwise comaprison***

'***SINGLE POS CANDIDATE***
If iPOSCandidatesCount2 = 1 Then
PlaneParameter(u) = POSCandidates2(u, 4) * (POSCandidates2(u, 1) * 1000# - sSURFACE(i, 2)) +
POSCandidates2(u, 5) * (POSCandidates2(u, 2) * 1000# - sSURFACE(i, 3)) + POSCandidates2(u, 6) *
(POSCandidates2(u, 3) * 1000# - sSURFACE(i, 4))
   If Round(PlaneParameter(u), DecimalPlaces) = 0 Then
      sSymmetryPlane(u, iArrayCounter(u)) = sSURFACE(i, 0)
      dSymmetricArea(u) = dSymmetricArea(u) + sSURFACE(i, 1)
      SymmetryPlaneCount(u) = SymmetryPlaneCount(u) + 1
      iArrayCounter(u) = iArrayCounter(u) + 1
   End If

'***MULTIPLE POS CANDIDATES***
Else

'***Check if face centroid of each stand-alone faces (except last one) is coincident with AOS***
For v = 0 To (iAOSCandidatesCount2 - 1)
dAxisVector1(0) = AOSCandidates2(v, 4)
dAxisVector1(1) = AOSCandidates2(v, 5)
dAxisVector1(2) = AOSCandidates2(v, 6)

dAxisVector2(0) = sSURFACE(i, 5)
dAxisVector2(1) = sSURFACE(i, 6)
```

```
dAxisVector2(2) = sSURFACE(i, 7)

vAxisVector1 = dAxisVector1
vAxisVector2 = dAxisVector2

Set AxisVector1 = swMathUntility.CreateVector(vAxisVector1)
Set AxisVector2 = swMathUntility.CreateVector(vAxisVector2)
Set CrossProductVector = AxisVector1.Cross(AxisVector2)

If Round(CrossProductVector.GetLength, DecimalPlaces2) = 0 Then

vAPPoint(0) = sSURFACE(i, 2) - AOSCandidates2(v, 1) * 1000#
vAPPoint(1) = sSURFACE(i, 3) - AOSCandidates2(v, 2) * 1000#
vAPPoint(2) = sSURFACE(i, 4) - AOSCandidates2(v, 3) * 1000#

vAPPointTemp = vAPPoint
Set APVector = swMathUntility.CreateVector(vAPPointTemp)
Set CrossProductVector = APVector.Cross(AxisVector1)
PointToAxisDistance = CrossProductVector.GetLength / AxisVector1.GetLength
   If Round(PointToAxisDistance, DecimalPlaces2) = 0 Then
      sSymmetryAxis(v, iArrayCounter2(v)) = sSURFACE(i, 0)
      dSymmetricArea2(v) = dSymmetricArea2(v) + sSURFACE(i, 1)
      SymmetryAxisCount(v) = SymmetryAxisCount(v) + 1
      iArrayCounter2(v) = iArrayCounter2(v) + 1
   End If
End If
Next v

For u = 0 To (iPOSCandidatesCount2 - 1) '***Loop all POS candidates***

'***Check if face centroid of each stand-alone faces (except last one) is coincident with POS***
   PlaneParameter(u) = POSCandidates2(u, 4) * (POSCandidates2(u, 1) * 1000# - sSURFACE(i, 2)) +
POSCandidates2(u, 5) * (POSCandidates2(u, 2) * 1000# - sSURFACE(i, 3)) + POSCandidates2(u, 6) *
(POSCandidates2(u, 3) * 1000# - sSURFACE(i, 4))
   If Round(PlaneParameter(u), DecimalPlaces) = 0 Then
      sSymmetryPlane(u, iArrayCounter(u)) = sSURFACE(i, 0)
      dSymmetricArea(u) = dSymmetricArea(u) + sSURFACE(i, 1)
      SymmetryPlaneCount(u) = SymmetryPlaneCount(u) + 1
      iArrayCounter(u) = iArrayCounter(u) + 1
   End If

For j = i + 1 To (iSurfaceCount - 1) '***Loop faces 2nd time for paiwise comaprison***

'***Check if face centroid of last stand-alone face is coincident with POS/AOS***
If i = (iSurfaceCount - 2) And j = (iSurfaceCount - 1) Then '
   PlaneParameter(u) = POSCandidates2(u, 4) * (POSCandidates2(u, 1) * 1000# - sSURFACE(j, 2)) +
POSCandidates2(u, 5) * (POSCandidates2(u, 2) * 1000# - sSURFACE(j, 3)) + POSCandidates2(u, 6) *
(POSCandidates2(u, 3) * 1000# - sSURFACE(j, 4))
   If Round(PlaneParameter(u), DecimalPlaces) = 0 Then
      sSymmetryPlane(u, iArrayCounter(u)) = sSURFACE(j, 0)
      dSymmetricArea(u) = dSymmetricArea(u) + sSURFACE(j, 1)
      SymmetryPlaneCount(u) = SymmetryPlaneCount(u) + 1
      iArrayCounter(u) = iArrayCounter(u) + 1
   End If

