Abstract

OLTIKAR, AKHIL MANOHAR. Computer-Aided Engineering of Plywood Upholstered Furniture Frames (Under the direction of Dr. Denis Cormier)

Until the early 1900s, furniture was built by hand, one piece at a time. The industrial revolution and modern manufacturing technology has changed all of that. Today, as the furniture industry moves firmly into the next century, computerized systems and automated manufacturing have become more common to the industry. This thesis represents an effort to analyze the current practices in computer-aided design of upholstered furniture, specifically plywood frame furniture, and to develop new procedures for reducing the lead-time in upholstery product development. Different 3-D modeling techniques for designing plywood furniture frames and their features have been developed and implemented. A plywood frame feature library has been created, and geometric relations needed to fully constrain each feature type have been developed. This reduces modeling time and also increases consistency in the solid models. A new reverse engineering procedure, using an articulating arm, has been proposed, implemented, and tested for 3-D digitization of plywood frames. The proposed methodology eliminates some of the traditional processes currently followed in the industry, thus making the product development faster and more streamlined. Further, an algorithm has been developed, implemented and tested for automatically mirroring plywood upholstery frame assemblies in a CAD system. The algorithm considerably reduces the modeling lead-time in the product development process. Finally, some future work that considers currently available 3-D CAD technologies has been recommended which would help close the gap between upholstery designers and manufacturers.

Computer–Aided Engineering of Plywood Upholstered Furniture Frames

By

AKHIL MANOHAR OLTIKAR

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Master of Science

Industrial Engineering

Raleigh, NC

2000

APPROVED BY

Dr. C.Thomas Culbreth	**Dr. Thomas L. Honeycutt**

Dr. Denis Cormier
Chair of the Advisory Committee

Personal Biography

Akhil Oltikar was born on 18th October 1976 in Mumbai, India. He did his schooling at St.Francis D'Assisi High School followed by junior college at Mithibai College, Mumbai. After schooling, he joined the D.J. Sanghvi College of Engineering, Mumbai (University of Mumbai) in 1995 for further education. After graduating from University of Mumbai, with a degree in Bachelor of Engineering (Production Engineering), Akhil moved to North Carolina State University, Raleigh, USA, for graduate studies in August 1999. After completion of his Master's degree in Industrial Engineering, he plans to work with Remix Technologies Inc. in New York.

Acknowledgements

The author would like to express deepest gratitude to his advisor Dr. Denis Cormier for his continuous encouragement, support and invaluable guidance throughout the course of this research project. He would like to thank his committee members, Dr. C.Thomas Culbreth and Dr. Thomas Honeycutt for their enthusiastic involvement and being on this committee.

Thanks to Mr. Glenn Kahn and Mr. Mike Parris at Klaussner Industries for invaluable guidance during this research.

Thanks to Mr. John H. Summey, Parikshit Kulkarni, Jason Low and Sumeet Patil for lending helping hand during the course of this research project.

Finally, the author expresses thanks to his family members for the love and support they have given him throughout his entire school career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Until the early 1900s, furniture was built by hand, one piece at a time. The industrial revolution and modern manufacturing technology changed all that. Production methods changed to building interchangeable parts, where large numbers of parts were produced that could be assembled later without hand fitting. Machines were developed that had the accuracy and repeatability to produce these parts. Today, as the furniture industry moves firmly into the next century, computerized systems and automated manufacturing have become more common. It is no longer a rarity for furniture and cabinet manufacturers to rely on computer-aided design systems for a majority of their output. More and more manufacturers are setting up direct links between these CAD systems and the CNC machinery they have on the factory floor.

Upholstering is essentially the process of covering the chair or sofa with certain fabrics to enhance its comfort and beauty. Formerly, upholstering included such as activities as making drapes, curtains, wall hangings and mattresses as well as the furniture with fabrics. Today, however, specialization has brought about a division of labor so that upholstering is usually limited to manufacturing and repair of overstuffed furniture. The establishment of mass production operations furthers sub-divides the upholstery trade, so that an upholsterer rarely completes an entire piece of furniture. He or she usually performs one or more operations on an assembly-line basis. The procedures vary in different companies all the way from custom building to strictly mass production methods.

