Mold Feature Recognition using Accessibility Analysis for Automated Design of Core, Cavity, and Side-Cores and Tool-Path Generation of Mold Segments

by

Rajnish Bassi

A thesis
presented to University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2012

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Injection molding is widely used to manufacture plastic parts with good surface finish, dimensional stability and low cost. The common examples of parts manufactured by injection molding include toys, utensils, and casings of various electronic products. The process of mold design to generate these complex shapes is iterative and time consuming, and requires great expertise in the field. As a result, a significant amount of the final product cost can be attributed to the expenses incurred during the product's design. After designing the mold segments, it is necessary to machine these segments with minimum cost using an efficient tool-path. The tool-path planning process also adds to the overall mold cost. The process of injection molding can be simplified and made to be more cost effective if the processes of mold design and tool-path generation can be automated.

This work focuses on the automation of mold design from a given part design and the automation of tool-path generation for manufacturing mold segments. The hypothesis examined in this thesis is that the automatic identification of mold features can reduce the human efforts required to design molds. It is further hypothesised that the human effort required in many downstream processes such as mold component machining can also be reduced with algorithmic automation of otherwise time consuming decisions.

Automatic design of dies and molds begins with the part design being provided as a solid model. The solid model of a part is a database of its geometry and topology. The automatic mold design process uses this database to identify an undercut-free parting direction, for recognition of mold features and identification of parting lines for a given parting direction, and for generation of entities such as parting surfaces, core, cavity and side-cores. The methods presented in this work are analytical in nature and work with the extended set of part topologies and geometries unlike those found in the literature. Moreover, the methods do not require discretizing the part geometry to design its mold segments, unlike those found in the literature that result in losing the part definition. Once the mold features are recognized and parting lines are defined, core, cavity and side-cores are generated. This work presents algorithms that recognize the entities in the part solid model that contribute to the design of the core, cavity and side-cores, extract the entities,

and use them in the design of these elements. The developed algorithms are demonstrated on a variety of parts that cover a wide range of features.

The work also presents a method for automatic tool-path generation that takes the designed core/cavity and produces a multi-stage tool-path to machine it from raw stock. The tool-path generation process begins by determining tool-path profiles and tool positions for the rough machining of the part in layers. Typically roughing is done with large aggressive tools to reduce the machining time; and roughing leaves uncut material. After generating a roughing tool-path for each layer, the machining is simulated and the areas left uncut are identified to generate a clean-up tool-path for smaller sized tools. The tool-path planning is demonstrated using a part having obstacles within the machining region. The simulated machining is presented in this work.

This work extends the accessibility analysis by retaining the topology information and using it to recognize a larger domain of features including intersecting features, filling a void in the literature regarding a method that could recognize complex intersecting features during an automated mold design process. Using this information, a larger variety of new mold intersecting features are classified and recognized in this approach.

The second major contribution of the work was to demonstrate that the downstream operations can also benefit from algorithmic decision making. This is shown by automatically generating roughing and clean-up tool-paths, while reducing the machining time by machining only those areas that have uncut material. The algorithm can handle cavities with obstacles in them. The methodology has been tested on a number of parts.

Acknowledgements

I am thankful to my supervisor, Prof. Sanjeev Bedi, whose guidance, support, and encouragement helped me to dig deeper at every step of my research. I sincerely believe that this thesis would not have been possible without the active and loving support of Prof. Bedi. I would also like to thank him for giving me complete freedom in research while guiding me along a path to achieve the objectives effectively. Furthermore, Prof. Bedi always encouraged me to perform my work to full potential, and appreciated those efforts regardless of the results. I am very thankful to him for showing faith in me and helping me to develop professionally and personally.

I would like to thank my committee members, Prof. Jan Huissoon, Prof. Michael Worswick, and Prof. Stephen Mann, for agreeing to be on my committee and for providing me valuable feedback on my comprehensive proposal. I would also like to thank Prof. Hoda ElMaraghy for agreeing to be my external examiner.

I would like to thank my parents for their consistent encouragement and support. I would also like to thank my colleagues, Adel Izadbakhsh, Gerardo Salas, Hussain Hahari, Kandarp Patel and Tarun Kanadala, who helped me in shaping to my ideas and discussing the latest advancements in the field.

Finally and most importantly, I would like to thank my wife, Saloni, whose support and love has played a significant role in the successful completion of my thesis.

Dedication

This work is dedicated to
my daughter Anusha.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Injection molding is the most widely-used plastic parts manufacturing process that can produce near-net shaped products on a large scale with good surface finish, dimensional accuracy and low cost. These molded parts have a vast domain of usage in today's market and typical examples from our day-to-day life include toys, utensils, casings of various electronics and electrical appliances.

Injection molding of a part requires at least two mold segments – *core* and *cavity*. Core and cavity, shown in Figure 1-1, are male and female portions of the molding and give shape to the inside and outside form of the molded parts, respectively [1]. During the molding process, the core and cavity are brought in contact with each other and the molten plastic is injected at high pressure into the space formed (called *impression*) between the core and the cavity. The molten plastic adopts the shape of the space formed within core and cavity. After the solidification process, the two halves are separated in the opposite direction (called *parting direction*) and the molded part is removed with the assistance of pins (called *ejector pin(s)*).

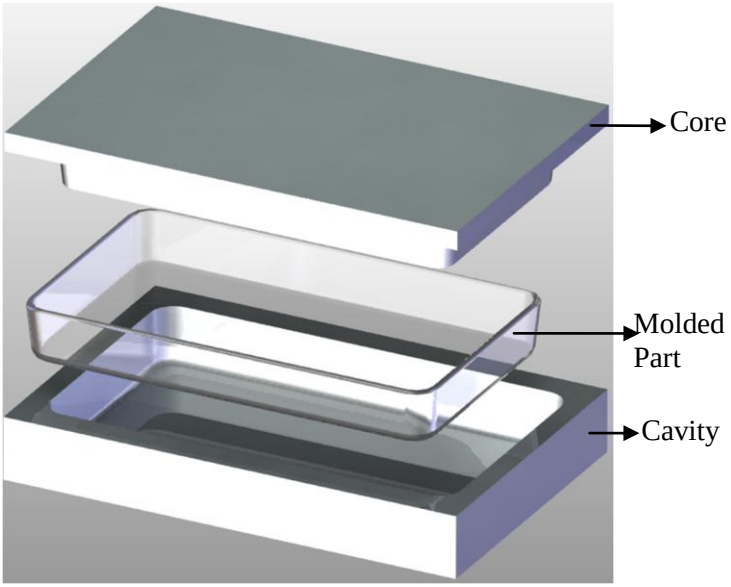


Figure 1-1: Core and cavity of a molded part.

However, the complexity of the molding process increases if the part has some surfaces that make its removal from core/cavity impossible after solidification. Figure 1-2 shows a part having a through hole in one wall. The surfaces forming the through hole will not be moldable using only core and cavity. The region formed by such surfaces is called an *undercut*. There are several types of undercuts that are discussed in detail in Chapter 4. Depending upon the undercut type, additional components are required to mold the part. The use of these additional components results in an increase of the manufacturing cost by contributing to increased mold design complexity and increased production cycle time.

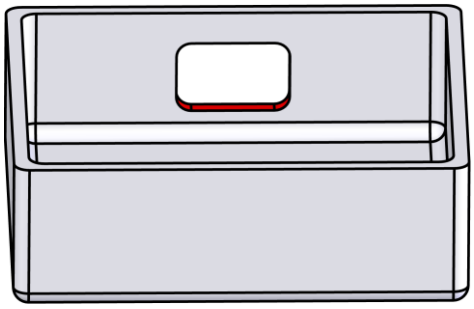


Figure 1-2: Injection molded part with an undercut feature.

Mold design is a complex, iterative and error-prone process. The design of mold components requires an extensive knowledge about their functions and structure. To facilitate the mold design, there are many commercial software systems (Mold Wizard®, ProMold®, IMold®, MoldFlow®, etc.) available to automate the design process, and there are many efforts reported in literature as well. However, most of the work is done in the field of flow analysis, cooling, shrinkage, warpage and stress analysis; few attempts have been made to automate the process of initial design and manufacturing of injection molds.

1.1 Injection Molding Process

Injection molding processes can differ substantially in design and operation. However, most of the injection molding processes generally include *plastication*, *injection*, *packing*, *cooling*, and *mold resetting* stages [2]. During the plastication stage, granules are plasticized through the combined effect of heat conduction from the heated barrel and the heating caused by the rotation of an internal screw. During the filling stage, the molten

polymer is forced from the barrel of the molding and into the mold cavity. After the mold cavity is filled with the molten polymer and is allowed to cool down and contract, more material is injected into the cavity during the packing stage. Typically 1% to 10% of additional material is injected during the packing stage. After the polymer ceases to flow, additional time is provided for the resin to solidify in the cavity and to become rigid for ejection during the cooling stage. During the mold resetting stage, the mold is opened and the molded part is removed from it. A depiction of the injection molding process is given in Figure 1-3.

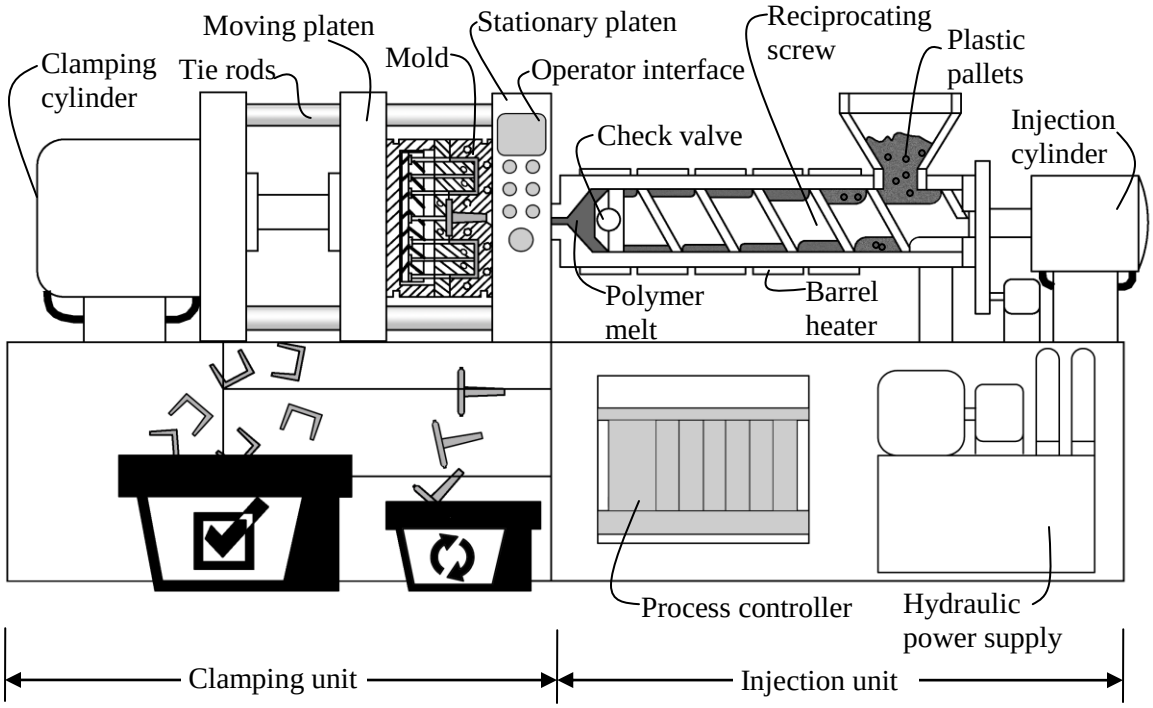


Figure 1-3: Depiction of injection molding process [2].

1.2 Injection Molding Tools

An injection mold part with undercuts requires additional elements other than the core and cavity, namely, *side-core/cavity* and *split-core/cavity*. Side and split cores are used to shape undercuts resulting from the material removal from the base face, whereas side and split cavities are used to shape part surfaces that bulge out of the base face. The difference between a side-core/cavity and a split-core/cavity is that a side-core/cavity has no obstacle in its way while separating from the mold assembly, whereas additional

mechanisms are required to remove the split-core/cavity from the mold assembly because of the obstacles in the way. A part shown in Figure 1-4(a) has one undercut due to material removal from the base face. The assembly of core, cavity and side-core required to mold the part is shown in Figure 1-4(b).

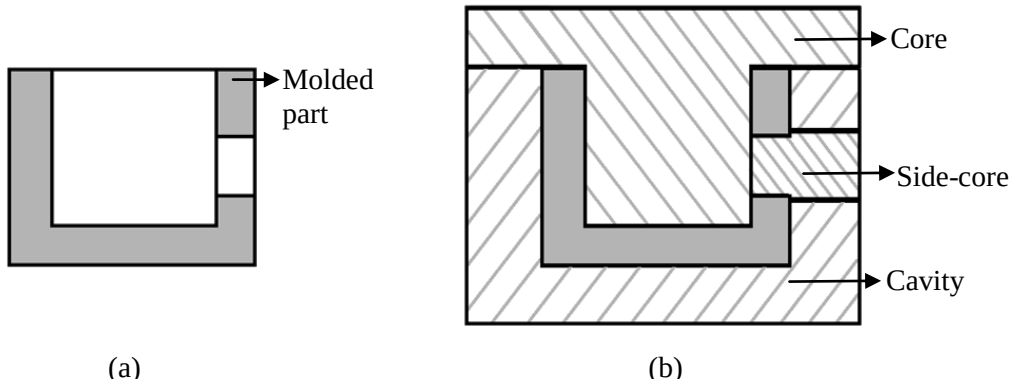


Figure 1-4: (a) An injection molded part and (b) aseembly of mold tools to shape the part.

The manufacturing process of the mold tools is dependent on the part intricacies and the material. To automate the manufacturing process, the material characteristics must be considered. The materials used for mold tools are required to exhibit certain characteristics for achieving the high functionality of the mold, e.g., high wear resistance, high corrosive resistance, good dimensional stability, and good thermal conductivity. During the processing of high-temperature plastics, the cavity wall temperature can approach 250ºC. Therefore, mold elements are required to be built up of tool steels with appropriate tempering properties. Non-compliance with this requirement results in microstructure changes within the material with temperature. This can lead to dimensional instability [3]. Other than tool steel, heat-treatable aluminium-zinc-magnesium-copper alloy has also proven useful for injection molds used to produce prototypes or small to medium quantities. Due to the material properties, the machining of mold tools is difficult and the material removal rate is small. Therefore, machining is usually performed in layers.

1.3 Molding Nomenclature

The commonly used terms in the field of molding, shown in Figure 1-5, are explained here and are used in the rest of the report.

- Parting direction: A pair of opposite directions ($\pm \vec{d}$) along which the core and cavity separate from the mold assembly.

  A part can be molded with one or more parting directions. The mold designer must select the most appropriate one before the design can proceed. An inappropriate parting direction selection can result in complicated tools that cost more to make and maintain. A parting direction that requires the minimum number of mold toolings is preferred. The designer must identify all feasible parting directions to ensure that the most economical is selected.

- Parting lines: A closed continuous curve on the mold surfaces that separates the core and cavity surfaces.

  A part can be molded with one or more different parting lines. The parting lines influence the core and cavity shapes. A number of criteria have been discussed in the literature [4] to generate the optimal parting lines.

- Parting surfaces: Mating surfaces of the core and cavity [5].

  Parting surfaces are generated for a given parting direction and parting lines. The parting surfaces are generated by extruding the parting lines outwards to the outside boundary of core and cavity. The parting surfaces are used as splitting surfaces to cut the containing box (bounding box) of a molded part into two halves [5].

- Bounding box: A rectangular box enclosing the part.

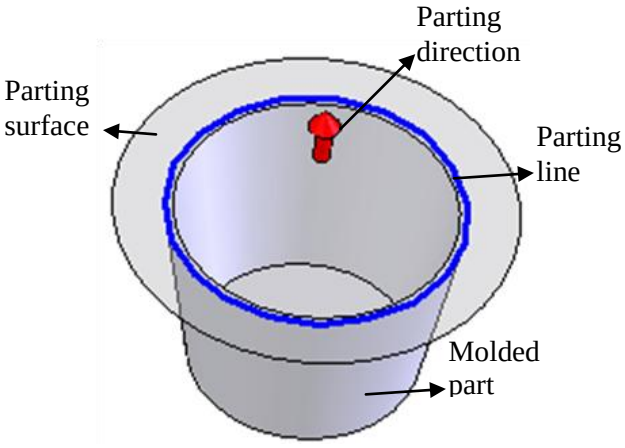


Figure 1-5: Molding terminologies.

The selection of the molding parameter is an important and time consuming step. If all possible parameters can be identified and evaluated, the mold design process can be made more efficient.

1.4 Molding Features

Features encapsulate the engineering significance of portions of the part geometry and as such are applicable in part definition and reasoning about the part in a variety of applications [5]. Therefore, features are related to some physical and geometric aspects of a part and are defined in different ways for different processes. For example, machining features can be a slot or fillet, molding features can be holes or bosses, and measuring features are a datum or reference plane. Therefore, recognition of a feature is solely dependent on the application.

In injection molding, the *features* (also known as *undercut features*) are the recesses or projections on the part that prevents its removal from the mold along the parting direction. Depending upon the type of the features, different types of additional mold tool elements, side-core and side-cavity, are required. These additional mold tool elements are also called *local-tools*. These local-tools are removed after molding the part in a direction other than the parting direction. The withdrawal direction of these local-tools is called the *undercut/release direction*. The ability to programmatically identify these features from the solid models can be useful in automatic design of the core, cavity, and local-tools.

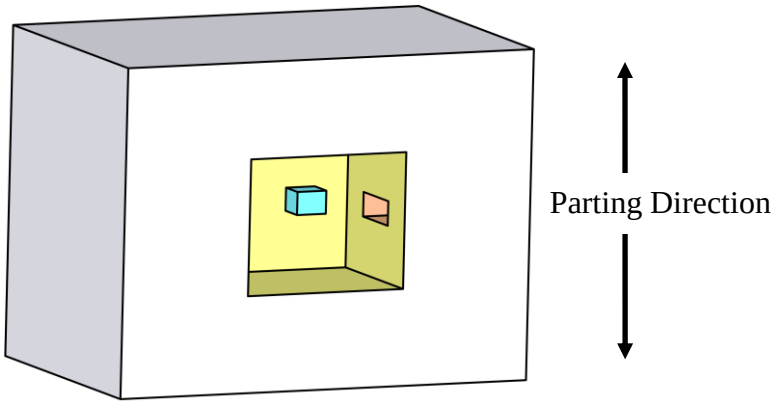


Figure 1-6: Injection molded part with intersecting features.

In practice, the features are present within each other or intersect and such features are known as *intersecting features*, shown in Figure 1-6. Depending upon the topology, the sub-features of the intersecting feature may require separate local-tools. Therefore, it becomes essential to recognize each sub-feature for designing the mold elements. However, the complexity of the feature recognition process increases with the intersections between various mold features.

1.5 Need

The injection molding process is well known for low production time and efficient material utilization [6]. The molded parts require little or no secondary operations; the process is therefore also known as a near net shaped manufacturing process. Moreover, the molded parts have good surface quality and high accuracy [7]. Due to these characteristics, the applications of injection molded parts can be found in most of the major industrial sectors, including automotive field, rail, transport, defence and aerospace, medical and healthcare, electrical and electronics, telecommunication, building and infrastructure, and furniture. Furthermore, the injection molding industry is expected to grow with innovation in new application areas, advancements in plastic materials, and utilization of computers in design and manufacturing. The industry must continue to incorporate the latest advancements in technology to grow and improve.

Mold making used to be largely experience-based and rely heavily on craftsmen's skills that were acquired through years of practice. The younger generation is reluctant to go through long training programs, so the supply of trained manpower is diminishing rapidly. With the advancements in CAD/CAM technologies, mold-making knowledge has been transferred largely from individuals to knowledge-based intelligent computer systems. However, the involvement of computers in the mold industry is relatively slow when compared to the manufacturing sectors [5].

In mold shops, mold design accounts for about 20% of total work effort and CAM programming accounts for about 8%. Out of the mold design work, about half of the work is associated with the design of core and cavity [8]. There have been attempts reported in the literature to automate the mold design and manufacturing. However, more work is required to automate the design of complex mold segments and to reduce the

machining time. The current methods of automated mold design are limited to simple parts with no tolerance for arbitrary intersections between the mold features.

The design of these molding tools is a critical and cumbersome process, and it consumes significant time for each new product. To make the manufacturing economically feasible, it becomes imperative for the manufacturer to make a large number of parts from the same mold and divide the cost. A better approach would be to shorten the design and manufacturing lead time.

Furthermore, the mold designer needs to work iteratively on the prototypes for the customer approval and cost analysis. In the iteration process, the mold designer modifies the part geometry to make part molding easier and each modification requires customer approval to ensure that the design still preserves the intent of the product. A mold design software system that can evaluate the various mold design parameters can speed up the customer approval process.

The hypothesis examined in this thesis is that the automatic identification of parting lines, generation of parting surfaces, recognition of mold features, and building of core, cavity and local-tools can reduce the human efforts required to design molds. It is further hypothesised that the human effort required in many downstream processes such as machining of core, cavity and local-tools can also be reduced with algorithmic automation that uses the geometric and topological data from the part definition to make otherwise time consuming decisions.

Automated mold and die designing and tool-path planning offer potential ways of reducing the design and manufacturing time. It is the objective of this research to show that it is possible to aid the mold design process with feature recognition algorithms and to automate the design process. It is further possible to envisage a geometry recognition algorithm that selects entities and groups them with respect to their topological relationship. The grouping can be machined with an algorithmically selected tool, machining parameters and footprint. This entire step can be automated and merged with the mold design process.

The objective of the dissertation is to determine the existing developments by reviewing the literature, and to advance the automatic design process. Another goal of the work is to show that tool-path planning for machining the designed mold segments

can be done automatically and efficiently by limiting the tools' movement to areas where material remains to be cut.

1.6 Introduction to Mold Design and Tool-Path Automation

The automation of the mold design process can be achieved through two methodologies. In the first method, part information used in reasoning about design, performance and manufacturing is embedded into the part model. The information is stored in the form of features. The features are used to relate geometry and topology algorithmically. Since features are used to design the part, its entities are labelled and grouped, easing the design of core, cavity, side-cores/cavities and tool-path. This method is known as *Feature-based Design*. Since features must be engineered in the early stages of the design, the method is best suited for large companies with engineering and programming resources. This method is used infrequently by small and medium scale enterprises that do not have these resources. In such companies, it is common practice to design the part independently of the subsequent steps. Once the part is designed, it is processed to identify the geometries that share a topological connection and the process is called *Feature Recognition*. A feature recognition-based approach is used in this work to automate the mold design segments.

To automate the tool-path generation process, to reduce the manual input, and to shorten the manufacturing lead time, a software system is required that can extract the tool-path information from the part geometry and topology. The tool-path generation process can be divided into two stages: tool-path profiling and tool-path positioning. The tool-path profile is the profile formed by tool movement while machining on a plane perpendicular to the tool axis, whereas tool-path position involves the tool movement along the tool axis. The tool-path profiling phase ensures that no uncut region should be left on a mold segment for a given tool size. On the other hand, tool-path positioning ensures that no extra material should be cut from the work-piece. With this division it is possible to merge tool-path footprints without the worry of gouging. The tool positioning process takes into account the entire geometry of the part and is thus gouge-free. This concept is used in this work.

In this work, the solid model of part is used in two different formats: *B-rep* and *STL*. The B-rep format is used to recognize the mold features, design mold segments, and

generate tool-path profile, whereas the STL format is used for tool positioning. The characteristics of both of these file formats are discussed in Appendix A.

1.7 Thesis Layout

First, a literature review covering the mold feature recognition, mold segment generation, and tool-path profiling methods is presented in Chapter 2. The shortcomings of the published work described in the chapter will point out the necessity of the research presented in this thesis.

The Boolean-based and pixel-based approaches to analyze the accessibility (visibility) of the part surfaces are presented in Chapter 3. The advantages and disadvantages of both methodologies are discussed as well.

Chapter 4 focuses on the recognition of various intersecting mold features. First, the mold features are classified based on the geometry and topology. Then methodologies to recognize the various features are discussed.

The process of core, cavity and side-core design from a given part, based on the recognized mold features, is discussed in Chapter 5.

The manufacturing information of the mold segments is extracted and the methodologies for generation of tool-path profiles for rough and clean-up machining are explained in Chapter 6.

In Chapter 7, conclusions and recommendations for future improvements are presented.

Chapter 2

Literature Review

Injection molding is on the critical path of any new product development due to the lead time required for mold design and manufacturing. There have been many attempts to automate the mold design process. However, their capabilities are limited and can be substantially enhanced. Similarly, in the field of mold machining, various strategies are reported in literature for generating the tool-path to automate the manufacturing process. The various aspects of the reported works in the fields of mold feature recognition, mold segment design, and mold manufacturing are discussed in this chapter.

2.1 Mold Design

The mold design process involves the determination of parting direction and parting lines, the formation of parting surfaces, and the construction of mold segments. To automate the design process, feature recognition is an important step that requires topological and geometrical analysis of the given part. In this review, various approaches for determining the parting direction are presented in Section 2.1.1. The methodologies for generating the parting lines and parting surfaces are discussed in Section 2.1.2. The various approaches for mold features recognition and mold segments generation are discussed in Sections 2.1.3 and 2.1.4, respectively.

2.1.1 Parting Direction Determination

Most of the methodologies developed for automatically determining the parting direction make use of visibility/accessibility analysis. A direction from which all the part surfaces are fully accessible is preferred as the parting direction because the part can be molded without side-cores. Hui and Tan [9] proposed a method to determine the parting direction by evaluating a set of candidate parting directions using a ray tracing method. They used semi-infinite rays originated from surface grid points towards a candidate parting direction to classify them as obstructed or unobstructed. To estimate the amount of obstacles along a candidate parting direction, they introduced the concept of a blocking factor, where the blocking factor is defined as the ratio of total number of obscured points

to the total number of test points. The direction with the minimum of blocking factor is chosen as the parting direction. However, the authors have heuristically generated the candidate parting directions along the normals of planar faces and axes of cylindrical surfaces. Therefore, the global optimality cannot be guaranteed. Moreover, the accuracy of the algorithm is dependent on the distance between surface grid points.

Woo [10] proposed a procedure to determine the visibility map (V-map) of a surface from its Gauss map (G-map), where both the V-map and the G-map represent a region on a unit sphere. Each point of a G-map donates a direction from which at least one point of the surface is visible, whereas each point of a V-map represents a direction from which entire surface is visible, as shown in Figure 2-1.

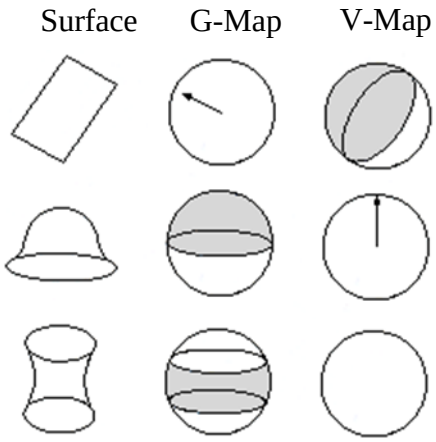


Figure 2-1: Surfaces and their corresponding G-Maps and V-Maps [11].

Chen et al. [11] used the visibility maps to determine the best parting direction. The best parting direction is determined by locating a pair of antipodal points that are contained by the maximum number of V-maps. Later, Chen et al. [12] used the two levels of visibility (complete and partial) to determine de-moldability of polyhedral surfaces. They further divided the partially visible surfaces into those that can be molded with and without side-cores. However, their approach cannot be used for parts having non-planar surfaces.

Dhaliwal et al. [13] determined the global accessibility cones for all facets of a polyhedral object. In their work, the boundary of a unit sphere is partitioned into a finite number of spherical triangles. The accessibility of each part facet is determined from the vertices of all the spherical triangles. The accessibility data is arranged in the matrix form

where rows represent spherical triangles and columns represent part facets. Each entry in the matrix describes whether a facet is accessible from a spherical triangle or not. If the accessibility of a facet from a spherical triangle is obstructed due to other part facets, the obstructing part facets are also stored at the corresponding entry in the matrix. This approach helps in determining the global accessibility instead of local accessibility as in the work of Woo [10]. However, accessibility determination using the STL format inherits the problems incorporated in the STL format itself. Priydarshi and Gupta [7] evaluated the accessibility of part facets from a set of candidate parting directions to automate the multi-piece permanent mold design process. The accessibility of each facet along the chosen candidate parting direction is determined by checking the obstruction of each facet with the rest of the facets on the object. For near-vertical facets, they slightly rotated the viewing direction in such a way that near-vertical facets become front-facing facets. Using the accessibility information of each facet along each candidate parting direction, the part boundary is divided into different mold piece regions. Out of these mold piece regions, a minimum number of mold piece regions are selected such that the entire part boundary can be covered. Their methodology is applicable to polyhedral objects as the free-form surfaces are approximated using the number of linear surfaces.

More recently, there have been attempts to determine accessibility with the aid of computer graphics hardware. Khardekar et al. [14] proposed a graphics hardware-based approach to test the moldability of geometric parts using two-piece permanent molds. Their methodology also identified the undercut faces of a part by using the depth texture capabilities of graphics hardware, that is, by using the depth values at each pixel location. Their running times grow only linearly with respect to number of facets in the solid model. The efficiency of their algorithms lies in the fact that groups of candidate parting directions are identified such that if any one direction in the group is undercut-free, all are, or if any one direction is not undercut-free, none are. Priyadarshi and Gupta [15] used the results of accessibility analysis determined by graphics hardware capabilities to identify various mold piece regions. To determine the accessibility, a given part is illuminated by two directional light sources located at infinity in the positive and negative parting directions. The regions that are lit by the upper and lower lights are marked as *core* and

cavity respectively. The regions in the shadow were marked as *undercuts*. However, they have assumed that each facet belongs to one mold piece region only.

2.1.2 Parting Lines and Parting Surfaces Generation

Ravi and Srinivasan [4] established the criterion for the generation of optimal parting surface based on nine factors: projected area, flatness, draw, draft, undercuts, dimensional stability, flash, machining surfaces, and feeders. Some of these factors are important for de-moldability while others are important for aesthetic and dimensional stability of the mold part. However, they did not address how to generate the parting surface based on the criterion. To determine the optimum parting line of molded parts, Weinstein and Manoochehri [16] developed an in-order tree structure whose leaves represent the surfaces formed by one mold half. The parting line follows the external edges of a set of surfaces in a given leaf. The edges forming the optimum parting line are determined by using an objective function. In their work, the objective function is defined as a function of parting line complexity, draw depth, number of undercuts, number of unique side-cores, and machining complexity. However, their approach determined the parting lines based on surfaces in convex regions only, whereas in practice the parting lines can pass through concave regions as well. Their approach cannot handle curved and free-form surfaces.

Fu et al. [17] classified the part surfaces into three groups (core, cavity and local-tools molded surfaces) based on their geometrical and topological characteristics and then determined parting lines using external edges of core- or cavity-molded surfaces. In this process, surfaces are considered part of the core or the cavity based on their visibility from the core and cavity sides of a given parting direction. However, if a surface *S* can be part of core as well as cavity, then it is assigned to core or cavity depending on the type of the majority of its adjacent surfaces, i.e., the surface *S* will be assigned to core if the majority of its adjacent surfaces are part of core. However, their methodology can give ambiguous results if the surface *S* has an equal number of core and cavity adjacent surfaces. Madan et al. [18] classified the part surfaces into three categories based on the results of the scalar product (positive, negative, and zero) of the surface normal with the parting direction. The largest edge loop between positive and negative surface forms the

parting line, if the positive and negative surfaces are directly connected with each other. Otherwise, the parting line is determined by identifying an edge loop on the surfaces separating the positive and negative surfaces. However, the implementation aspects of identifying the edge loop are not discussed. Chakraborty and Reddy [19] determined the parting line and best pair of parting directions for a two-piece permanent mold from an STL part. They also classified the facets into three regions, positive, negative and perpendicular, by testing the scalar product of the facet normal and the parting direction. The perpendicular facets are grouped with positive and negative facets in many combinations to determine the parting line having the maximum flatness factor, where the flatness factor is the ratio of the projected length of the parting line to the original length of the parting line. However, the authors did not consider the presence of protruded undercuts on the perpendicular region.

Wong et al. [20] proposed a methodology based on slicing a CAD model to determine the optimal parting lines. In their method, parting directions are determined heuristically along principle axes, and then the algorithm slices the part using uneven slicing planes that are normal to a chosen parting direction. The possible parting lines for each parting direction are determined by using the intersection profile of the part on each slicing plane. They developed optimization criteria to determine the best parting line among the alternatives. Their optimization criteria is based on flatness of the parting line, draw distance, projected area, undercut volume, undercut length and relative positions of external undercuts to the mold. However, the parting direction is determined heuristically and it cannot be guaranteed that a correct parting direction is selected. Rubio et al. [21] also proposed an approach for determining parting line and part de-moldability based on part slicing. Their methodology first involved the slicing of the part by planes that are normal to a given parting direction. Then they analysed the intersection profile on each slicing plane to divide the part into different mold zones (i.e., core, cavity and side-cores) and determined the parting lines. However, their methodology cannot determine the optimal parting lines.

2.1.3 Mold Features Recognition

To automate the side-core design process, it is important to recognize the mold features. In the molding context, a feature can be classified broadly as a depression or a protrusion. For automatic feature recognition, Nee et al. [22] classified molding features based on their topology and geometry. The undercut features are classified based on whether the feature can be molded using side-cores or form-pins. Then the features are identified based on the characteristics of edge-loops connecting target and adjacent surfaces, and their release directions are determined. To determine the optimum parting direction, the number and volume of undercuts in each release direction are considered. Their method can handle cylindrical, conical, spherical, revolved and B-spline surfaces. Later, Fu et al. [23] recognized undercut features having blending surfaces (fillets) by considering the virtual edges of target surfaces, where virtual edges of target surfaces constitute the external edge-loop of the target surface and its blending surfaces. The release direction and the draw range of the undercuts are computed based on the V-map of the surfaces constituting the undercut. However, the criteria to identify the blending surfaces can be used only for cylindrical-shaped surfaces.

Yin et al. [6] developed a methodology to recognize interactive depression undercut features and to determine an optimal parting direction using volume decomposition. In this approach, regularized difference between the part and its convex hull were performed to obtain maximal connected components (MCCs). An MCC is identified as an interactive potential undercut feature (INPUF) if its freedom cone is zero i.e., if an MCC cannot be translated to any direction without obstruction with part in 3-D. The volume of INPUF is decomposed into convex cells and reconstructed by connecting the small cells using a non-directional blocking graph. Then the optimal parting direction is determined by minimizing the number of potential undercut features. Subsequently, undercut features in the optimal parting direction are identified from the set of potential undercuts. However, this approach cannot be used for parts having free-form or ruled surfaces. Later, Bidkar and McAdams [24][25] decomposed the part into small elements (cubes) and their accessibility and relationship are used to recognize features and assess part moldability. Each cube is classified into two categories based on its solid volume: *solid* and *void*. The mesh size is determined by the smallest feature that can be produced

by that manufacturing process. Appropriate and variable mesh size is necessary to make the approach robust.

Joshi and Chang [26] introduced a graph isomorphism-based approach for recognizing polyhedral machining features by using an attributed adjacency graph (AAG), where an AAG represents the low level information of B-rep model by nodes (each node refers to a unique face), arcs (each arc refers unique edge) and arc attributes (each arc's attribute is *0* or *1* based on the angle between faces sharing that arc/edge). They recognize machining features (such as pockets, slots, steps, blind steps, and polyhedral holes) from a B-rep solid model. Later, Ganter and Skoglund [27] presented a graph-based approach to extract three types of side-core mold features: internal voids, single and multi-surface holes, and boundary perturbations. They also generated core-prints of the casting pattern by determining the convex hull of recognized concave features. However, their approach is limited to depression undercuts. Ye et al. [28] proposed a hybrid method (graph-based combined with hint-based) to recognize interacting mold undercut features by searching the cut-set of undercut sub-graph. Face and edge properties are used as hints to recognize interacting features. Their approach can successfully recognize undercut features from parts having free-form surfaces. However, the hybrid method cannot determine the feature volume and release direction. Kumar et al. [29] proposed a polyhedron face adjacency graph to recognize undercut features of parts having Bézier surfaces. To obtain the polyhedron face adjacency graph, they divided the Bézier surfaces into their planar polyhedron surfaces. Optimal parting lines are generated while considering the flatness and draw distance. Recently, Zhang et al. [30] proposed an approach to recognize depression and protrusion mold features based on the curvature properties of the entities in B-rep model. To identify the features of B-rep model with sharp edges and vertices, they replaced the sharp edges and vertices with the curvature surfaces according to their geometrical and topological information in the original model. However, replacing the vertex and edges with the curvature surfaces can result in the loss of small features.

2.1.4 Mold Segments Design

There are a few attempts published to automatically generate the mold components and the reported work is limited to simple mold parts. Hui and Tan [9] proposed a concept

using solid sweep and regularized Boolean operations to generate the core and cavity of an injection molded part but limited to parts having no through holes. Shin and Lee [31] developed a procedure for side-core extraction. The interference faces are identified by applying polygon overlap tests. Then side-cores are generated through Euler operations. However, their method can generate ambiguous results when two or more undercuts are overlapping. Later, Fu et al. [17] generated core and cavity blocks based on given parting lines. But the generation of local-tools for undercut features were not covered. Recently, Fu [32] has expanded the approach [17] to design core, cavity and side-cores. The release direction of undercuts is determined and the undercut with same release direction are grouped. The side-core main body and side-core head are generated and unionized to form the side-core. However, intersecting features are not handled in their approach. Ye et al. [33] extended their own approach [28] of using attributed adjacency graphs to recognize interactive undercuts and to generate side-cores. To form a side-core, a bounding box of the blockage portion of the undercut feature was created and trimmed with all the faces of undercut feature. Then the side face of the bounding box was swept along the release direction to get a linkage portion. The union of linkage portion and blockage portion formed the side-core for undercut feature. However, this approach cannot handle intersecting features [19]. Banerjee and Gupta [34] presented an algorithm to automatically generate the shapes of side-actions using the STL format. They defined side-actions as secondary mold pieces that are removed from the part using translation directions other than mold opening directions. Given a mold opening direction, a side-action translation motion is generated by determining the collision free 2-D translation space for horizontal and vertical accessibility for each undercut facet. In their approach, a side-action is considered to be retracted in a plane perpendicular to the mold opening direction only. However, the approach gives an optimal solution only if each connected undercut region on the part requires three or fewer side-actions. Recently, Ran and Fu [35] recognized the inner and outer undercut features based on the topological relationship between the geometric entities. In their approach, they identified the part surfaces that can be mouldable using core, cavity, side-cores and internal pins using the part edge characteristics. After identifying the surfaces requiring internal pins, the authors determined whether there would be enough space for internal pins translation without

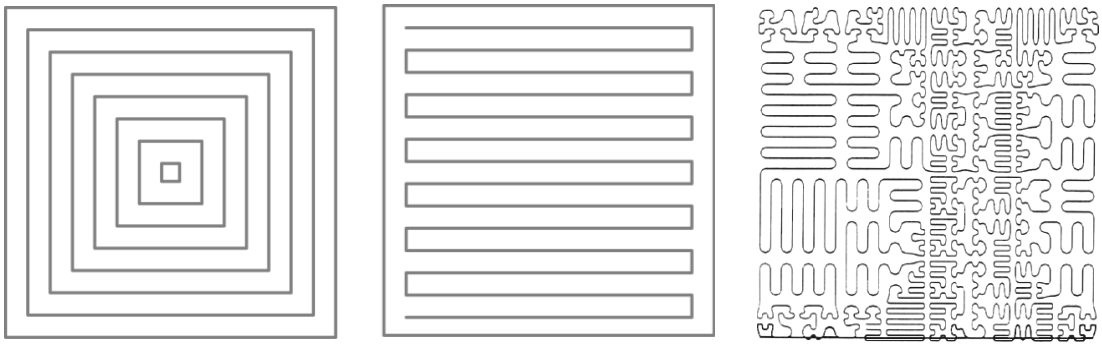
interfering with each other or with the core. If no interference is detected, internal pins are designed. However, the protrusion undercut features cannot be identified with this approach.

2.2 Tool-Path Generation

A lot of work has been done in the field of tool-path generation for machining sculptured surfaces. There are several tool-path planning methods [36][37][38][39], including direction-parallel, contour-parallel offset, iso-parametric, constant scallop height, and space-filling curves. The functional requirements of the tool-path planning methods are to machine efficiently in the minimum time, to produce fine surface quality without tool-marks, and to machine without gouging. The research done to realize these requirements in practice is discussed in this section, with a focus on die-cavity machining.

2.2.1 Tool-Path Profiling Methods

For cavity machining, there are three main types of tool-path strategies [40]: contour-parallel offset (CPO), direction-parallel (DP), and space filling curves (SFC), shown in Figure 2-2. In contour-parallel offset machining, the cutting is performed along curves offset to the boundary curve. The cutting is performed along line segments parallel to each other in direction-parallel machining. The concept of machining using space filling curves is relatively new. Space filling curves have been used for recursive algorithms and to draw images on computer screen.



(a) (b) (c)

Figure 2-2: Different types of tool-path profiles: (a) Contour-parallel, (b) Direction-parallel, and (c) Space filling curve [41].

The contour-parallel method is preferred for pocket machining because it is able to avoid alternating feed direction and results in better surface finish [42]. The curve offsetting problem has been attempted in three different methods: Voronoi diagram, pair-wise offset, and pixel-based approach.

Persson [43] proposed the use of the Voronoi diagram for determining the offset curves of the arbitrary shaped pocket boundary curves consisted of lines and arcs. They partitioned the whole pocket area into sub-areas using bisectors of the bounding curve entities. The offset curves are achieved by simply connecting the offset segment of each sub-area. However, the algorithm cannot generate a tool-path for pockets having islands within. Using the Voronoi diagram, Held et al. [44] developed CPO tool-path for pockets having islands within. To generate the connected components of the offset curve, the neighbourhood relations between monotonic pouches, called *proximity maps*, are used. Monotonic pouches are sub-areas in which the distance from the boundary decreases monotonically while moving away from the innermost point of the pouch. Although the approach is efficient, it can construct offset curves only when the number of islands is much less than the total number of contour elements of the boundary. Later, Jeong and Kim [40] constructed a Voronoi diagram of free-form shaped pocket for generating contour parallel offset tool-paths. To construct the Voronoi diagram, the curve is divided into segments at the points where its curvature is positive and a local maximum. The bisectors curves are formed by sequence of discreet points to divide the pocket area into sub-areas. Then a CPO tool-path is generated by connecting the offset curves of each curve segment.