   If bSecondCount2 = False Then
   bSecondCount2 = True
   For v = 0 To (iAOSCandidatesCount2 - 1)
```

```
   dAxisVector1(0) = AOSCandidates2(v, 4)
   dAxisVector1(1) = AOSCandidates2(v, 5)
   dAxisVector1(2) = AOSCandidates2(v, 6)

   dAxisVector2(0) = sSURFACE(j, 5)
   dAxisVector2(1) = sSURFACE(j, 6)
   dAxisVector2(2) = sSURFACE(j, 7)

   vAxisVector1 = dAxisVector1
   vAxisVector2 = dAxisVector2

   Set AxisVector1 = swMathUntility.CreateVector(vAxisVector1)
   Set AxisVector2 = swMathUntility.CreateVector(vAxisVector2)
   Set CrossProductVector = AxisVector1.Cross(AxisVector2)

   If Round(CrossProductVector.GetLength, DecimalPlaces2) = 0 Then

   vAPPoint(0) = sSURFACE(j, 2) - AOSCandidates2(v, 1) * 1000#
   vAPPoint(1) = sSURFACE(j, 3) - AOSCandidates2(v, 2) * 1000#
   vAPPoint(2) = sSURFACE(j, 4) - AOSCandidates2(v, 3) * 1000#

   vAPPointTemp = vAPPoint
   Set APVector = swMathUntility.CreateVector(vAPPointTemp)
   Set CrossProductVector = APVector.Cross(AxisVector1)
   PointToAxisDistance = CrossProductVector.GetLength / AxisVector1.GetLength
      If Round(PointToAxisDistance, DecimalPlaces2) = 0 Then
         sSymmetryAxis(v, iArrayCounter2(v)) = sSURFACE(j, 0)
         dSymmetricArea2(v) = dSymmetricArea2(v) + sSURFACE(j, 1)
         SymmetryAxisCount(v) = SymmetryAxisCount(v) + 1
         iArrayCounter2(v) = iArrayCounter2(v) + 1
      End If
   End If
   Next v
   End If
End If

If ((sSURFACE(i, 1) / sSURFACE(j, 1)) >= LowerRange) And ((sSURFACE(i, 1) / sSURFACE(j, 1)) <=
UpperRange) Then

NewPoint1 = sSURFACE(i, 2)
NewPoint2 = sSURFACE(j, 2)
dSurfMidPoint(0) = (NewPoint1 + NewPoint2) / 2
NewPoint1 = sSURFACE(i, 3)
NewPoint2 = sSURFACE(j, 3)
dSurfMidPoint(1) = (NewPoint1 + NewPoint2) / 2
NewPoint1 = sSURFACE(i, 4)
NewPoint2 = sSURFACE(j, 4)
dSurfMidPoint(2) = (NewPoint1 + NewPoint2) / 2

dCentroidPoint(0) = sSURFACE(i, 2) - dSurfMidPoint(0) 'centroid point 1
dCentroidPoint(1) = sSURFACE(i, 3) - dSurfMidPoint(1) 'centroid point 1
dCentroidPoint(2) = sSURFACE(i, 4) - dSurfMidPoint(2) 'centroid point 1

dCentroidPoint2(0) = sSURFACE(j, 2) - dSurfMidPoint(0) 'centroid point 2
dCentroidPoint2(1) = sSURFACE(j, 3) - dSurfMidPoint(1) 'centroid point 2
dCentroidPoint2(2) = sSURFACE(j, 4) - dSurfMidPoint(2) 'centroid point 2
```

```
dNormalPoint(0) = sSURFACE(i, 5) 'normal point 1
dNormalPoint(1) = sSURFACE(i, 6) 'normal point 1
dNormalPoint(2) = sSURFACE(i, 7) 'normal point 1

dNormalPoint2(0) = sSURFACE(j, 5) 'normal point 2
dNormalPoint2(1) = sSURFACE(j, 6) 'normal point 2
dNormalPoint2(2) = sSURFACE(j, 7) 'normal point 2

vCentroidPoint = dCentroidPoint
vCentroidPoint2 = dCentroidPoint2
vNormalPoint = dNormalPoint
vNormalPoint2 = dNormalPoint2

Set CentroidVector = swMathUntility.CreateVector(vCentroidPoint)
Set CentroidVector2 = swMathUntility.CreateVector(vCentroidPoint2)

dEuclideanDistance1 = Sqr((sSURFACE(i, 2) - POSCandidates2(u, 1) * 1000#) ^ 2 + (sSURFACE(i, 3) -
POSCandidates2(u, 2) * 1000#) ^ 2 + (sSURFACE(i, 4) - POSCandidates2(u, 3) * 1000#) ^ 2)
dEuclideanDistance2 = Sqr((sSURFACE(j, 2) - POSCandidates2(u, 1) * 1000#) ^ 2 + (sSURFACE(j, 3) -
POSCandidates2(u, 2) * 1000#) ^ 2 + (sSURFACE(j, 4) - POSCandidates2(u, 3) * 1000#) ^ 2)

dA = POSCandidates2(u, 4) * 1000#
dB = POSCandidates2(u, 5) * 1000#
dC = POSCandidates2(u, 6) * 1000#
dD = -(dA * POSCandidates2(u, 1) * 1000# + dB * POSCandidates2(u, 2) * 1000# + dC *
POSCandidates2(u, 3) * 1000#)