Upholstered furniture production in the US registered over $10 billion in 1999 [CSIL, 1999]. The US records a large deficit in international trade in the upholstered furniture sector, with Canada and Mexico being the major trading partners. Production of upholstered furniture is broken down by covering materials (leather, micro fiber) and by price range. Upholstered furniture accounts for about 37% of household furniture sold in the US, and consumption is expected to grow at an annual average rate of 3% in the next years. Upholstered furniture consumption is broken down by product (stationary upholstery, stationary chairs, recliners, swivel chairs, rocking chairs, and motion-

upholstered sofa beds, futon), state, type of household. [CSIL - Centro Studi Industria Leggera, Furniture Industry Research,1999].

Thus upholstered furniture currently represents one of the most competitive segments of the furniture industry. The industry is characterized largely by high production volumes and low profit margins. To gain a competitive advantage, these companies are constantly examining new methods and technologies in design, manufacturing and marketing. One such technology is the solid modeling CAD (computer-aided design) system. New solid modeling CAD applications in designing and manufacturing of upholstered furniture have caught the eye of the furniture industry. These applications have tremendously contributed to reducing the lead times.

1.1 3-D CAD Solid Models

Historically, 3-D CAD solid models have not been used in upholstery product development. With recent technological advances in CAD, there is great utility for CAD solid models. Some of the benefits of solid models are described below.

- *Toolpath Generation* - CAD solid models are used to generate CNC router tool paths to machine plywood frame components.
- *Assembly interference detection* – Most solid modeling CAD systems have the ability to detect assembly interferences before the item is ever built. This can dramatically reduce shop floor disruptions on the first cutting of an item [Cormier and Unnanon, 2000].
- *Virtual testing* – finite element analysis programs can be used to obtain preliminary performance predictions before a frame is ever made. This is quite valuable, as frame designs that clearly will not perform can be eliminated from consideration early in the design stage.
- *Photo-realistic computer renderings* – rather than asking a customer to imagine what a sofa will look like using a small fabric swatch, a retailer can now create a photo-realistic computer rendering of that sofa.

1.2 Upholstery Product Development

At the present time, the upholstery industry is shifting away from hardwood stick-built frames towards plywood frames. Plywood construction has numerous advantages. With traditional stick-built frames, components for each style of frame are stored in bins. When an order for a particular style is received, the stored parts are picked up manually from the bins for that frame style. This process is time consuming and increases the labor costs. As the components are stored in bins, an enormous amount of money is tied up in inventory.

Conversely, plywood frames do not require bin stock. The components for plywood frames are cut from one or more sheets of plywood, using a CNC router, just prior to frame assembly. As plywood is ideal for notching applications, plywood components typically use interlocking joinery. These joints provide excellent frame strength and give frame assemblers positive locating surfaces for each component. Plywood frames are built from side-to-side using the interlocking notch joints. Stick-built frames, on the other hand, make frequent use of butt joints in which the assembler staples a component in its approximate location. Plywood frames provides labor efficiencies because the CNC equipment eliminates many manual operations.

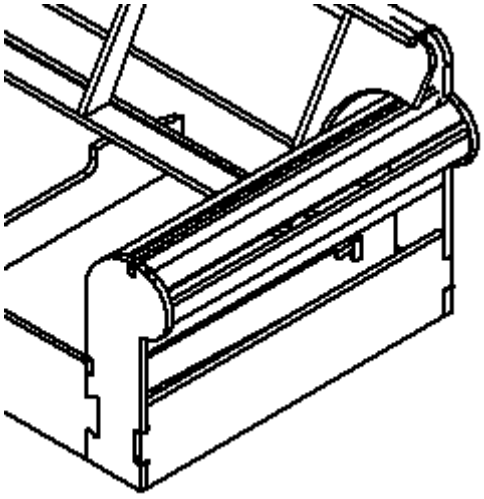


Figure 1.1: Plywood Frame with Interlocking Joints

1.2.1 Product Development Process

The product development process typically starts with the designer creating small hand drawn sketches, full-scale drawings or foam models of new styles. A company may also purchase a competitor's product and then introduce it with minor design modifications. Once the initial stylistic design concept is generated, the more practical issues of frame design and construction are considered. Figure1.2 shows a product development process that is typically followed in manufacturing upholstered furniture.