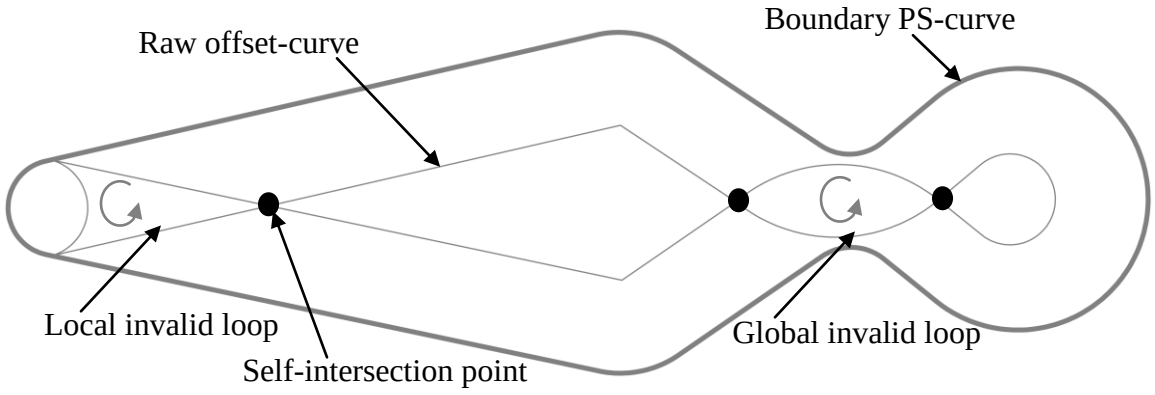


Figure 2-3: Geometric entities in pair-wise offset approach [46].

Hansen and Arbab [45] introduced the pair-wise offset approach for generating contour-parallel offset tool-paths. In their approach, an offset segment for each boundary element is generated and the gaps between the offset segments are closed by the trimming arcs. Then all pair-wise intersections in these raw offset curves are detected and invalid loop segments are removed. However, the process of invalid loops removal is prone to numerical errors. Choi and Park [46] extended the pair-wise offset approach [45] by removing local invalid loops, shown in Figure 2-3, before constructing a raw offset curve. After forming the raw offset curves, the algorithm detects and removes the global interfering regions to generate the valid offset curves. However, the algorithm cannot automatically generate tool-path for cavities having islands. Later, Park and Choi [47] expanded the algorithm to generate tool-paths for cavities having islands by checking the inclusion relationship between the raw offset curves. They also identified the uncut regions that exist between two successive tool-paths using the cutter radius and offset distance, and generated the tool-path for these uncut regions. The types of uncut regions are shown in Figure 2-4. However, the uncut regions left due to the presence of obstacles cannot be identified using this approach. Chuang and Yau [49] determined the contour-parallel offset tool-path for triangulated surface models. They determined the local interfering regions by using segment-segment intersection operations, and removed these regions by using circles tangent to both ends of the interfering segments. The global interfering regions are detected by checking the loop direction and are removed by polygon Boolean operations.

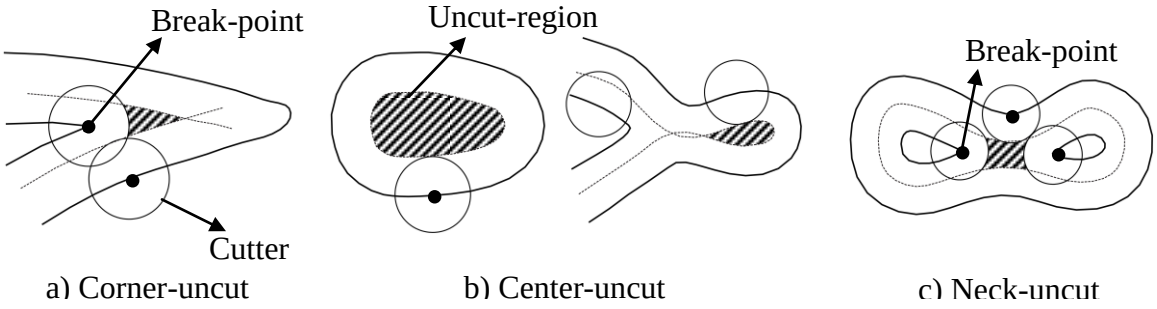


Figure 2-4: Uncut regions in contour-parallel offset tool-path [48].

Choi and Kim [48] used a pixel-based approach to generate a CPO tool-path for die/cavity pocketing. The cutter movement is simulated along the boundary curves and

the pixels within the area swept by the tool are marked. The offset curve is formed by identifying the unmarked pixels directly connected to the swept area pixels. After generating the offset curve, the uncut regions are detected and cleanup tool-paths are generated for these regions. To reduce memory usage and to improve efficiency while detecting uncut regions, the regions left uncut by cutter are first approximated by using the cutter geometry, and then the cutter movement is simulated in 2D within the limited region. The method is efficient, but the islands within the cavity boundary cannot be handled with this method. Later, Park and Choi [50] gave an algorithm for extraction of the boundary of the area to be cut from Z-map model. The authors developed a connectivity-net between the nodes of a binary image obtained from the Z-map. From the connectivity-net, boundaries are extracted and the inclusion relationship among the boundaries is identified. However, the Z-map of the cutting area is not determined automatically and is taken as an input.

Other than CPO tool-paths, the direction-parallel tool-path is commonly used in the pocket machining because it is simple and can easily handle complex compound surfaces in a seamless manner. Using space filling curves, Cox et al. [51] proposed tool-path generation methodology for machining sculptured surfaces, where a space-filling curve in a unit square is a continuous curve that passes through every point of the square as the order of the curve increase to infinity. To generate the tool-path, the curves developed on the unit square are mapped to the sculptured surfaces. The scallops resulting from this tool-path are non-oriented and material removal is more than the zigzag tool-path. However, this tool-path is not used in practice because of frequent change of feed direction.

2.2.2 Tool Positioning Methods

Interference and gouge-free machining is an important functional requirement for tool-path generation. The gouging can be avoided by determining the cutter location data (CL data) from the part surfaces. A number of methods have been reported in the literature for determining the CL data. First, CL data can be determined from the offset surface, called a CL surface, of the surface model. Second, the cutter contact point is determined from the surface model and the cutter location data is calculated from the cutter contact point

using the tool geometry. Third, a polygon model is constructed from the surface model and the CL data is directly calculated from the polygon model.

Molds and dies usually consist of multiple sculptured surfaces and can be modelled on a computer by using continuous surfaces or triangular polygons or point clouds. The work reported in the literature to generate tool-paths from compound surface models, STL models and scanned models is discussed in this section.

To machine compound surfaces, Choi et al. [52] generated NC milling tool-paths for compound surfaces where a compound surface is a collection of topologically unrelated surface elements specified in the domain of interest [53]. The cutter contact points are determined by finding the intersection points of a line at a given x and y, pointing towards the z-axis (the tool-axis) with the primitives of a CSG (constructive solid geometry) part. To determine the tool-path, a grid is defined over the domain and the cutter contact point is obtained for each grid point. However, the generated tool-path is not gouge-free. Later, Choi and Jun [54] proposed a methodology for interference-free machining of sculptured surfaces using ball end cutter. The cutter passes are generated on the surfaces using iso-parametric curves and the cutter location points are calculated from the cutter contact data using the cutter geometry. For a given cutter radius, the interference points are identified by analyzing the surface normal of the points adjacent to the cutter-contact points. The interfering points are removed to generate a gouge-free tool-path. Suh and Lee [55] obtained the cutter location points for machining pockets with convex or concave free-form surfaces bounded by lines, arcs, and free-form curves. First, the authors generate the valid offset profiles and then they sweep the profiles in the offset direction to yield the ruled surfaces. The ruled surfaces are intersected with the bottom surface of the pocket to get a series of contact points. The cutter location points are obtained by using the geometry of ball end milling cutter. Cho et al. [56] developed the tool-path for multi-patch surfaces for three-dimensional machining operations in the parametric domain. In their approach, the surface patches are intersected with 2D planes whose normals are perpendicular to the tool-axis. The cutter location points are determined for the curve resulting from the intersection. However, the cutter location points are obtained for only ball end milling tools.

Tekeuchi et al. [57] generated the NC data/cutter location (CL) data for the part surfaces from their offset surfaces calculated by using the inverse offset method. In the inverse offset method, the offset surface is generated by the center of the inverse tool when the tool moves on the surface of the work piece. Tang et al. [58] developed an algorithm to offset the part surfaces as well as the boundary curves of these surfaces. The offset surfaces are intersected with vertical planes to obtain the intersection curves. The curves are polygonalized to determine their upper envelopes which are used as the tool-paths. The authors have generated offset surfaces to accommodate different tool profiles, namely, ball-end, bull-nose, and flat-end. Later Choi et al. [59] discussed the advantages of the configuration-space (C-space) approach in the tool-path generation process for high speed machining of dies and molds. In the C-space approach, the entire space is divided using offset surfaces into three sub-spaces: free space, machining space, and gouging space. The free space corresponds to the volume in which the tool does not collide with any other object, machining space corresponds to the volume in which the tool should traverse for cutting, and the gouge space is the space in which tool movement results in gouging. The surfaces offset to the raw stock surfaces and the die-cavity surfaces are used to divide the space. Then the tool-path is generated after determining the step-length and path interval. Park [60] proposed a method for efficiently obtaining the tool-path data from the CL surface. The CL surface is in the form of a triangular mesh and contains the invalid triangles. In their approach, the CL surface is sliced with a vertical plane, and interference-free CL data points are obtained by intersecting the resulting line-segments with vertical lines. The authors improved the efficiency by avoiding the computation of unnecessary intersection points between the line-segments and the vertical lines. However, their work is focused on obtaining a tool-path using a ball end mill.

Kim and Choi [61] proposed a guide surface based method for clean-up machining of the part where the guide surface is a surface offset to the surfaces of the machining regions to be cut. The machining regions are the uncut regions after the finishing operation. The machining regions are automatically located along concave corner areas. Iso-parametric curves are defined on the guide surface and are projected on

to the machining region to obtain the clean-up tool-path. However, the method cannot detect the uncut volumes that are not part of concave regions.

Hwang [62] triangulated the compound surfaces to generate an interference-free tool-path using a ball-end milling tool. First, the surface points are obtained by sectioning the compound surface with parallel section planes. These surface points are converted into a triangular mesh and interference-free cutter location points are obtained by checking the interference with vertices, triangles, and edges of facets under the tool shadow. The machining tolerance is maintained within the limits by controlling the forward step length. Later, Hwang and Chang [63] extended the methodology proposed by Hwang [62] to machine the compound surfaces using filleted and flat end mills. Chuang et al. [64] modified Hwang's method [63] and proposed a simple method that can be applied to different tool shapes, namely, ball, fillet and flat-end cutters. The generalized calculations for the cutter locations determination for filleted-end cutters are applied to ball and flat end cutter by changing the fillet radius. Recently, Patel [65] gave a methodology to determine the cutter location point for different cutter shapes by analyzing the geometry. To improve the efficiency while determining the vertices, edges and triangles under the tool shadow, they suggested the bucketing method in which the entire machining region is divided into a number of sub-regions (buckets) and only the triangles in the required buckets are checked.

Lin and Liu [66] suggested a methodology to generate the tool-paths from data points. They constructed a Z-map model by linking the points adjacent to each other for roughing and finishing. For the roughing operation, the Z-map model is sliced with horizontal planes, and cut-areas are identified by using a pixel-based approach. For determining the cutter location for the finishing tool-path, the height of cutter middle point is adjusted to avoid interference with data points under the tool shadow. The method is robust but requires a large amount of memory and excessive computation to achieve the desired precision. Park and Chung [67] gave another approach to generate the tool-paths directly from the measured data. They used the point sequence curves to determine the cut area for rough machining using the procedure given by Park and Choi [50]. To obtain the finishing zigzag tool-path, the point data model is sliced with vertical planes and the point sequence curves under the tool-shadow are offset in the tool-axis

direction and then projected onto the slicing plane. The intersection points among these projected curves are determined and the highest parts of the projected curves are determined by making use of the intersecting points, as suggested by Park [60]. This method can generate only zigzag finishing tool-path for a ball end mill cutter type. Yau and Hsu [68] generated tool-paths from the point data by lowering the cutter along the tool-axis until there is no interference between the tool and the surface points. The cutter contact point is used to find the cutter location point using the cutter geometry. The step errors are estimated and extra cutter location points are inserted if the errors exceed the machining tolerance. Recently, Yingjie and Liling [69] generated a guide surface from an input point cloud, and planned the tool-path on the guide surface as point sequences. These points are projected onto the point cloud to obtain a set of approximate projecting points. Then the cutter location points are calculated from the projected points. However, the approach cannot generate accurate tool-path for surfaces with sharp corners.

2.3 Scope and Goals of Present Work

The literature review presented above indicates that many approaches have been taken to automate the mold design and manufacturing processes. However, there are still issues to be addressed to fully automate the mold design and manufacturing process.

To determine the mold parting direction, most of the work is done by analyzing the accessibility of the part surfaces from different directions. The global accessibility is analyzed either by throwing discretized rays in the parting direction [9][11] or by checking the accessibility of each facet in STL format [7][13] or by using graphics hardware [14][15]. As discussed earlier, the accuracy of the ray-based approach is dependent on the distance between the grid points. Moreover, the number of grid points becomes enormous as the complexity of the solid increases, and the time taken for accessibility analysis may be unacceptable. In approaches based on the STL format, the part surfaces are approximated by a number of triangles and the topological information stored in the B-rep model is not considered, which is important for recognizing mold features and extracting machining information. The graphics hardware-based approaches have the same drawback, as the part input for these approaches are in the STL format.

Many attempts to recognize features have been reported. However, the complexity of solving feature recognition problems is limited to simple features with no

tolerance for arbitrary interactions between the features. Furthermore, there are only a few attempts to automate the design of mold segments.

There have been many tool-path generation approaches for die-cavity machining. As discussed earlier, contour-parallel offset machining is preferred for pocket machining as it is able to avoid alternating feed direction, and results in better surface finish. However, little work [45][46][49] has been done in identifying and discarding invalid local and global loops. Furthermore, the roughing tool-path leaves some uncut regions, shown in Figure 2-4. There has been little work [47][48] to identify and machine these uncut regions. Machining efficiency can be improved by identifying and limiting the cutter movement to uncut regions.

The literature survey shows that automatic design of mold is an active area. The current weaknesses are that the automation process does not recognize the intersecting mold features, and part surfaces' accessibility information is not combined with geometrical and topological information during feature recognition. The main problem with automatic machining is that the uncut regions formed due to presences of obstacles are not identified, therefore these regions are not considered during tool-path planning. The identification of these uncut regions is important to limit the tool-movement within the region boundaries and machine efficiently.

Based on the above analysis, the primary goal of the present work is to determine a methodology for automatic mold design methods that does not depend on descretizing the part. The second goal is to determine a method for the development of tool-path planning that can accommodate intersecting and complex mold features. To fulfill these goals, the objective of current work can be stated as follow:

- To analyse the accessibility of a part to identify a parting direction that does not require side-core/cavity and to do so without discretizing the part.
- To develop automatic feature recognition methodology for tool design using the accessibility information and taking the part geometry as an input (in B-rep format) and its parting direction. Feature recognition must work on complex geometry of molded parts.
- To automate the design of core, cavity and side-cores.

- To automate the tool-path generation process for rough machining of mold components.
- To identify the regions left uncut by the roughing tool, and to automate the clean-up tool-path generation process only for these uncut regions.

The work done to meet the above outlined changes is discussed in the next few chapters.

Chapter 3

Determination of Undercut-Free Parting Directions

In injection molding, undercut-free parting directions are preferred. The lack of undercuts eliminates the use of side-cores, and the part can be produced economically using only core and cavity. A parting direction can be undercut-free if all of the part surfaces are fully accessible in that direction. Therefore, an undercut feature can be avoided by changing the parting direction; a mold surface can be inaccessible from some directions, but may be fully accessible from some other directions. The finding of these undercut-free parting directions is important for minimizing the number of molding elements, as well as for reducing the complexity of the molding process.

During the mold design process, the parting direction is kept in mind before designing most of the molding parts [34]. Therefore, a system is required that can aid the designer in identifying an undercut-free parting direction out of a given set of directions, if one exists. The objective of the present work is to determine the accessibility of each face of the part and to evaluate part de-moldability in a given set of directions.

In this work, each direction from the set of directions is evaluated in sequence to determine the part de-moldability and is called a *considered/candidate parting direction*. Two different methodologies are proposed for analysing the accessibility of part surfaces. In the first methodology, an algorithm analyses the accessibility of each surface using sweep and Boolean operations. The second methodology is based on coloring each surface of the part differently and then analyzing the accessibility of each surface using a pixel-based approach. The accessibility is analyzed in both of these approaches while retaining the information stored in the B-rep model. Therefore, in addition to determining an undercut-free parting direction, the results of the accessibility analysis can be combined with the B-rep information to be utilized in feature recognition. A comparison of both approaches is discussed in detail at the end of this chapter.

This chapter is organized as follows. First, the surfaces are classified based on the orientation with respect to the parting direction and various terms are defined in Section 3.1. Then, the surfaces are classified based on accessibility in Section 3.2. An overview

of the technical approach is discussed in Section 3.3. The procedure for the Boolean-based accessibility analysis approach is discussed in Section 3.4 and the procedure for the pixel-based accessibility analysis approach is discussed in Section 3.5. Both of these approaches can be used to determine whether the considered parting direction is undercut-free or not. The methodology to determine an undercut-free parting direction is discussed in Section 3.6. The comparison of the both of these approaches is given in section 3.7. The approaches have been implemented on test parts and the results are discussed in Section 3.8.

3.1 Orientation-Based Surface Classification

In two-piece conventional molding, core and cavity separate in opposite directions during de-molding. The two sides of the parting direction correspond to the separation directions of the core and the cavity, respectively. One of the two sides of the parting direction is called the positive parting direction ($+\vec{d}$) and the other is called the negative parting direction ($-\vec{d}$). The orientation of the surface normal with respect to the positive side ($+\vec{d}$) of the candidate parting direction is used for orientation-based classification of each surface, discussed in this sub-section.

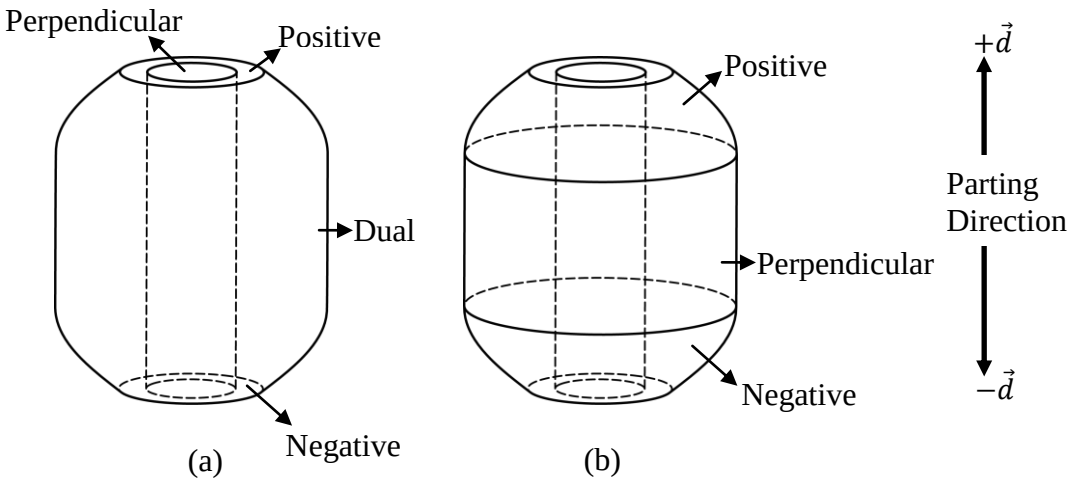


Figure 3-1: Orientation-based surface classification.

The part surfaces (either free-form or planar) are classified into four categories (shown in the Figure 3-1(a)), namely, positive, negative, perpendicular and dual, based on the cosine of the angle (θ) between the surface normal and the candidate positive

parting direction $+\vec{d}$. In case of non-planar surfaces, the surface normal at a number of surface points are evaluated to determine their orientation. The criteria for the orientation-based classification are given in Table 3-1. Most industrial parts have dual surfaces that exhibit the characteristics of positive, negative and perpendicular surfaces at different surface points. Such surfaces are divided into positive, negative and perpendicular surfaces by generating the silhouettes in the considered parting direction $\vec{d}$, as depicted in Figure 3-1(b).

Surface Type	*Cosine of angle (θ)*
Positive	*cosine* (θ) > 0 at all surface points
Negative	*cosine* (θ) < 0 at all surface points
Perpendicular	*cosine* (θ) = 0 at all surface points
Dual	*cosine* (θ) > 0 at some, but not all, surface points and *cosine* (θ) $\leq$ 0 at other surface points

Table 3-1: Orientation-based surface classification criteria.

3.2 Accessibility-Based Surface Classification

To determine de-moldability of a part/object (*O*) with respect to the candidate parting direction ($\vec{d}$), the surfaces of the part are classified based on their accessibility. The terms related to accessibility, shown in Figure 3-2, are defined next.

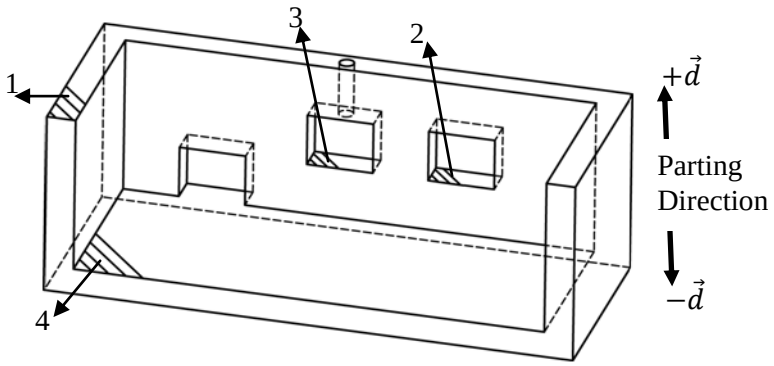


Figure 3-2: Accessibility-based surface classification of positive and negative surfaces: (1) *Fully-Accessible*, (2) *Fully-Inaccessible*, (3) *Outer-Boundary-Inaccessible* (*Partially-Accessible*), and (4) *Partial-Outer-Boundary-Accessible* (*Partially-Accessible*).

- A surface *S* is *Fully-Accessible* from a direction $\vec{d}$ if translation of *S* to infinity in the direction $\vec{d}$ does not cause any intersection with the interior of the object *O*.
- A surface *S* is *Fully-Inaccessible* from a direction $\vec{d}$ if translation of *S* to infinity in the direction $\vec{d}$ results in the intersection of all points on *S* with the interior of the object *O* at least once.
- A surface *S* is *Partially-Accessible* from a direction $\vec{d}$ if translation of *S* to infinity in the direction $\vec{d}$ results in the intersection of some, but not all, points on *S* with the interior of the object *O*.
- A surface *S* is *Outer-Boundary-Inaccessible* from a direction $\vec{d}$ if translation of *S* to infinity in the direction $\vec{d}$ results in the intersection of some, but not all, inside points and all boundary points of *S* with the interior of the object *O*.
- A surface *S* is *Partial-Outer-Boundary-Accessible* from a direction $\vec{d}$ if translation of *S* to infinity in the direction $\vec{d}$ results in the intersection of some,

but not all, inside points and some, but not all, boundary points of *S* with the interior of object *O*.

To facilitate the analysis of a *Fully-Inaccessible* surface during the Boolean-based approach, the surface is categorized into two types: *Type1* and *Type2*. If an inaccessible surface is shadowed by a connected set of interfering surfaces, it is classified as *Type1*, shown in Figure 3-3(a); otherwise, it is classified as *Type2*, shown in Figure 3-3(b) and (c).

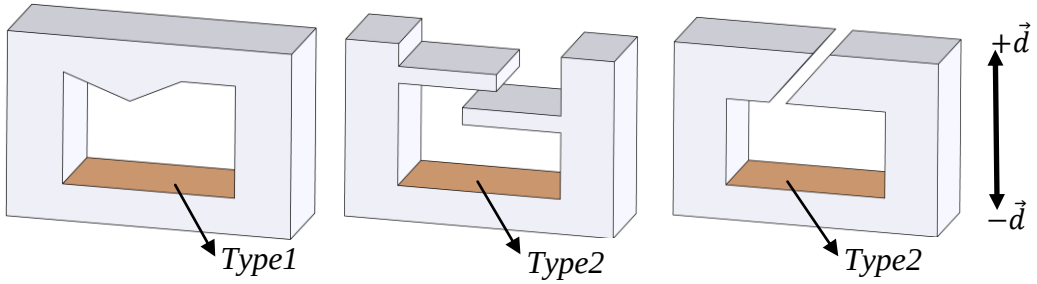


Figure 3-3: Inaccessible surfaces classification: (a) Type1, (b) and (c) Type2 inaccessible.

3.3 Overview of Technical Approach

To determine the de-moldability of a part in the considered parting direction $\vec{d}$, faces of the B-rep model are first classified based on their orientation with respect to the direction $+\vec{d}$, as discussed in Section 3.1, then classified based on accessibility. Orientation-based classification is required to avoid unnecessary operations in the accessibility analysis, e.g., surfaces classified as negative cannot be accessible from the $+\vec{d}$ direction, and it is therefore not required to check their accessibility from the positive direction. As a result, the orientation-based surface classification is utilized to reduce the lead time in accessibility analysis, as illustrated in Figure 3-4.

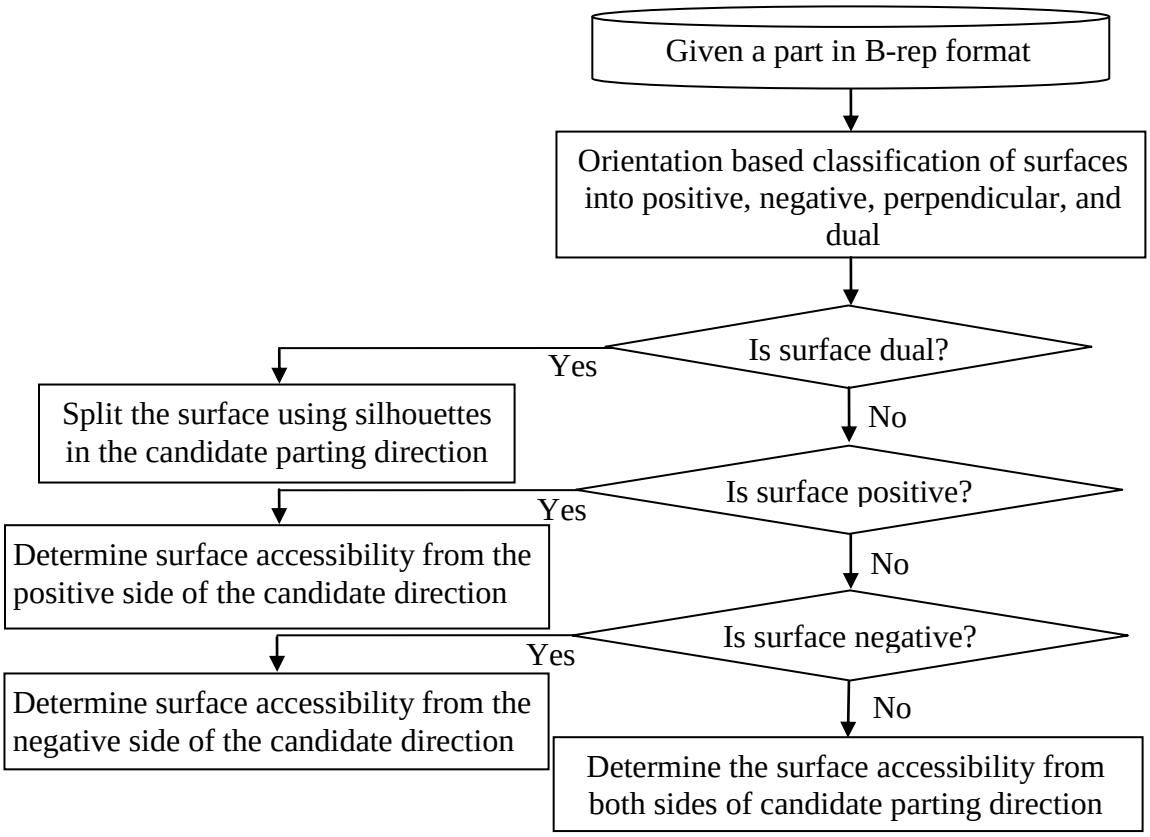


Figure 3-4: Role of orientation-based surface classification in accessibility analysis procedure.

During accessibility analysis, positive and negative surfaces are mainly classified into three types, namely, *Fully-Accessible*, *Partially-Accessible*, and *Fully-Inaccessible*, shown in Figure 3-2. Depending upon the accessibility of the outer edge boundary of the *Partially-Accessible* surfaces, these surfaces are further classified into two types: *Partial-Outer-Boundary-Accessible* and *Outer-Boundary-Inaccessible*. The further classification of the *Partially-Accessible* surfaces is useful for splitting them into *Fully-Accessible* and *Fully-Inaccessible*, if required during the feature recognition process.

The perpendicular surfaces can be fully accessible from both positive and negative parting directions, if not obstructed by any other surface. Therefore, perpendicular surfaces are analyzed in both $+\vec{d}$ and $-\vec{d}$ directions and are classified in the each direction into three categories: *Fully-Accessible*, *Partially-Accessible*, and *Fully-Inaccessible*, shown in Figure 3-5. A *Partially-Accessible* perpendicular surface cannot be *Outer-Boundary-Inaccessible*; therefore, such a surface is not classified further.

The results of surface classification are used as input to determine if the considered parting direction is undercut-free and de-moldability is possible without side-cores and cavities, as well as to recognize mold features.

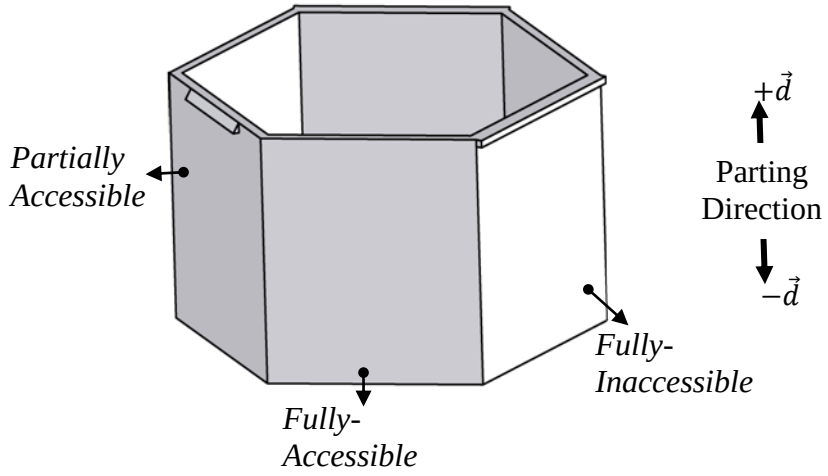


Figure 3-5: Accessibility-based classification of perpendicular surfaces from $+\vec{d}$ direction.

3.4 Boolean-Based Accessibility Analysis Approach

The accessibility analysis is carried out by selecting one surface at a time from a valid solid body and sweeping it in the considered parting direction to check its accessibility. The detailed methodology to classify a surface based on its accessibility is illustrated using a test part, shown in Figure 3-6, consisting of surfaces with different levels of accessibility from the considered parting direction ($\vec{d}$). The methodology adopted for the positive and negative surfaces is presented first, followed by methodology adopted for the perpendicular surfaces.

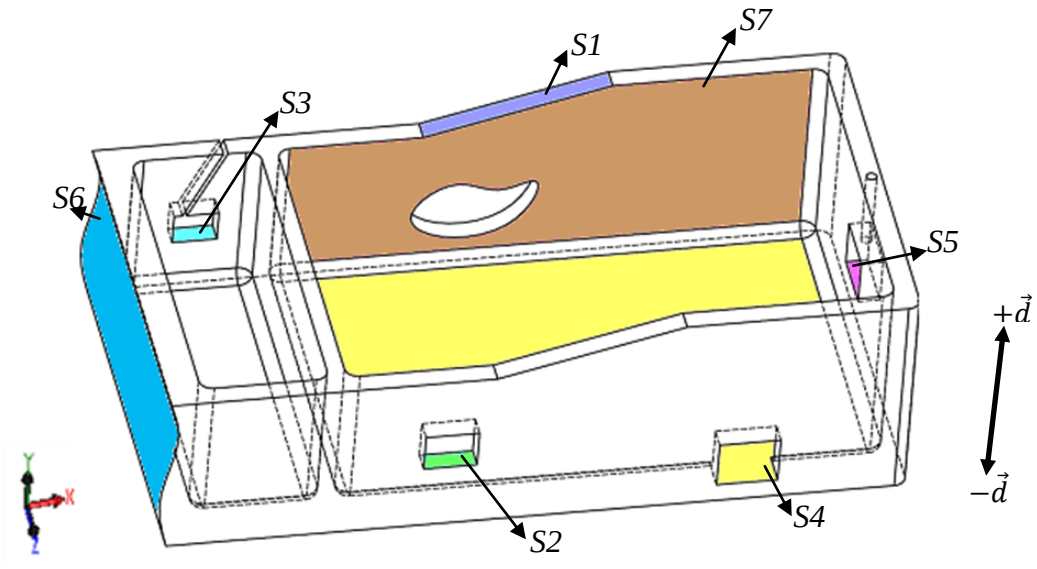


Figure 3-6: Test part to illustrate accessibility analysis procedure considering parting direction along Y-axis.

3.4.1 Positive and Negative Surfaces

Positive and negative surfaces are checked for accessibility in $+\vec{d}$ and $-\vec{d}$ directions respectively using the same methodology. A series of sweeping and regularized Boolean operations, along with the geometrical and topological constraints, are utilized to determine the accessibility level of the considered surface.

3.4.1.1 Fully-Accessible Surfaces

The considered surface is selected and swept along the $+\vec{d}$ direction, if the surface is positive, by a distance more than the maximum dimension (say twice the maximum dimension) of the part in that direction. In case of the negative surface, the surface is swept along the $-\vec{d}$ direction. The sweeping operation results in a valid solid body *B_Swept1*, shown in Figure 3-7.

The swept body (*B_Swept1*) corresponds to the space swept by the considered surface during the de-molding process. The considered surface will be fully accessible only if there is no interference between *B_Original* and *B_Swept1*. To check the interference, a regularized Boolean subtraction operation is performed between *B_Original* and *B_Swept1* as given by the following equation:

$$B\_Result = B\_Original \text{ -* } B\_Swept1. \quad (3\text{-}1)$$

The above regularized Boolean operation represents the subtraction of *B_Swept1* from *B_Original* to generate a resulting body (*B_Result*), as shown in Figure 3-7(c). After the regularized Boolean operation, the volume of *B_Result* is compared with that of *B_Original*. If both bodies have the same volume, then the considered surface is classified as *Fully-Accessible* and no further analysis is performed for this surface.

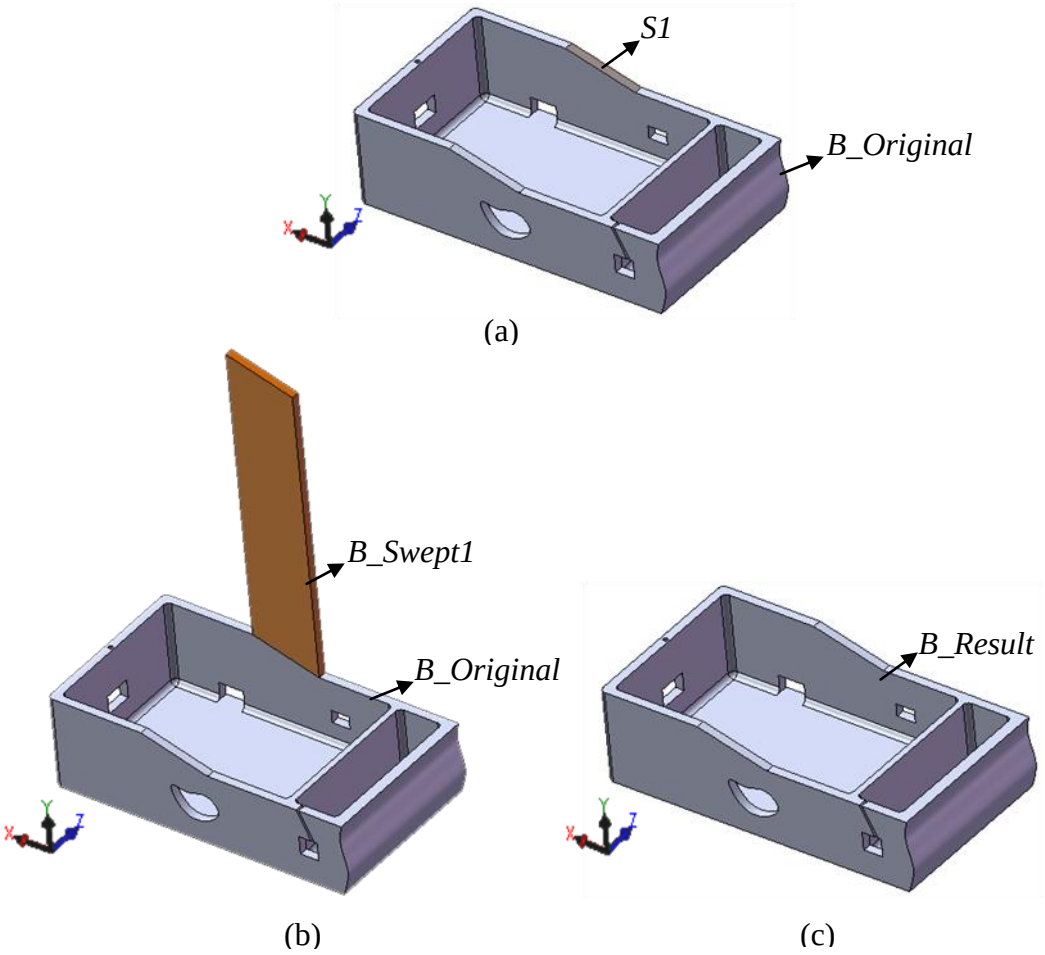


Figure 3-7: Accessibility analysis procedure for a *Fully-Accessible* surface, demonstrated using surface *S1* of Figure 3-6: (a) surface *S1* of *B_Original* (b) *B_Original* and *B_Swept1*, and (c) *B_Result* as per Eqn. 3-1.

For example, sweeping of surface *S1* in the $+\vec{d}$ direction, in Figure 3-7, results in a new swept body *B_Swept1*. *B_Swept1* does not interfere with *B_Original*. As a result,

the resulting body (*B_Result*) has the same volume as that of *B_Original* and the surface *S1* is classified as *Fully-Accessible* in the corresponding direction.

3.4.1.2 Fully-Inaccessible (Type1) Surfaces

If the surface under consideration is not classified as *Fully-Accessible*, then it is tested for the conditions of full inaccessibility and partial accessibility, shown in Figure 3-8, from the direction under consideration. Now, a regularized Boolean subtraction similar to Eqn. 3-1 is used, but *B_Original* is subtracted from *B_Swept1* and is expressed as

$$B\_Result1 = B\_Swept1 \text{ -* } B\_Original. \qquad (3\text{-}2)$$

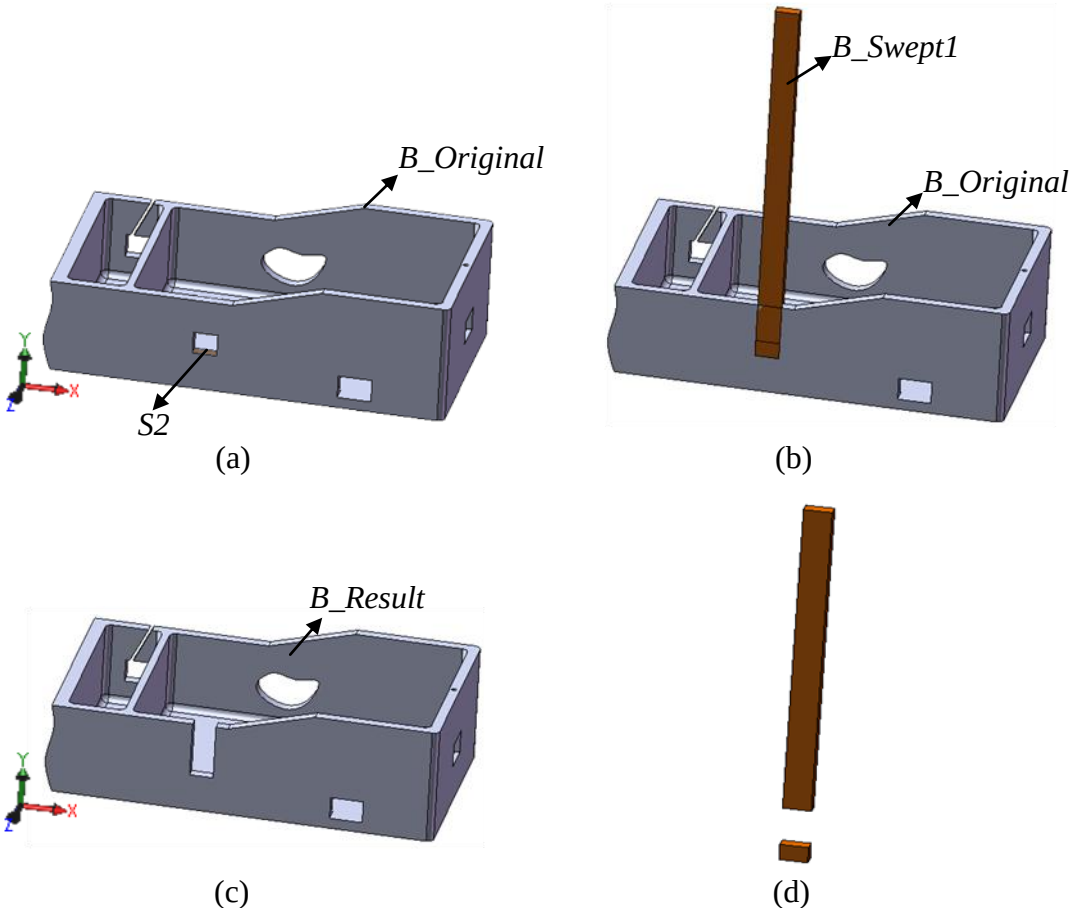


Figure 3-8: Accessibility analysis procedure for a *Fully-Inaccessible* surface, demonstrated using surface *S2* of Figure 3-6: (a) surface *S2* of *B_Original*, (b) *B_Original* and *B_Swept1*, (c) *B_Result* as per Eqn. 3-1, and (d) *B_Result1* as per Eqn. 3-2.