PointToPlaneDistance = (Abs((dA * dSurfMidPoint(0)) + (dB * dSurfMidPoint(1)) + (dC *
dSurfMidPoint(2)) + dD)) / Sqr(dA ^ 2 + dB ^ 2 + dC ^ 2)

Set NormalVector = swMathUntility.CreateVector(vNormalPoint)
Set NormalVector2 = swMathUntility.CreateVector(vNormalPoint2)

Set CrossProdNormVect = NormalVector.Cross(NormalVector2)
Set CrossProdCentVect = CentroidVector.Cross(CentroidVector2)

'***Checking the status of two face pairs, whether they are coplanar, parallel or else***
'***********PlaneParameter = An * (x1 - x0) + Bn * (y1 - y0) + Cn * (z1 - z0)***********
PlaneParameter1 = dNormalPoint(0) * (dCentroidPoint(0) - dCentroidPoint2(0)) + dNormalPoint(1) *
(dCentroidPoint(1) - dCentroidPoint2(1)) + dNormalPoint(2) * (dCentroidPoint(2) -
dCentroidPoint2(2))

   If Round(CrossProdNormVect.GetLength, DecimalPlaces2) = 0 Then
      If Round(PlaneParameter1, DecimalPlaces2) = 0 Then
         FaceStatus = "COPLANAR"
         dPt3(0) = dCentroidPoint(0) - dCentroidPoint2(0)
         dPt3(1) = dCentroidPoint(1) - dCentroidPoint2(1)
         dPt3(2) = dCentroidPoint(2) - dCentroidPoint2(2)
         vPt4 = dPt3
         Set VectorOperation = swMathUntility.CreateVector(vPt4) 'coplanar vector
      Else
         FaceStatus = "PARALLEL"
         Set VectorOperation = NormalVector
      End If
   Else
      FaceStatus = "ELSE"
      Set VectorOperation = NormalVector.Subtract(NormalVector2)
   End If
```

```
   If Round(CrossProdCentVect.GetLength, DecimalPlaces2) = 0 Then
      FaceStatus2 = "PARALLEL"
      Set VectorOperation2 = CentroidVector.Subtract(CentroidVector2)
   Else
      FaceStatus2 = "ELSE"
      Set VectorOperation2 = CentroidVector.Subtract(CentroidVector2)
   End If

   Set NormalizedVector = VectorOperation.Normalise
   Set NormalizedVector2 = VectorOperation2.Normalise
   vArrayVector = VectorOperation.ArrayData
   vArrayVector2 = VectorOperation2.ArrayData
   vArrayNormVector = NormalizedVector.ArrayData
   vArrayNormVector2 = NormalizedVector2.ArrayData

'***Checking whether the midpoint of face pairs is coincident with the POS***
'*****PlaneParameter = An * (x1 - x0) + Bn * (y1 - y0) + Cn * (z1 - z0)******

PlaneParameter(u) = POSCandidates2(u, 4) * (POSCandidates2(u, 1) * 1000# - dSurfMidPoint(0)) +
POSCandidates2(u, 5) * (POSCandidates2(u, 2) * 1000# - dSurfMidPoint(1)) + POSCandidates2(u, 6) *
(POSCandidates2(u, 3) * 1000# - dSurfMidPoint(2))

'***Checking whether the cross product of the face normals resultant vector (vector operation) and the
normal POS vector is zero***
      If Round(PlaneParameter(u), DecimalPlaces2) = 0 Then
         dNormalPoint3(0) = POSCandidates2(u, 4) 'POS normal point 1
         dNormalPoint3(1) = POSCandidates2(u, 5) 'POS normal point 1
         dNormalPoint3(2) = POSCandidates2(u, 6) 'POS normal point 1
         vNormalPoint3 = dNormalPoint3
         Set NormalVector3 = swMathUntility.CreateVector(vNormalPoint3)
         Set CrossProdNormVect2 = NormalizedVector.Cross(NormalVector3)
         Set CrossProdCentVect2 = NormalizedVector2.Cross(NormalVector3)

            If Round(CrossProdNormVect2.GetLength, DecimalPlaces2) = 0 Or
Round(CrossProdNormVect2.GetLength, DecimalPlaces2) = 1 Then
            If Round(CrossProdCentVect2.GetLength, DecimalPlaces2) = 0 Then
            sSymmetryPlane(u, iArrayCounter(u)) = sSURFACE(i, 0) & "-" & sSURFACE(j, 0)
            dSymmetricArea(u) = dSymmetricArea(u) + sSURFACE(i, 1) + sSURFACE(j, 1)
            SymmetryPlaneCount(u) = SymmetryPlaneCount(u) + 2
            iArrayCounter(u) = iArrayCounter(u) + 1
            End If
            End If
      End If
End If
End If
Next j
Next u
End If
Next i
End If
Erase sSURFACE()
Line1:
bSecondCount2 = False
Next s
```