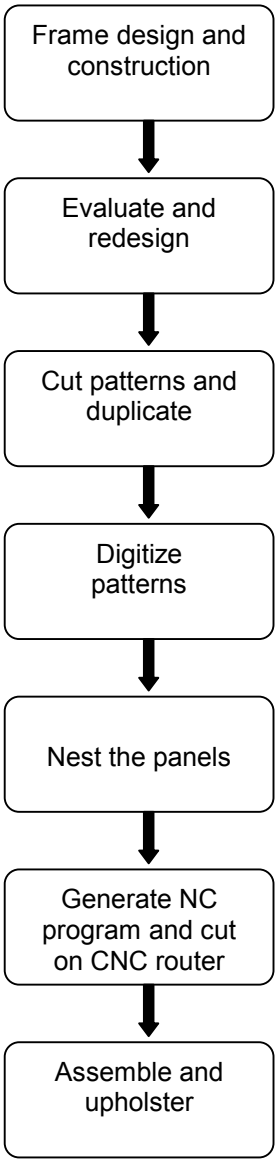


Figure 1.2: Upholstery Product Development Process

The sample piece of upholstered furniture is constructed iteratively in the model shop until all departments, including marketing, are satisfied with the design. There can be several redesigns, and each piece of plywood, foam, cardboard and fabric is typically hand cut to fit. Each piece is cut in duplicate. One copy is used in the assembled frame, while the other copy serves as a digitizing pattern once the design is finalized. After a design is approved, the outline of each pattern (plywood, fabric, foam and cardboard) is traced onto paper using a digitizing table. The electronically digitized files serve as an input to a panel-nesting program (Fig.1.3). Nesting has several benefits. One of the most important benefits is increased material utilization. Nesting increases throughput and programming productivity. It improves scheduling and control, it allows accurate cost estimates, and it generally results in lower cost per part.

Figure 1.3: Panel Nesting

(Source: www.sigmanest.com)

As shown in Figure 1.3, a variety of parts can be nested on a sheet of plywood, particleboard or MDF. Those parts are cut out, normally using a CNC router. The nesting software has a built-in function to generate NC program for cutting the panels on a CNC router. Lastly, full sheets of material are cut up into the individual parts.

In simple terms, nested manufacturing eliminates many machining operations used with hardwood stick-built frames in favor of one machine: a router. Once the plywood

components are machined they are joined with staples and glue. The cushion springs are dropped in or strung across a frame with bored spring holes. After assembly, thick cardboard is attached over each chair arm, padding is added and the backs are covered with an upholstery base cloth. The frame is now ready for upholstering

Although the use of plywood upholstery frames has many advantages as mentioned above, the use of plywood is still relatively new to the industry. Unfortunately, there is virtually no published literature pertaining to plywood upholstery frames. This thesis seeks to address this problem by focussing on computer-aided engineering and development of plywood frames. The thesis starts with review of upholstery frame construction and literature related to frame design in Chapter 2. Different types of constructions and upholstered furniture styles have been reviewed in this chapter. Chapter 3 presents a feature library for various plywood frame features. Methodology for modeling frame features and creating a feature library has been documented in this chapter. A new reverse engineering procedure, using an articulating arm, is described in Chapter 4. Chapter 5 details a procedure to mirror a plywood furniture frame assembly in a CAD system. The algorithm used for this procedure has been described. Collectively, Chapters 3 through 5 contain different computer-aided design procedures and technologies that have been developed for reducing upholstery product development time and increasing productivity.

2 Literature Review

2.1 Frame Constructions

Furniture is produced in many different styles and shapes. It might be difficult to classify the furniture according to any system other than one based on function. However, it is relatively easy to identify the structural support system used in manufacturing the furniture. Essentially all the furniture is constructed from one or more of only three basic components. These include bar-like frame members such as rails and legs; flat or curved panels such as table tops, case sides, and chair seats; and finally, shell components such as those used in molded chair construction. The complete structure is built piece by piece and may be classified according to the type of the element that predominates. In the case of sofas and lounge furniture, the predominate structure is usually a frame.

A frame can be defined as an assemblage of components that are joined together at a specified number of points termed “joints”. Joints, which are flexible and free to rotate, are called pinned joints. Joints that do not allow relative movement are called rigid joints. Most of the vertical parts of a frame are called *posts* and most horizontal parts are called *rails*. These are the main parts of a frame and give the piece of furniture its shape and support. Some vertical and horizontal pieces have no other function except to provide a place to tack down upholstery fabric.