The above Boolean operation may result in a number of bodies depending on whether the surface is partially-accessible or fully-inaccessible. If *B_Result1* consists of two or more disjoined bodies, then the surface is *Fully-Inaccessible* (*Type1*) from the chosen direction, e.g., the surface considered in Figure 3-8 is classified as *Fully-Inaccessible* because the Boolean operation, as per Eqn. 3-2, on *B_Swept1* results in two disjoined bodies, as shown in Figure 3-8(d).

3.4.1.3 Fully-Inaccessible (Type2) and Partially-Accessible Surfaces

If the Boolean operation, as per Eqn. 3-2, results in a single body, as shown in Figure 3-9(d), then the considered surface can be *Fully-Inaccessible (Type2)* or *Partially-Accessible*. To analyze the level of inaccessibility, the surfaces of *B_Result1* are classified based on their orientation with respect to the considered positive parting direction. The negative surfaces of *B_Result1*, except the surface corresponding to the considered surface, are shown in Figure 3-9(e). Dual surfaces in *B_Result1*, if any exist, are segmented into positive and negative surfaces using the silhouettes in the considered direction.

To simplify the explanation, the considered surface is assumed to be positive. For a positive surface, the negative surfaces (one at a time) of the resulting body (*B_Result1*) are swept in the $+\vec{d}$ direction, as shown in Figure 3-9(f). Negative surfaces of *B_Result1* are swept in the previous operation because these correspond to positive surfaces of the original body that can throw their shadow on the considered surface. The sweeping distance of the surface is taken more (say 20%) than the maximum dimension of the body to which it belongs. The above operation results in a new body (*B_Swept2*) corresponding to each negative surface of *B_Result1*, as shown in Figure 3-9(f). A regularized Boolean operation on *B_Result1* and *B_Swept2* is performed as follows:

$$B\_Result2 = B\_Result1 \text{ -* } B\_Swept2. \qquad (3\text{-}3)$$

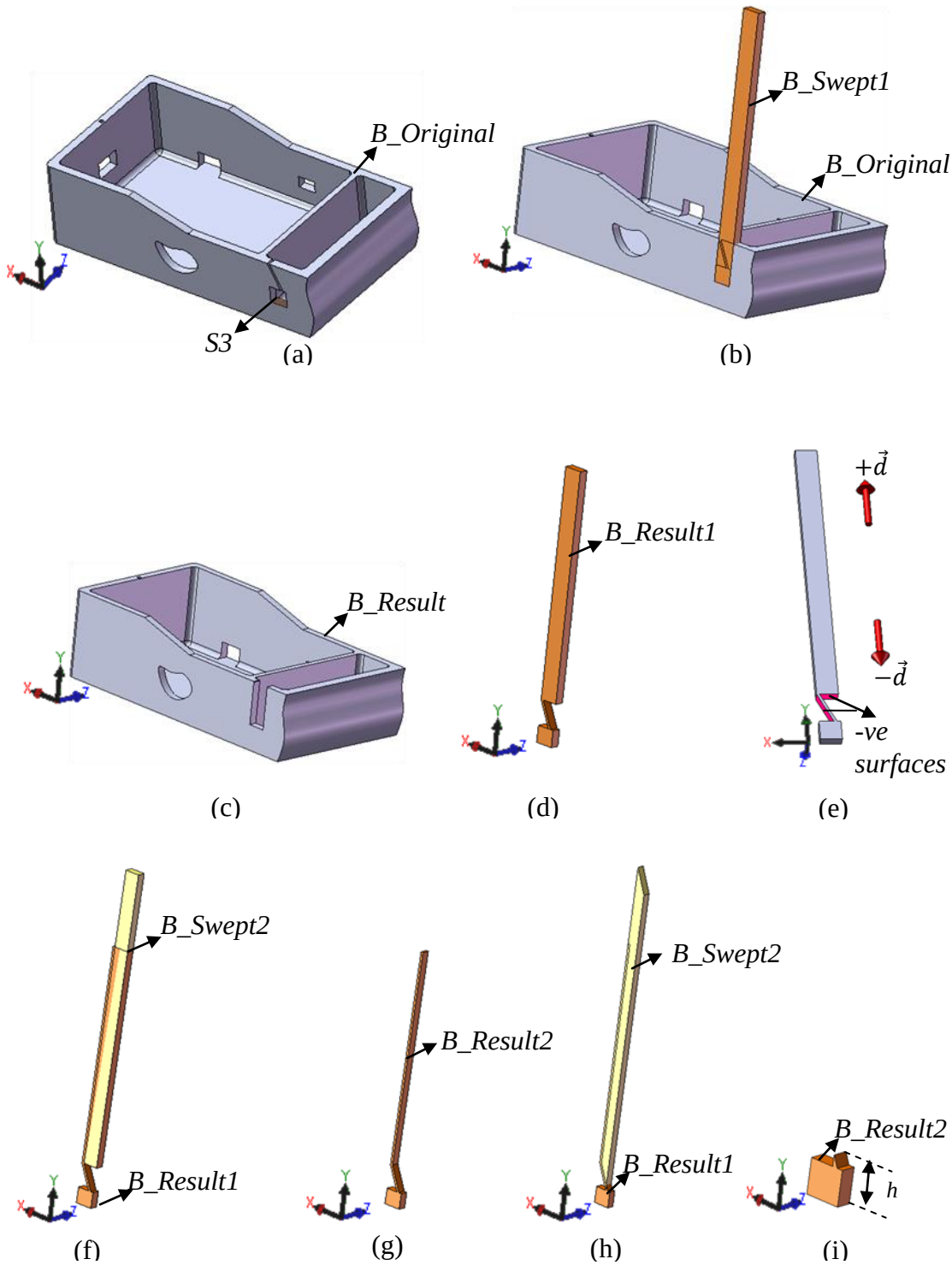


Figure 3-9: Accessibility analysis procedure for a *Fully-Inaccessible* surface, demonstrated using surface *S3* of Figure 3-6: (a) surface *S3* of *B_Original*, (b) *B_Original* and *B_Swept1*, (c) *B_Result* as per Eqn. 3-1, (d) *B_Result1* as per Eqn. 3-2, (e) -ve surfaces of *B_Result1*, (f) *B_Result1* and *B_Swept2*, (g) *B_Result2* as per Eqn. 3-3, (h) *B_Result1* and *B_Swept2*, and (i) *B_Result2* as per Eqn. 3-3.

The above operations, including sweeping of negative surfaces and Boolean subtraction from the resulting body (*B_Result1*), are repeated until there is any negative surface on the *B_Result2* except the surface corresponding to considered surface. At the end of each cycle, if any negative surface is found on *B_Result2*, then *B_Result2* is renamed as *B_Result1* and the sweeping and Boolean subtraction operations are repeated for that surface, shown in Figure 3-9(h). Next, if the height (*h*) of *B_Result2*, shown in Figure 3-9(i), is less than that of *B_Swept1* in the considered direction, then it is concluded that the considered surface is *Fully-Inaccessible* (*Type2*); otherwise it is classified as *Partially-Accessible*.

3.4.1.4 Classification of Partially-Accessible Surfaces

For further classification of *Partially-Accessible* surfaces, the outer edge boundary of the positive surface, shown using bold lines in Figure 3-10(g), that is at the uppermost level of *B_Result2* in the positive direction, is compared with outer edge boundary of the considered surface. If there is a full or partial overlapping of the edges when projected on a plane in the considered parting direction, then the surface is classified as *Partial-Outer-Boundary-Accessible*, otherwise, the considered surface is classified as *Inaccessible-Outer-Boundary*.

Figure 3-10 depicts the accessibility analysis procedure for a *Partially-Outer-Boundary-Accessible* surface. It is observed that outer edge boundary of the uppermost positive surface of *B_Result2*, shown in Figure 3-10(g), partially overlaps the outer boundary of the considered surface when their outer-boundary edges are projected on a plane in the parting direction. Whereas in Figure 3-11, the outer edge boundary of uppermost positive surface of *B_Result2*, shown in Figure 3-11(g), does not overlap the outer boundary of the considered surface, when their outer-boundary edges are projected in the parting direction. Therefore, the considered surface is of *Inaccessible-Outer-Boundary* type.

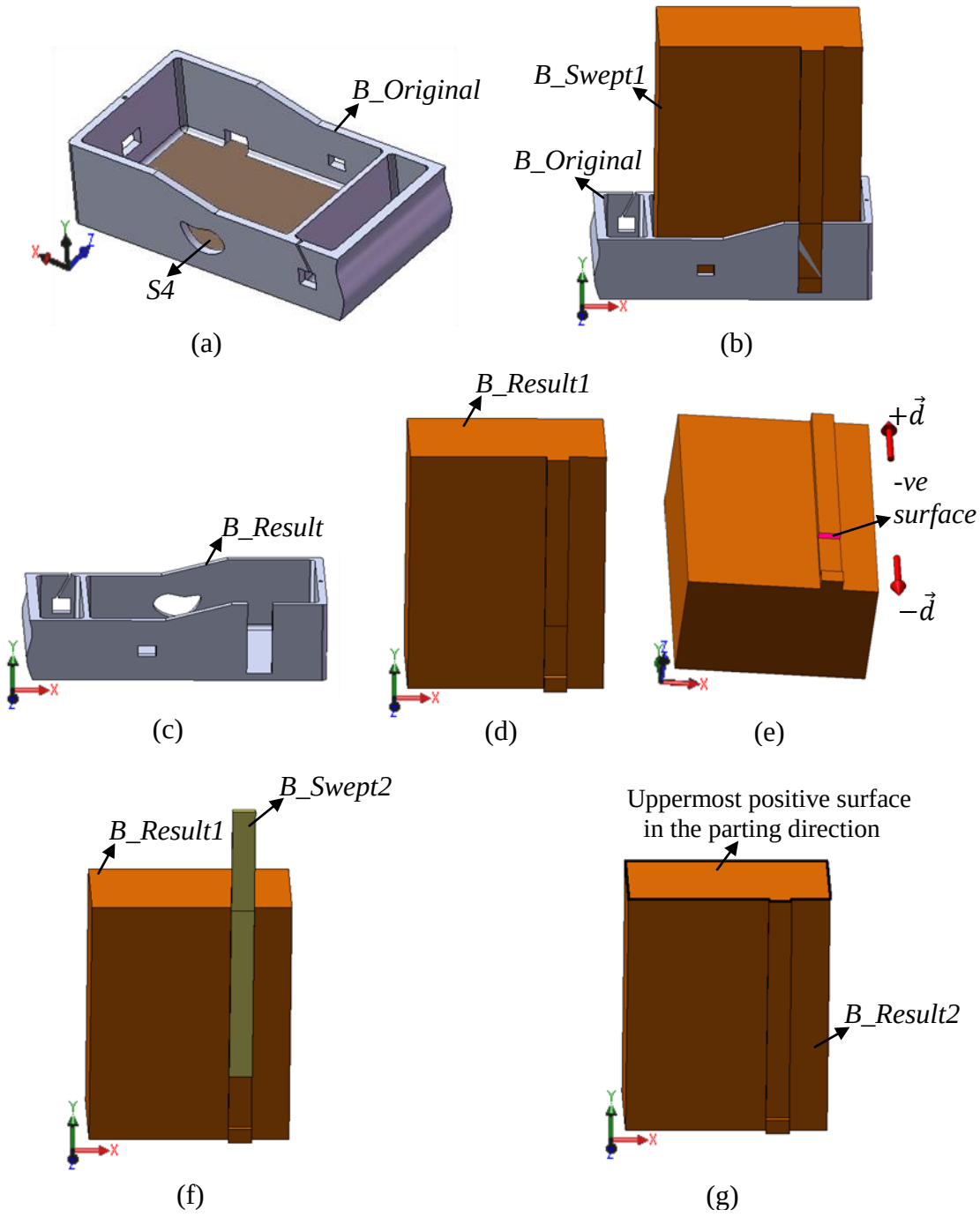


Figure 3-10: Accessibility Analysis procedure for a *Partial-Outer-Boundary-Accessible* surface, demonstrated using surface *S4* of Figure 3-6: (a) surface *S4* of *B_Original*, (b) *B_Original* and *B_Swept1*, (c) *B_Result* as per Eqn. 3-1, (d) *B_Result1* as per Eqn. 3-2, (e) -ve surface of *B_Result1*, (f) *B_Result1* and *B_Swept2*, and (g) *B_Result2* as per Eqn. 3-3.

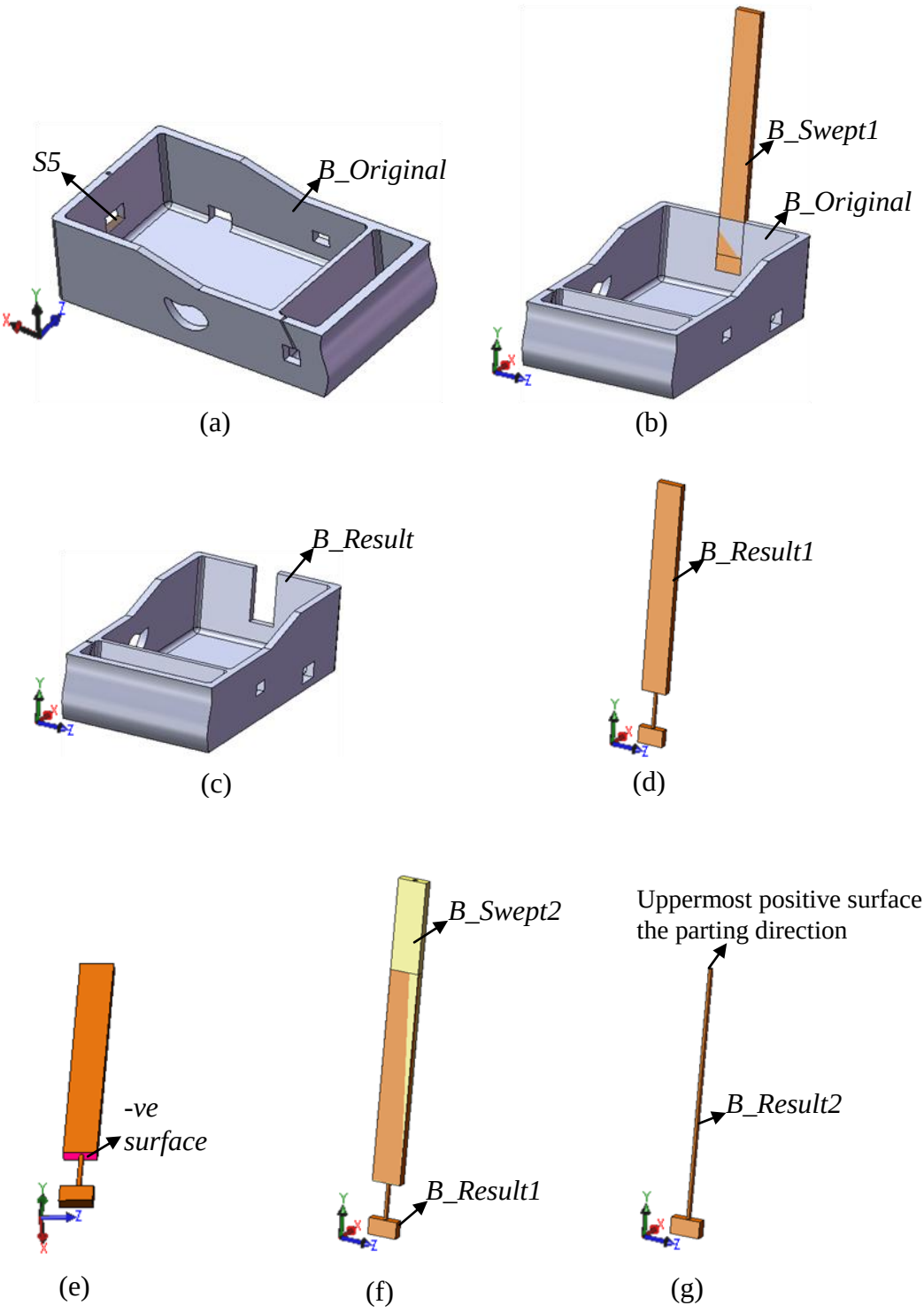


Figure 3-11: Accessibility analysis procedure for an *Outer-Boundary-Inaccessible* surface, demonstrated using surface *S5* of Figure 3-6: (a) surface *S5* of *B_Original*, (b) *B_Original* and *B_Swept1*, (c) *B_Result* as per Eqn. 3-1, (d) *B_Result1 a*s per Eqn. 3-2, (e) *-ve* surface of *B_Result1*, (f) *B_Result1* and *B_Swept2*, and (g) *B_Result2* as per Eqn. 3-3.

3.4.2 Self-Occluded Free-form Surfaces

The dual free-form surface is analyzed by splitting it into positive and negative surfaces by generating their silhouettes in the considered parting direction. The surface *S6*, shown in Figure 3-6, resulted in two negative surfaces and one positive surface after the splitting. The accessibility of positive and negative surfaces is analyzed in the positive and negative parting direction, respectively, using the procedure described in Section 3.4.1. The procedure for the positive segment of the surface *S6* is illustrated in Figure 3-12.

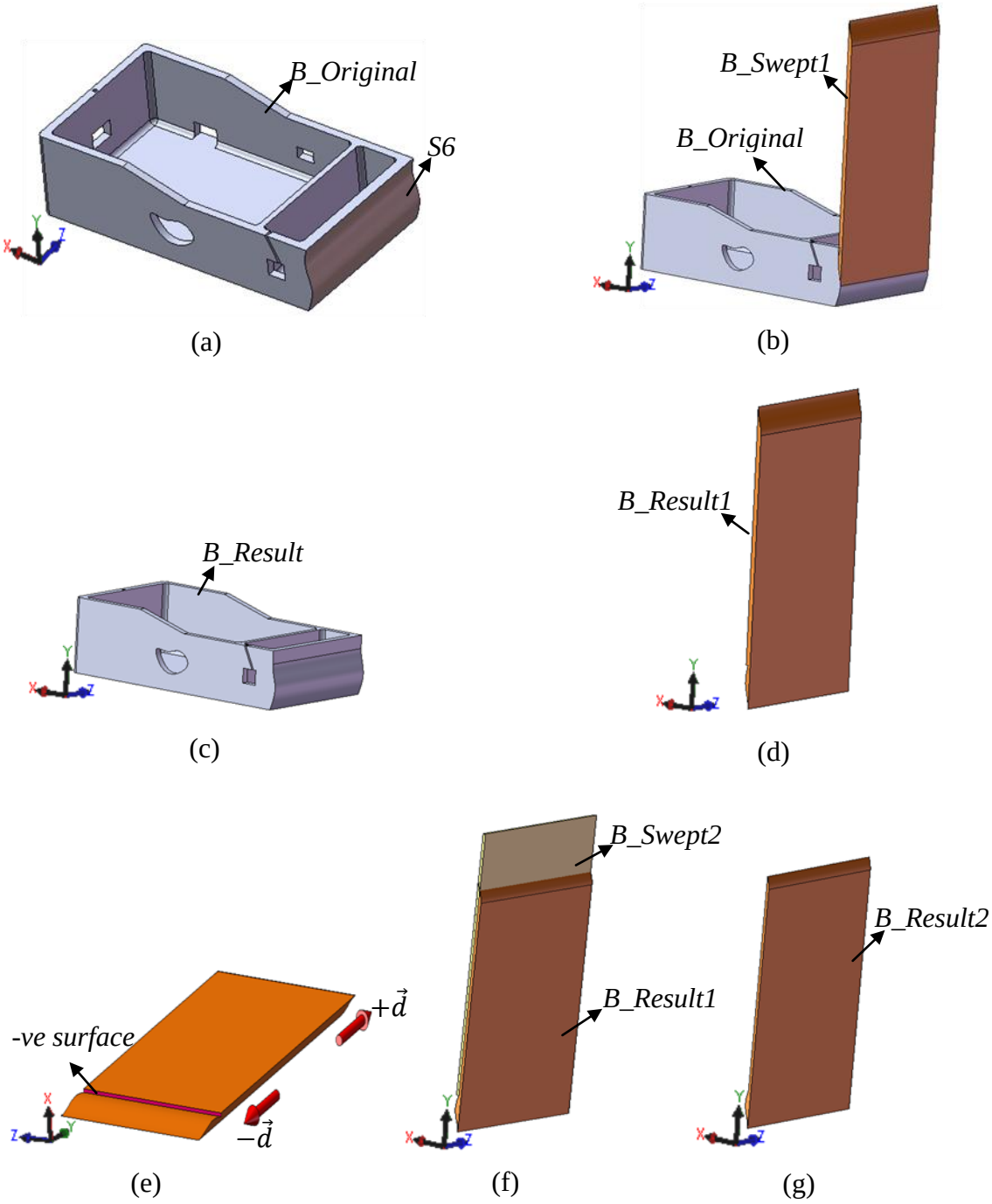


Figure 3-12: Accessibility analysis procedure for a dual free-form surface, demonstrated using surface *S6* of Figure 3-6: (a) surface *S6* of *B_Original*, (b) *B_Original* and *B_Swept1*, (c) *B_Result* as per Eqn. 3-1, (d) *B_Result1 a*s per Eqn. 3-2, (e) *-ve* surface of *B_Result1*, (f) *B_Result1* and *B_Swept2*, and (g) *B_Result2* as per Eqn. 3-3.

3.4.3 Perpendicular Surfaces

Perpendicular surfaces can be accessible from positive as well as negative parting directions; therefore, these are analyzed in both directions. Moreover, a perpendicular surface might not be visible, but can be accessible. To determine the accessibility of the perpendicular surfaces, Priyadarshi and Gupta [7] slightly rotated the near-vertical facets of a STL part such that the near-vertical facet becomes a front-facet in the parting direction. Later, Priydarshi et al. [15] determined the accessibility of near-vertical facets by thickening the concave contour edges. The methodology developed to determine the accessibility of the perpendicular faces in the B-rep format is presented next.

3.4.3.1 Fully-Accessible Surfaces

To check the accessibility of the considered surface from the parting direction $\pm\vec{d}$, an offset surface is created at distant ε towards the outward surface normal of the considered surface, as shown in Figure 3-13(b). The value of ε is decided by the minimum space required normal to parting direction to mold the part. The offset surface is thickened by the amount ε towards the inward surface normal of the considered surface, as shown in Figure 3-13(c), to generate a new solid body (*B_Offset*). The accessibility of positive and negative surfaces of *B_Offset* is evaluated to determine the accessibility of the perpendicular surface.

The surfaces of *B_Offset* are classified based on the orientation with respect to the considered parting direction $+\vec{d}$. If *B_Offset* has any dual surface, the surface is segmented into positive and negative surfaces using the silhouettes in the direction $\vec{d}$.

To analyze the accessibility from the $+\vec{d}$ direction, all the positive surfaces of *B_Offset* are swept by a distance more than the maximum dimension of the part (twice the maximum dimension) in the $+\vec{d}$ direction. This operation results in a number of swept solid bodies that are represented as *B_Swept1, B_Swept2 …… B_Sweptn*, as shown in Figure 3-13(d). A regularized Boolean operation is performed on these swept bodies as follows:

$$B\_Combined = B\_Swept1 +^* B\_Swept2 +^* B\_Swept3 +^* \ldots\ldots +^* B\_Sweptn. \quad (3\text{-}4)$$

The resulting body, *B_Combined*, is the result of regularized Boolean addition operation of all the swept bodies, shown in Figure 3-13(e). Next, the accessibility

analysis procedure for the perpendicular surfaces is similar to that of positive/negative surfaces if *B_Swept1* is replaced by *B_Combined*.

A regularized Boolean subtraction operation is performed between *B_Original* and *B_Combined*, shown in Figure 3-13 (f), as per the following equation:

$$B\_Result = B\_Original \text{ -* } B\_Combined. \quad (3\text{-}5)$$

If the resulting body (*B_Result*) has same volume as that of *B_Original*, then the considered perpendicular surface is classified as *Fully-Accessible* in the corresponding direction. For example, volume of *B_Result* and *B_Original* is the same for the considered perpendicular surface in Figure 3-13; therefore, it is classified as *Fully-Accessible*.

3.4.3.2 Fully-Inaccessible (Type1) Surfaces

If the perpendicular surface is not classified as fully accessible, then it is analyzed for full-inaccessibility and partial-accessibility from the considered direction. Similar to Eqn. 3-2 for positive/negative surfaces, a regularized Boolean subtraction is utilized in which *B_Original* is subtracted from *B_Combined* as follows:

$$B\_Result1 = B\_Combined \text{ -* } B\_Original. \quad (3\text{-}6)$$

The above Boolean operation may result in a number of bodies depending on whether the surface is partially accessible or fully inaccessible. If *B_Result1* consists of two or more disjoined bodies, then it is concluded that the surface is *Fully-Inaccessible (Type1)* from the considered direction; otherwise, the surface can be *Fully-Inaccessible* (*Type2*) or *Partially-Accessible*.

3.4.3.3 Fully-Inaccessible (Type2) and Partially-Accessible Surfaces

To determine if the considered surface is *Fully-Inaccessible* (*Type2*) or *Partially-Accessible*, the procedure adopted is same as that for the positive/negative surfaces, discussed in Section 3.4.1. The *Partially-Accessible* perpendicular surface is not further classified into *Partial-Outer-Boundary-Accessible* and *Outer-Boundary-Inaccessible* because a partially accessible perpendicular surface cannot be *Outer-Boundary-Inaccessible*.

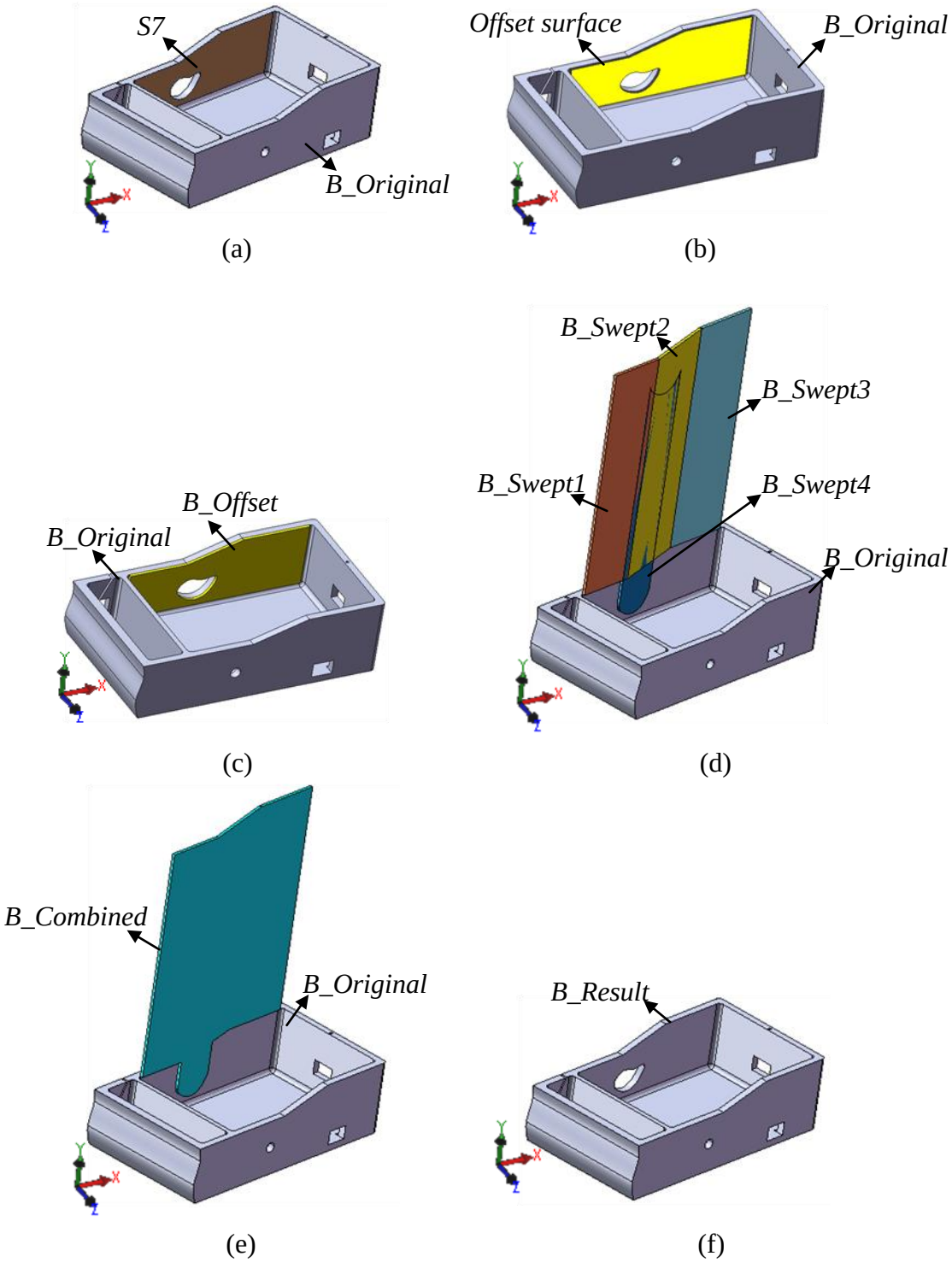


Figure 3-13: Accessibility analysis procedure for a perpendicular surface, demonstrated using surface *S7* of Figure 3-6: (a) surface *S7* of *B_Original*, (b) *B_Original* and *Offset* surface, (c) *B_Original* and *B_Offset*, (d) *B_Original* and swept bodies, (e) *B_Original* and *B_Combined*, and (f) *B_Result* as per Eqn. 3-5.

3.4.4 Discussion

The Boolean-based accessibility analysis approach evaluates the accessibility of part surfaces while preserving the topological data stored in the B-rep format. The time taken for the analysis process is dependent upon the number of geometric operations involved for each part surface. For example, the number of Boolean and sweeping operations involved in the accessibility analysis of *Partially-Accessible* surfaces is more than the number of operations involved in the case of *Fully-Accessible* surfaces. The regularized Boolean and sweeping operations are computationally complex and time consuming. Therefore, time taken by the algorithm is dependent upon the type and number of the part surfaces.

In comparison to STL-based approaches, the method treats all surface entities in their algebraic form instead of decomposing the surfaces into small triangles. Though the STL-based approaches have time gains due to linear computations for each triangle, the number of triangles can be large for a part with sculptured surfaces, and the computation time can be large. In contrast, the proposed method deals with fewer entities for parts with sculptured surfaces.

The benefits of the method are that results of accessibility analysis can be combined with topological data to recognize mold features. On the other hand, one disadvantage is that the geometric operations involve time-consuming regularized Boolean operations. Therefore, there is a need to further reduce the time required for analyzing accessibility by eliminating computationally complex Boolean operations while maintaining the information stored in B-rep format.

3.5 Pixel-Based Accessibility Analysis Approach

In a mold part, the accessibility of a surface is affected by other part surfaces. A surface is *Fully-Inaccessible* or *Partially-Accessible* if it is shadowed by some obstructing part surfaces in the considered parting direction. Thus, a shadowed surface (*Fully-Inaccessible* or *Partially-Accessible*) can be detected by comparing its visible surface area in the presence of all part surfaces with the visible surface area in isolation of other part surfaces.

To compare the visible surface areas, two 2D images for each surface are generated. One image has all the part surfaces and the other image has only the

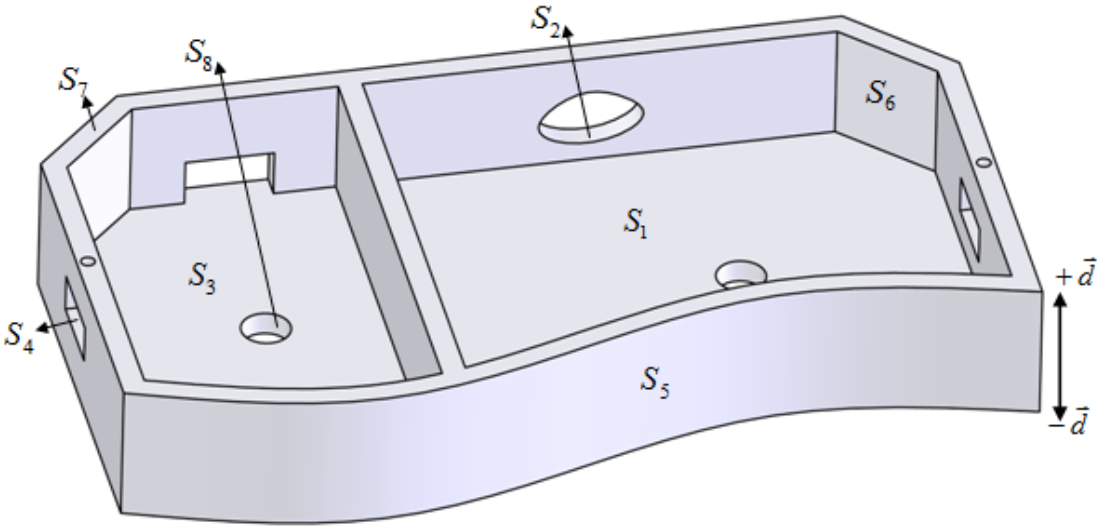
considered surface in the absence of all other part surfaces. The numbers of pixels occupied by the considered surface in both of the images are compared to classify the surface based on accessibility. However, perpendicular surfaces do not occupy any pixel on the computer screen. To check their accessibility, the surface is projected onto a plane normal to the considered parting direction, and the resulting 2D curve profile is obtained. The accessibility of this 2D profile is analyzed by overlapping it to the other part surfaces.

3.5.1 View-port Set-up

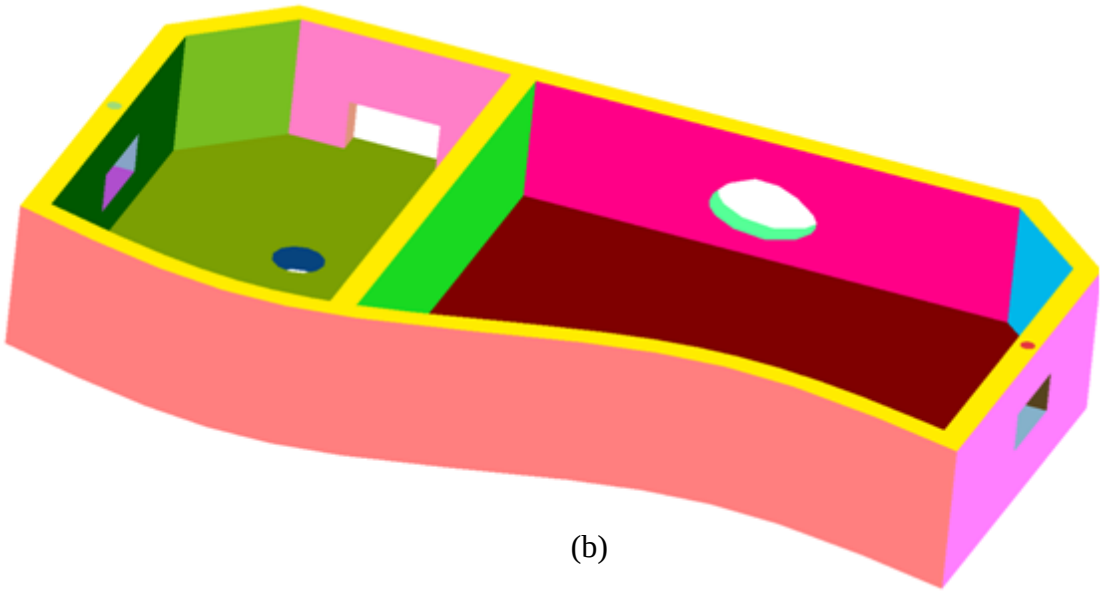
In this methodology, positive and negative surfaces are analyzed from the positive and negative sides of the parting direction, respectively. Whereas perpendicular surfaces can be accessible from both sides of the parting direction, their accessibility is analyzed from both the positive and negative parting directions. The analysis process starts by generating a 2D image of the complete part. To generate the image, the following steps are taken:

- assign a unique color to each surface, shown in Figure 3-14(b);
- fix the view-port size; and
- orient the part and adjust the scaling factor.

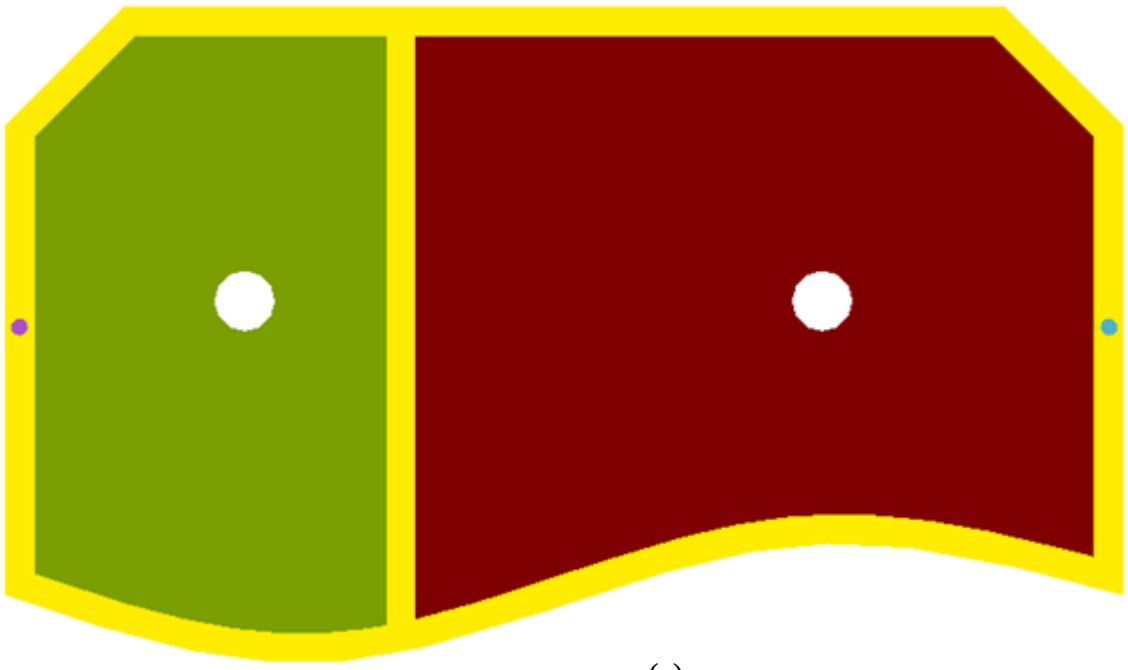
The part is first oriented in a way so that the parting direction coincides with the normal of the view-port. Then, it is scaled to fit within the view-port. For example, the part is oriented so that $+\vec{d}$ and the normal vector of view-port point in the same direction, as shown in Figure 3-14(c), while evaluating the accessibility from the positive side of the parting direction ($\vec{d}$). The view-port size, orientation, and scaling factor are maintained for analyzing all of the positive surfaces. A similar procedure is adopted for negative surfaces.



(a)



(b)



(c)

Figure 3-14: 3D model of a part for illustrating pixel-based accessibility analysis approach. (a) *Fully-Accessible* (S_1, S_5, S_6, S_7, S_8), *Fully-Inaccessible* (S_2), *Partial-Outer-Boundary-Accessible* (S_3), and *Outer-Boundary-Inaccessible* (S_4) surfaces of the part; (b) Part surfaces with unique colors; and (c) Part orientation while evaluating accessibility from $+\vec{d}$.

3.5.2 Accessibility Analysis of Positive and Negative Surfaces

The methodology adopted for positive and negative surfaces is the same except that the part is oriented differently as explained earlier. Therefore, in the following section, the methodology is explained only for the positive surfaces, considering that the positive parting direction ($+\vec{d}$) coincides with the normal vector of the view-port.

Let $S = \{S_1, \dots\dots, S_n\}$ be the positive surfaces of a part with the corresponding area of $\{A_1, \dots\dots, A_n\}$ where A_i is the surface area of the i^{th} surface in the absence of any other surfaces in the image plane, n is the total number of positive surfaces. Let $A_v = \{A_1^v, \dots\dots, A_n^v\}$ where A_i^v is the surface area of the i^{th} surface in the presence of all the part surfaces in the image plane. The method to determine A_i and A_i^v are discussed later. Once A_i and A_i^v are calculated, the part surfaces are classified into three principal categories based on the following relationship between A_i and A_i^v:

$$S_i = \begin{cases} Fully - Accessible, & if\ A_i^v = A_i \\ Fully - Inaccesible, & if\ A_i^v = \emptyset \\ Partially - Accessible, & Otherwise \end{cases}$$

The methodology is explained using a test part shown in Figure 3-14. After a unique color is assigned to each part surface, the part is oriented as described earlier. The image of the part is generated (named *PartImage*) and the area occupied by each surface color is determined by multiplying the numbers of pixels occupied by the color with the area of each pixel, i.e.,

$$A_i^v = N_i^v \times A_p,$$

where N_i^v is number of pixels occupied by the color of the i^{th} surface and A_p is the area of each pixel. However, the area occupied by each pixel is constant and its effect cancels out while comparing the area of two surfaces. Therefore, the area of each pixel is considered as the comparison unit. Next, an image of each individual surface is generated in isolation from the other part surfaces, while maintaining the same view-port size, orientation and scaling factor. The example of this action is shown in Figure 3-15(a). The areas occupied by the surface in both of the images are compared. If the areas are the same, the surface is classified as *Fully-Accessible*.

If the area occupied by the surface in *PartImage* is zero, then the surface is *Fully-Inaccessible*. Otherwise, if the areas occupied by the surface in the *PartImage* and the individual surface image do not exactly match one another, the surface is classified as *Partially-Accessible*.

To further classify the *Partially-Accessible* surfaces into *Outer-Boundary-Inaccessible* and *Partial-Outer-Boundary-Accessible*, the outer boundary of surface in surface's individual image is compared with the boundary of surface in *PartImage*.

Let $S^p = \{S_1^p, \dots\dots, S_q^p : q < n\}$ be the *Partially-Accessible* surfaces and let their outer boundaries in the image in the absence of other part surfaces be $\{C_1, \dots\dots, C_q\}$ and the corresponding outer boundaries in the presence of all the part surfaces in the *PartImage* be $\{C_1^v, \dots\dots, C_q^v\}$. The two subcategories of *Partially-Accessible* surfaces are defined as follows:

$$S_j^p = \begin{cases} Partially - Outer - Boundary - Accessible, & if\ C_j \cap C_j^v \neq \emptyset \\ Outer - Boundary - Inaccessible, & otherwise \end{cases}$$

The methodology has been explained using the surface S_3 shown in Figure 3-14(a). The images of outer-boundary of the surface S_3 in the absence of other part surface, shown in Figure 3-15(b), and in the presences of all part surfaces, shown in Figure 3-15(c), are generated separately. The locations of pixels representing the surface boundary in these two images are compared. As some of the boundary pixels in both the images have same pixel locations, the surface is classified as *Partially-Outer-Boundary-Accessible*.