Construction of the Sofa Support System

There are various frame styles used in sofa and lounge furniture, but there is no clear classification or categorization of these constructions. The various constructions, which go into the basic structural subsystems of which the total frame is composed, are readily identifiable. There are three subsystems - the seat system, the back system and the side frame system. Each of these sub-systems can be seen in Figure 2.1

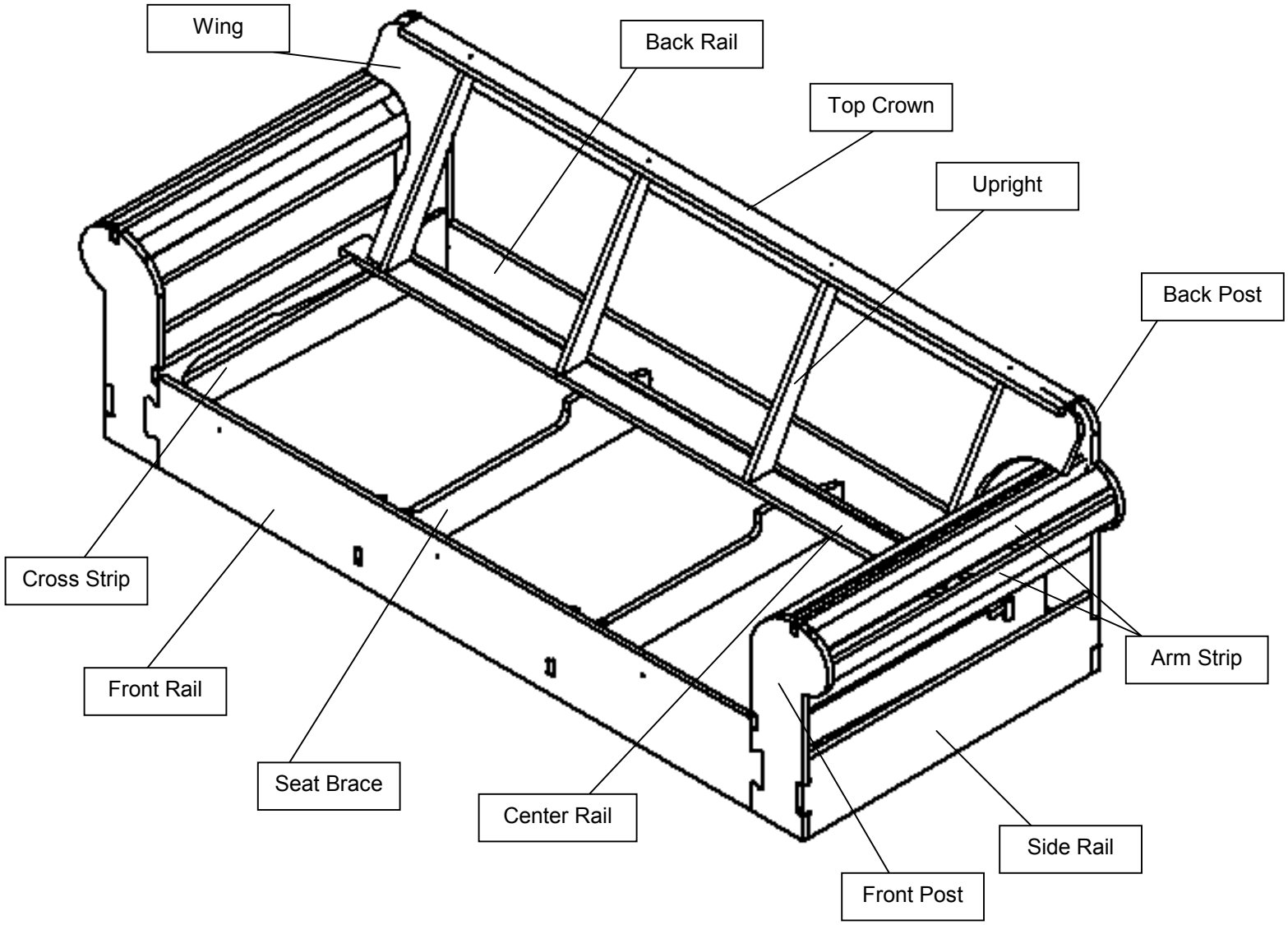


Figure 2.1: Typical Sofa Frame Construction

A. Seat support system

The seat support system for a sofa consists of the front rails, back rails and seat braces.