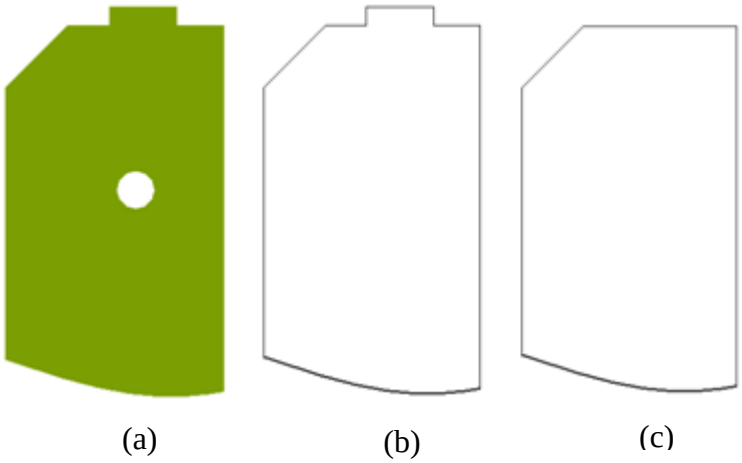


Figure 3-15: Part surface S_3 (from Figure 3-14) for illustrating accessibility analysis procedure of *Partially-Accessible* surfaces. Surface S_3 (a) in the absence of other surfaces, (b) its outer boundary in surface's individual image, and (c) its outer boundary in *PartImage.*

3.5.3 Determination of Obstructing Surfaces

If a surface is classified as *Fully-Inaccessible* or *Partially-Accessible*, then it must be obstructed by other part surfaces. The list of obstructing surfaces of each considered surface is used for feature recognition. To determine the list of surfaces obstructing the accessibility of S_i, all the surfaces with the same orientation as S_i are first identified in the B-rep model. Then an image of S_i in the presence of one of the identified surfaces is generated and analyzed. The identified surface obstructs the accessibility only if the areas

occupied by the surface S_i in the newly-generated image and in the individual surface image do not match. This process is repeated for all of the remaining identified surfaces and a list of obstructing surfaces is generated. For example, while determining the obstructing surfaces for the surface S_3, images of S_3 are separately generated with all the other positive surfaces. The image generated by the surfaces S_3 and S_7 is shown in Figure 3-16.

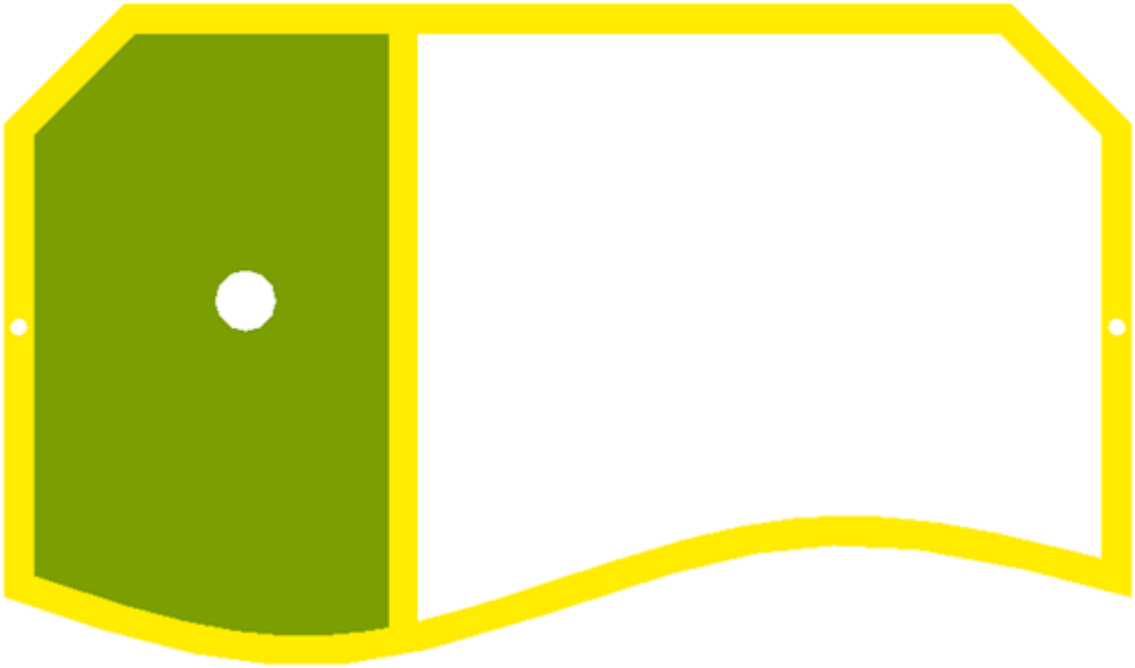


Figure 3-16: Image of part surfaces S_3 and S_7 (from Figure 3-14) for illustrating obstructing surfaces identification procedure.

3.5.4 Accessibility Analysis of Perpendicular Surfaces

In the proposed methodology, the accessibility of perpendicular surfaces is determined by using the bounding edges of the surfaces in B-rep format. In B-rep format, each edge separates only two adjacent faces from each other. The edge is represented using mathematical equations and is displayed on screen in vector format. Before classifying a perpendicular surface, all bounding edges of the surface are categorized based on the topology as discussed below.

3.5.4.1 Edge Classification

The bounding edges of a perpendicular surface are classified into two categories, namely concave and convex, depending on the angle between the two neighboring faces separated by the edge. A concave edge is formed if the angle between two neighboring faces on material side (θ) is more than 180 degrees, whereas, a convex edge is formed when the angle (θ) is less than 180 degrees, as shown in Figure 3-17.

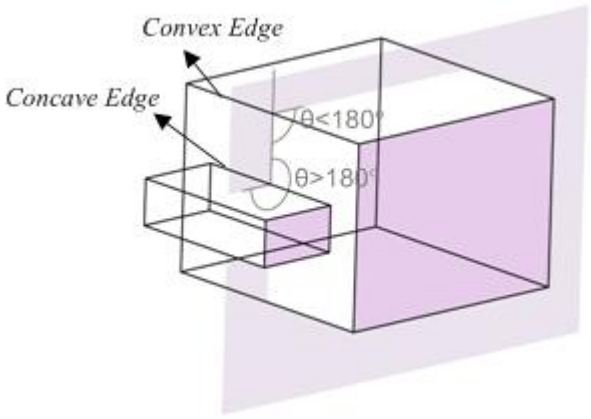


Figure 3-17: Concave and convex edges.

A face can be bounded by a number of edge loops. There can be only one outer bounding edge loop, and the rest of the loops form inside edges within the surface. Each edge of the perpendicular surface is further classified depending upon whether it is part of an inner or outer bounding edge loop. Therefore, if an edge belongs to outer bounding edge loop, it is classified as outer-bounding edge; otherwise, it is classified as inner-bounding edge. The inner and outer bounding edges for the perpendicular surface S_i are shown in Figure 3-18.

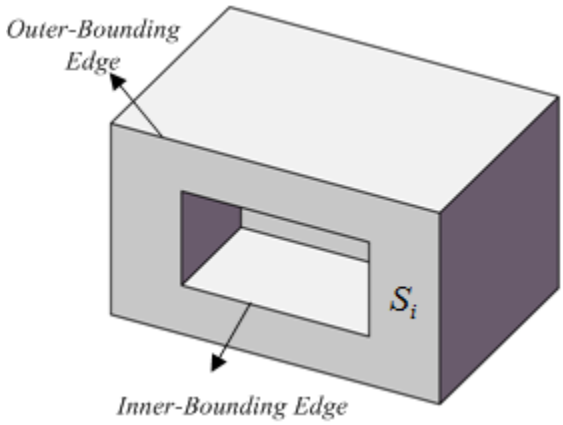
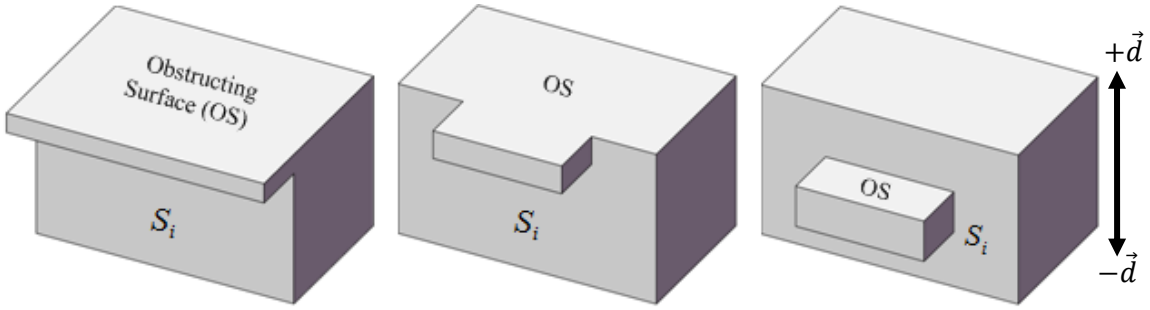


Figure 3-18: Outer and Inner bounding edges of the part surface S_i.

3.5.4.2 Topology based Preliminary Analysis

The accessibility of a perpendicular surface (S_i) can be obstructed by: *non-adjacent non-perpendicular* surfaces, shown in Figure 3-19(a); *non-perpendicular surfaces adjacent to outer boundary*, shown in Figure 3-19(b); and *non-perpendicular surfaces adjacent to inner-boundary*, shown in Figure 3-19(c). This observation is used to minimize the number of checks during accessibility analysis.



(a) (b) (c)

Figure 3-19: Different topologies to obstruct the accessibility of a perpendicular surface. Perpendicular surface S_i obstructed by (a) *non-adjacent non-perpendicular surface*, (b) *non-perpendicular surface adjacent to outer-boundary*, and (c) *non-perpendicular surface adjacent to inner-boundary*.

While analyzing the accessibility of a perpendicular surface from the considered side (positive or negative side) of the candidate parting direction, *non-adjacent non-perpendicular* surfaces completely obstruct the accessibility of the perpendicular surface

if the perpendicular surface does not have any adjacent non-perpendicular surface that is connected through an outer-bounding edge and oriented in the considered side of the parting direction ($\vec{d}$). Therefore, such perpendicular surfaces are classified as *Fully-Inaccessible* from that side of $\vec{d}$. For example, the perpendicular surface S_i, shown in Figure 3-19(a), does not have any adjacent positive surface connected through an outer-bounding edge; therefore, the perpendicular surface is inaccessible from the positive side of the considered parting direction. Such a surface is not further analyzed.

If a perpendicular surface (S_i) has any non-perpendicular surfaces adjacent to inner-boundary and connected through concave edges as shown in Figure 3-19(c), then the accessibility of the perpendicular surface is partially obstructed by the non-perpendicular surfaces. Therefore, the analysis process starts with an assumption that a perpendicular surface with an inner-bounding concave edge is *Partially-Accessible*; otherwise it is considered as *Fully-Accessible*.

3.5.4.3 Image-Processing based analysis

In this process, all edges of the perpendicular surface are assigned a unique color. It should be noted that a perpendicular surface can be accessible from both positive and negative sides of a parting direction, and such surfaces are analyzed from both sides of the candidate parting direction. While analyzing perpendicular surfaces from the positive side of the candidate parting direction, part orientation, scaling factor and viewport size must be the same as during the analysis of positive surfaces. The same is true while analyzing the accessibility of perpendicular surfaces from the negative side of the candidate parting direction.

An image of the face is generated in the form of a 2D curve called a *Face Profile*. For example, the *Face Profile* of a cylindrical surface on the image plane will be a circle, as shown in Figure 3-20. Similarly, the *Face Profile* of a planar surface will be a straight line. Since the *Face Profile* occupies pixels of the non-perpendicular surfaces adjacent to the perpendicular surface, the *Face Profile* is offset by a small amount towards the non-material side of the perpendicular surface.

The focus of the further analysis is to generate an offset profile to the *Face Profile* and overlap the part surfaces over the offset profile. The offset profile is created towards

the non-material side of the perpendicular surface. To keep the approximations as close to the original as possible, the offset is limited to one pixel.

Figure 3-20: *Closed Face Profile* of surface S_8, shown in Figure 3-14.

As discussed earlier, an offset profile (called *Offset Profile*) of the *Face Profile* is required towards the outward surface normal (non-material side) of the perpendicular surface. To determine the outward surface normal of the perpendicular surface from the *Face Profile*, the following two characteristics of the *Face Profile* are used: first, it separates the material side and non-material sides; second, the outward surface normal points toward the non-material side of the *Face Profile*.

Figure 3-21: *Open Face Profile* of surface S_5, shown in Figure 3-14, and its magnified view.

To determine the material and non-material side of the *Face Profile*, the profile is classified into three categories, the closed, open and intersecting profiles. A closed profile divides the image into two regions separating material and non-material sides. The closed profile formed by the edges of the surface S_8 is shown in Figure 3-20. An open profile is a curve that does not intersect within the image boundaries even if it is extended using tangents at the end points (shown in Figure 3-21), whereas an intersecting profile results if the *Face Profile* intersects within the image boundaries if it is extended using the tangents at the end points (shown in Figure 3-22).

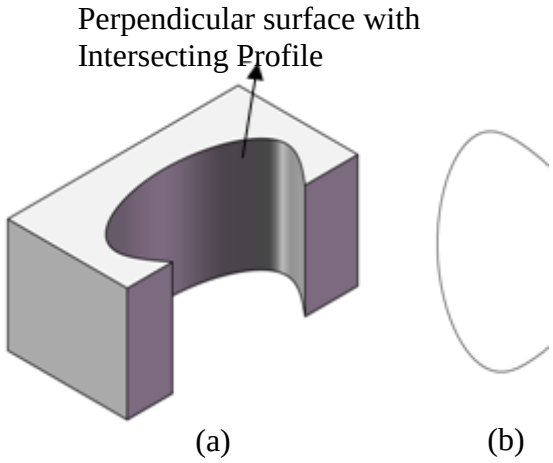


Figure 3-22: (a) Perpendicular surface with intersecting profile and (b) *Face Profile* on the image plane.

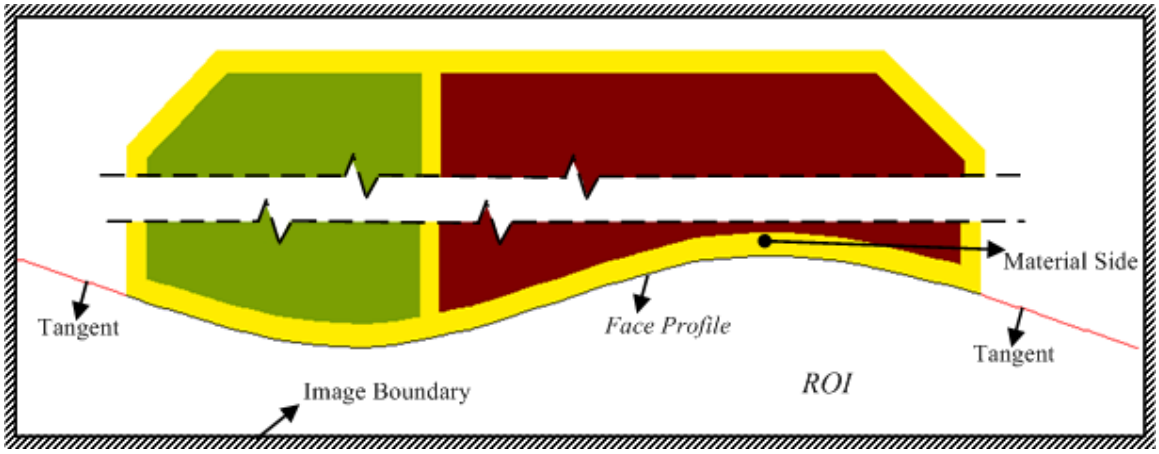


Figure 3-23: Procedure to determine tangents to *Face Profile* curve illustrated using surface S_5, shown in Figure 3-14.

In case of an open or intersecting *Face Profile*, the image is divided into two regions by determining the tangents at the end points of the profile, shown in Figure 3-23. The *Offset Profile* should be in one of these two regions. However, determination of the tangent based on the two end pixels of the profile can result in an inaccurate tangent due to the discrete nature of pixels. For example, the tangent determination using the two end pixels of a *Face Profile*, shown in magnified view in Figure 3-21, would result in an

inaccurate tangent. To overcome the above problem, a curve is passed through the discrete pixels and tangents at curve end points are calculated.

Furthermore, any positive/negative surface connected to the perpendicular surface through an outer bounding edge would also be in one of these two regions. To determine the region of the *Offset Profile*, characteristics of the edge connecting the perpendicular surface with one of its adjacent positive/negative surfaces are used. If the edge connecting the perpendicular face and its adjacent positive/negative face is concave, the *Offset Profile* is taken to be in the same region of positive/negative face; otherwise, the *Offset Profile* is taken in the other region and the region is called a Region of Interest (*ROI*).

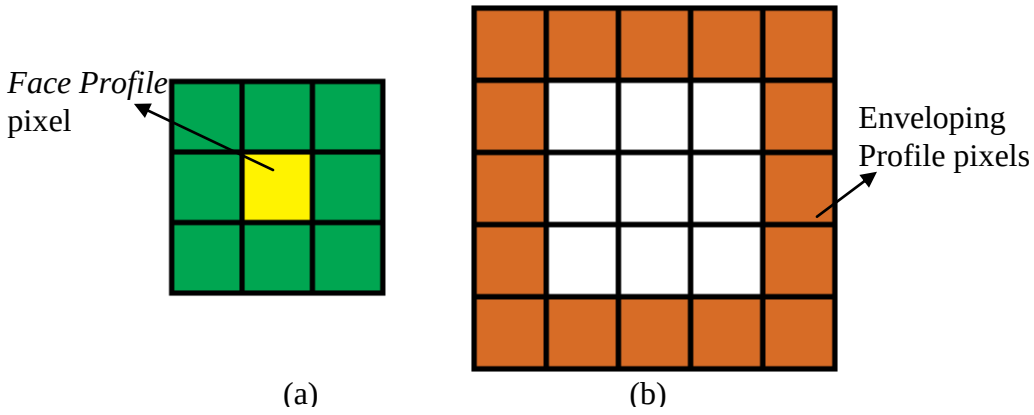


Figure 3-24: Procedure to determine enveloping profile pixels (a) pixels adjacent to a pixel of *Face Profile*, (b) and corresponding pixels of enveloping profile.

To determine the *Offset Profile*, the surrounding pixels of each pixel of the *Face Profile* are selected, as shown in Figure 3-24(a). Next, pixels surrounding these surrounding pixels are extracted as shown in Figure 3-24(b). These pixels form one enveloping profile in the case of *Open* and *Intersecting Face Profile*s and two enveloping profiles in the case of a *Closed Face Profile*. This enveloping profile is offset from the *Face Profile* by one pixel.

In the case of *Open* and *Intersecting Face Profiles*, a segment of the enveloping profile that overlaps the *ROI* is determined. The segment of enveloping profiles for surface S_5 is shown in Figure 3-25(a). It should be noted that the segment contains some extra pixels on both of its ends. These extra pixels are discarded using the curve normals

at the end points, shown in Figure 3-25(b). This resulting profile is the *Offset Profile* and would be towards the outward surface normal of the perpendicular surface. In case of a *Closed Face Profile*, one of the two profiles overlaps the *ROI* and is considered as an *Offset Profile*.

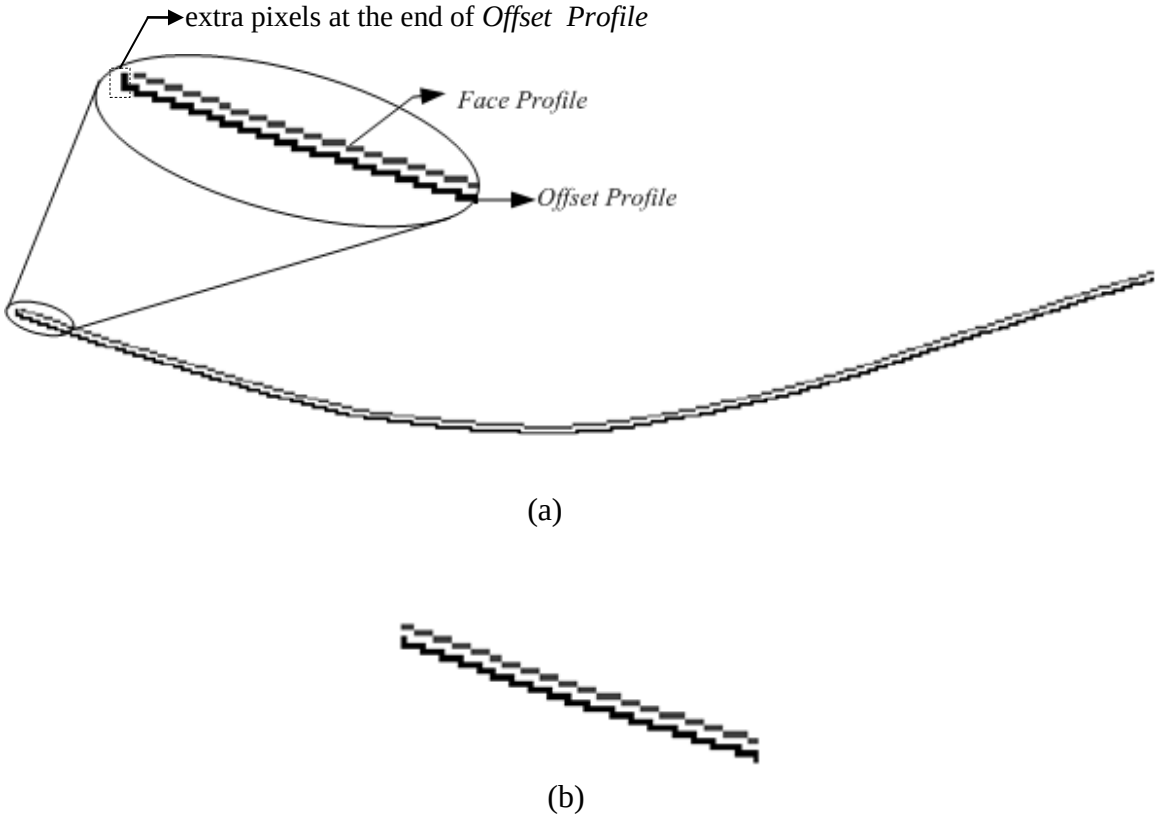


Figure 3-25: Pixels of *Face* and *Offset Profile* (a) *Offset Profile* with extra pixels at its ends, and (b) after discarding the extra pixels.

Now the *Offset Profile* is overlapped with its non-perpendicular adjacent surfaces connected through the outer-bounding convex edges and the obstructing surfaces of all the non-perpendicular adjacent surfaces. If all the pixels of an offset profile are covered, then the perpendicular surface is classified as *Fully-Inaccessible*. If none of the offset profile pixel is overlapped, then the perpendicular surface is *Fully-Accessible*, otherwise, the surface is *Partially-Accessible*.

3.6 Determination of Undercut-Free Parting Direction

In a part model, if the accessibility of a positive surface is obstructed, then the accessibility of at least one of the negative surfaces will be obstructed as well. Similarly, if all positive surfaces are *Fully-Accessible* from the positive side of the parting direction, then all negative surfaces will be *Fully-Accessible* from the negative side of parting direction. This topological characteristic is used to minimize the number of checks and time complexity for finding the undercut-free parting direction. The undercut-free parting direction is determined by evaluating the accessibility of the positive or negative surfaces, whichever are fewer in number.

3.7 Discussion

Two accessibility analysis approaches, Boolean-based and pixel-based, are discussed in this chapter. Both of these approaches evaluate the accessibility of part surfaces while preserving the geometrical and topological information stored in the B-rep model. Both of the approaches work well, but also have some drawbacks which are discussed here.

The Boolean-based approach is based on sweeping the part surfaces and performing regularized Boolean operations on the swept bodies and the given part. However, the regularized Boolean operation fails if the resulting body has invalid topology. Further research is required to avoid the invalid topologies. Moreover, the Boolean and sweep operations used in this approach are computationally complex. To solve these problems, the pixel-based approach was introduced.

In the pixel-based approach, the pixels used to represent each surface on the computer screen are analyzed. However, if there are too few pixels to represent the solid model, then there is a possibility that some small surfaces may not occupy any pixel and consequently be classified as *Fully-Inaccessible*. This does not affect the output of accessibility analysis in most of the injection molded parts as these can be represented in full or larger scale on modern day computer screens. Furthermore, the algorithm can be extended to large parts by dynamically zooming the part during the analysis phase.

The time required for analyzing the accessibility of a part's surfaces by Boolean-based and pixel-based approaches depends on the number of part surfaces and number of operations involved for each surface. Furthermore, the number of operations involved for

a surface depends on the surface orientation and its accessibility level. The number of operations involved for a perpendicular surface is more than the number of operations required for a positive or negative surface with same level of accessibility. For example, the number of operations required to analyze the accessibility of a *Fully-Accessible* perpendicular surface is more than that of a *Fully-Accessible* positive surface. It is also observed that the pixel-based approach takes less time than the Boolean-based approach when they are tested on identical parts. Test results of two of the parts are discussed in the next section.

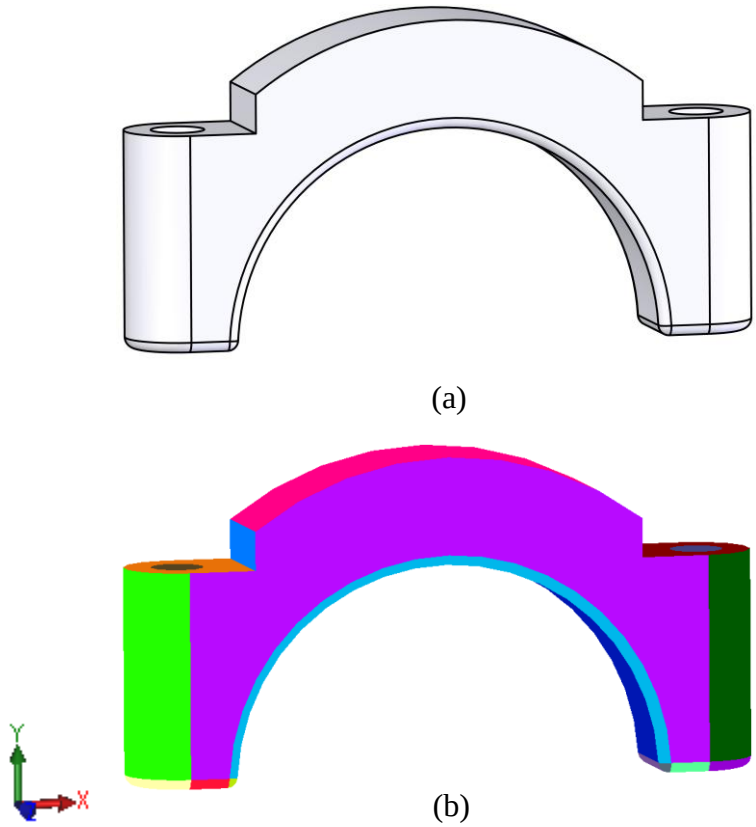
3.8 Implementation

The proposed methodologies were implemented to analyze the part accessibility from a set of directions and to determine the feasible mold parting directions. The algorithm for the Boolean-based accessibility analysis has been implemented using Solidworks® and VBA. The Solidworks functions are used to perform the sweep and Boolean operations. For the pixel-based accessibility analysis approach, the system has been implemented in a prototype test bed consisting of Solidworks, Microsoft Visual Basic, and Matlab®. The APIs provided by Solidworks are used to generate the surface images and to analyze the part topology. The images are analyzed using Matlab to evaluate the accessibility of part faces. All the programs are tested on a 1.6GHz Pentium IV processor and 1GB RAM machine running on Microsoft Windows XP.

The undercut-free parting direction determination algorithms have been implemented and tested using various examples. The parting direction in most industrial parts is towards the major axis of the coordinate system. Therefore, in these tests, the accessibility is analyzed along the three major axis of coordinate system.

The first case reported is a test part shown in Figure 3-26(a). This part is similar to an industrial part considered by Chen et al. [70] and consists of 27 faces. The accessibility of part surfaces is analysed using both the Boolean-based and the pixel-based approaches. The undercut-free parting direction is successfully found along the *Y*-axis. In pixel-based approach, 314 images are generated with 800×600 pixels. To compare the two approaches, the accessibility of all the surfaces along the *Y*-axis is analyzed using both methods. The Boolean-based approach took 28 seconds, whereas the

pixel-based approach took 25 seconds. In the pixel-based approach, the obstructing surfaces for each surface were also determined.

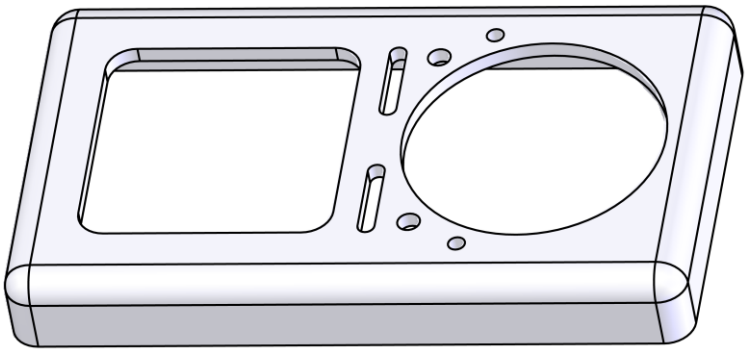


(a)

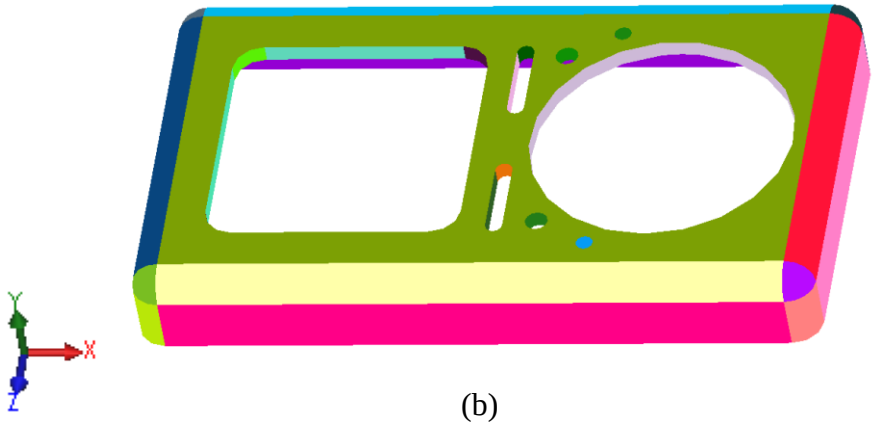
(b)

Figure 3-26: (a) Test part taken from the work of Chen and Rosen [70], and (b) after applying unique colors to the part faces.

The second case study, in Figure 3-27(a), consists of 56 faces and is geometrically similar to one of the industrial parts shown by Banerjee et al. [34]. The accessibility is analyzed along three major axes and all the surfaces were found to be *Fully-Accessible* along the *Y*-axis. The time taken to analyze all part surfaces along the *Y*-axis using the Boolean- and the pixel-based approaches is 120 seconds and 65 seconds, respectively. A total of 257 images with 800×600 pixels were generated for determining accessibility of each surface and the list of obstructing surfaces of each surface along the *Y*-axis.



(a)



(b)

Figure 3-27: (a) Test part taken from the work of Banerjee et al. [34], and (b) after applying unique colors to the part faces.

Chapter 4

Mold Feature Recognition

The recognition of mold features is important for automatic generation of mold segments. The features are defined in different ways in the literature for the purpose of their automatic recognition. The features/undercuts that cannot be molded by using core and cavity are broadly divided into two categories: *external undercuts* and *internal undercuts*, shown in Figure 4-1.

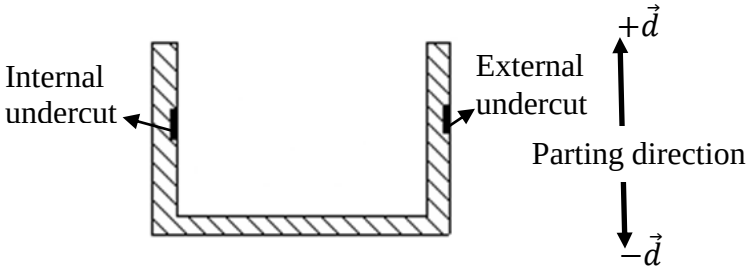


Figure 4-1: External and internal undercut.

External undercuts are the restriction regions of a part which prevent the withdrawal of the molding from the cavity and are molded with the use of side-cores [23]. *Internal undercuts* are the restriction regions of a part which prevent the withdrawal of the core from the molding [23].

The *external* and *internal* undercuts are further classified as *protrusion* and *depression* undercuts. *Protrusion* undercut signifies a material addition over the base surface of the part, whereas, the *depression* undercut signifies material removal from the base surface, shown in Figure 4-2. A depression undercut can be of two types: *blind* and *through*. A *blind* depression is connected only to either the core or the cavity surfaces. Whereas, a *through* depression undercut connects core and cavity surfaces, and *shut-off* surfaces are required to mold such undercuts. A *shut-off* surface is a surface patch that separates the core and cavity surfaces. The complexity of the feature recognition process increases with the intersections between various mold features. A nomenclature has been developed for such features and is discussed in Section 4.2.

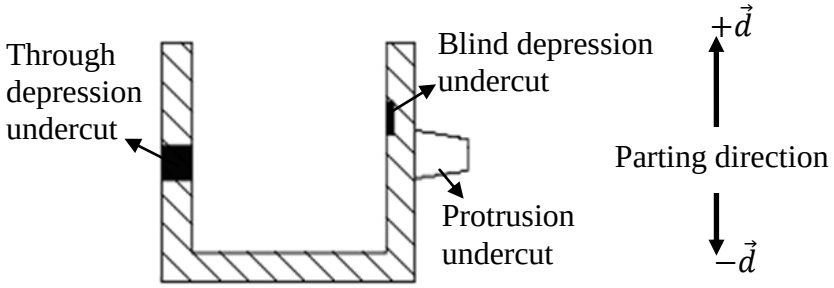


Figure 4-2: Different types of undercut features.

The problem of feature recognition is formulated in Section 4.1. The mold features are classified based on geometry, topology, and accessibility of the part faces into simple protrusion and intersecting depression features in Section 4.2. The part faces that constitute a simple protrusion are identified, and the procedure to recognise protrusion features is discussed in Section 4.3. Next, the intersecting depression features are recognized in Section 4.4 by using the release direction of depression features. To maintain the flow, in Section 4.4, the procedure for determining the release direction of the mold futures is detailed in Section 4.5. The face classification and feature recognition algorithms have been tested on many parts; four of these are described in Section 4.6 to demonstrate the proposed algorithms. A block diagram of the approach, developed in the present work is shown in Figure 4-3.

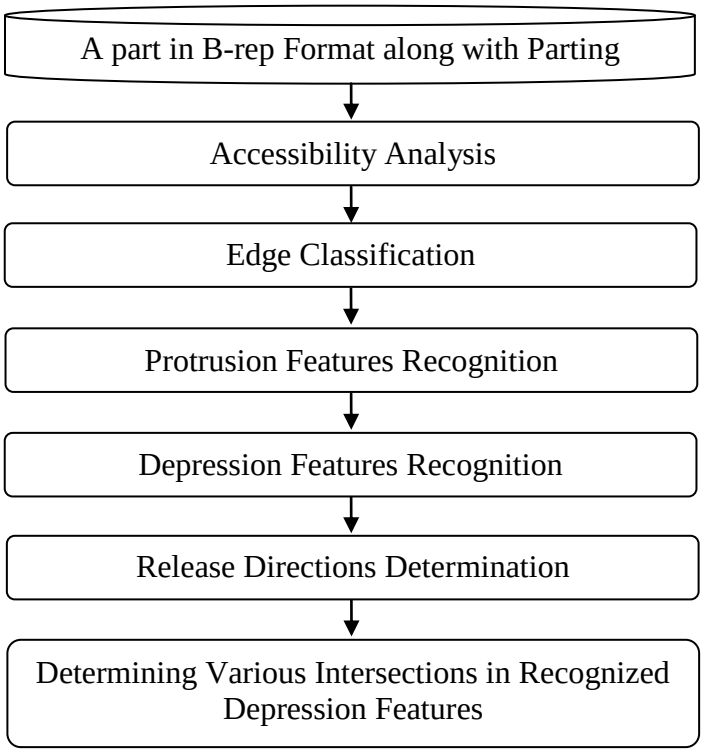


Figure 4-3: Mold features recognition procedure.

4.1 Problem Formulation

The objective of the work described in this section is to develop a system for the automatic recognition of side-core/cavity surfaces (and their release direction) of two piece permanent molds, ensuring the de-moldability of the given part along the given parting direction.

Input:

- A 3D part in the B-rep format
- A parting direction for de-moldability

Output:

- The recognition of the side-core/cavity surfaces

4.2 Classification of the Molding Features

Molding features/undercuts hinder the part de-moldability and require additional elements, other than the core and the cavity, to ensure de-moldability. Each feature is

recognized by using the geometric and topologic information embedded inside the B-rep format of the given solid part.

In the feature recognition module, the output of the face classification module (discussed in Chapter 3) is used as an input. In addition to the face classification, each edge is classified into three categories (convex, concave, and tangent) to recognize the features. In a valid solid model, an edge is shared by two faces only and is categorized by considering the angle (φ) between the two adjacent faces on the material side. The criteria for classifying an edge are given in Table 4-1.

Edge Type	Angle (φ)
Concave	>180°
Convex	<180°
Tangent	=180°

Table 4-1: Edge classification criteria.

The literature and observation of many molded components indicate that molding features can be broadly classified into protrusions, depressions, and their intersections. Two features are considered to be intersecting if their face-sets intersect to alter each other's shape and need different molding elements to ensure de-moldability. Intersecting features are further classified depending on the elements used during de-molding. To facilitate the recognition, the mold features are defined next.

As discussed earlier, *Protrusion* is the restriction region of the part during de-molding and signifies material addition over the base face (feature *FT6,* Figure 4-4); whereas, a *depression* signifies material removal from the base face (feature *FT8*, Figure 4-4). A depression undercut can be further classified as through (feature *FT7*, Figure 4-4) or blind (feature *FT1*, Figure 4-4).

The intersecting depression features are classified into two main categories: *Type-I intersection* and *Type-II intersection*. A *Type-I intersection* results in connecting the core and cavity features either to each other or to features moldable by using side-cores, and requires shut-off surfaces to ensure de-moldability (feature *FT5,* Figure 4-4). A *Type-*

II intersection is the result of the intersection of two depression features that are moldable by using only the side-cores (feature *FT8*, Figure 4-4).

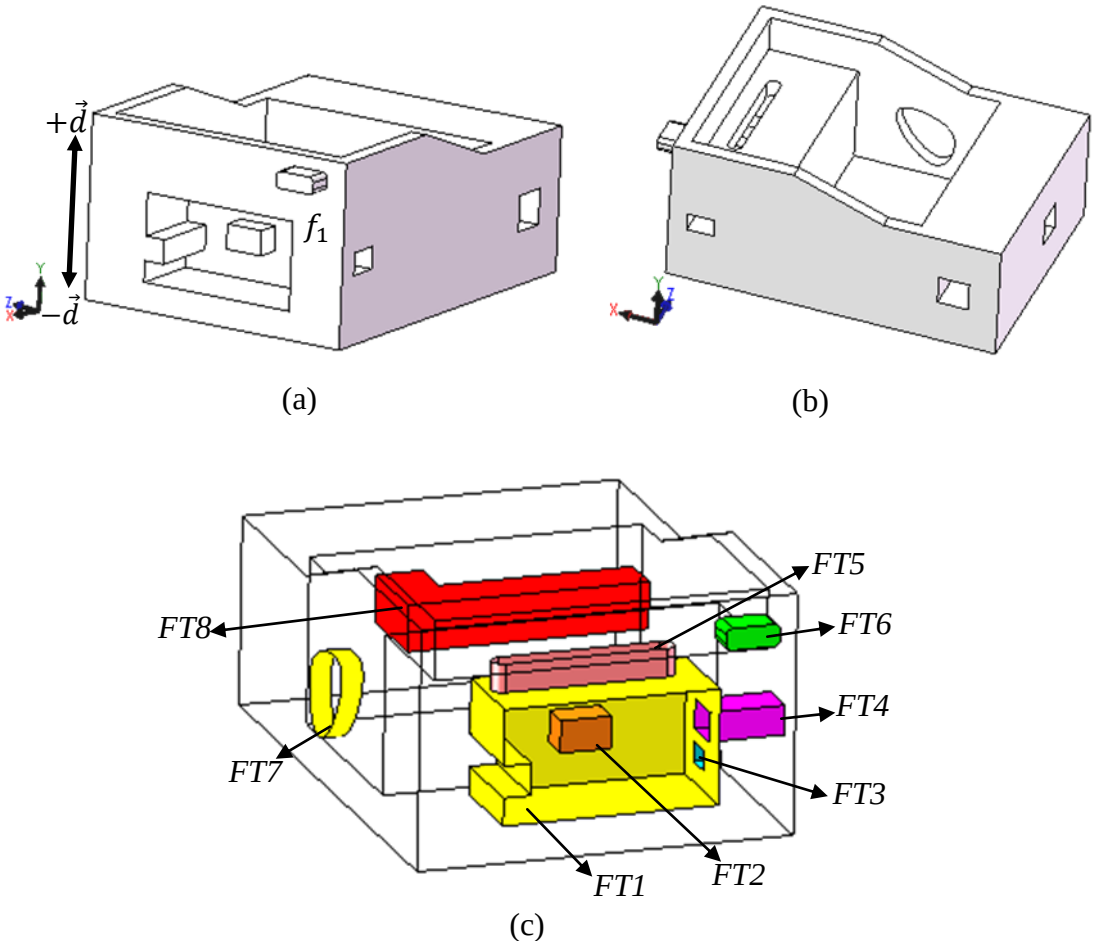


Figure 4-4: Nomenclature of molding features: (a) moldable part, (b) part in different view, and (c) feature nomenclature (*FT1* is a *Blind Depression Parent Feature*, *FT2* is a *Protrusion Child Feature*, *FT3* is a *Blind Depression Child Feature*, *FT4* is a *Through Depression Child Feature*, *FT5* is a *Type-I intersecting Feature*, *FT6* is a *Simple Protrusion Feature*, *FT7* is a *Simple Depression Feature*, and *FT8* is a *Unbounded-Type-II Intersecting Feature*).

Furthermore, *Type-II* intersecting features are classified into two categories, depending on the edge-boundary at the intersection of the two depression features. If one depression feature exists inside another depression feature, is bounded by closed convex edge loops, and requires side/split-core for de-moldability, then the features are called *child depression features* (features *FT3* and *FT4*, Figure 4-4) and *parent depression features* (feature *FT1*, Figure 4-4), respectively, and both are called *Bounded-Type-II*

intersecting features. Otherwise, if two depression features intersect with each other such that they are moldable by using different side-cores and the intersection boundary is not definable by a convex edge loop, then the two features are called *Unbounded-Type-II intersecting features* (feature *FT8*, Figure 4-4).

In the feature recognition procedure, the protrusions are recognized first, described in Section 4.3, as their presence sometimes affects the characteristics of the face(s) from which they originate, e.g., base face f_2 is partially accessible as shown in Figure 4-5. Then depression features are recognized, detailed in Section 4.4, by suppressing the information of protrusion features from the part. Next, *Type-I* intersecting features are recognized, in Section 4.4.2, prior to recognizing the *Type-II* intersecting features, in Section 4.4.3. At the end, the remaining faces that are not part of any of the recognized feature are grouped into individual features (core, cavity, and additional) according to their adjacency and accessibility, discussed in Chapter 5.

4.3 Recognition of Protrusion Features

Simple protrusions have concave and/or tangent bounding edges along the base faces (e.g., f_2 and f_3 are the base faces for the simple protrusion in Figure 4-5) and consist of a convex face-set. A convex face-set of the simple protrusion is denoted in green in Figure 4-5. However, the protrusion features can have concave faces (signified in red in Figure 4-5) due to the intersection with the depression(s), as well as protrusion(s). The recognition of intersecting protrusions is beyond the scope of this work.

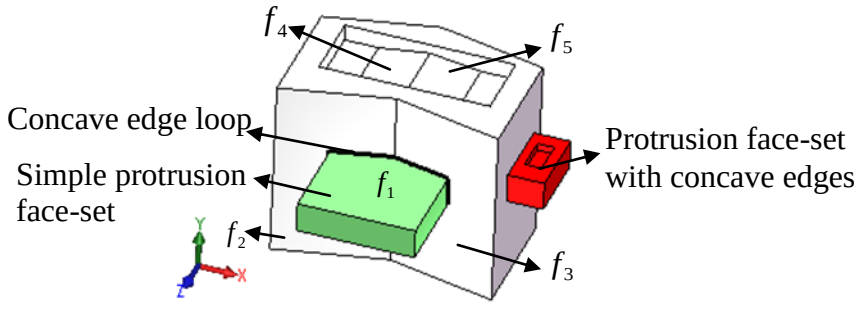


Figure 4-5: Simple protrusion face-set, considering parting direction along the Y-axis.

4.3.1 Procedure for Protrusion Features Recognition

A positive/negative face with at least one concave edge is chosen as the seed face to initiate the recognition process, e.g., face f_1 is chosen as the seed face for the simple protrusion in Figure 4-5. All the adjacent faces of the seed face, sharing one or more convex edges with it, are grouped in an array (*Protrusion_Face_Set*) along with the seed face. Next, the faces adjacent to these grouped faces that share the convex edge with them are also added to the same array. This process is continued until all the faces, grouped in the array, are processed (the grouped faces for the seed face f_1 are shown in green in Figure 4-5). However, if the faces grouped in the *Protrusion_Face_Set* are all positive or all negative, then the grouped faces cannot form a restriction region and are not classified as protrusion faces (e.g., group of f_4 and f_5 faces in Figure 4-5). These faces are not processed further. Next, all the concave edges of the faces in the *Protrusion_Face_Set* array are identified, stored in an array (*Protrusion_Bounding_Edges*), and arranged based on their connectivity. If the arranged concave edges form one closed loop (such a loop is shown in Figure 4-5 in bold), then the faces grouped in the *Protrusion_Face_Set* array are classified as the face-set of a simple protrusion.

4.4 Recognition of Depression Features

Depression undercuts have *Fully-Inaccessible*/*Partially-Accessible* faces and need side/split cores to be molded. In the feature recognition procedure, the existence of a depression undercut is indicated by the presence of *Fully-Inaccessible* or *Partially-Accessible* face(s) (e.g., face f_1 in Figure 4-6) that are identified during the accessibility analysis, as discussed in Chapter 3. The *Partial-Outer-Boundary-Accessible* faces are split manually into *Fully-Accessible* and *Fully-Inaccessible* faces. Any of the *Fully-Inaccessible*/*Partially-Accessible* faces can be used as a seed face for starting the recognition process. The faces that are identified as members of the protrusion face-set are not considered as the seed face. In addition, the seed face should not be obstructed by only the face-set of identified protrusions.

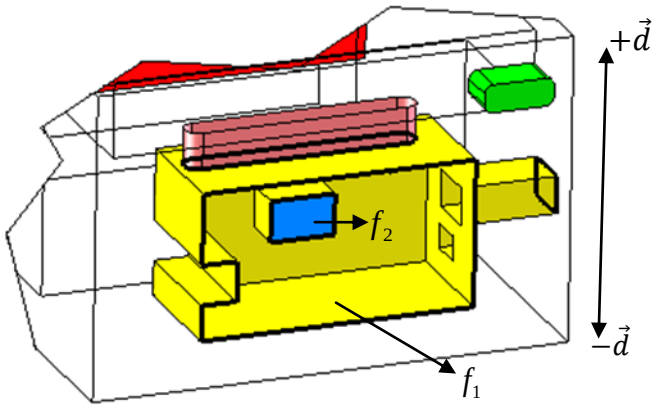


Figure 4-6: Bounding edge loops of depression undercut feature *FT1* of Figure 4-4.

4.4.1 Procedure for Identifying and Grouping Depression Faces

Once the seed face is identified, the recognition process of the depression feature is initiated. *Fully-Inaccessible* or *Partially-Accessible* faces, adjacent to the seed face, are grouped in an array (*Depression_Face_Set)* that also includes the seed face. Next, *Fully-Inaccessible*/*Partially-Accessible* faces, adjacent to the faces in the array, are also added to the same array. This process is continued until all the faces in the array have been processed. However, the faces, obstructed only by the faces forming the identified protrusions, are not considered as part of the undercut (e.g., face f_1 in Figure 4-4) and are not added to the array. (The faces, obstructed by identified protrusion only, are grouped in the category of the *Additional Faces* at the end of the feature recognition process. These *Additional Faces* should be considered while determining the parting line, and might need additional elements to ensure de-moldability).

Once the *Depression_Face_Set* array is populated, the bounding edges of the grouped faces are stored in an array called *Depression_Bounding_Edges*, and these edges are arranged according to their connectivity. In Figure 4-6, the faces of the *Depression_Face_Set* array, for seed face f_1, are shown in yellow and the edge loops of *Depression_Bounding_Edges* array are shown in bold lines. A depression feature can have more than one closed loop, based on the depression type.

It should be noted that in Figure 4-6, face f_2 is not included in the *Depression_Face_Set* array because it is fully-visible through the loop formed by the *Type-I* intersection. Thus, the aforementioned procedure does not include *Fully-*

Accessible faces that are part of the undercut in the given parting direction into *Depression_Face_Set* array of the considered undercut. The identification of such missing faces and their addition to the *Depression_Face_Set* array requires further processing.

4.4.1.1 Procedure for Adding Missing Faces to the Depression Faces

The convex hull faces of the part are determined by using the approach given in Appendix B. Then the part faces are temporarily categorized into three types: the faces included in the *Depression_Face_Set* array, convex hull faces, and remaining faces. The remaining faces are further sub-grouped based on their adjacency and then each sub-group is checked to see if it is enclosed by the faces of the *Depression_Face_Set* array. If the sub-group is enclosed, then the sub-grouped faces (e.g., face f_2 in Figure 4-6) are added to the *Depression_Face_Set* array.

Next, the *Type-I intersecting* features (e.g., feature FT5 in Figure 4-4) that are accessible in the parting direction and at the same time interact with the faces of the depression features that require the side-core, must be identified. These features form an interacting zone that needs to be shut off to ensure de-moldability. Moreover, the zone can be interpreted as an undercut opening, instead of an intersection, for calculating the release direction. Such intersections must be identified, and the corresponding closed loops need to be removed from the *Depression_Bounding_Edges* array. In this work, the following method is adopted to recognize such intersecting features.

4.4.2 Procedure for Type-I Intersecting Features Recognition

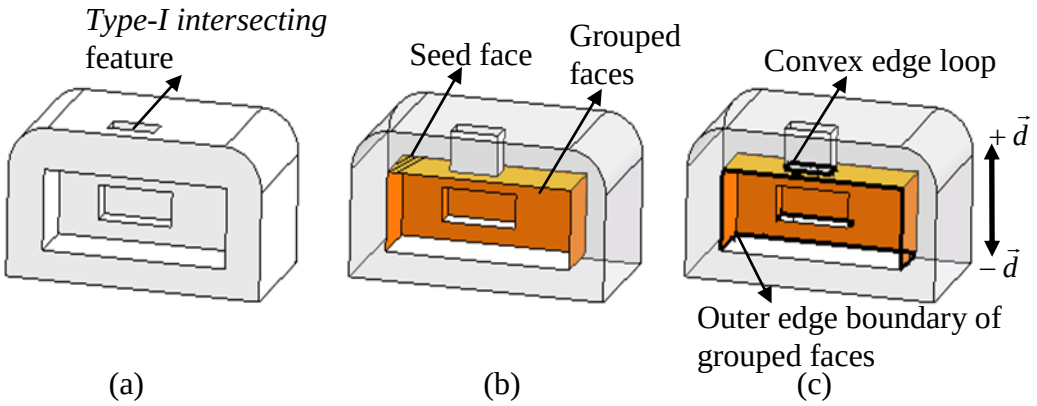


Figure 4-7: *Type-I Intersecting* feature recognition procedure: (a) part with *Type-I Intersecting* feature, (b) seed and grouped faces (to avoid clutter, the face obstructing the feature is hidden in figure), and (c) convex edge loop of *Type-I intersecting* feature.

To recognize the *Type-I intersecting* features accessible from the $+\vec{d}$ direction, as shown in Figure 4-7(a), a negative face of the *Depression_Face_Set* array that is nearer from outside the part in the $+\vec{d}$ direction is identified, and the face is used as a seed face, in Figure 4-7(b). All the negative and perpendicular faces adjacent to the negatively oriented seed face are selected from the *Depression_Face_Set* array and are sub-grouped into an array (*TypeI_Bounding_Faces*). Negative and perpendicular faces, adjacent to faces of *TypeI_Bounding_Faces* array, of *Depression_Face_Set* array are also grouped to *TypeI_Bounding_Faces* array, as shown in Figure 4-7(b). If the *Type-I* intersection is present, then there is a closed convex edge boundary within the outer edge boundary of the grouped faces, shown using bold lines in Figure 4-7(c). A similar approach is adopted for recognizing *Type-I intersecting* features; those are accessible from the $-\vec{d}$ direction. The convex edge loop needs to be shut-off to ensure de-moldability, and the loop edges are stored in an array called *TypeI_Bounding_Edges*. Once the *Type-I intersecting* features are identified, the *Type-II intersecting* features are detected by the following procedure.

4.4.3 Procedure for Identifying the Presence of Type-II Intersecting Features

To determine the type of depression feature (blind, through, or intersecting), the faces in the *Depression_Face_Set* array are further processed by determining the release direction and performing the accessibility analysis with respect to the release direction. It is evident from the earlier discussions that the presence of more than one loop in the *Depression_Bounding_Edges* array indicates that the depression under consideration can be through, intersecting or a combination of these two; otherwise it is blind with or without a child.

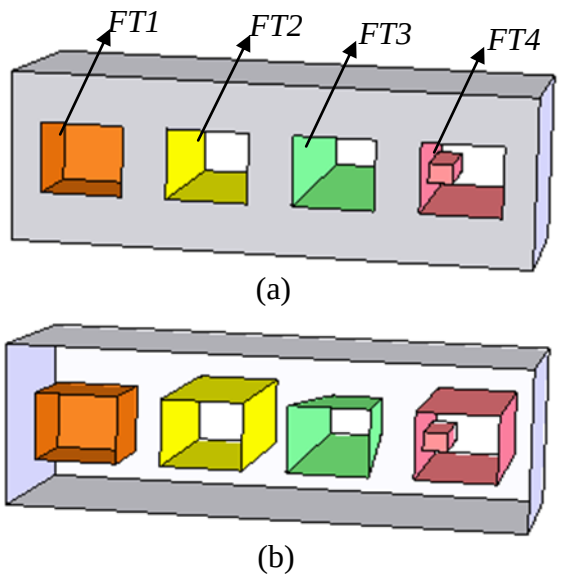


Figure 4-8: (a) Different types of depression features and (b) faces obstructing the features are hidden to show cross-sections.

Let the depression under consideration be formed by the faces belonging to set ***U*** and stored in a *Depression_Face_Set* array, as stated in Section 4.4.1. The corresponding *Depression_Bounding_Edges* array contains the number of edge loops and related information, as stated in Section 4.4.1. The methodologies adopted to classify the depressions are as follows.

- The possible range of the release direction(s) of each undercut is determined first, to classify the depression. The release direction is determined for each edge loop present in the *Depression_Bounding_Edges* array. The detailed procedure for determining the release direction is presented in Section 4.5 to maintain continuity.

- Let U_i be the set indicating depression faces fully accessible along the release direction determined by using loop l_i, where *i* varies from 1 to *N*, where *N* is the total number of loops in each *Depression_Bounding_Edges* array. Note that ***U*** contains all the faces forming the undercut, including the protrusion face-sets that are present in that undercut. The undercut type is determined by using set theory as follows.

  Let *N* be the total number of loops present in *Depression_Bounding_Edges* array

  $U = \{f : f$ is a face of the depression undercut under evaluation$\}$

  $U_i = \{f \in U : f$ is visible in the release direction determined by using loop l_i and $1 \leq i \leq \mathrm{N}\}$

i. If $N = 1$ and $U_1 = U$, then the undercut under evaluation is simple and blind (*FT1*, Figure 4-8).

ii. If $N = 2$ and $U_1 = U_2 = U$, then the undercut is through and accessible from both directions (*FT2*, Figure 4-8).

iii. If $N = 2$ and U_1 or $U_2 = U$ and the other one is $\{\varnothing\}$, then the undercut is through but accessible from one side only (*FT3*, Figure 4-8).

iv. If $N \geq 2$ and $U_1 \cup U_2 \cup \ldots . \cup U_N = U$ and $U_i \neq U$, then the undercut is a *Type-II* intersecting feature. However, it can be a bounded (*FT4*, Figure 4-4) or unbounded (FT8, Figure 4-4) *Type-II* intersecting feature. Further analysis is required to classify the undercut feature and to evaluate the convex edge boundary in the case of the *Bounded-Type-II* intersecting feature. The detailed procedure is presented in Section 4.4.4.

v. For any *N* and $U_1 \cup U_2 \cup \ldots .. \cup U_N \neq U$ and non-visible surfaces are obstructed only by the protrusion present in that undercut (*FT4*, Figure 4-8), then the undercut under evaluation needs a split core to be molded.

vi. For any *N* and $U_1 \cup U_2 \cup \ldots .. \cup U_N \neq U$, then the undercut under evaluation has an inaccessible child depression features (FT3, Figure 4-4). Once the presence of a child is identified, the edge boundary between the parent and child is determined. The details are presented in the following Section 4.4.4.

Once the presence of intersecting depression features is identified by using these conditions, the following procedure is followed to determine the type of intersection and to extract the edge boundary between the parent and child for the *Bounded-Type-II* intersecting features.

4.4.4 Procedure for Type-II Intersecting Features Recognition

All the possible loops, consisting of convex edges other than the ones present in *Depression_Bounding_Edges* array or *TypeI_Bounding_Edges* array, are determined, as portrayed in Figure 4-9 (the procedure to determine the convex edge loops is presented in Appendix C). One edge can exist in multiple loops, e.g., edge e_1 in Figure 4-9(a) is present in two convex edge loops as shown the by bold lines in Figure 4-9(b) and (c). If there is no convex edge loop and condition '*iv'* mentioned in Section 4.4.3 is satisfied, then all the requirements of the *Unbounded-Type-II* intersection are fulfilled, and the feature is identified. This type of feature must be divided into more than one feature for de-moldability, e.g., feature *FT8* in Figure 4-4. Their automatic division is outside the scope of this work.

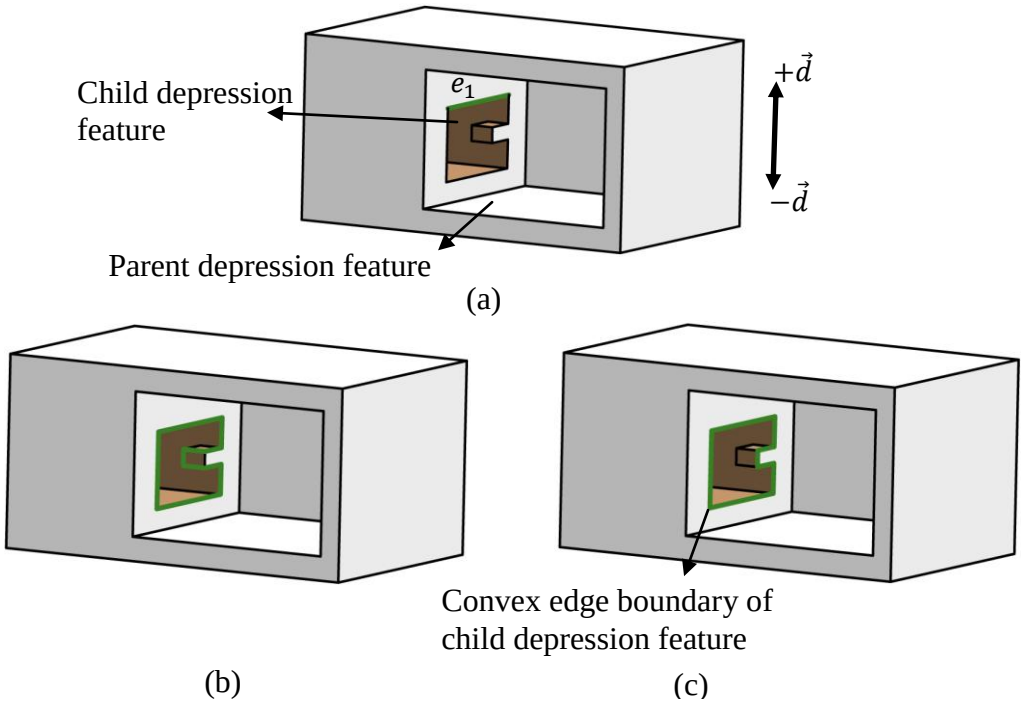


Figure 4-9: Convex edge loops for seed edge e_1 are shown using bold lines in (b) and (c).

Next, the undercut faces, inaccessible from the release direction, are grouped based on adjacency. If any undercut face shares concave edges with the group of inaccessible faces, then that face is also added to the group. Now the convex edge loops are evaluated for each inaccessible face group to determine the boundary between the parent and the child. The loop, shown in bold in Figure 4-9 (c), separating the accessible and inaccessible faces of the undercut under evaluation, is identified. To identify the faces of the child undercut, the undercut faces on the inaccessible side of the loop are grouped. To determine if the child undercut is through or blind, the bounding edges of the group of faces, forming the child undercut, are extracted from the part geometry. If these edges are present in the *Depression_Bounding_Edges* array, then the child undercut is a through depression. Otherwise, the child undercut is a blind depression.

If the child feature is not a through depression, then its release direction is determined by using the edge loop separating the parent and child features. However, the retraction space availability for releasing the split-core during de-molding is not checked.

If the child undercut is classified as a through depression, its accessibility is checked along the release direction that is calculated by using the corresponding edge loop of the *Depression_Bounding_Edges* array.

Once the features are identified, including the *Type-I* and *Type-II* intersecting features, the remaining faces of the part are grouped as core, cavity and additional features by using the accessibility analysis results and their adjacency.

4.5 Determination of the Undercut Release Direction

The release direction for the protrusion and depression features is required to ensure de-moldability of the side-core and cavity. For the protrusion undercuts, if the face set is fully-accessible along the $\pm\vec{d}$ direction, the undercut is considered moldable by using the core and cavity; otherwise, a release direction for the feature is generated. The release direction for depression undercuts is determined with the help of the information present in the *Depression_Face_Set* and *Depression_Bounding_edges* arrays (whereas the *Protrusion_Face_Set and Protrusion_Bounding_Edges* arrays are used for determining the release direction of protrusion undercuts) by using the procedure presented next.

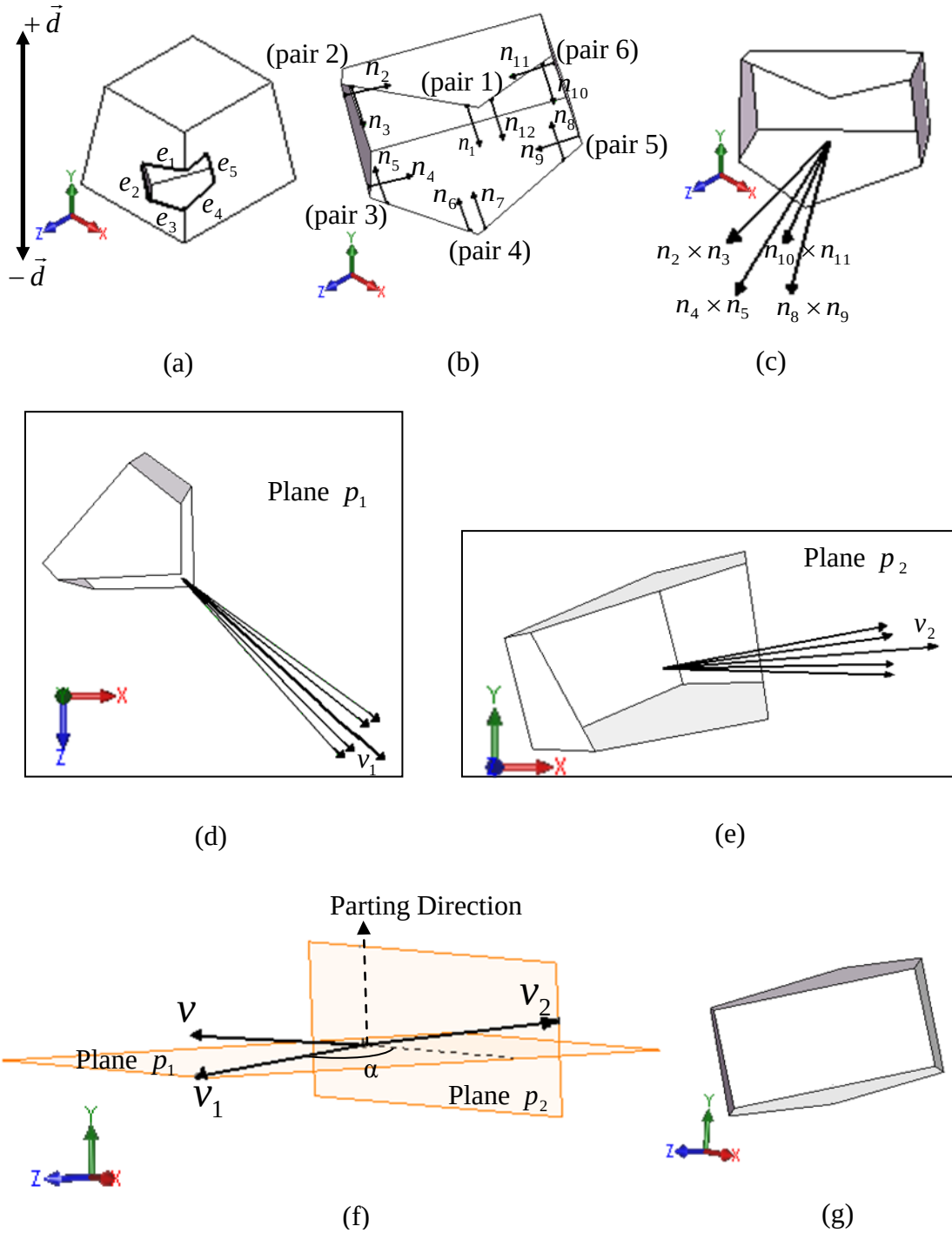


Figure 4-10: Procedure for release direction determination.

4.5.1 Procedure for Determining Undercut Release Direction

The undercut release direction is determined for each bounding edge loop present in the *Depression_Bounding_edges* array of the considered undercut. If the depression undercut is of the through type, the release direction is determined for both openings of the undercut.

The procedure for determining the release direction for an edge loop starts with the identification of undercut faces adjacent to the bounding loop edges, shown in Figure 4-10(a). For each edge, surface normals of its adjacent undercut face in the proximity of both of its vertices are determined, as shown in Figure 4-10(b). In an edge loop, each vertex is shared by two edges. Therefore, each vertex has two normal vectors adjacent to it, forming a pair. The cross product of each pair of normal vectors is determined, as shown in Figure 4-10(c). However, if the normal vectors of a pair are parallel or opposite to each other, then that pair is not considered; this would be the case with edge pair 1 and 4 in Figure 4-10(b).

The purpose of further work is to determine a center vector of all the cross product vectors in 3D. The objective is simplified by solving it in 2D, and cross-products are analyzed on two planes that are perpendicular to each other. The detailed procedure is presented in the following section.

These cross-product vectors are translated to a common point (Figure 4-10(c)) and projected on to a plane (p_1) perpendicular to the parting direction ($\pm\vec{d}$), as shown Figure 4-10(d). If the number of projected cross-product vectors is fewer than four, the two outer most vectors are selected and their average vector (v_1) is calculated. Otherwise, the two innermost vectors are averaged to determine v_1, as depicted in Figure 4-10(d).

Now the cross product vectors are rotated about the parting direction onto a plane p_2, perpendicular to p_1, in such a way that all the rotating vectors are pointing towards one hemisphere, in Figure 4-10(e). An average vector (v_2) of the rotated vectors is determined by using a similar procedure that was followed for v_1.

Finally, the release direction vector v is determined by rotating vector v_2 about the parting direction by an angle equivalent to the minimum angle between v_1 and p_2, shown in Figure 4-10(f).

Such a procedure is graphically illustrated in Figure 4-10 with the help of a depression undercut (Figure 4-10(a)), and the parting direction is chosen to be parallel to the Y-axis. Figure 4-10(b) and Figure 4-10(c) show the normal vectors and their corresponding cross product vectors, respectively. The cross product vectors are then projected onto plane ZX (plane p_1) and the vector v_1 is determined by averaging the innermost two vectors, as illustrated in Figure 4-10(d). Next, the cross-product vectors are rotated about the parting direction, i.e., the Y-axis, onto the XY plane (plane p_2) in such a way that all the vectors point towards the +X direction, and then their average vector v_2 is determined, as depicted in Figure 4-10(e). Finally, the release direction is determined by rotating vector v_2 about the Y-axis by angle α equivalent to that of the minimum angle between vector v_1 and plane p_2, as shown in Figure 4-10(f). It can be seen in Figure 4-10(g) that all of the undercut faces are fully visible from the release direction.

4.6 Implementation

To recognize mold features, including intersecting ones, the mold feature recognition methodology presented here was implemented using Visual Basics 6.0 and Solidworks 2007. Solidworks has a large set of user routines that can be used to obtain geometrical and topological data from a B-rep model. The newly developed feature recognition system was executed on an Intel Pentium IV CPU 1.75GHz, 1 GB RAM with the Windows XP operating system. The mold feature recognition system requires the user to input the part information in B-rep format and specify the parting direction. The system then recognizes the depression and protrusion features, and determines the release direction of the undercut features. An efficient data structure was developed to store and retrieve the necessary information. The developed system was tested on various parts with a variety of surfaces, including free-form surfaces, to validate its effectiveness.

Four case studies are presented in this section. The first case study, in Figure 4-11(a), is geometrically similar to the benchmark part considered by Ye et al. [33]. Figure 4-11(b) illustrates the features that are recognized by the newly developed system. The proposed algorithm successfully recognizes five depression features and six protrusion features, along with their release directions (shown by the dotted arrows). Similar feature information is reported in the work Ye et al. [33]. However, similar to the

reported work [33], there is a need to split the face f (Figure 4-11(a)) into base face and undercut faces to recognize the protrusion features, in Figure 4-11(c).

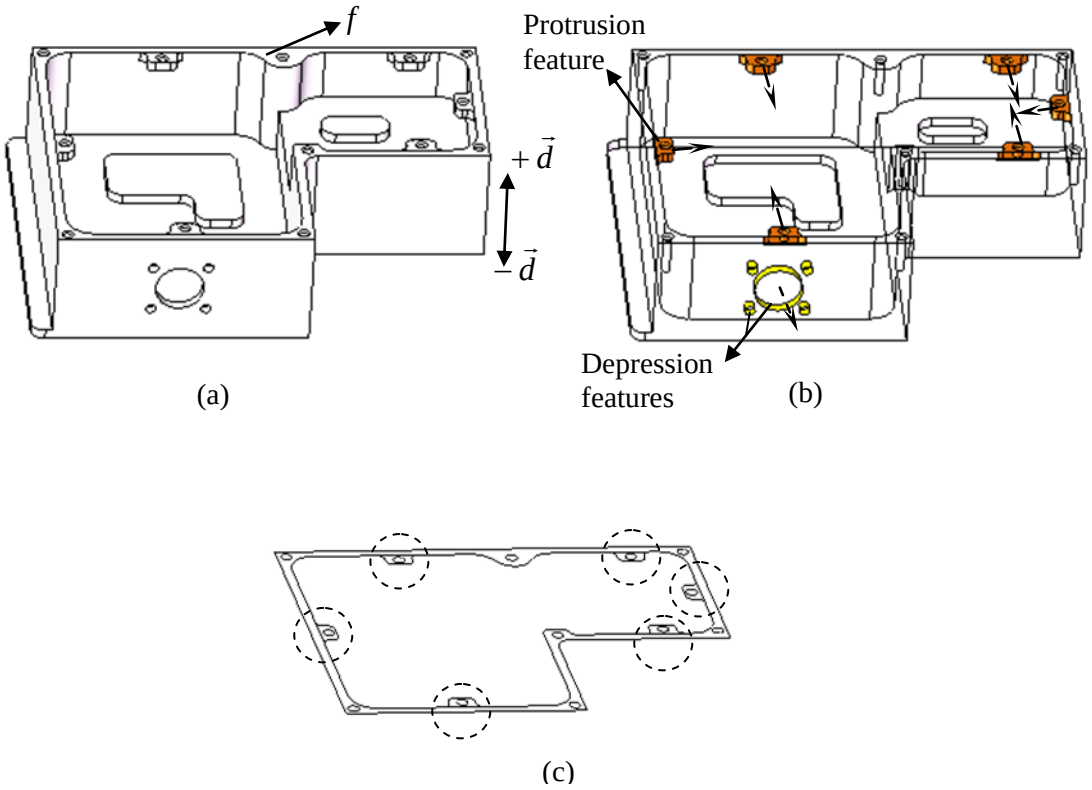


Figure 4-11: Test part 1 from the work of Ye et al [33]: (a) part model, (b) recognized features and release directions, and (c) splitting required for recognizing protrusions.

The second case study, represented in Figure 4-12(a), is geometrically similar to an automobile part (the switch housing) [3] used in luxury cars. This injection molded part has three obliquely positioned holes. The proposed methodology successfully recognizes three depression features with their release directions, as denoted in Figure 4-12(c). Of three identical depression features, the release direction for one of the depression features is shown in Figure 4-12(c) as a dotted arrow.

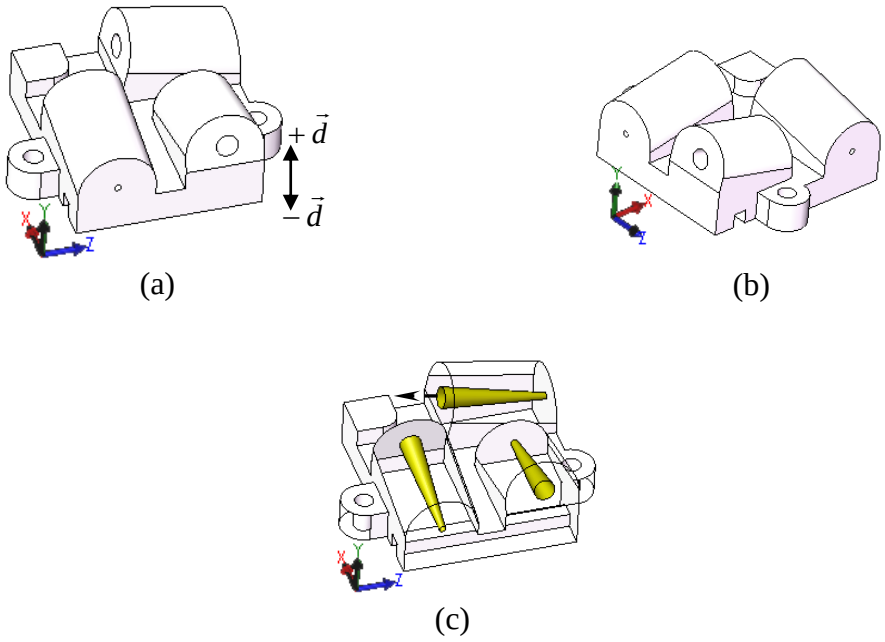


Figure 4-12: Test part 2, an injection molded part from [3]: (a) part model, (b) part model in another view, and (c) recognized features and release direction.

The third case study, in Figure 4-13(a) is geometrically similar to an industrial part (the throttle knob). This part has one depression feature and one through hole in the parting direction. The face f, in Figure 4-13(a), of the depression feature is partially accessible (i.e., the face is partially outside the depression undercut, and is successfully identified during the accessibility analysis in the present work as a *partial outer boundary accessible surface*). The system prompts the user to split such faces, and the splitting is done interactively. Once f is split into the two faces, in Figure 4-13(c), the algorithm successfully recognized the depression features, along with their release direction, shown in Figure 4-13(d).

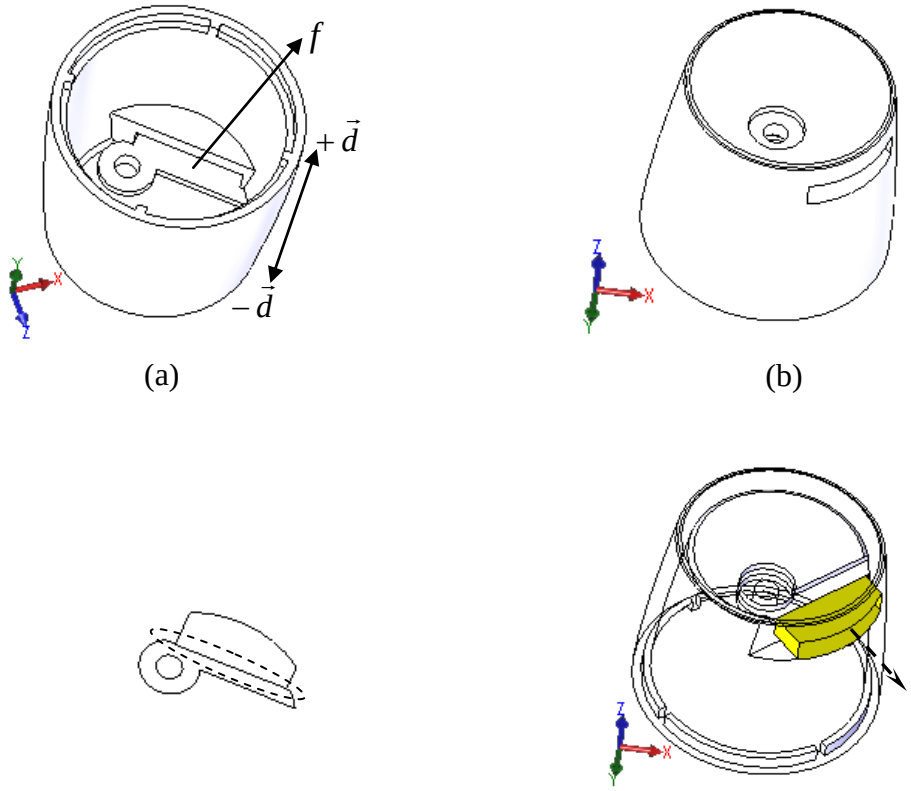


Figure 4-13: Test part 3, an injection molded industrial part: (a) part model and parting direction along the *Y*-axis, (b) part model in different view, (c) face split, and (d) recognized feature and its release direction shown by dotted arrow.

The geometry of the fourth case study is selected to explain the relative merits of the present system with respect to previous work. The part in Figure 4-14(a) has free-form surfaces as well as protrusion, depression, and intersecting depression features. The feature *FT1*, in Figure 4-14(b), is a depression feature (commonly known as a *slot* in machining) with one blind depression, one through depression, one protrusion child, and one *Type-I* intersecting feature. In this depression feature, face f_1 in Figure 4-14(b) is fully accessible from the parting direction. This face has been successfully recognized as part of the depression undercut. The faces of the *Type-I* intersecting features are accessible from the parting direction, and are successfully identified as part of the core surfaces. The feature *FT2* is an *intersecting* depression feature and is intersecting with two features. One of these is a *Type-I intersection* and the other one is a *Bounded-Type-II*

intersection. All three features are recognized and their release directions are determined. The feature *FT3* is a simple depression undercut. The feature *FT4* is a depression feature with an opening in the parting direction and its release direction is determined along the cylinder axis. The feature *FT5* is a result of the intersection of two depression features and does not have any convex edge loop that can separate the two intersecting features. This feature has been identified as an *Unbounded-Type-II* intersection and the release directions are determined for both features involved. The solid arrows in Figure 4-14(c) show the release directions of the undercut features.

The present system recognizes the various types of intersecting features that have not been reported in the published work. The features recognition approach uses the accessibility information of part surfaces along the parting direction, part geometry and topology. The contributing factor that facilitated the recognition of intersecting features is that the topological and geometrical information of the part (in B-rep format) is preserved during the accessibility analysis, for use in the feature recognition module. The sub-features of the intersecting features are identified. The edge boundaries and release direction of the features are also identified and used in the mold segments design.

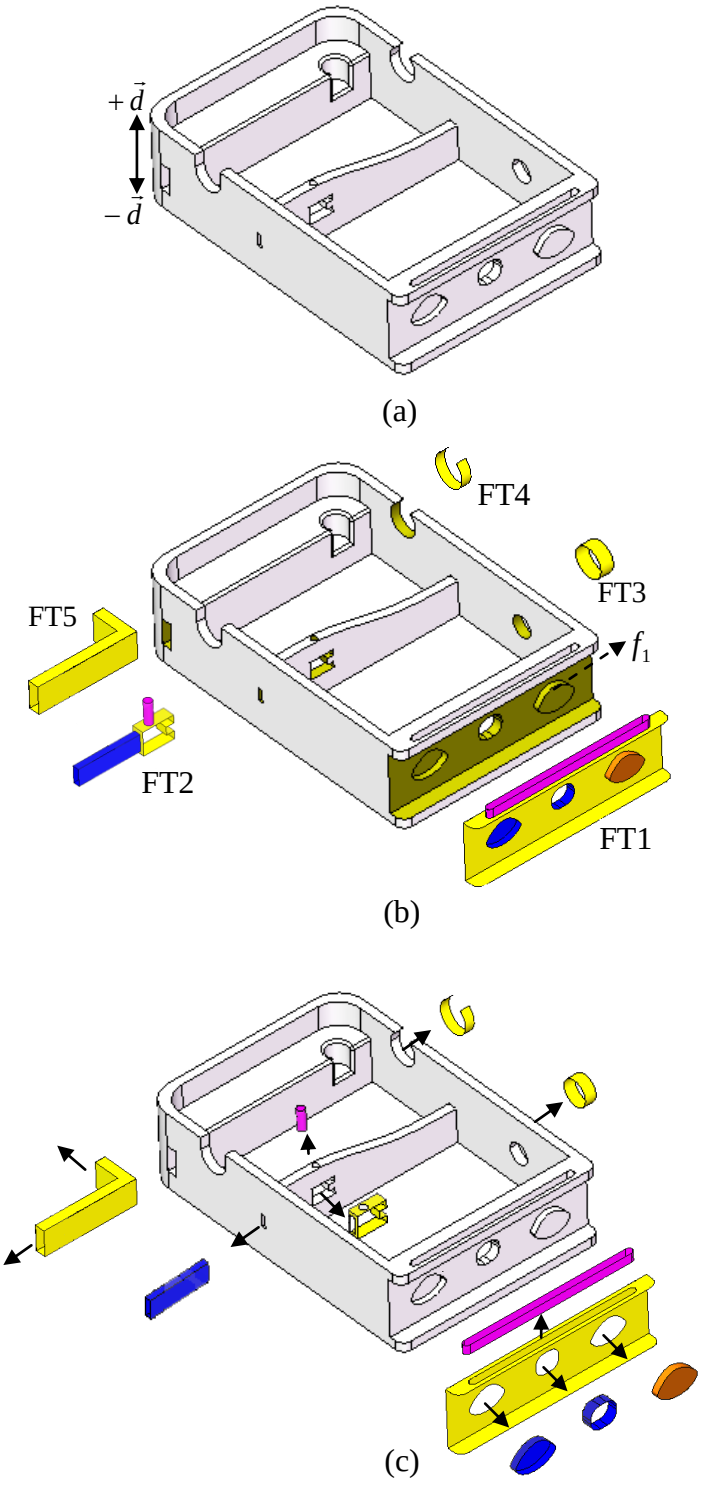


Figure 4-14: Test part 4, an injection molded part with intersecting features: (a) part model, (b) recognized features, and (c) intersecting features along with their release direction shown by solid arrow.

Chapter 5

Core, Cavity and Side-Cores Design

In Chapter 4, a method for identifying the mold features was presented. The surfaces forming these mold features can only be molded using side-cores and side-cavities. The part surfaces not requiring side-core/cavity are classified into core, cavity and core/cavity surfaces, where the core/cavity surfaces are the surfaces that can be molded by core as well as cavity. The process of automated design involves determination of edges forming the parting line, generation of parting surfaces, building of core and cavity blocks, and generation of core, cavity, and side-cores/cavities.

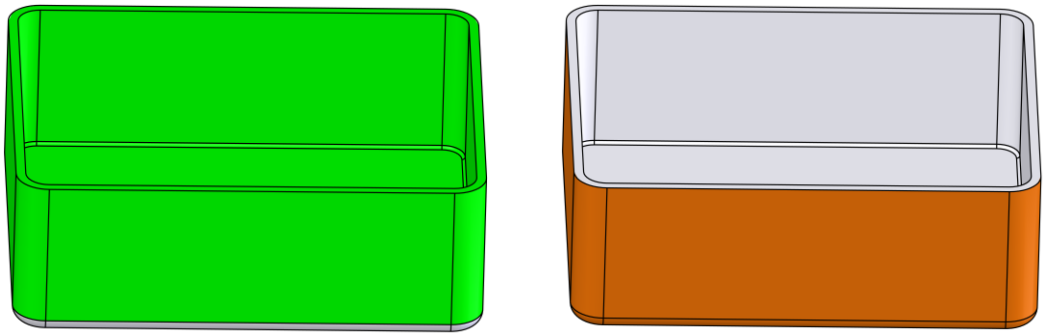
In this chapter, the steps involved in designing core, cavity, and side-cores automatically are discussed. The methodology of classifying the part surfaces, other than the mold feature surfaces, into core and cavity surfaces to identify the edges forming the parting lines is discussed in Section 5.1. Generation of parting surfaces and building of mold blocks are discussed in Section 5.2. The process of generating the mold segments is discussed in Section 5.3. The various aspects of the methodology are discussed in Section 5.4. The methodology has been tested on industrial parts and the results are discussed in Section 5.5.