Front Rail Construction

Front Rail to Front Post: Figure 2.1 shows a sofa in which the arm is flush with the front rail. In this construction, the front rail is joined to the lower inside edge of the front post The joint can be a dovetail, notch & tenon joint, or a butt joint. These joints are fully explained in Chapter 3.

Front Rail to Side Rail: In cases where the arm is set back from the front rail (the so-called T-front) the front rail is attached directly to the side rail (Figure 2.2). This right-angle joint can be a butt joint or a tenon-and-notch joint.

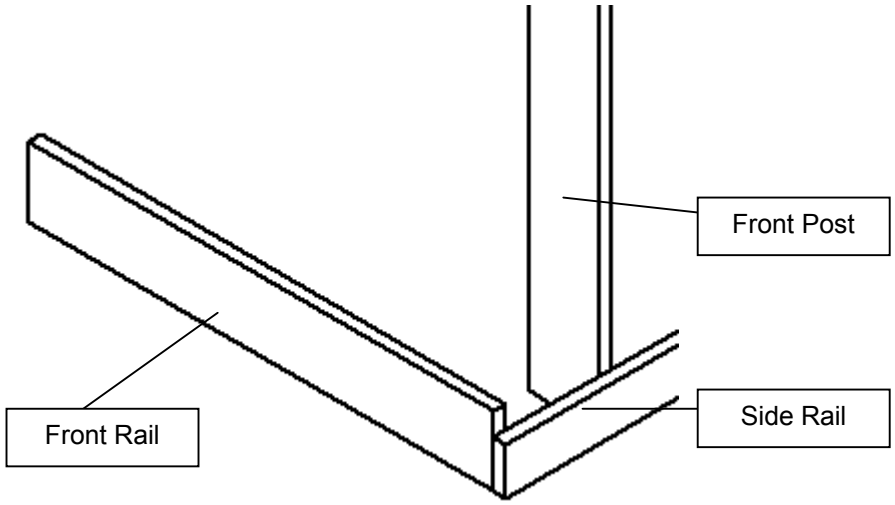


Figure 2.2: Front Rail to Side Rail Construction (T-Front)

Back Rails Construction

The back of a plywood sofa can be a single unified panel as shown in Figure 2.3, or it can be an assembly as shown in Figure 2.1. Single piece panels require less assembly labor, however they adversely affect plywood panel utilization. Much higher utilization is possible with multiple-piece back assemblies, however, assembly labor costs are incurred. In cases where an assembly is used, the back rail is joined to the lower inside edge of the back post, using dovetail or butt joints.

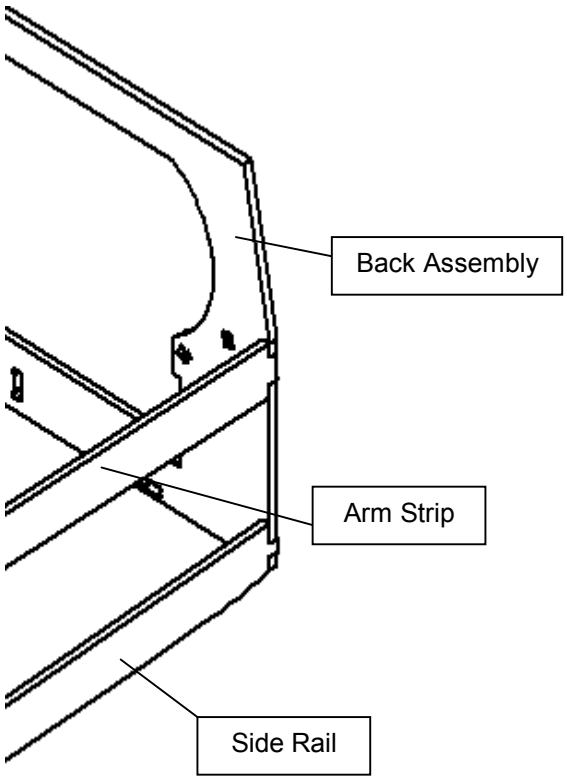


Figure 2.3: Back Assembly

Seat Brace Construction

Seat Braces used in the seat support system can be laid flat or turned on edge. Generally speaking, strength is greatest when the seat braces are turned on edge. Seat braces can either be straight or curved. Seat Braces are most commonly attached to the front and back rails through a mortise- tenon joint. Figure 2.4 shows part of a frame with end assemblies removed for clarity. The number of seat braces used depends on the length of the sofa or love seat. Typically, sofas with three seat cushions will employ two seat braces. Love seat with two cushions will use one seat brace. Chairs with one seat cushion will not require any seat brace.