5.1 Identification of Mold Parting Lines

Mold parting lines separate the core-molded surfaces from the cavity-molded surfaces. In this work, the parting lines are considered to be passing through the edges of the part surfaces.

The process starts with identification of a *Fully-Accessible* positive surface (called a *Seed* surface), provided that the surface should not be part of any mold feature. The adjacent *Fully-Accessible* positive and perpendicular surfaces, which are not part of mold features, are grouped with the *Seed* surface. The process is repeated for all the surfaces in the group; and positive and perpendicular surfaces adjacent to group surfaces are added to the group. The grouped positive and perpendicular surfaces for a sample part are shown in Figure 5-1(a). The edges separating the group of *Fully-Accessible* positive and perpendicular from the negative or inaccessible surfaces are extracted, shown in Figure

5-1(c). The edges separating the grouped positive surfaces from the depression features' surfaces are not considered because a parting line requires that its adjacent surfaces should be fully accessible from the parting direction; therefore, it cannot pass through these mold features. The rest of the identified edges are arranged in closed loops. There can be more than one closed loop, as shown in Figure 5-2(c). The edges of the outermost loop are used as parting lines and rest of the loops are closed using *Shut-off* surfaces to mold the part.



(a) (b)

(c)

Figure 5-1: Grouping of perpendicular surfaces with positive and negative surfaces. (a) *Positive* and *perpendicular* grouped faces are shown in green, (b) *negative* and *perpendicular* grouped faces are shown in brown, (c) Edge boundary of *positive* and *perpendicular* grouped faces are shown in red; and edge boundary of *negative* and *perpendicular* faces are shown in blue.

Similarly, negative and perpendicular surfaces are grouped and bounding edge loops are extracted. If any surface is a member of both groups, that surface can be molded either by core or cavity and is called a *core/cavity surface*. The core, cavity and

core/cavity surfaces of the parts, used to demonstrate the capabilities of the mold feature recognition methodology in Chapter 4, are shown in Figure 5-3. The loops obtained by grouping the surfaces are then analyzed. One parting line edge loop is obtained by grouping the positive/perpendicular surfaces and the other loop is obtained by grouping the negative/perpendicular surfaces, shown in Figure 5-1(c). If both of the loops consist of same edges, the resulting parting line is unique and optimal. However, if the loops consist of different edges, the user has the option to select the required parting line based on topological requirements. By default, the edge loop bounding the group of negative and perpendicular surfaces is used as the parting line.

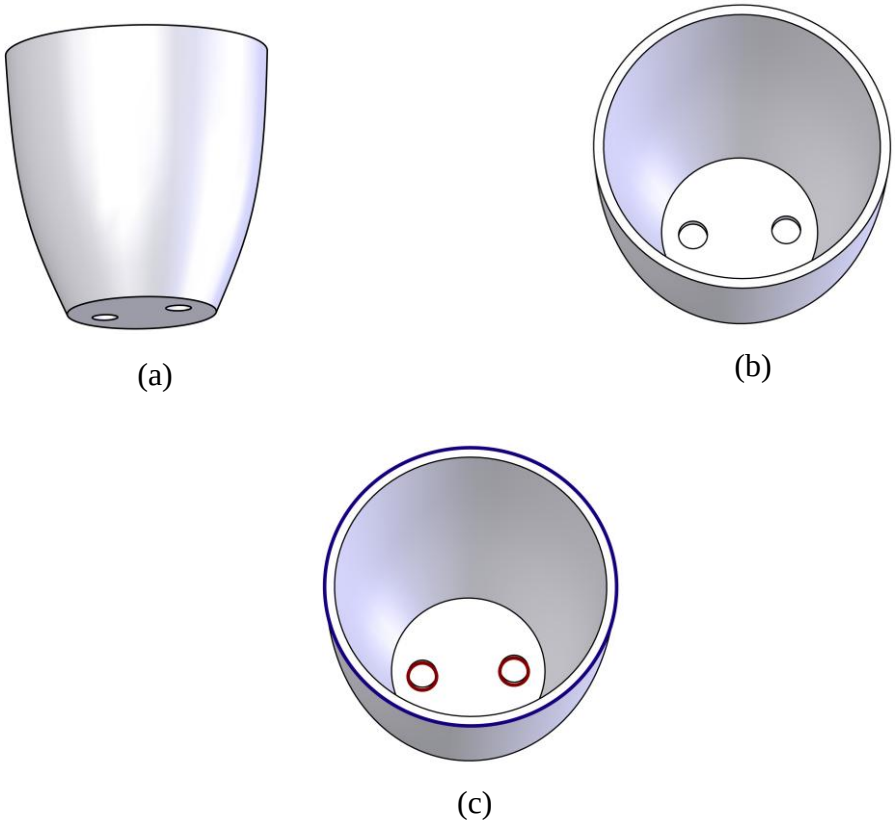


Figure 5-2: Procedure for identifying parting lines of a molded part. (a) Molded part, (b) molded part in different orientation, and (c) edge loop in blue corresponds to parting line and red edge loops require *Shut-off* surfaces.

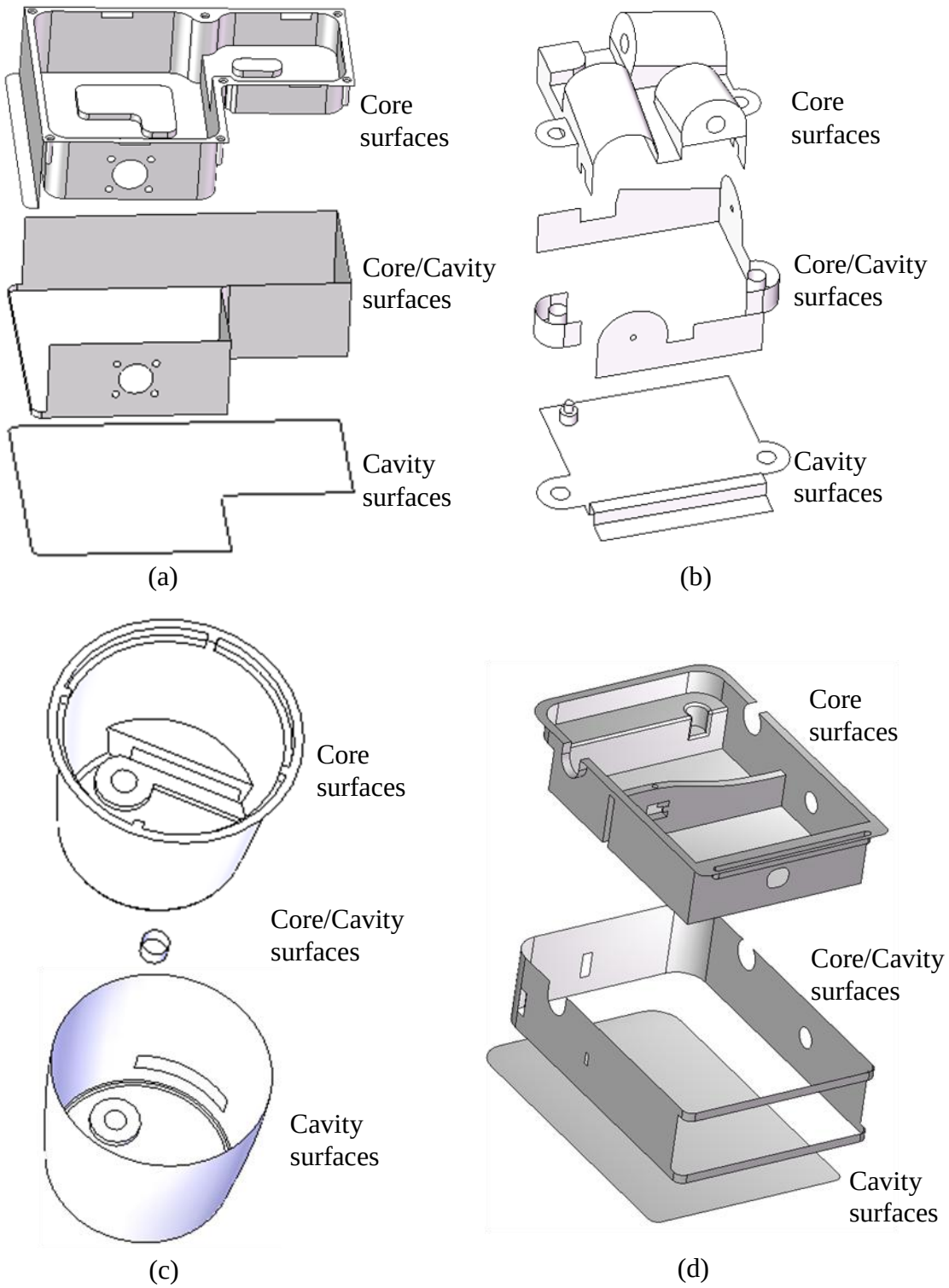


Figure 5-3: Core, cavity, and core/cavity surfaces of parts, shown in Figure 4-11, Figure 4-12, Figure 4-13, and Figure 4-14, are shown in (a), (b), (c), and (d) respectively.

5.2 Generation of Core and Cavity Block

The parting line is used for generating the parting surfaces. The parting line is extruded perpendicular to the parting direction to form the parting surface. The rest of the loops are closed with *Shut-off* surfaces, shown in Figure 5-4. To generate the mold blocks, a box is modelled around the part and the part is subtracted from the box using a regularized Boolean operation. The resulting box is split into two parts, core and cavity blocks, using the parting surfaces, as shown in Figure 5-5.

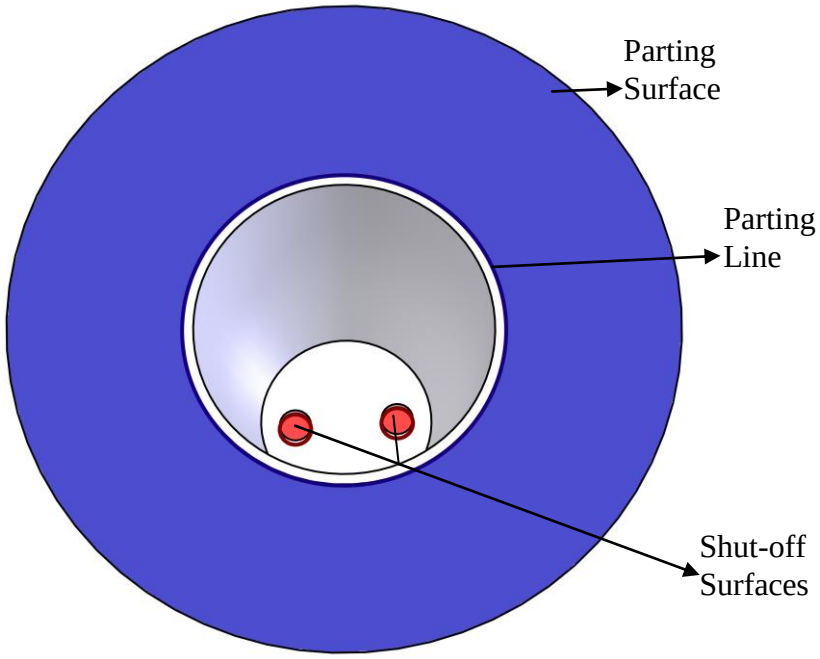


Figure 5-4: Parting line, parting surface and shutoff surfaces for part shown in Figure 5-2.

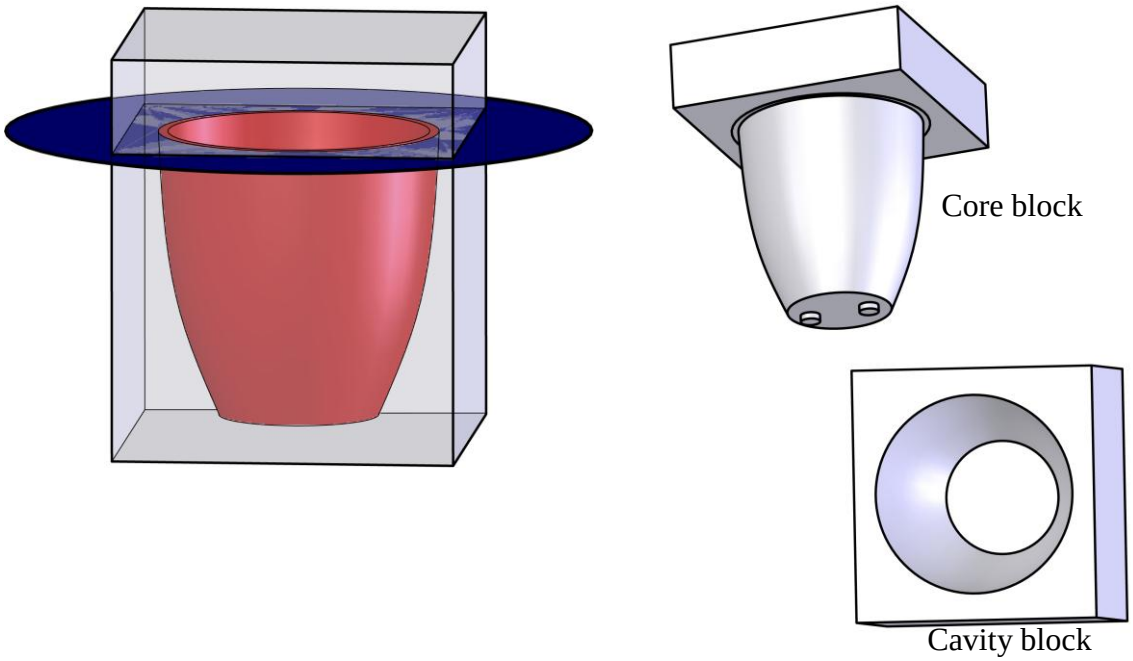


Figure 5-5: Generation of core and cavity block.

5.3 Generation of Mold Segments

To generate mold segments for each feature, a plane is formed normal to its release direction at a vertex of the feature's bounding edges, shown in Figure 5-6(c). The vertex should have a minimum distance along the feature release direction from the feature center. All the vertices of the bounding edges of the feature are projected onto the plane and a rectangle tightly enveloping these projected points is computed. A rectangle 10% bigger than the enveloping rectangle is sketched on the plane. Another rectangle, about one micrometer bigger than the recently drawn rectangle, is also drawn on the same plane, shown in the enlarged view in Figure 5-6(c). The area between the two rectangles is swept in the feature release direction to form a new body, and the body is Boolean subtracted from the cavity block to extract the volume formed by the side-core. The side-core is shown in Figure 5-6(d). After extracting the solid volume of all the side-cores, the cavity-block is used as a cavity.

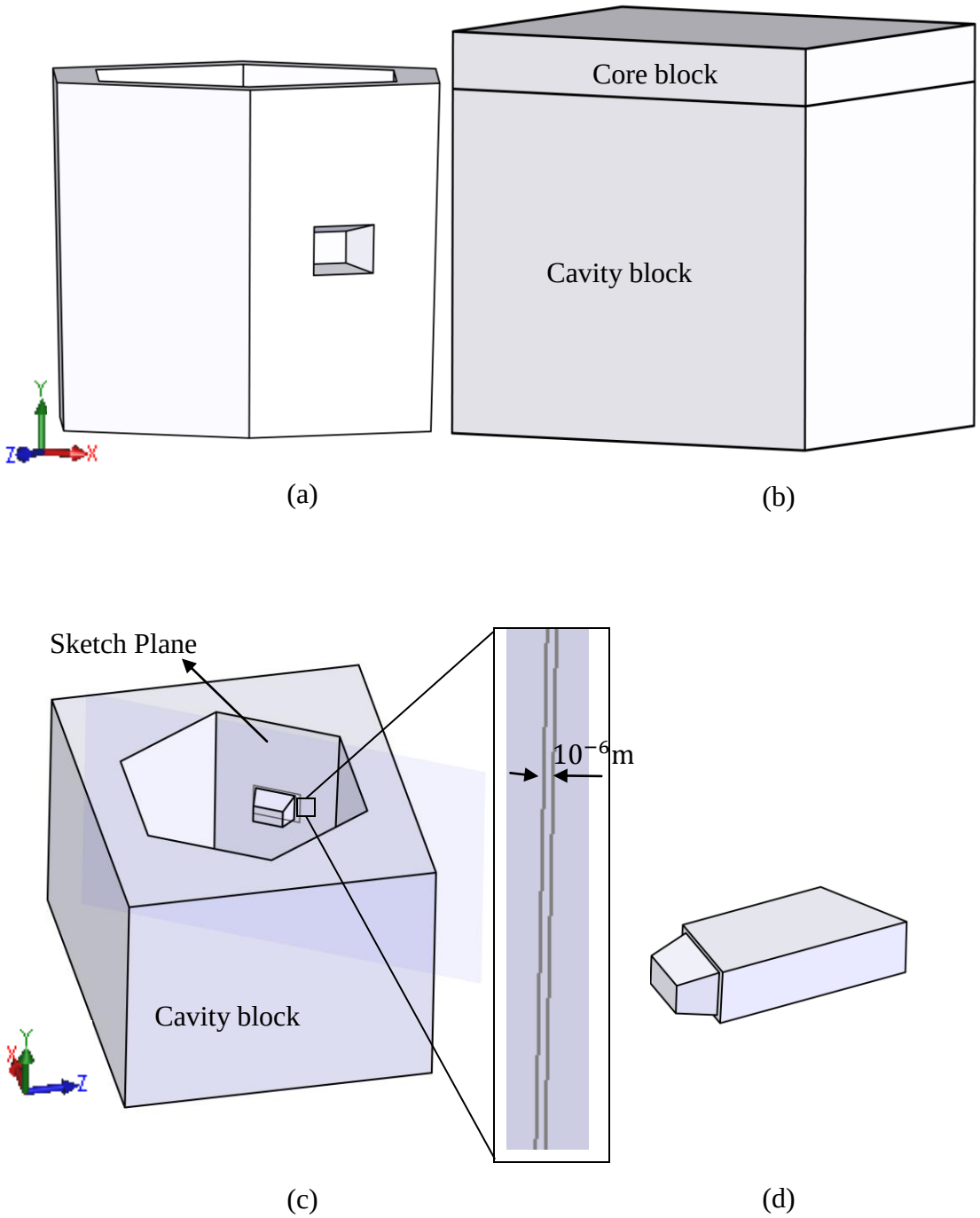


Figure 5-6: Procedure for extracting side-cores (a) Molded part, (b) core and cavity blocks of the part, (c) 2D sketch to extrude-cut the cavity block and its detailed view, and (d) side-core.

5.4 Discussion

To generate the side-cores of the recognized features, a simple approach is developed by assuming that the depression features are not intersecting. Intersecting features require further splitting of the generated side-cores to be used in the molding process and require further work.

The software system can generate side-cavities for the protruded features using the same approach. However, the side-cavities generated by extrude-cutting, as is done in case of side-cores, cannot guarantee part de-moldability using the segments. The reason is that the protruded feature may collide with the cavity surfaces while de-molding. Moreover, the use of side-cavities can be avoided by properly selecting the parting line. This requires further work.

5.5 Implementation

The mold segments are generated after recognizing the mold features, as discussed in Chapter 4. The feature based automatic mold design system successfully identified the edges forming the parting line, generated parting surfaces and mold segments. The system has been tested on complex industrial parts. The first test part is a throttle-knob having *Type-I* feature intersecting with depression undercut. There is also a through hole in the parting direction, therefore two edge loops are identified which separate the core and cavity surfaces. The inner edge loop, identified because of the through hole, requires a shut-off surface to generate the mold segments. The outer edge loop is used as the parting line. The parting surfaces are generated using the parting line. The generated mold segments, core, cavity, and side-core, are shown in Figure 5-7. The second test part is a flange sprocket, shown in Figure 5-8. The part has a core/cavity surface, one which can be molded using core as well as cavity. Therefore, the part has two edge loops that can be used as parting lines. The edge loop identified by grouping positive and perpendicular surfaces is used to generate the parting surface. The part also has a through hole in the parting direction and requires to be shut off before generating the core and cavity blocks. The core and cavity generated to mold the part are shown in Figure 5-8 (c) and (d).

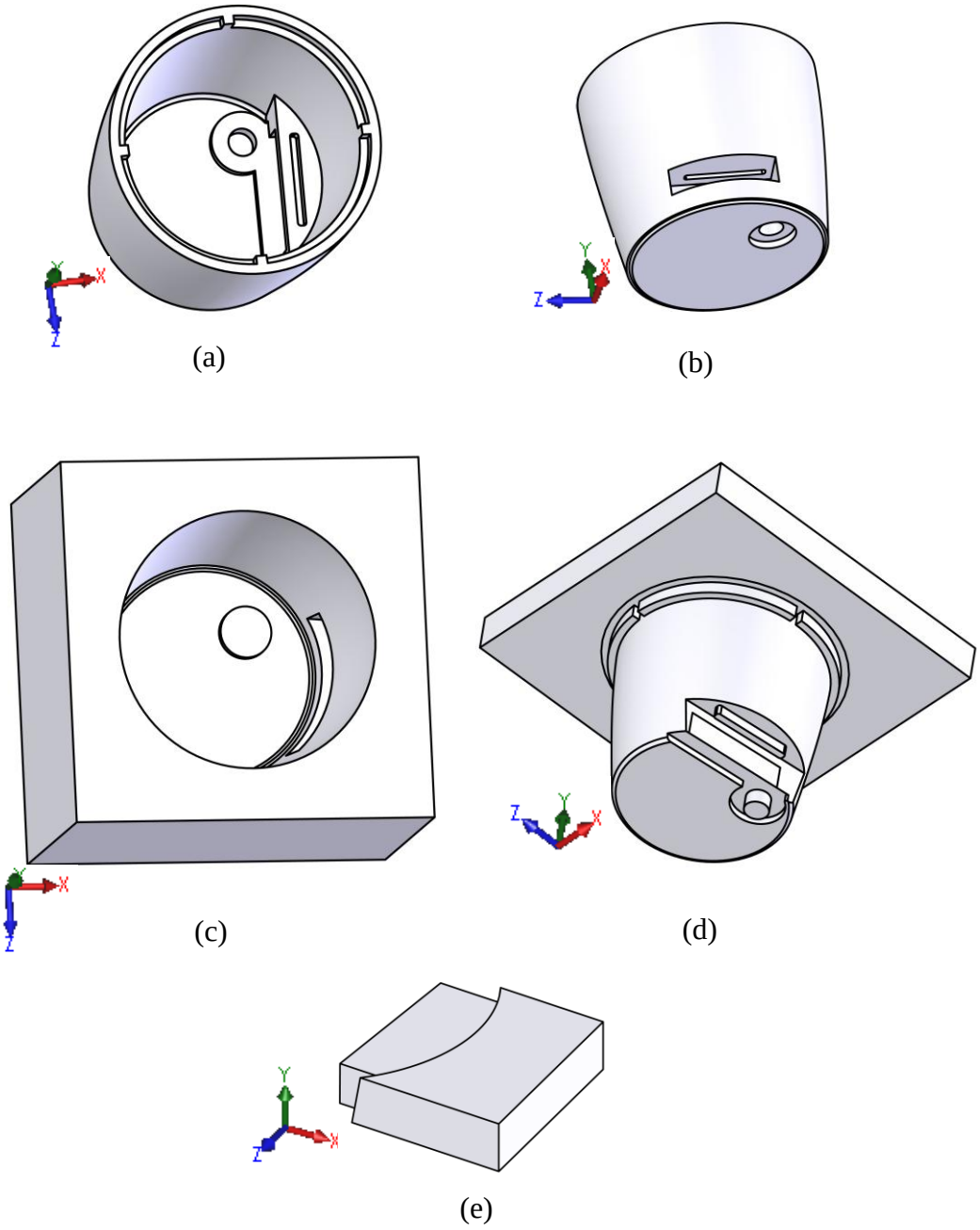


Figure 5-7: An injection molded industrial part and its alternate view are shown in (a) and (b). The generated cavity, core and side-core are shown in (c), (d), and (e).

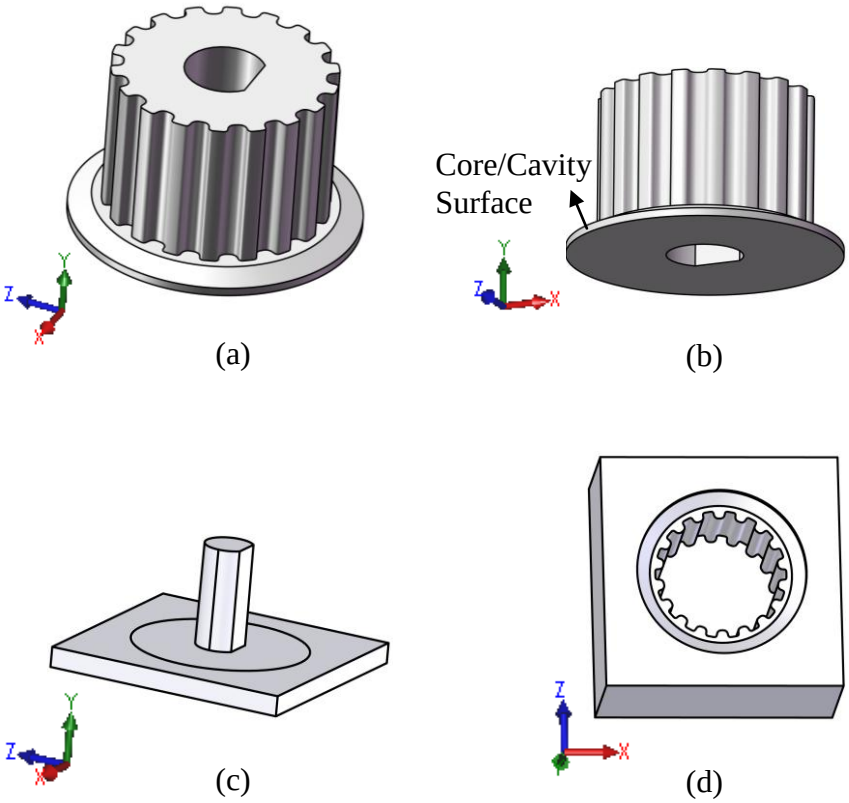


Figure 5-8: (a) An injection molded industrial part and, (b) its alternate view. The core and cavity generated to mold the part are shown in (c) and (d) respectively.

Chapter 6

Tool-Path Generation

Once the core, cavity and side-cores have been designed, they must be machined. A tool-path is required for machining these mold segments. Manual tool-path generation of large mold segments, such as a bumper facia, can take as large as 10% of total production time. This can add up to about 200 hours and increase the cost greatly as it requires an experienced and trained operator to produce the tool-path. The machining of mold segments is usually done in layers because mold materials need to have good wear resistance and dimensional stability, and these segments are usually made of tool steel [3].

An automatic method for tool-path planning would benefit the mold production as it could reduce the tool-path planning time and the associated cost. In this work tool-path planning is divided into two parts. The first part is the selection of a tool-path footprint and the second part is the positioning of the tool along the path in a gouge-free manner. There can be a number of footprint strategies to generate the tool-path. The most commonly used footprint strategies for mold machining are zigzag and contour parallel offset (CPO) [71]. The contour parallel offset tool-path is preferred over the zigzag for machining complex shapes as it consumes less time and provides better surface finish [42]. The gouge-free tool positioning is based on modelling tool positioning as dropping a ball on a triangulated surface and simulating it in software. This method has been used by Patel et al. [72] and modified here for this work.

The machining is usually done in two main stages: roughing and finishing. It is economical to use a large tool for rough machining, removing the maximum amount of material. Therefore, rough machining is usually followed by clean-up machining to cut the material left by roughing tool due to its large size. The clean-up pass is performed with a smaller-radius tool to machine the material left uncut by the roughing passes. During the clean-up pass, the tool moves over the entire work area to cut the left-over material. If regions left uncut after rough machining can be identified, unproductive tool movements can be avoided. A few researchers [48] have worked on automatic identification of the uncut areas; however, their approach approximates the uncut regions

using the tool-path footprints and cannot identify uncut regions due to the presence of obstacles within the machining area. Therefore, a robust approach that can identify the uncut regions in the presence of obstacles is required.

This chapter is organized into a number of sections. The overview of the approach is given in Section 6.1. The procedure to slice the part and extract the point sequence curves is discussed in Section 6.2. In the Section 6.3, the methodology used to generate valid offset curves is discussed. Then, uncut regions are extracted using the methodology discussed in Section 6.4. The algorithm has been implemented on a test part and the results are discussed in Section 6.5.

6.1 Overview

In this work, the procedure starts with slicing the part into a number of layers. Each layer has a number of edge loops. These loops can be internal loops or external loops. The internal loops are comprised of half-edges that are connected to each other in a clockwise manner, and the external loops are connected in a counter-clockwise manner.

Once the part is sliced into equidistant layers, the layers are processed from top to bottom. For each layer, the machining boundaries are extracted in the form of edge loops. These machining boundaries are used to generate CPO tool-path profiles for rough machining of the layer. To determine the tool-position, the tool is dropped at each offset point from a height well above the part until it touches the part surface. The tool-path is constructed by moving the tool between the determined tool positions. The regions left uncut are identified by the tool-geometry and graphics hardware-based depth-buffer information of the part, and the region boundaries are extracted. The region boundaries are further offset to generate a contour parallel offset tool-path for clean-up machining of each region.

In rough machining, the tool-path footprint must cover portions beyond the outer most contour (loop). The extent of the region to be covered outside the outermost contour depends on the stock shape. For this work, the stock is assumed to be a rectangular block that covers the entire part. Other stock shapes can be accommodated readily.

6.2 Extraction of Point Sequence Curves

In this work, the B-rep model of the part is evenly sliced with planes that are normal to the tool axis, as shown in Figure 6-1. The part shown in Figure 6-1 will continue to be used in the subsequent work to demonstrate the methodology. It has a rectangular base with a defined protrusion and a clear cavity. This part has been cut into nine layers as shown in the Figure 6-1(b). Next, the machining and obstacle loops are identified for each layer. A machining loop bounds the area that should be machined, whereas an obstacle loop bounds the area into which a tool must not enter. For each layer, the machining and obstacle loops consist of edges formed after slicing the part with the cutting planes.

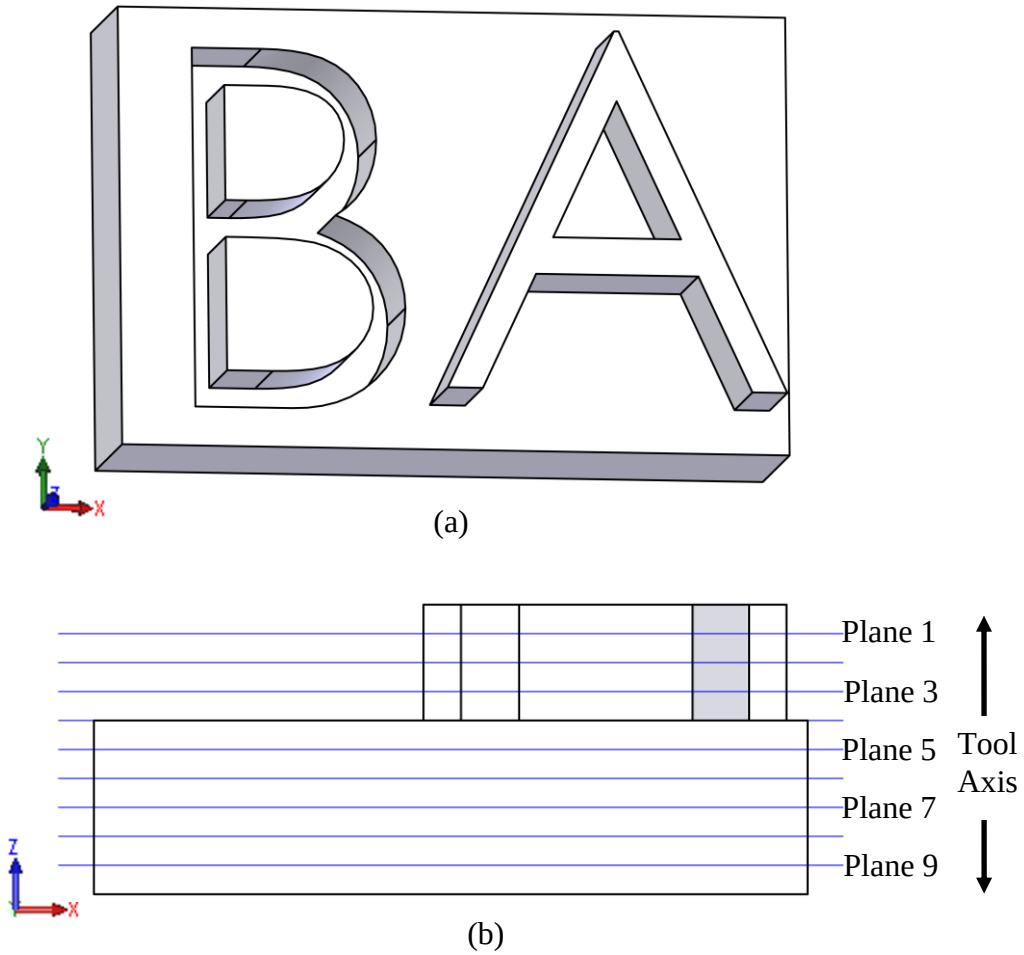


Figure 6-1: Slicing of the part with planes having normals parallel to the tool axis (a) sample part and (b) slicing planes.

To identify the machining and obstacle loops on each layer, the part topology is used to determine whether an edge loop is a machining loop or an obstacle loop. It is observed that if an edge loop forms an inner loop, then the area within the loop boundary should be machined and the loop is identified as a machining loop. On the other hand, if the edge loop is an outer loop, it represents an obstacle on that layer and the area within that loop should not be machined and the loop is identified as an obstacle loop. The points along the edges of the loops are extracted and are arranged in a sequence. The edge points of the machining loop are arranged in a clockwise orientation, whereas the edge points of the obstacle loop are arranged in an anti-clockwise orientation. The orientation of the points on the edge loops formed after slicing the part shown in Figure 6-1 with *Plane 1* is shown in Figure 6-2.

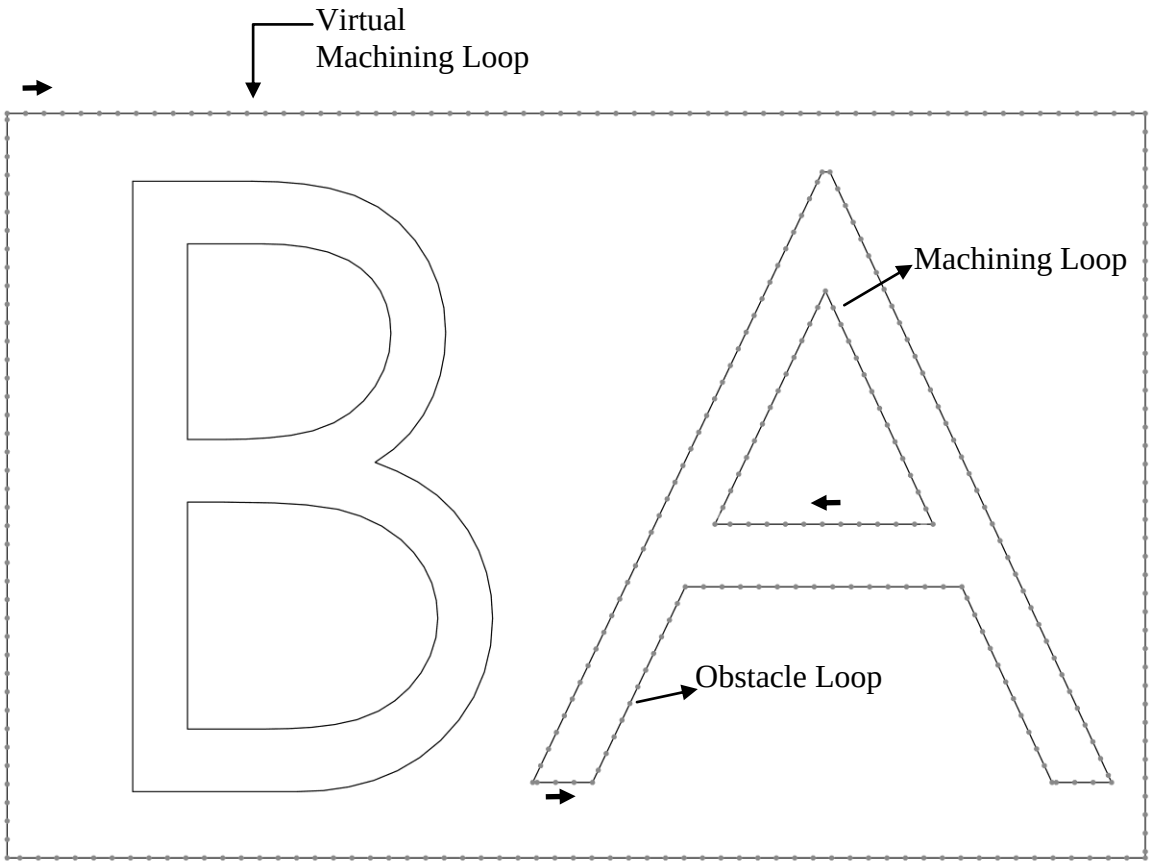


Figure 6-2: Boundary edge points, after slicing the part shown in Figure 6-1 with plane 1, arranged in clockwise order for machining area and in anticlockwise order for obstacles, shown using thick arrows.

It is possible that a machining area may not be bounded by any edge loop. For example, out of the two machining areas, shown in Figure 6-3, on the layer formed after slicing the part shown in Figure 6-1 with plane 1, the bigger area has no bounding edge loop. Such a case can occur only if an obstacle loop is not bounded by any machining loop on the given layer. If any machining area is identified without a bounding loop, then the part bounding box is sliced with the cutting plane; the new edges loop formed after slicing is considered as the virtual bounding loop of the machining area. The points on the virtual machining loop are also arranged in clockwise orientation, as shown in Figure 6-2.

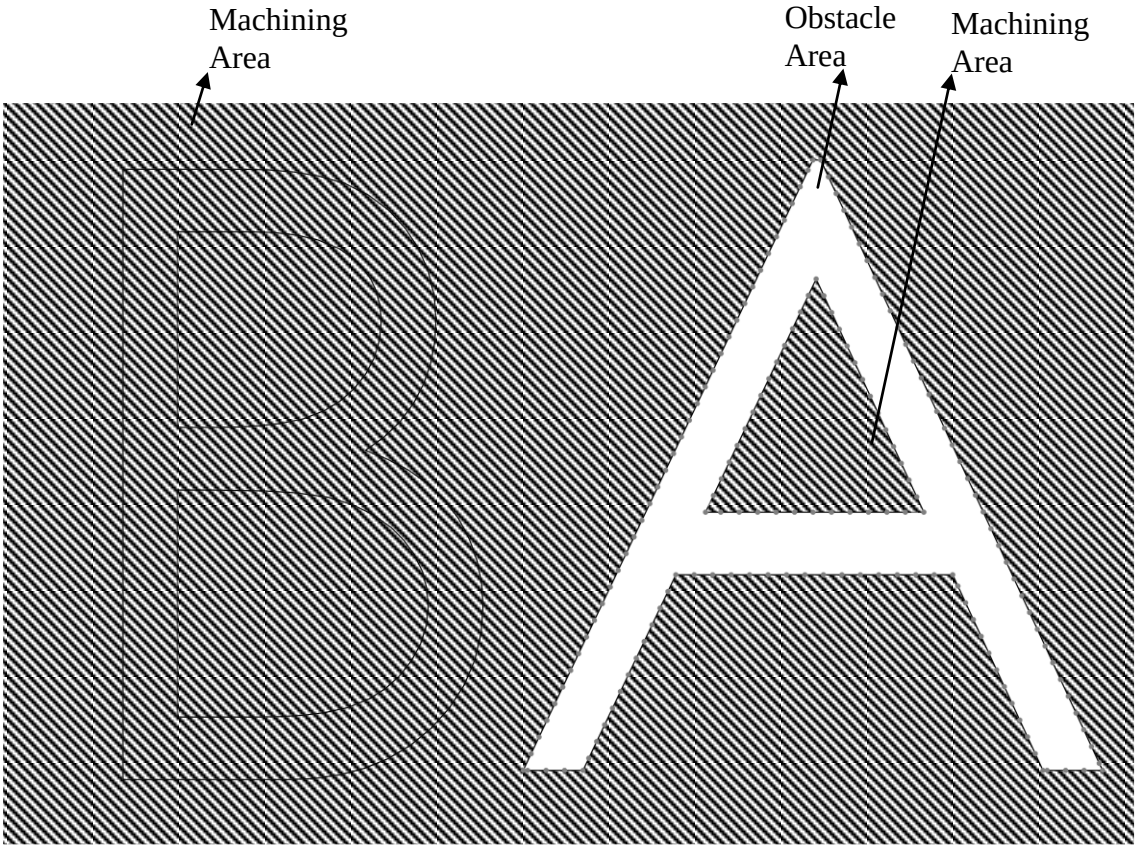


Figure 6-3: Machining and obstacle areas identified after slicing the part shown in Figure 6-1 with plane 1.

6.3 Generation of Offset Loops

The machining loops are offset inward to generate the contour parallel offset tool-path. Each offset loop is further offset until all the points of the new offset loop are invalid

(explained in Section 6.3.2). An offset point of a loop is invalid if it is either outside the loop or is closer than the offset distance to any point on the loop.

To generate an offset loop (called a *Child Loop*) within the boundary of a point sequence loop (called a *Parent Loop*), each point of the *Parent Loop* is rotated clockwise by 90° on the sliced plane about its predecessor point, shown in Figure 6-4. This can also be easily explained using vectors. Vectors are formed from each point to its next point in the loop and the vectors are rotated clockwise by 90° about the machining axis. The length of the new rotated vector is adjusted and is made equal to the offset distance. The rotated vector is called the *Offsetting Vector*. The end points of *Offsetting Vectors* are joined in clockwise sequence to construct a new offset loop. These end points are called the *Offset Points*.