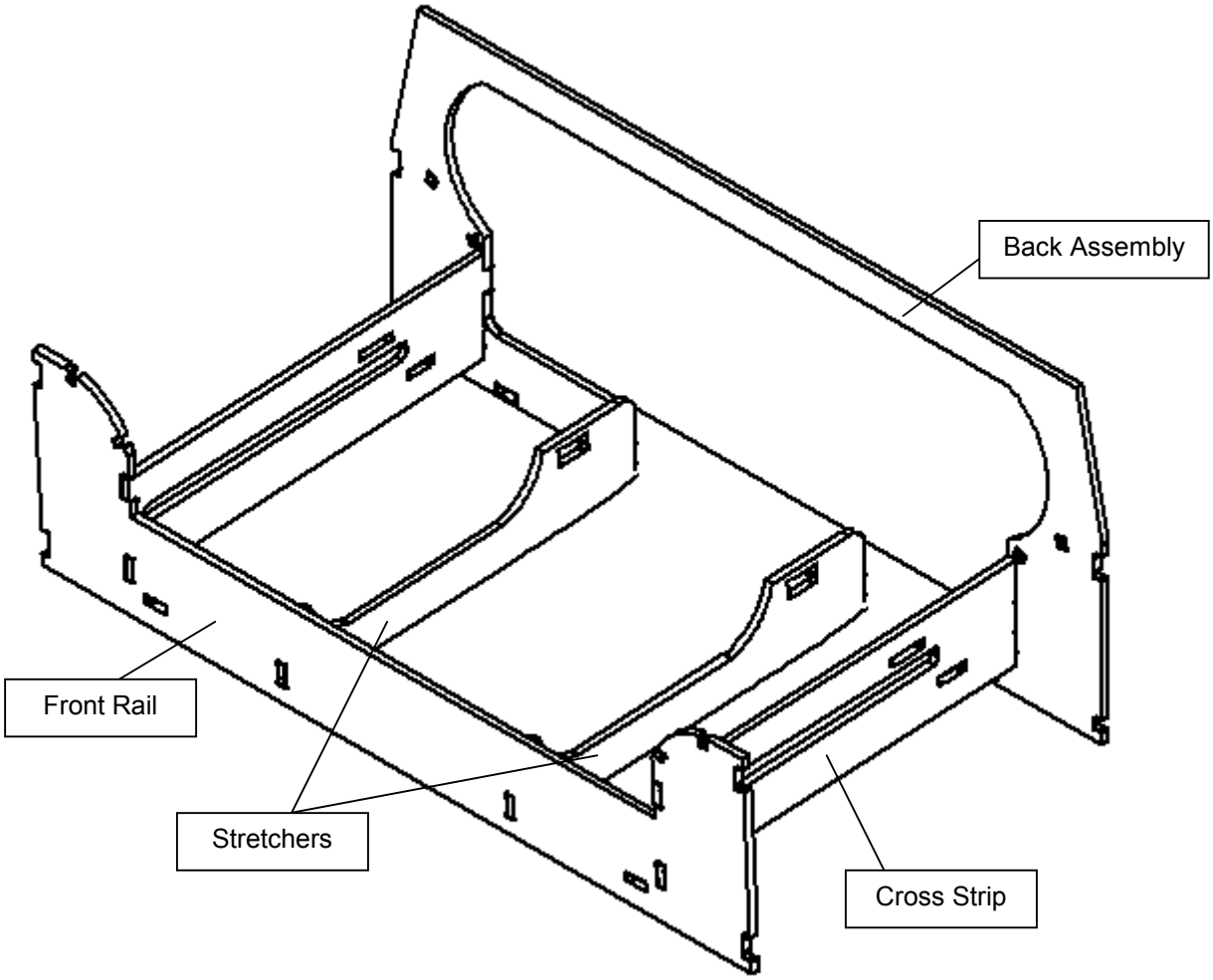


Figure 2.4: Stretcher Construction

B. Back Support Constructions

The back support system typically consists of a top rail, a center rail, back interior uprights, and backrest interior uprights.