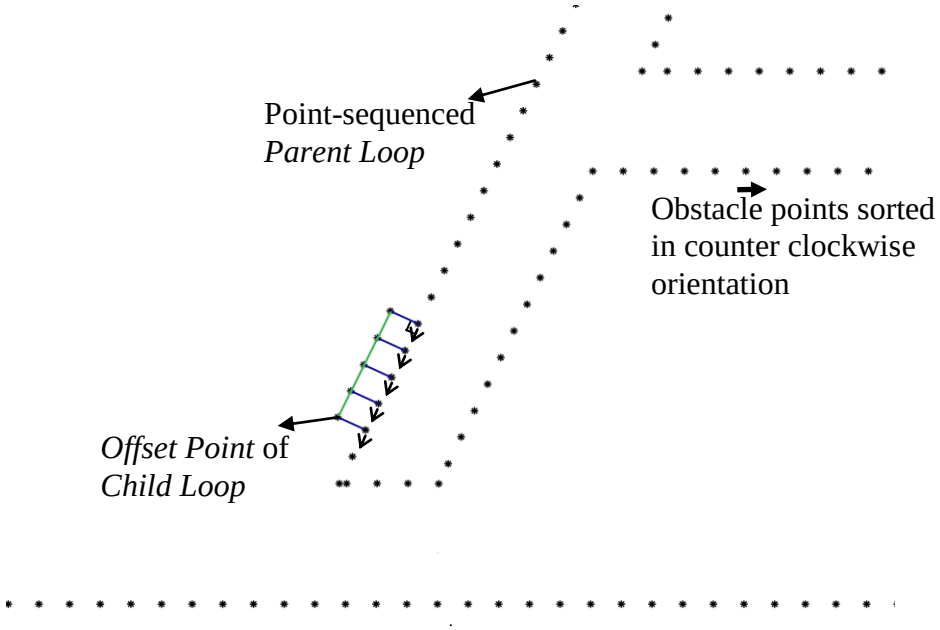


Figure 6-4: Generating an offset loop of a point sequenced loop.

If two adjacent points of the *Parent Loop* belong to a convex region, their *Offsetting Vectors* will not be parallel and will point away from each other, as shown in Figure 6-5; and the tool movement along such an offset loop can overcut the material.

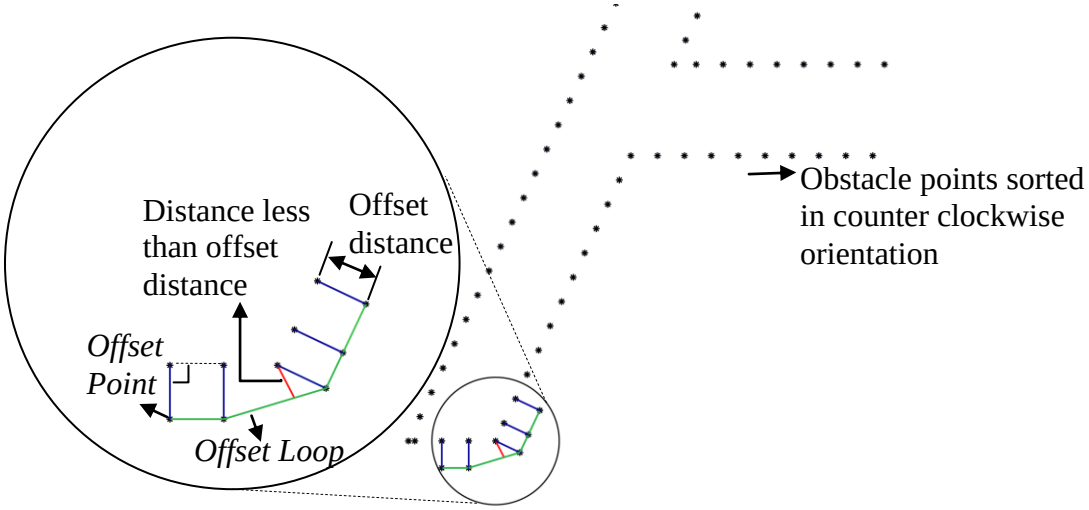


Figure 6-5: Overcutting due to angle between *offsetting vectors*.

To overcome the problem of overcutting, extra vectors are placed between the *Offsetting Vectors* of the two adjacent points of the *Parent Loop*. This is shown in Figure 6-6. First, the *Offsetting Vector* of the predecessor point is placed at the current point. More points are placed in between the two *Offsetting Vectors* by circular interpolation if the angle between two vectors is more than an allowed limit (say Φ degrees), shown in Figure 6-6.

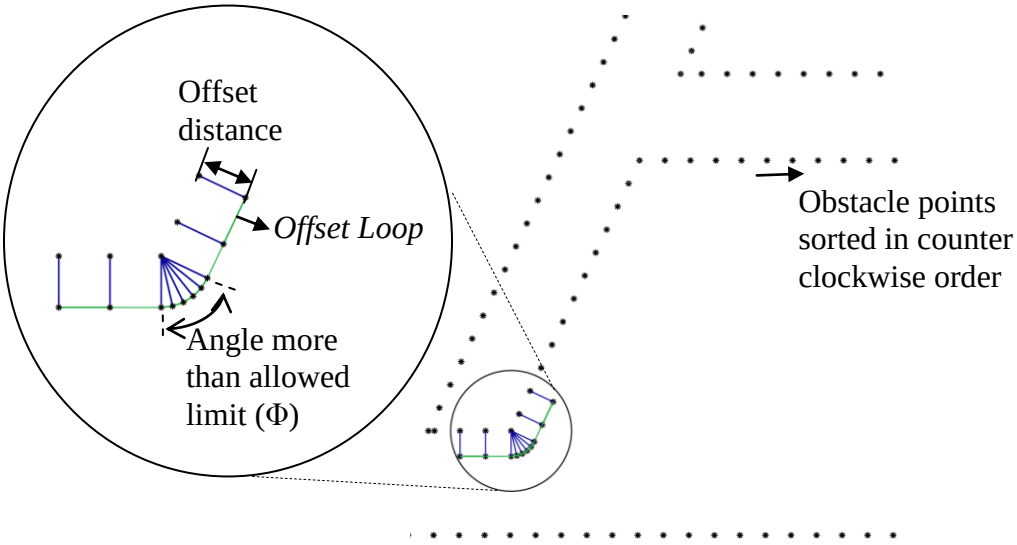


Figure 6-6: Insertion of extra offset vectors for smooth offset loop.

6.3.1 Tool-Path Generation for parts with Obstacles

After determining an *Offset Point*, the algorithm checks its distance from all the points on obstacle edge loops, if exists. If the distance is less than tool radius, the offset point is invalidated and tool movement is considered to be obstructed by the obstacle. The tool should now move along the obstacle to complete the offset loop. To achieve that, the algorithm finds a valid point on the obstacle loop that should be the next offset point. The obstacle loop now becomes the *Parent Loop* and the previous *Parent Loop* is treated as an obstacle loop.

The obstacle may divide the machining area into two or more regions. To find such a region after invalidating the offset point, proceeding points of the *Parent Loop* are traversed until a valid point, having distance greater than or equal to the tool radius from any obstacle point is found. This point can be the starting point of a new loop, and is called a *Seed Point*, shown in Figure 6-7.

After generating the offset loop, the *Seed Point* is discarded if its offset point is already used in the newly-generated offset loop. This situation occurs if the obstacle does not divide the machining region into two or more segments.

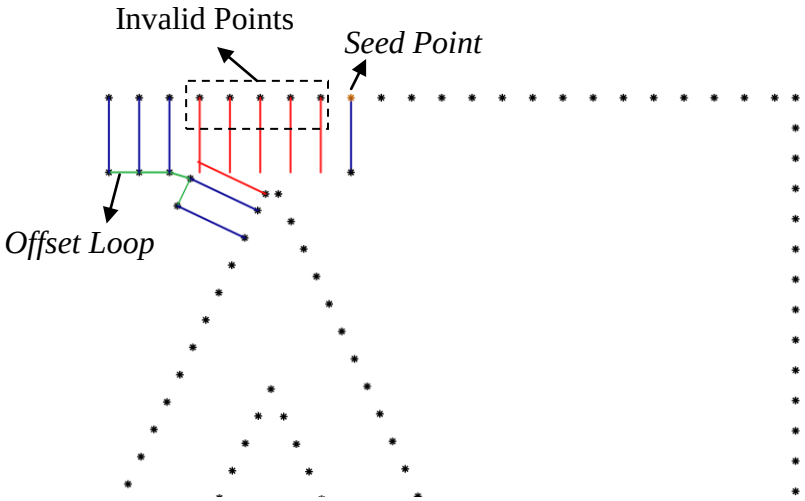


Figure 6-7: Incorporating obstacles while generating offset loops.

6.3.2 Removal of Invalid Loop Segments

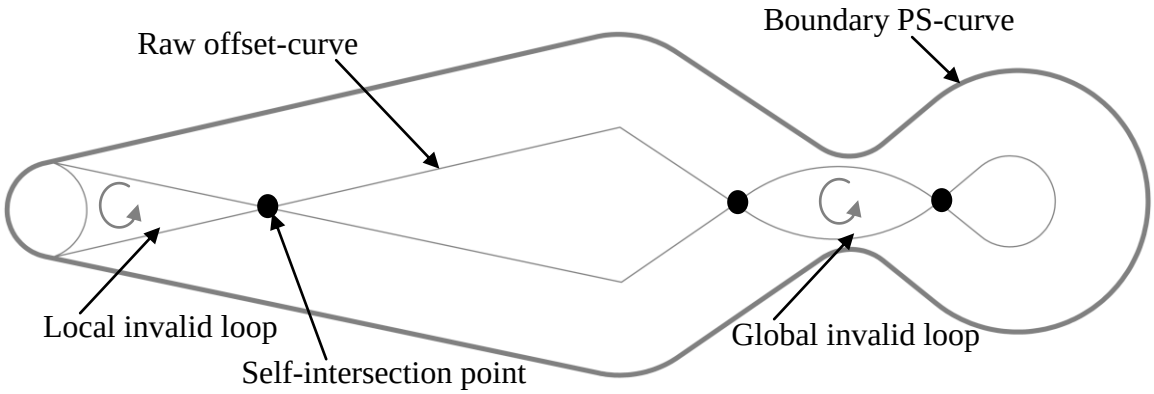


Figure 6-8: Local and global invalid loop segments [46].

The offset loop may have invalid loop segments as shown in Figure 6-8. There are two types of invalid loops: *local* and *global*. A local invalid loop is bounded by a single self-intersection point, whereas a global invalid loop is bounded by a pair of self-intersection points [46]. The invalid loop segments result in overcutting of material and must be removed. To identify the invalid segments of an offset loop, the invalid offset points whose distance from *Parent Loop* is less than offset distance are determined. The adjacent invalid points are grouped. If the group of invalid points form a closed loop, then the group/segment is identified as a local invalid loop. Otherwise, the start and end

points of each group is matched with the start and end points of other groups. If a match is found, the invalid segments are considered to form a global invalid loop. For each global invalid loop, the offset loop is divided into two separate loops.

6.4 Identification of Uncut Regions

The roughing tool can leave some uncut material due to tool size, step size or part topology. The identification of these uncut regions is important in that it allows the clean-up tool's motion to be restricted to the areas where work is required, avoiding unnecessary tool movements. The procedure can be divided into four stages:

- model a raw part (before any machining) and a finished part for each layer,
- determine tool position for each point of the roughing tool-path,
- compute the points on cut-surface generated by tool movement, and
- identify uncut regions, if they exist.

6.4.1 Procedure to model a Raw and a Finished Part for Each Layer

Since the machining is done in layers, the raw part used for machining the layer and the required finished part are modelled using the procedure discussed in the following paragraph.

For identifying the regions left uncut on the n^{th} layer after rough machining, the base surface of the part is swept up to the n^{th} and $(n+1)^{th}$ planes, as shown in Figure 6-9, and two separate parts in B-rep format are generated. For example, to determine the uncut region after rough machining the second layer, the base surface is swept up to *Plane 2* and *Plane 3.* The part formed by sweeping the base surface of the bounding box up to n^{th} plane represents the raw stock for rough machining and is called the *Stock.* Whereas the part formed by sweeping the base surface of the bounding box up to the $(n+1)^{th}$ plane represents the required finished part after machining the layer and the part is named the *Finished Part*.

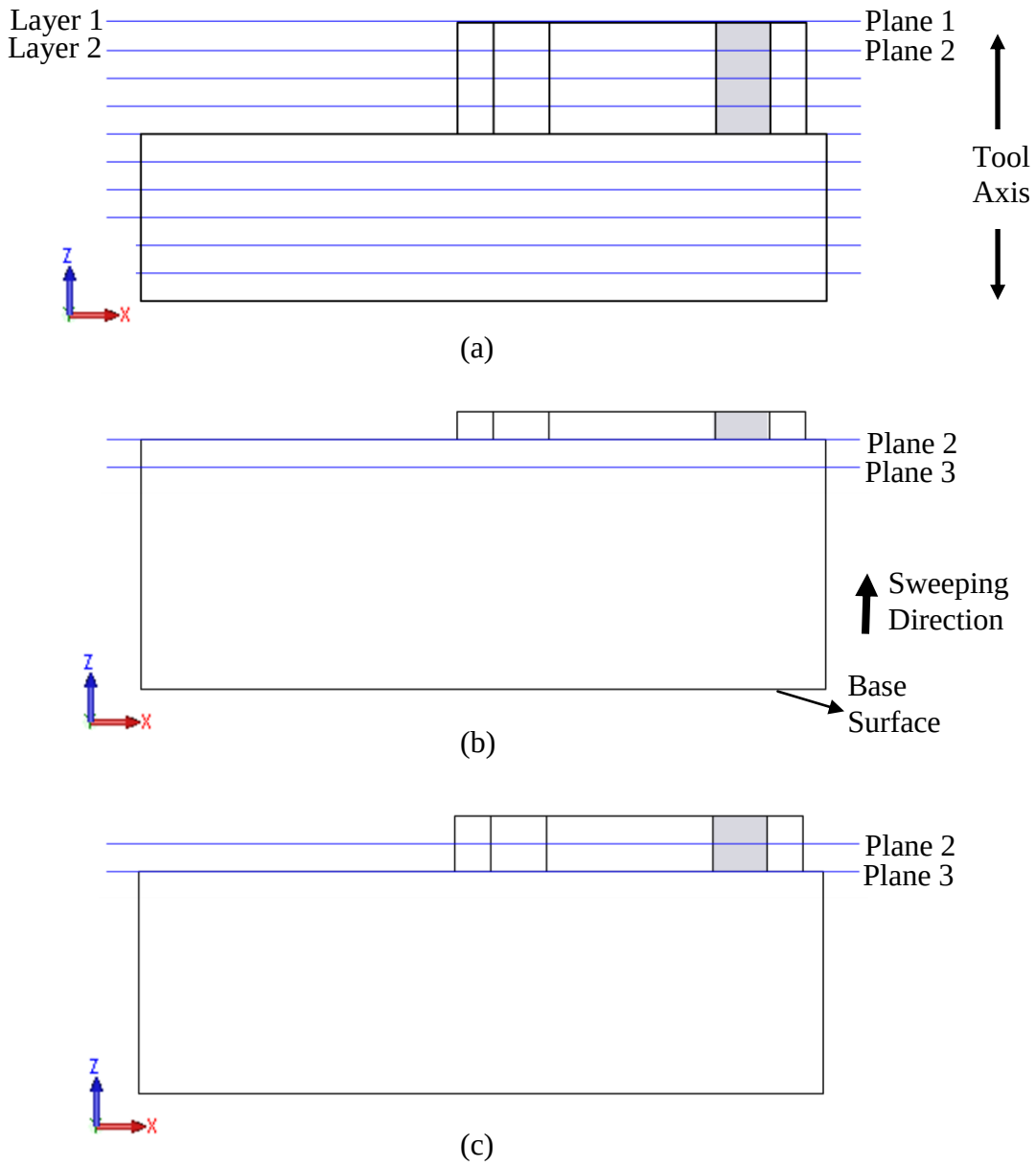


Figure 6-9: Sweeping of bottom surface of the part bounding box to identify uncut regions (a) side-view of the part shown in Figure 6-1, (b) part formed by sweeping the base surface up to plane 2, and (c) part formed by sweeping the base surface of the part up to plane 3.

6.4.2 Determination of Tool Positions for Roughing Tool-Path

To identify uncut regions, the tool position for each point on tool-path is first determined using the drop-ball methodology developed by Patel et al. [72]. In the drop-ball methodology, the tool-profile is dropped along the tool axis from a distance *height* above

the part until it first touches the part facets, shown in Figure 6-10. From the facet and tool contact point, the tool position is determined using the tool geometry.

In this work, the B-rep model *Finished Part* for each layer is converted into the STL format and the tool is dropped along the tool axis for each point of tool-path profile and the tool position is determined using the drop-ball method. The tool positions are determined for all points on the roughing tool-path footprints.

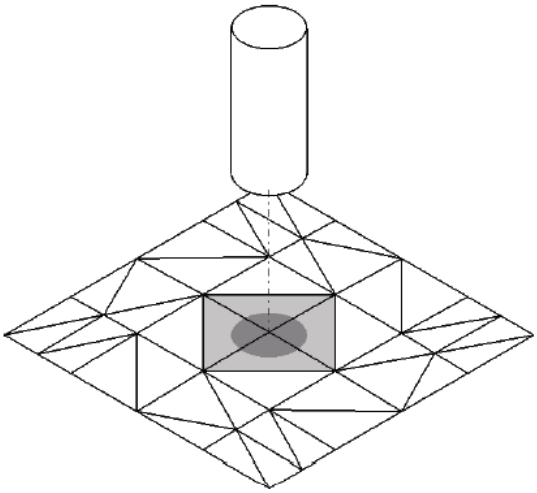


Figure 6-10: Drop-ball method for determining tool-positions [72].

6.4.3 Determination of Cut-Surface and Identification of Uncut Regions

In the cutting operations, the roughing tool movements along the determined tool positions cut the *Stock* and the cut-surface gets generated. The points on the cut surface are compared to the *Finished Part* to determine the uncut regions. In this work, the points on the *Finished Part* are determined from its z-buffer data and the z-buffer data is determined by rendering the part after orienting it so that machining axis is perpendicular to the view-port. Similarly, the z-buffer data of *Stock* is determined using the same procedure. The z-buffer data of the *Stock* is used to determine the points on the cut-surface. The viewport size is kept the same while extracting the z-buffer data of the *Stock* and *Finished Part.*

To determine the points on cut surface, the *Stock/Raw Part* is rendered to compute the depth buffer information. The depth buffer information at each pixel location can be treated as a virtual line, parallel to the tool axis, from negative infinity to depth buffer

reading for the pixel. The pixel locations are mapped to the given part's coordinate space by using the relative position of the pixels in the viewport.

The next step is to determine the intersection point between these virtual/vertical lines and the swept surface formed by the tool while moving from one tool position to the next tool position. For a ball end nose tool, the swept surface between two tool positions consists of one cylinder and two spheres at the end, as shown in Figure 6-11.

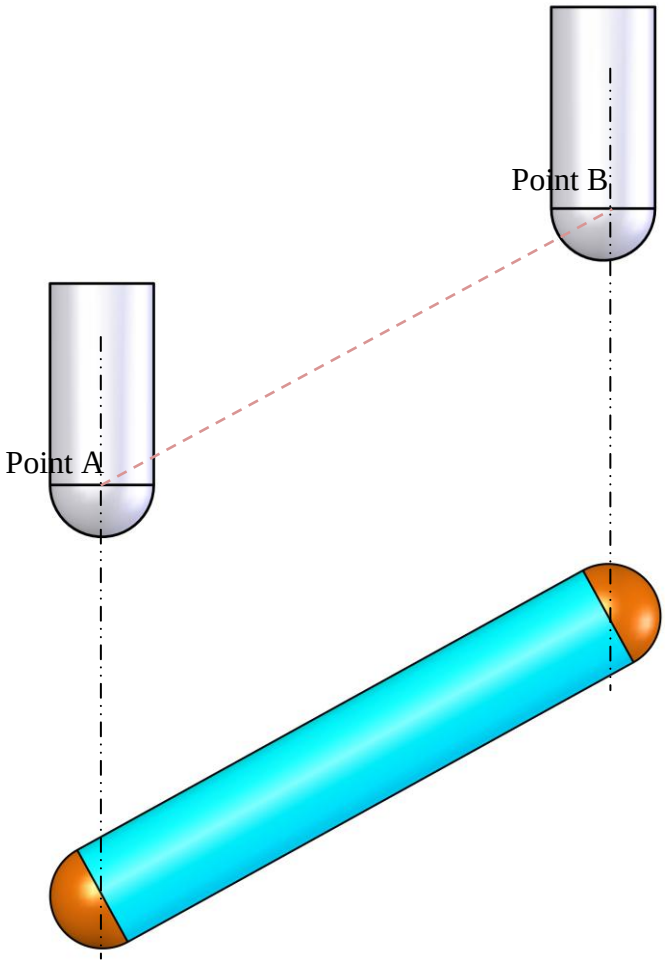


Figure 6-11: Surface swept by tool while moving from point A to B.

The vertical lines are intersected with swept surfaces formed between the adjacent tool positions, using the procedure given in Appendix D, and the intersection point having maximum depth along the tool axis is recorded for each vertical line. Next, the *Finished Part* is rendered and the depth buffer value at each pixel location is determined. The distance between the *Finish Part*'s depth data and the intersection point for each

pixel is used to determine whether a pixel location requires a clean-up cut or not. If the distance is more than an allowed limit, the pixel location is considered to require clean-up.

All pixel locations requiring a clean-up pass are grouped into different regions based on their adjacency. The region boundaries are extracted and a contour parallel offset tool-path is generated for each region, using the procedure discussed for the roughing tool-path.

6.5 Discussion

The presented work generates the tool-path footprints for rough machining and identifies the areas left uncut in the rough machining. As the uncut areas are determined by mapping the pixel positions to the part's coordinate space, the accuracy of the algorithm to determine the uncut area depends upon the number of pixels that are used to render the part. If too few pixels are used to render the part, some uncut positions may be left unidentified. On the other hand, too large of a view-port would result in an increase in the computation time. Further work to determine the optimal view-port size is required.

During the rough machining, the tool-path footprints must cover the area outside the outermost loop of the sliced part. For this work, a rectangular stock is assumed and the area outside the outermost loop is covered by enlarging the rectangle size. However, other shapes can be readily accommodated by outward offsetting of the outer most loops.

Cutting forces are not considered while calculating the tool-path. Therefore, the machining parameters (including spindle speed and feed rate) are not determined in this work.

6.6 Implementation

The methodology to generate contour parallel offset tool-path for rough machining, to identify the regions left uncut by rough machining, and to generate the clean-up tool-path were tested on the part shown in Figure 6-12. The algorithm was developed using Solidworks VBA, C++, and OpenGL. Solidworks VBA was used to implement the algorithm for extracting the point sequence machining and obstacle loops from a B-rep model, and for generating raw and finished models for each layer. The algorithms for generating contour parallel offset tool-path for rough and clean-up machining and for identifying uncut regions were implemented using C++.

The part is shown from different viewing angles in Figure 6-12(a), (b), and (c). In Figure 6-12(d), the outside faces of the part are made transparent to show the cavity profile. The part has one main cavity, which consists of a free-form surface. There is one obstacle in the center of the cavity. The obstacle has two small cavities within its boundary.

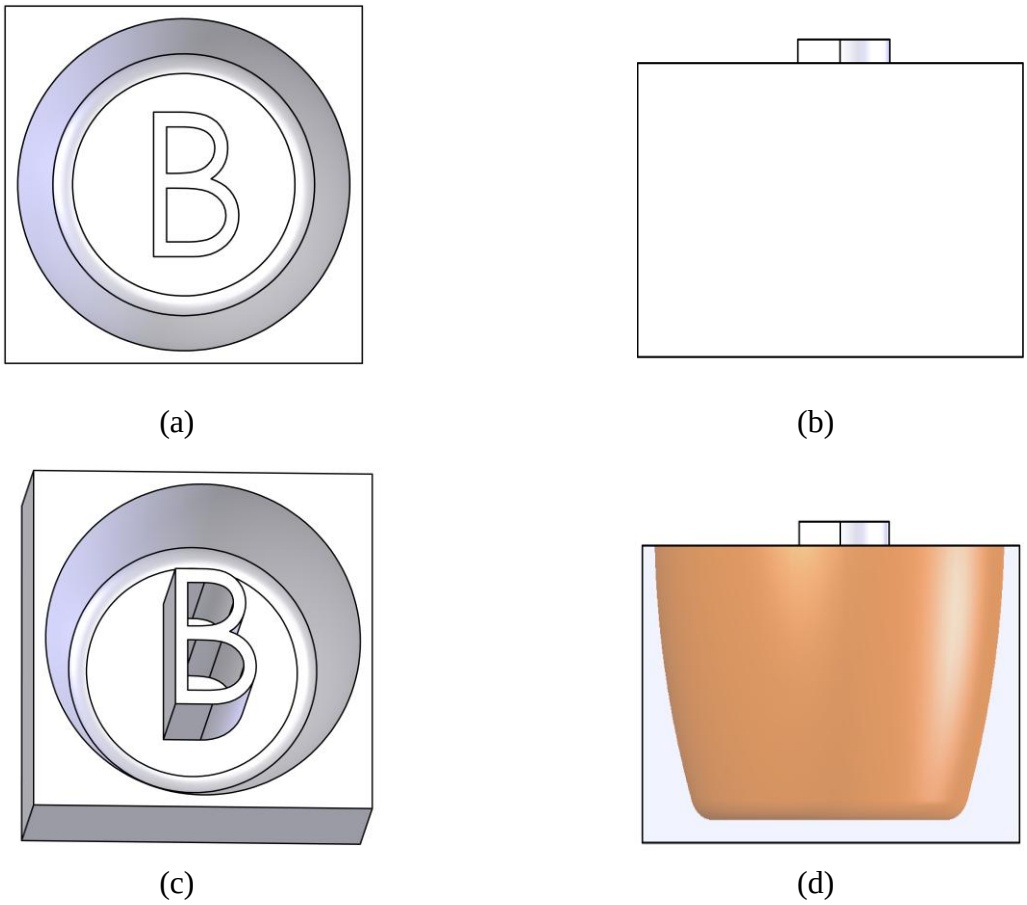
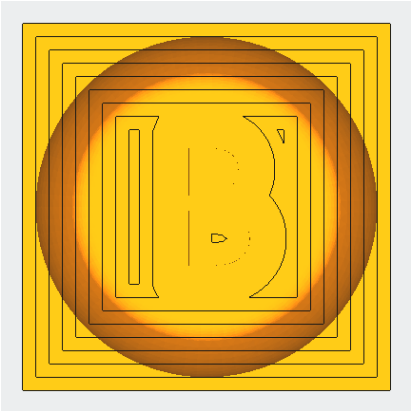


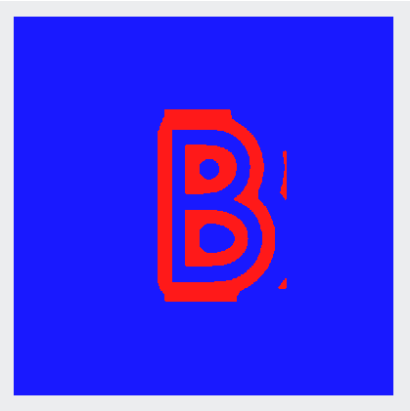
Figure 6-12: Test part used for generating roughing tool-path, identifying uncut regions, and generating clean-up tool-path. The part is shown in different views in (a), (b), and (c). The outside surfaces of the part are made transparent in (d) to show the curve profile.

The part dimensions are 85mm × 85mm × 54mm. To identify the uncut areas, the part is rendered as an image with 600 × 600 pixels. The tool diameters for roughing and clean-up tool-path are 3/8" are 1/8" respectively. The part is sliced into 13 layers of 4mm depth along the machining axis. As the outermost cavity is tapered, the roughing and clean-up tool-path foot-prints are different for each layer. The foot-print of the

roughing tool-path for layer 1 is shown in Figure 6-13(a). After the rough machining, the areas requiring clean-up machining are shown in Figure 6-13(b). A point is considered to require clean-up machining if the height difference at the point between the swept profile and the *Finished Part* is more than 0.25mm. The boundaries of the uncut areas are determined in the form of point sequence curves. The boundary points are further offset to determine clean-up tool-path footprints. The clean-up tool-path footprints for the layer are shown in Figure 6-13 (c). The roughing and clean-up tool-paths are generated for rest of the layers are shown in Figure 6-14 to Figure 6-25.



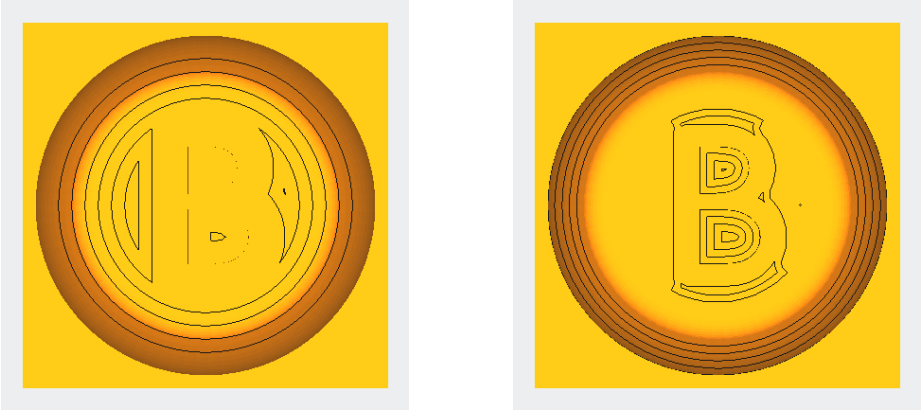
(a)



(b)

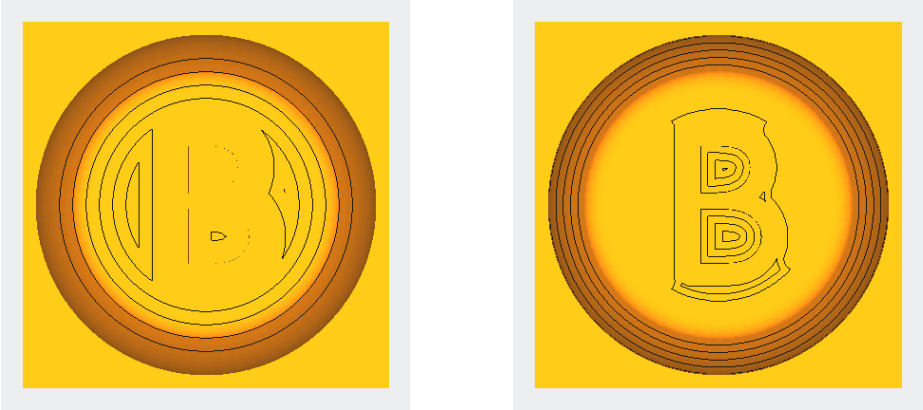
(c)

Figure 6-13: (a) Roughing tool-path for machining layer 1, (b) areas requiring clean-up machining are shown in red, and (c) and clean-up tool-path.



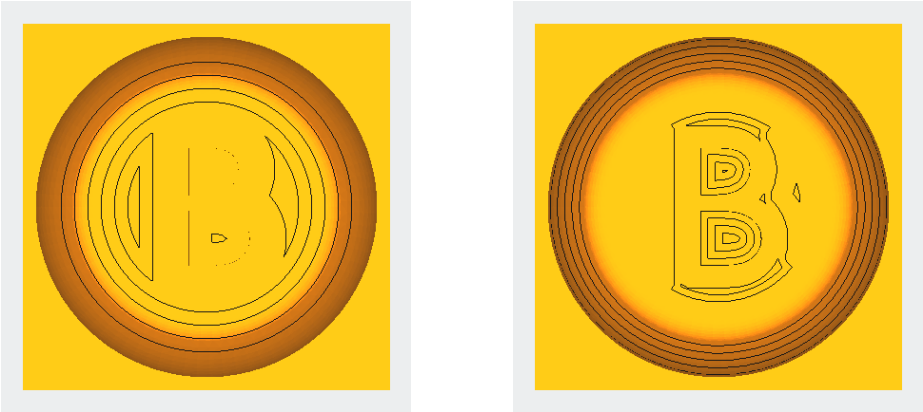
(a) (b)

Figure 6-14: Roughing and clean-up tool-paths for layer 2 are shown in (a) and (b) respectively.



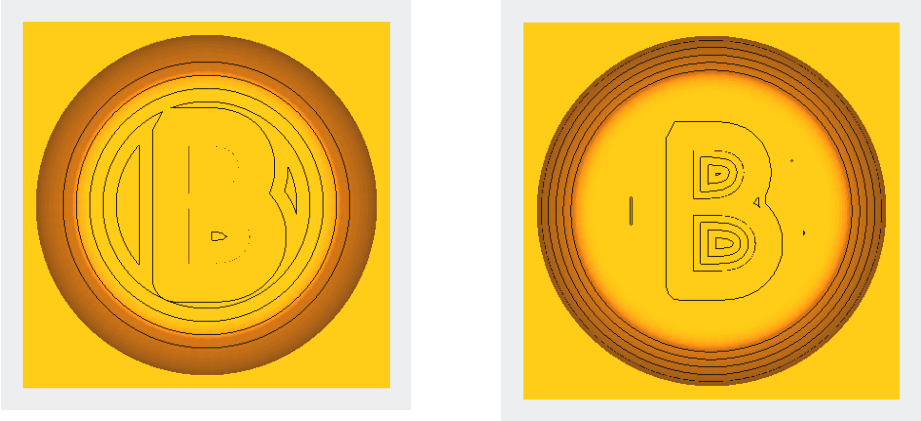
(a) (b)

Figure 6-15: Roughing and clean-up tool-paths for layer 3 are shown in (a) and (b) respectively.



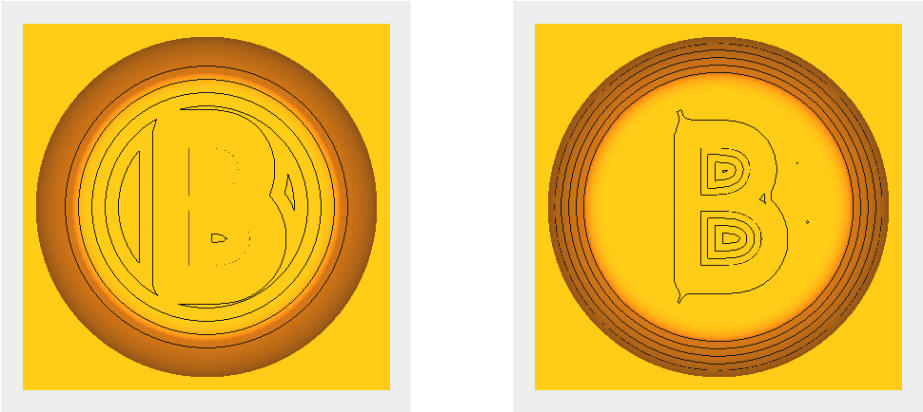
(a) (b)

Figure 6-16: Roughing and clean-up tool-paths for layer 4 are shown in (a) and (b) respectively.



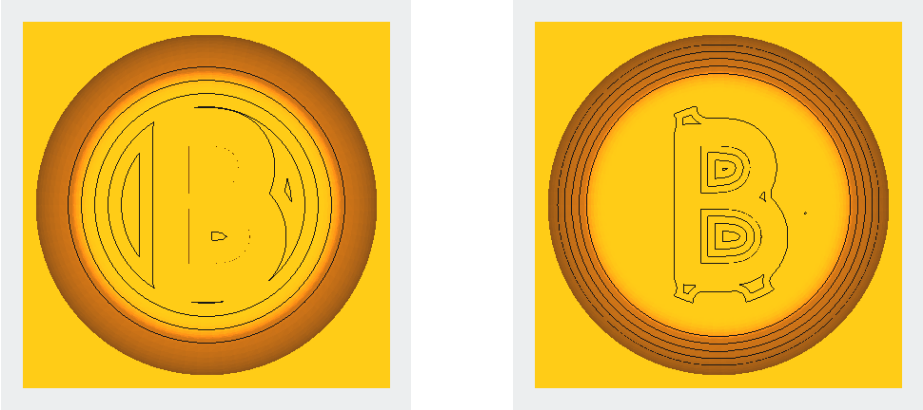
(a) (b)

Figure 6-17: Roughing and clean-up tool-paths for layer 5 are shown in (a) and (b) respectively.



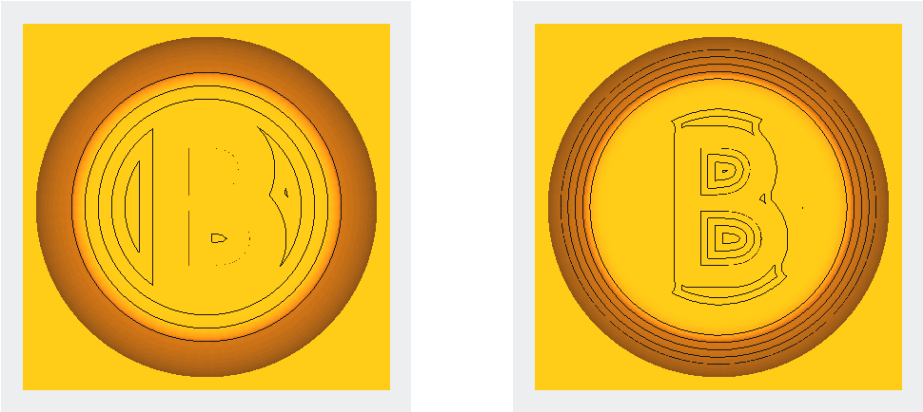
(a) (b)

Figure 6-18: Roughing and clean-up tool-paths for layer 6 are shown in (a) and (b) respectively.



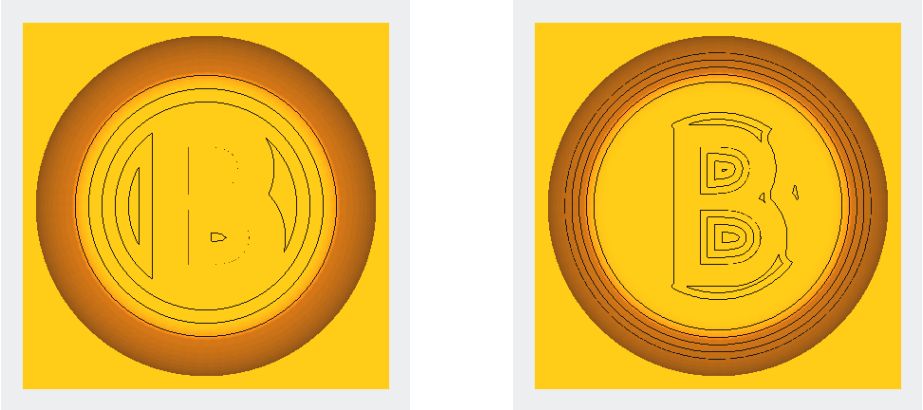
(a) (b)

Figure 6-19: Roughing and clean-up tool-paths for layer 7 are shown in (a) and (b) respectively.



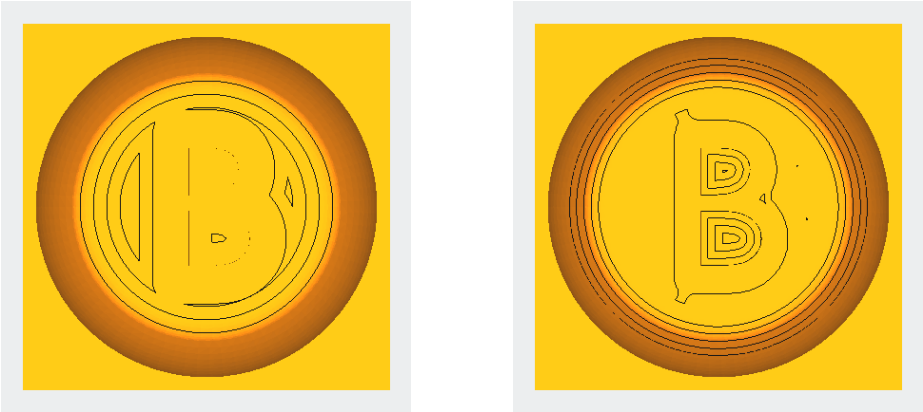
(a) (b)

Figure 6-20: Roughing and clean-up tool-paths for layer 8 are shown in (a) and (b) respectively.



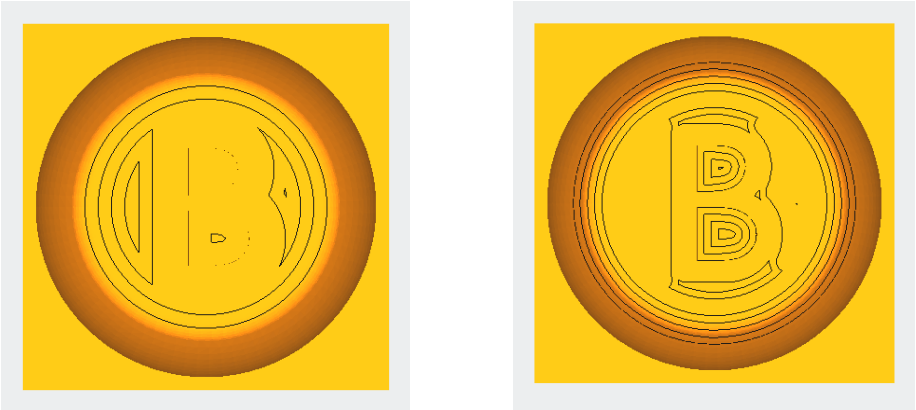
(a) (b)

Figure 6-21: Roughing and clean-up tool-paths for layer 9 are shown in (a) and (b) respectively.



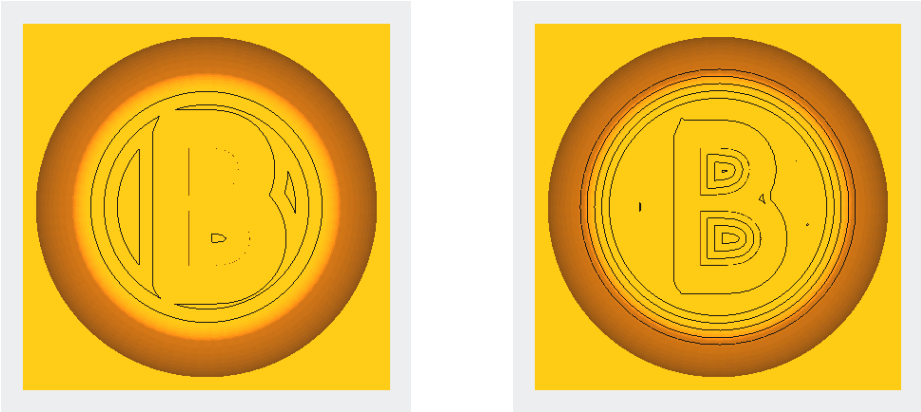
(a) (b)

Figure 6-22: Roughing and clean-up tool-paths for layer 10 are shown in (a) and (b) respectively.



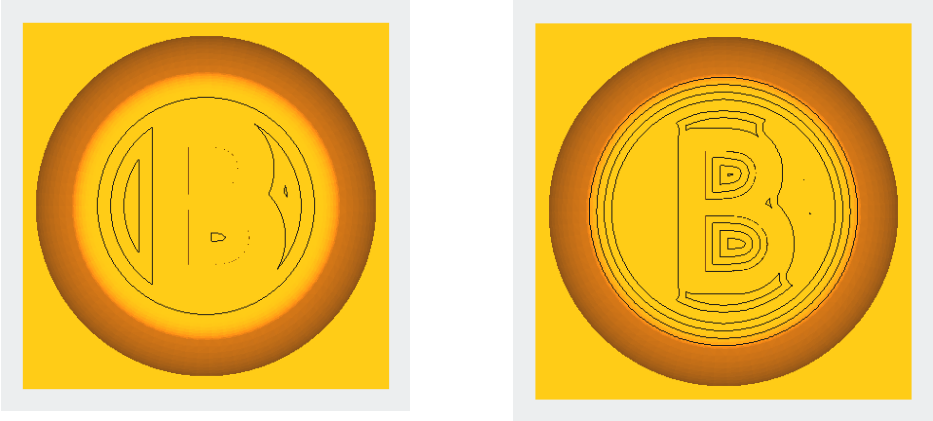
(a) (b)

Figure 6-23: Roughing and clean-up tool-paths for layer 11 are shown in (a) and (b) respectively.



(a) (b)

Figure 6-24: Roughing and clean-up tool-paths for layer 12 are shown in (a) and (b) respectively.



(a) (b)

Figure 6-25: Roughing and clean-up tool-paths for layer 13 are shown in (a) and (b) respectively.

Chapter 7

Conclusions and Future Direction

This chapter concludes the work accomplished in this thesis and provides a direction in which the work can be extended in the future. The work is focused mainly on automated design and tool-path generation of mold segments.

7.1 Summary and Conclusions

The goal of the work was to determine methodologies for automating the mold design.

1. To achieve this goal, the first step was to design an algorithm that can determine the accessibility of part surfaces without discretizing the part. Two different approaches, namely, the Boolean-based approach and the pixel-based approach, were developed for accessibility analysis. Both of these approaches can analyze the accessibility of planar, ruled and free-form surfaces while preserving the topological information of B-rep models. Therefore, it is possible in the current work to combine the geometrical and topological information with the results of accessibility analysis during feature recognition. The methodologies have been tested on a number of parts.
2. The second step in the process is to recognize geometric features conducive to mold design. The feature recognition module uses the geometry, topology and results of accessibility analysis. Intersecting depression and simple protrusion features are recognized, release direction of the undercut features is determined, and the edge boundaries between the intersecting features are also identified. The edge boundaries can be used to automate the designing of mold segments for intersecting features. The methodology has been tested on a benchmark part taken from the literature, and recognized all of the features reported in that research [33]. The methodology has also been tested on a number of industrial parts.
3. The third step, after recognizing the surfaces forming the mold features, is to identify the core and cavity surfaces. Parting line and parting surfaces are

generated to automatically design the core, cavity and side-cores. This methodology has also been tested on industrial parts.

The methodology outlined above was implemented and tested on a number of parts. The test parts have planar, ruled and free-form surfaces and have a number of intersecting features. The edge boundaries of the sub-features are recognized and the release direction of each sub-feature is determined. After identifying the core and cavity surfaces, the mold segments of the tested parts are generated.

Prior to this work, many attempts to analyze the accessibility of part surfaces [7][13][14] have been reported. The reported methodologies did not preserve the topological information during the accessibility analysis. Therefore, the accessibility analysis information could not be combined with topological information during mold feature recognition process. There were attempts [28][33] to recognize the intersecting mold features using part geometry and topology. However, these methodologies cannot be used to recognize *Type-I intersecting Features* and to determine the edge boundaries between various intersecting features. Thus, there was a void in the literature of a method that could recognize complex intersecting features during an automated mold design process. In this work, the accessibility information is combined with topologic and geometric information of the part during the feature recognition module. Using this information, a larger variety of new mold intersecting features are classified and recognized in this approach. This enhances the domain of parts for which the molds can be designed automatically.

The second major goal of the work was to demonstrate that the downstream operations can also benefit from algorithmic decision making. In this work, the design process that was selected to demonstrate this was the machining process. All mold segments must be machined and any efficiency gain in this area would be beneficial. To meet this goal, this work focused on automatically generating roughing and clean-up tool-paths while reducing the machining time by limiting the tool movements within those areas that have uncut material. To do this, the mold segment is sliced into equal layers and the geometry is used to produce a gouge-free tool-path for each layer. During the rough machining of the layer, the material is removed aggressively with large tools. This leaves behind uncut material. The algorithm processes the machined area and

identifies the uncut regions. In the next step only these uncut regions are machined with smaller tooling, thereby adding to the tool-path efficiency and reducing the danger of breaking a tool. Once the layer has been machined to satisfaction, the next layer is machined.

The algorithm is capable of recognizing any shape of uncut region, even in the presence of obstacles. The tool-path can handle cavities within obstacles and any number of obstacles. The methodology has been tested on a number of parts, and results of a part after slicing it with a number of layers are discussed in Chapter 6. The part work has one obstacle within a large cavity.

There have been works [47][48] reported in the literature that can identify the regions left uncut after machining. However, these methodologies cannot identify the regions left uncut due to the presence of obstacles within the machining regions. In this work, tool-movement is simulated to determine the points on the tool's swept surface. These surface points are compared with those of the required machined surfaces to identify the left-over uncut regions. Therefore, the regions left uncut due to obstacles within machining regions, concave corners within the machining region, and large tool-size can be identified with this approach. The machining efficiency can be improved by limiting the cutter movement within the uncut regions during the clean-up machining. It is demonstrated that algorithmic analysis of process information and geometry/topology can be successfully used to automate downstream processes and to improve manufacturing process planning efficiency.

The objectives of this work (the automation of the mold design process, and the improvement in the efficiently of mold segments machining by reducing the unnecessary tool movements) were met successfully.

7.2 Future Direction

As the B-rep model stores the topological information of part models, the accessibility analysis of a B-rep model helps in connecting the accessibility information with the topological information. Two approaches have been developed in this work: the Boolean-based approach and the pixel-based approach. Both of these approaches suffer from some drawbacks. The Boolean-based approach involves computationally expensive Boolean operations that can fail if the resulting body has invalid topology. In the pixel-based

approach, if the surface is too small to occupy any pixel, then the surface accessibility cannot be successfully analyzed using this approach. An efficient and robust approach is required that can analyze the accessibility of part surfaces while preserving their topological information.

The feature recognition approach recognizes simple and intersecting depression features. However, the new system cannot recognize protrusion features with concave edges due to their intersection with other features. Moreover, it is necessary to split the "*Unbounded-Type-II*" intersecting feature to generate the mold elements, since it does not have any defined boundary. Further work is required to identify intersecting protrusion features and to automatically split the "*Unbounded-Type-II*" intersecting depression features.

To generate the mold components, feasible mold parting lines are identified. However, the identification of optimal parting lines requires further work. The side-cores of depression features are generated with an assumption that undercut will have ruled property in the release direction. This assumption is not valid in the case of intersecting depression features.

To generate the tool-path for clean-up machining, the areas left uncut during rough machining are identified. The part is rendered and depth buffer information is used to determine the pixels requiring clean-up machining. The pixels' positions are mapped to the part's coordinate space. If too few pixels are used to render the part, some of the points requiring clean-up machining may be left unidentified. On the other hand, if too large a view-port size is used to render the part, the computation time will increase. Further work is required to optimize the viewport size.

References

[1] Pye RGW (1989) Injection Mould Design: a textbook for the novice and a design manual for the thermoplastics industry. Longman Scientific & Technical in association with the Plastics and Rubber Institute, ISBN 978-0-582-01611-8.

[2] Kazmer DO (2007) Injection Mold Design Engineering. Hanser Gardner Publications, ISBN 978-3-446-41266-8.

[3] Gastrow H (1993) Injection molds, 108 Proven Designs. Hanser Gardner Publications, ISBN 978-1-569-90028-4.

[4] Ravi B, Srinivasana MN (1990) Decision criteria for computer-aided parting surface design. Computer-Aided Design 22(1): 11-18.

[5] Fuh JYH, Zhang YF, Nee AYC, Fu MW (2004) Computer-Aided injection mold design and manufacture. Marcel Dekker Publications, New York, ISBN 0-8247-5314-3.

[6] Yin ZP, Ding H, Xiong YL (2001) Virtual prototyping of mold design: geometric mouldbaility analysis for near-net-shape manufactured parts by feature recognition and geometric reasoning. Computer-Aided Design 33: 137-154.

[7] Priyadarshi A, Gupta SK (2004) Generating algorithms for automated design of multi-piece permanent molds. Computer-Aided Design 36: 241-260.

[8] Christman A (2001) Mold design – a critical and rapidly changing technology. MMS Online, Gardner Publication, Inc.

[9] Hui KC, Tan ST (1992) Mould design with sweep operations – a heuristic search approach. Computer-Aided Design 24(2): 81-91.

[10] Woo TC (1994) Visibility maps and spherical algorithms. Computer-Aided Design 26(1): 6-16.

[11] Chen LL, Chou SY, Woo TC (1993) Parting directions for mould and die design. Computer-Aided Design 25(12): 762-768.

[12] Chen LL, Chou SY (1995) Partial visibility for selecting a parting direction in mold and die design. Journal of Manufacturing Systems 14(5): 319-330.

[13] Dhaliwal S, Gupta SK, Huang J, Priyadarshi A (2003) Algorithm for computing global Accessibility cones. Journal of Computing and Information Science in Engineering 3: 200-209.

[14] Khardekar R, Burton G, McMains S (2006) Finding feasible mold parting directions using graphics hardware. Computer-Aided Design 38: 327-341.

[15] Priyadarshi A, Gupta SK (2006) Finding mold-piece regions using computer graphics hardware. In: Geometric modeling and processing. Lecture notes in computer science, vol 4007: 655-662.

[16] Weinstein M, Manoochehri S (1997) Optimum parting line design of molded and cast parts for manufacturability. Journal of manufacturing systems 16(1): 1-12.

[17] Fu MW, Nee AYC, Fuh JYC (2002) The application of surface visibility and moldability to parting line generation. Computer-Aided Design 34: 469-480.

[18] Madan J, Rao PVM, Kundra TK (2007) Die-casting feature recognition for automated parting direction and parting line determination. Journal of Computing and Information Science in Engineering 7: 236-248.

[19] Chakraborty P, Reddy NV (2009) Automatic determination of parting directions, parting lines and surfaces for two-piece permanent molds. Journal of Material Processing Technology 209: 2464-2476.

[20] Wong T, Tan ST, Sze WS (1998) Parting line formation by slicing a 3D CAD model. Engineering with Computers 14: 330-343.

[21] Paramio MA Rubio, Garcia JM Perez, Chueco J Rios, Idoipe A Vizan, Sevillano JJ Marquez (2006) A procedure for plastic parts demoldability analysis. Robotics and Computer-Integrated manufacturing 22: 81-92.

[22] Nee AYC, Fu MW, Fuh JYH, Lee KS, Zhang YF (1997) Determination of optimal parting directions in plastic injection mold design. Annals of CIRP 46: 429-432.

[23] Fu MW, Fuh JYH, Nee AYC (1999) Undercut feature recognition in an injection mould design system. Computer-Aided Design 31: 777-790.

[24] Bidkar RA, McAdams DA (2004) Feature recognition for injection-molded and die-cast parts. DETC2004-57754. Proc. of 2004 ASME DET and CIE conf., ASME, Salt Lake City.

[25] Bidkar RA, McAdams DA (2010) Methods of automated manufacturability analysis of injection-molded and die-cast parts. Res Eng Design 21: 1-24.

[26] Joshi S, Chang TC (1988) Graph-based heuristics for recognition of machined features from a 3D solid model. Computer-Aided Design 20(2): 58-66.

[27] Ganter MA, Skoglund PA (1993) Feature extraction for casting core development. Journal of Mechanical Design 115(4): 744-50.

[28] Ye XG, Fuh JYH, Lee KS (2001) A hybrid method for recognition of undercut features from moulded parts. Computer-Aided Design 33: 1023-1034.

[29] Kumar N, Ranjan R, Tiwari MK (2007) Recognition of undercut features and parting surface of molded parts using polyhedron face adjacency graph. Int J Adv Manuf Technol 34: 47-55.

[30] Zhang C, Zhou X, Li C (2010) Feature extraction from freeform molded parts for moldability analysis. International Journal of Advanced Manufacturing Technology 48:273–282.

[31] Shin KH, Lee KW (1993) Design of side cores of injections moulds from automatic detection of interference faces. Journal of Design and Manufacturing 3: 225-236.

[32] Fu MW (2008) The application of surface demoldability and moldability to side-core design in die and mold CAD. Computer Aided Design 40: 567-575.

[33] Ye XG, Fuh JYH, Lee KS (2004) Automatic undercut feature recognition for side core design of injection molds. Journal of Mechanical Design 126:519-526.

[34] Banerjee AG, Gupta SK (2007) Geometric algorithms for automated design of side actions in injection moulding of complex parts. Computer-Aided Design 39: 882-897.

[35] Ran JQ, Fu MW (2010) Design of internal pins in injection mold CAD via the automatic recognition of undercut features. Computer-Aided Design 42: 582-597.

[36] Dragomatz D, Mann S (1997) A classified bibliography of literature on NC milling path generation. Computer Aided Design 29(3): 239-247.

[37] Hatna A, Grieve RJ, Broomhead P (1998) Automatic CNC milling of pockets: geometric and technological issues. Computer Integrated Manufacturing Systems 11(4): 309-330.

[38] Lasemi A, Xue D, Gu P (2010) Recent development in CNC machining of freeform surfaces: A state-of-the-art review 42: 641-654.

[39] Marshall S, Griffiths JG (1994) A survey of cutter path construction techniques for milling machines. International Journal of Production Research 32(12): 2861-2877.

[40] Jeong J, Kim K (1998) Tool path generation for machining free-form pockets using Voronoi diagrams. International Journal of Advanced Manufacturing Technology 14: 876-881.

[41] Griffiths JG (1994) Toolpath based on hilbert's curve. Computer Aided Design 26(11): 839-844.

[42] Pang J, Narayanaswami R (2004) Multiresolution offsetting and loose convex hull clipping for 2.5D NC machining. Computer Aided Design 36: 625-637.

[43] Persson H (1978) NC machining of arbitrary shaped pockets. Computer Aided Design 10(3):169-174.

[44] Held M, Lukacs G, Andor L (1994) Pocket machining based on contour-parallel tool path generated by means of proximity maps. Computer Aided Design 26(3): 189-203.

[45] Hansen A, Arbab F (1992) An algorithm for generating NC tool paths for arbitrary shaped pockets with islands. ACM transactions on graphics 11(2): 152-182.

[46] Choi BK, Park SC (1999) A pair-wise offset algorithm for 2D point-sequence curve. Computer Aided Design 31: 735-745.

[47] Park SC, Choi BK (2000) Uncut free pocketing tool-paths generation using pair-wise offset algorithm. Computer Aided Design 33: 739-746.

[48] Choi BK, Kim BH (1997) Die-cavity pocketing via cutting simulation. Computer Aided Design 29(12): 837-846.

[49] Chuang CM, Yau HT (2004) A new approach to z-level contour machining of triangulated surface models using fillet endmills. Computer Aided Design 37: 1039-1051.

[50] Park SC, Choi BK (2000) Boundary extraction algorithm for cutting area detection. Computer Aided Design 33: 571-579.

[51] Cox JJ, Takezaki Y, Ferguson HRP, Kohkonen KE, Mulkay EL (1994) Space-filling curves in tool-path applications. Computer Aided Design 26(3): 215-224.

[52] Choi BK, Lee CS, Hwang JS, Jun CS (1988) Compound surface modelling and machining. Computer Aided Design 20(3): 127-136.

[53] Duncan JP, Mair SG (1983) Sculptured surfaces in engineering and medicine. Cambridge University Press, Cambridge, UK, ISBN 978-0-521-23450-4.

[54] Choi BK, Jun CS (1989) Ball-end cutter interference avoidance in NC machining of sculptured surfaces. Computer-Aided Design 21(6): 371-378.

[55] Suh YS, Lee K (1990) NC milling tool path generation for arbitrary pockets defined by sculptured surfaces. Computer Aided Design 22(5): 273-284.

[56] Cho JH, Kim JW, Kim K (2000) CNC tool path planning for multi-path sculptured surfaces. Int. J. Prod. Res. 38(7): 1677-1687.

[57] Takeuchi Y, Sakamoto M, Abe Y, Orita R (1989) Development of personal CAD/CAM system for mold manufacture based on solid modeling techniques. Annals CIRP 38(1): 429-432.

[58] Tang K, Cheng CC, Dayan Y (1995) Offset surface boundaries and 3-axis gouge-free surface machining. Computer Aided Design 27(12): 915-927.

[59] Choi BK, Kim DH, Jerard RB (1997) C-space approach to tool-path generation for die and mould machining. Computer Aided Design 29(9): 657-669.

[60] Park SC (2004) Sculptured surface machining using triangular mesh slicing. Computer Aided Design 36: 279-288.

[61] Kim BH, Choi BK (2000) Guide surface based tool path generation in 3-axis milling: an extension of guide plane method. Computer Aided Design 32: 191-199.

[62] Hwang JS (1992) Interference-free tool-path generation in the NC machining of parametric compound surfaces. Computer Aided Design 24(12): 667-676.

[63] Hwang JS, Chang TC (1998) Three-axis machining of compound surfaces using flat and filleted endmills. Computer Aided Design 30(8): 641-647.

[64] Chuang CM, Chen CY, Yau HT (2002) A reverse engineering approach to generating interference-free tool paths in three-axis machining from scanned data of physical models. International Journal of Advanced Manufacturing Technology 19: 23-31.

[65] Patel K (2010) Web based automatic tool path planning strategy for complex sculptured surfaces. Master's Thesis, University of Waterloo, Canada.

[66] Lin AC, Liu HT (1998) Automatic generation of NC cutter path from massive data points. Computer Aided Design 30(1): 77-90.

[67] Park SC, Chung YC (2002) Tool-path generation from measured data. Computer Aided Design 35: 467-475.

[68] Yau HT, Hsu CY (2009) Generating NC tool paths from random scanned data using point-based models. International Journal of Advanced Manufacturing Technology 41: 897-907.

[69] Yingjie Z, Liling G (2011) Adaptive tool-path generation on point-sampled surfaces. Precision Engineering 35 (2011) 591– 601.

[70] Chen Y, Rosen DW (2003) A reverse glue approach to automated construction of multi-piece molds. Journal of Computing and Information Science in Engineering 3:219-230.

[71] Park SC, Choi BK (2003) Free-form die-cavity pocketing. International Journal of Advanced Manufacturing Technology 22: 864-872.

[72] Patel K, Bolanos GS, Bassi R, Bedi S (2011) Optimal tool shape selection based on surface geometry for three-axis CNC machining. International Journal of Advanced Manufacturing Technology 57: 655-670.

[73] Mantyla M (1988) An introduction to solid modeling, Computer Science Press, New York, ISBN 0-88175-108-1.

[74] Fu MW, Fuh JYH, Nee AYC (1999) Generation of optimal parting direction based on undercut features in injection molded parts. 31: 947-955.

[75] Jacobs PF (1992) Rapid Prototyping and Manufacturing. Society of Manufacturing Engineers, ISBN: 0-87263-425-6.

Appendix A

Characteristics of B-rep and STL models

Solid modeling is an unambiguous representation of the solid parts of an object, and these models of solid objects are suitable for computer processing. Solid modeling was conceptualized with developments in the computer science, and has gone through different stages of development, namely, half-plane modeling, constructive solid geometry (CSG) modeling, polygonal modeling (includes STL modeling), and boundary representation (B-rep) modeling. B-rep and STL models are used in this research and their characteristics are discussed as follows.

A.1 B-rep Model Characteristics

To recognize the mold features, it is necessary to convert the low-level information (i.e., vertices, edges, and faces) of CAD models into useful high-level semantic mold features. In B-rep models, the information is stored under the terms *geometry* and *topology*. Geometry is used to bundle the geometric information of different entities (vertices, edges, and faces), whereas topology stores the information about the interconnections between different entities [73]. The various characteristics of B-rep model are listed here:

- A B-rep model is a function of vertices, edges, and surfaces, which can be represented as:

$$\text{B-rep model } (M) = f\,(V, E, S)$$

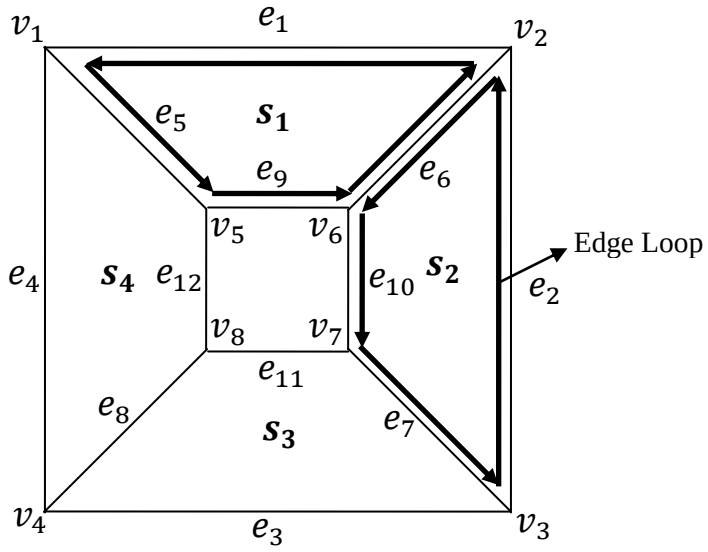


Figure A-1: Relationship between vertices (v_i), edges (e_i), and surfaces (s_i)

where *V*, *E*, and *S* represent the set of vertices, edges, and surfaces respectively [74]. The connection between various entities is shown in Figure A-1.

- Each edge is shared by two faces and has two vertices.
- Each face is bounded by edges. The boundary of connected edges is called a loop.
- A loop can be inner or outer. An outer loop forms an outermost edge boundary of a face, whereas an inner loop is an edge boundary of a face within the outer loop, shown in Figure A-2.

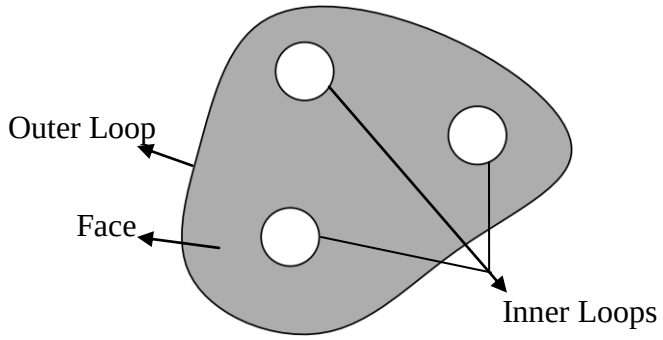


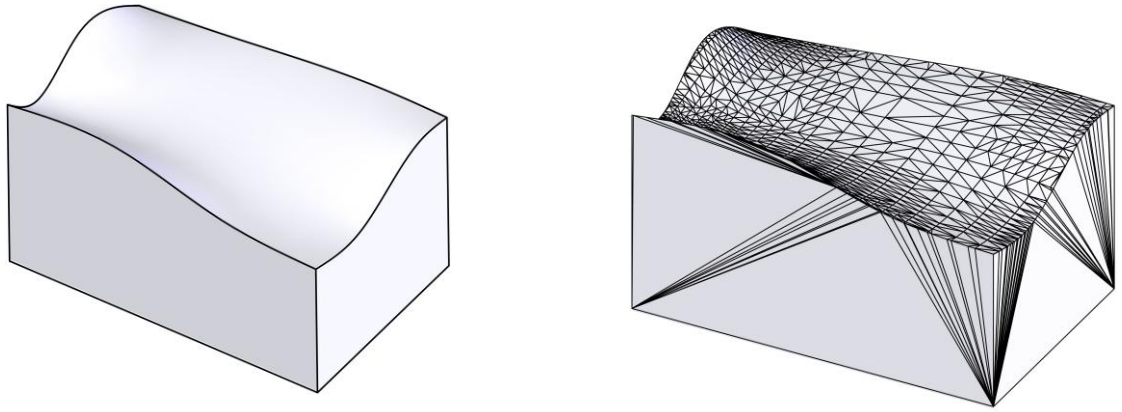
Figure A-2: Outer and Inner loops of a face.

- Each face has one closed outer loop and can have several inner loops.

- Each edge has positive and negative (opposite of positive) orientation. Positive and negative orientations of edge e_6, shared by surface s_1 and s_2, are shown by thick arrows in Figure A-1.
- Loops do not intersect with each other.

A.2 STL Model Characteristics

In this work, stereolithography (STL) files are used while extracting the manufacturing information of mold segments. In the STL file format, surfaces of three-dimensional objects are built by various two-dimensional triangles. Each triangle is defined using end points of the triangle edges and the outward normal of the triangle. The characteristics of STL file format are as follows:



(a) (b)

Figure A-3: (a) A solid part, and (b) its representation in STL format.

- Each triangle of the part (facet) is the boundary between the interior and exterior of the object.
- Vertices of a facet are listed in counter-clockwise order when viewed from outside of the object.
- Triangles in the STL file are connected with the other triangles at the vertices. This is known as the "vertex to vertex" rule [75]. In other words, a vertex of one triangle cannot lie on the edge of another triangle.

Appendix B

Determination of Part Faces Shared by its Convex Hull

B.1 Introduction

A 3D convex hull envelops the part, and each face of the convex hull subdivides space such that all the entities (faces, edges, and vertices) of the part lie wholly on or to one side of that face, as illustrated in Figure B-1. The purpose of determining the part faces that are shared by its convex hull is to identify *Fully-Accessible* missing faces of a depression undercut, as discussed in Section 4.4.1.

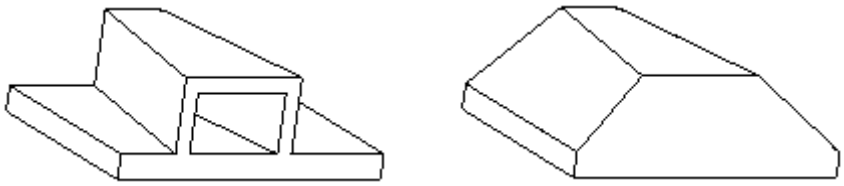


Figure B-1: (a) a non-convex part and (b) its convex hull [11].

B.2 Methodology for Determining the Faces that are Part of Convex Hull

Weinstein and Manoochehri [16] have proposed a methodology to identify the surfaces of the polyhedral part so that all the other surfaces are located on the material side of these surfaces. Their methodology has been extended to include the ruled, as well as free-form surfaces and is presented next.

Given a surface, each edge of its outer edge loop is parameterized and divided into a number of equal segments. The dot product of the surface normal at each segment's node with the vectors from the node to every vertex of other faces of the part is taken. If all the dot products are negative or zero, then the considered surface is determined to be part of the convex hull.

Appendix C

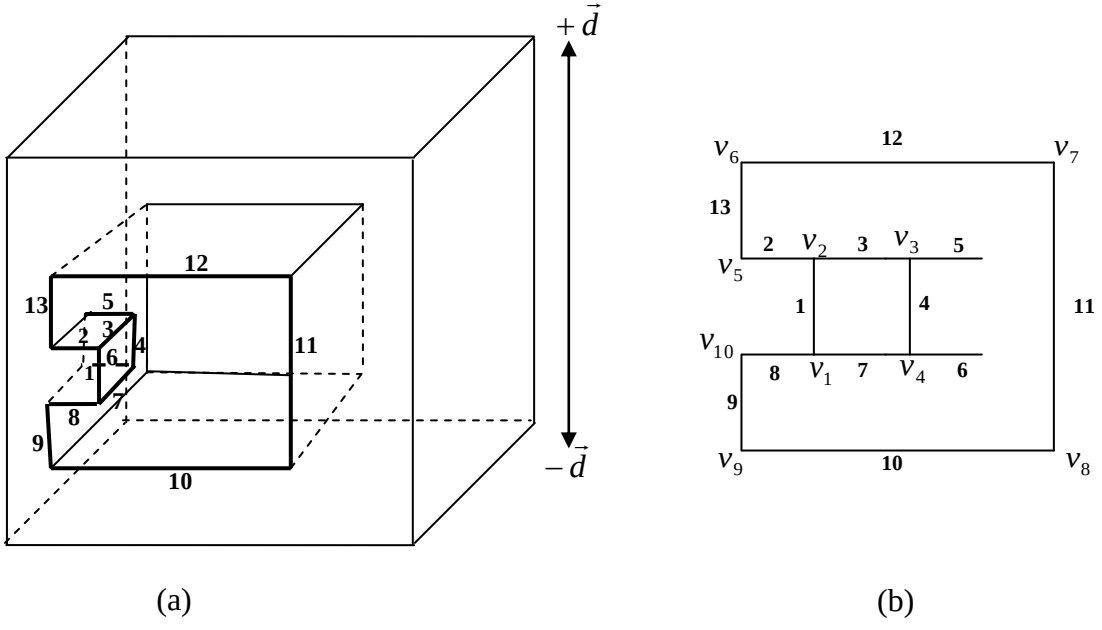
Determination of Closed Convex Edge Loops in a Set of Connected Surfaces

C1. Introduction

The identification of closed convex edge loops is required for determining the edge boundaries of intersecting features. In the process of convex edge loops determination, all convex edges connected to each convex edge are extracted first and stored in a database. Then, closed edge loops are determined using the database.

C2. Methodology for determining the closed convex edge loops

The Figure C-1 is used to illustrate the methodology to identify the convex edge loops within a set of connected surfaces. First, any of the convex edges may be used as a seed edge. In this case, edge ***1*** is used as seed edge.



(a) (b)

Figure C-1: (a) Convex edges of an undercut feature shown using thick lines, and (b) 2-D map showing their connectivity.

Now, all the convex edges within the set of connected surfaces are extracted using the procedure discussed below:

- One of the vertices (v_1) of seed edge (Edge ***1***) is identified. Edge ***1*** and vertex v_1 are added to the database under the column '*Next edge*' and '*Next Vertex*', as shown in line 1 of Table C-1.
- The convex edges connected to the seed edge through vertex v_1 are identified. The seed edge and vertex v_1 are placed under column '*Prev Edge*' and '*Prev Vertex*', respectively, for each of the connected convex edge and each identified convex is placed under '*Next Edge*'. The second vertex of each connected edge is placed under '*Next Vertex*'.
- The process is repeated and the convex edges connected to '*Next Edge*' of each row through '*Next Vertex*' are identified and stored in the database. If the '*Next Edge*' and '*Next Vertex*' are already evaluated to find the connected edges, then these are not evaluated again. In addition, the process is not repeated if the '*Next Edge*' is same as the seed edge, otherwise it would result in an infinite loop.

S. No	*Prev. Edge*	*Prev. Vertex*	*Next Edge*	*Next Vertex*	*S. No*	*Prev. Edge*	*Prev. Vertex*	*Next Edge*	*Next Vertex*
0.			1	v_1	13.	13	v_6	12	v_7
1.	1	v_1	8	v_{10}	14.	13	v_5	2	v_2
2.	1	v_1	7	v_4	15.	12	v_7	11	v_8
3.	8	v_{10}	9	v_9	16.	2	v_2	3	v_3
4.	7	v_4	4	v_3	17.	2	v_2	1	v_1
5.	9	v_9	10	v_8	18.	11	v_8	10	v_9
6.	4	v_3	3	v_2	19.	3	v_3	4	v_4
7.	10	v_8	11	v_7	20.	10	v_9	9	v_{10}
8.	3	v_2	2	v_5	21.	4	v_4	7	v_1
9.	3	v_2	1	v_1	22.	9	v_{10}	8	v_1
10.	11	v_7	12	v_6	23.	7	v_1	8	v_{10}
11.	2	v_5	13	v_6	24.	7	v_1	1	v_2
12.	12	v_6	13	v_5	25.	8	v_1	1	v_2

Table C-1: Database of convex edges.

To read the closed loops from the database, shown in Table C-1, the following procedure is adopted:

- Vertices connecting more than two convex edges are identified and are called *Junction Vertices*. A vertex is a junction vertex if it appears under column '*Prev. Edge*' and '*Prev. Vertex*' (of Table C-1) more than once.
- The connectivity between various edges is identified to form unique paths. A path can be *Open*, *Closed* or *Branched*.
    - A path is open if last edge and vertex are not same as seed edge and vertex. One case can be where the '*Next Edge*' is not connected to any other convex edge through the '*Next Vertex*' of the same row. Another case is where a vertex appears again in the path, but its corresponding next edge is not the seed edge. For example, path number 4, shown in Table C-2, is open path as the vertex v_1 appears under '*Next Vertex*', but the '*Next Edge*' is not the seed edge.
    - A path is a closed path if the last edge and vertex are the same as the seed edge and vertex. In other words, if the '*Next Edge*' and '*Next Vertex*' come out to be the seed edge and seed vertex, respectively. Path number 2 and 5 extracted using edge ***1***, as seed edge, are closed.
    - If the next vertex of a path is a *Junction* vertex, then the path travelled so far is a branched path. Each branch must be accessed separately to find convex edge loops.

In this example, two closed convex edge paths, 1-8-9-10-11-12-13-2 and 1-7-4-3, are formed by using edge ***1*** as the seed edge.

Prev. Edge	*Prev. Vertex*	*Next Edge*	*Next Vertex*	*Path No*	*Prev. Path No*	*Path Type*
		1	v_1	1	-	Branched
1	v_1	8	v_{10}	2	1	
8	v_{10}	9	v_9			
9	v_9	10	v_8			
10	v_8	11	v_7			
11	v_7	12	v_6			
12	v_6	13	v_5			
13	v_5	2	v_2			
2	v_2	1	v_1			Closed
1	v_1	7	v_4	3	1	
7	v_4	4	v_3			
4	v_3	3	v_2			Branched
1	v_1	7	v_4	4	3	
7	v_4	4	v_3			
4	v_3	3	v_2			
3	v_2	2	v_5			
2	v_5	13	v_6			
13	v_6	12	v_7			
12	v_7	11	v_8			
11	v_8	10	v_9			
10	v_9	9	v_{10}			
9	v_{10}	8	v_1			Open
1	v_1	7	v_4	5	3	
7	v_4	4	v_3			
4	v_3	3	v_2			
3	v_2	1	v_1			Closed

Table C-2: Paths formed by convex edges of set of connected surfaces.

Appendix D

Determination of an Intersection Point between a Vertical Line and Tool Swept Surface

For a ball-end tool, the volume swept by the tool while moving between two points can be represented by a cylinder and two spheres at its ends, shown in Figure 6-11. The pixels under the shadow of a swept surface formed by the tool-profile are identified and the virtual lines passing through these pixels are intersected with the profile.

To confirm that a pixel falls under the shadow of sphere, the horizontal distance between the pixel and the sphere center is determined. If the distance is less than the sphere radius, then the pixel is considered to be under the sphere shadow, and two intersection points between the vertical line passing through the pixel and sphere are found. Out of the two intersection points, one with maximum depth towards the tool axis is used for further comparisons and the other point is discarded.

To determine the intersection point between the cylinder and the virtual line, a number of steps are taken. First, it is determined whether the pixel location falls under the cylinder shadow or not. To do that, the minimum distance between the cylinder axis and the virtual line passing through the pixel is calculated. If the minimum distance is less than the tool radius, a point on the cylinder axis having minimum distance from the virtual line is determined, as shown in Figure D-1(a). If the point lies between end points of the cylinder axis (point A and B), then the pixel is considered under cylinder shadow.

If the pixel falls under the cylinder shadow, the intersection point between the vertical line passing through the pixel and the cylinder is found. To find the intersection point, the cylinder is sliced with a plane (shown in Figure D-1(a)) passing the point of minimum distance on cylinder axis from vertical line, and two end points of vertical line, shown in Figure D-1(b). An elliptical profile is formed on the plane, as shown in Figure D1(b). The minor radius of the ellipse is equal to the tool radius. To determine the major radius, the angle (β) between the cylinder axis and the horizontal plane is determined. The major radius is determined using the formula:

$$\text{Major Radius} = \frac{\text{Tool Radius}}{\text{cosine } (\beta)}$$

The point of intersection is determined by using the ellipse formula.

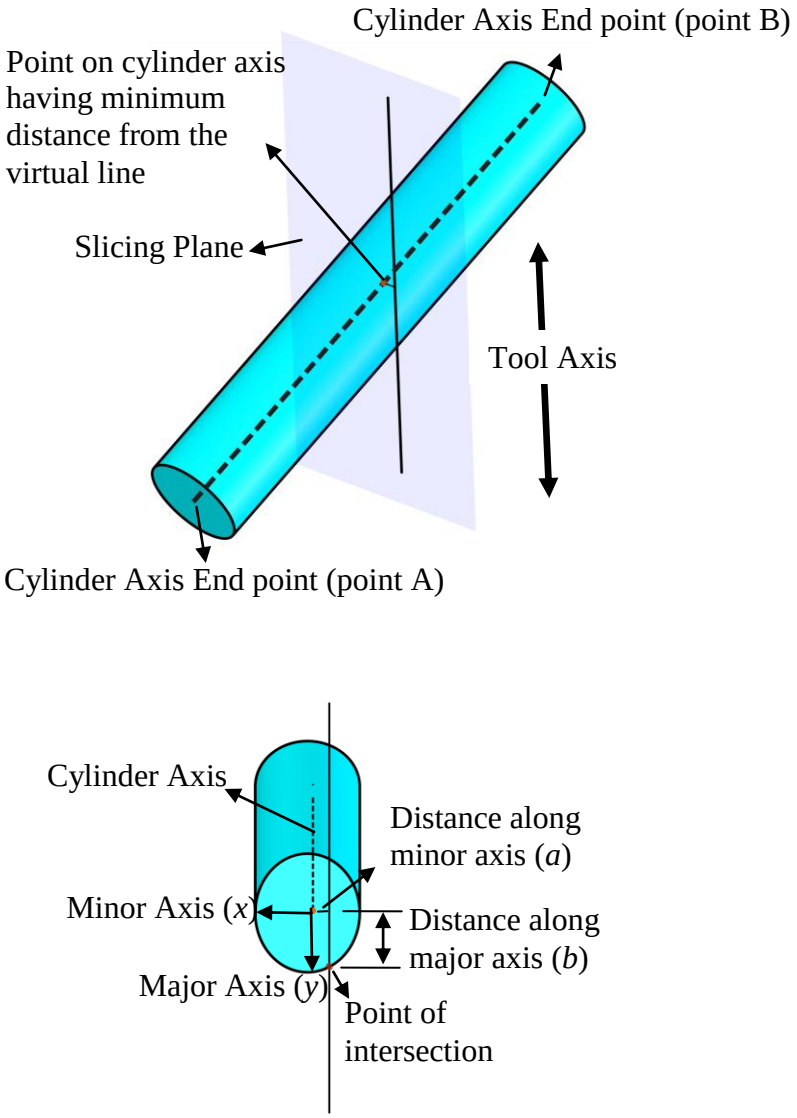


Figure D-1: (a) Cylinder-line intersection geometry and (b) after slicing the cylinder with the slicing plane.

In some cases, the virtual line may pass through close to the cylinder end points, as shown in Figure D-2. In such cases, the pixel is under the shadow of the cylinder, but the virtual line passing through the pixel does not intersect with the cylinder, and these cylinder-line intersection points must be discarded. To identify these cases, two planes having normal towards each other are passed through the end points of the cylinder. For

each plane, the dot product between plane normal and a vector connecting any point on the plane with intersection point is taken. If the value of a dot product is negative, the intersection point is discarded.

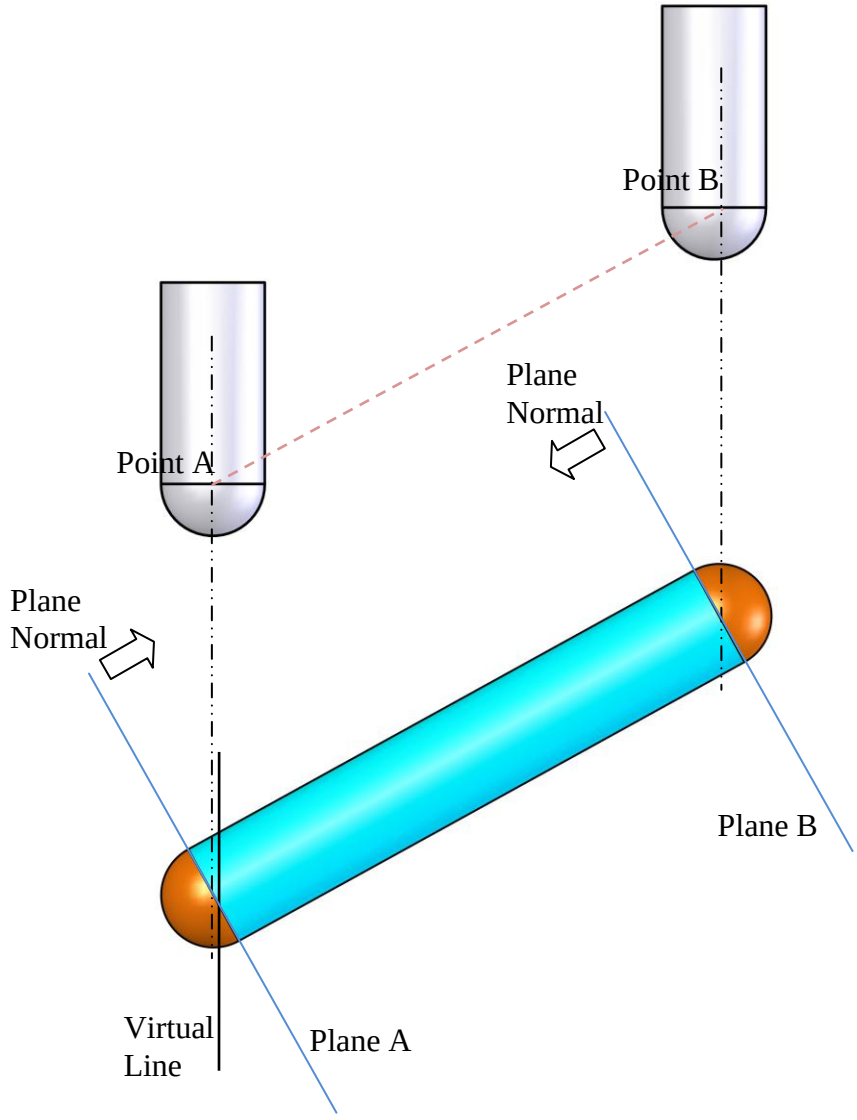


Figure D-2: Intersection point between virtual line and cylinder outside the zone formed by plane A and B.