Top Rail: In general, the top rail is laid flat. Built-up sections may also be used which have a T or L shape. In built-up sections, the top rails could provide resistance to front-to-back loads, and the bottom section provides resistance to vertical loads applied to the top rail.

Interior Uprights

Interior uprights are normally attached to the top rail through a mitered butt joint that is stapled in place.

Center Rail System

The center rail system consists of the center rail, side slats and backrest uprights. The center rail sometimes runs through the stretchers and the side rails. There can also be other members, such as center back braces, that reinforce the center rail system.

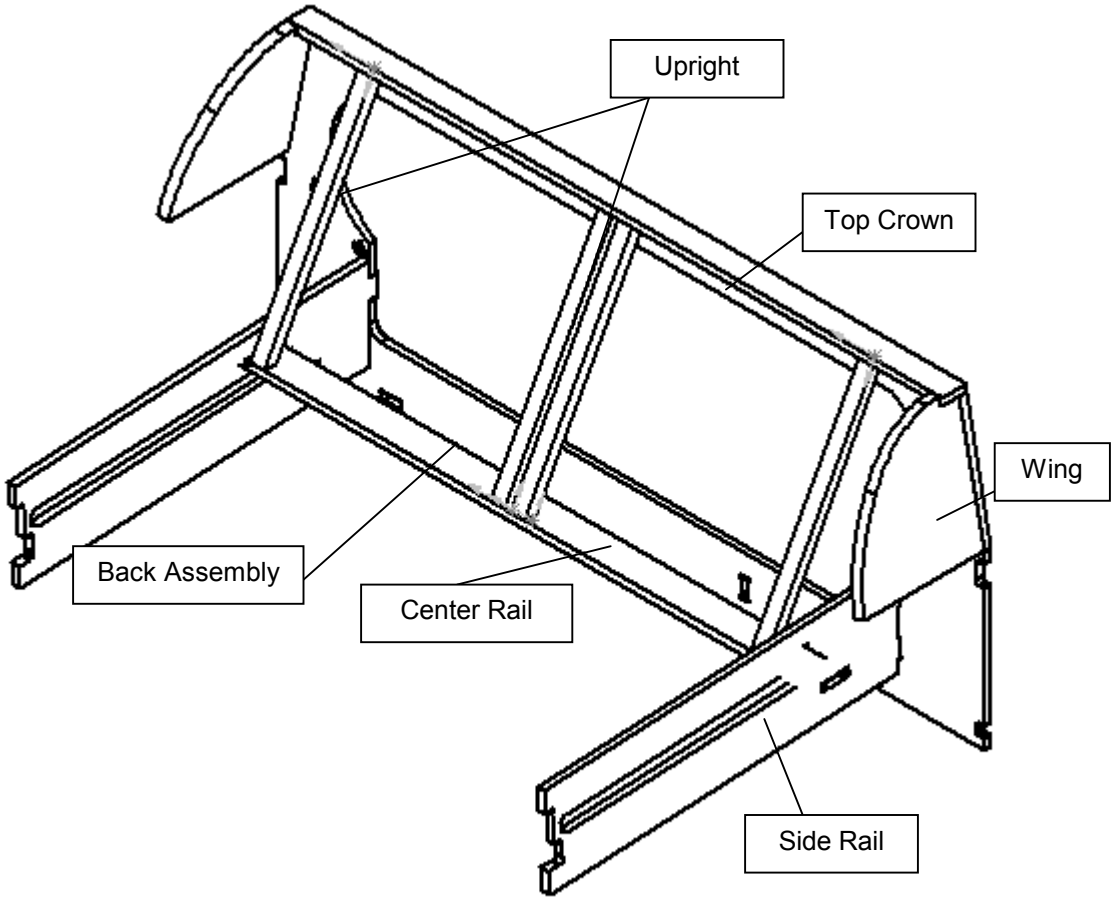


Figure 2.5: Back Support Construction

C. End Frame System

The end frame system typically consists of a front post, arm strips, a side rail and a cross strip. The side frame system can have two types of constructions. In the first, the front post is attached to the front rail, and all the other side rail members are attached to the front post. In the second, all the side rail members are joined with directly with the front rail, and the front post is attached to the side rails approximately six inches back from the front rail. This second style is used with T-front frames in which the arm is set back from the front of the item.

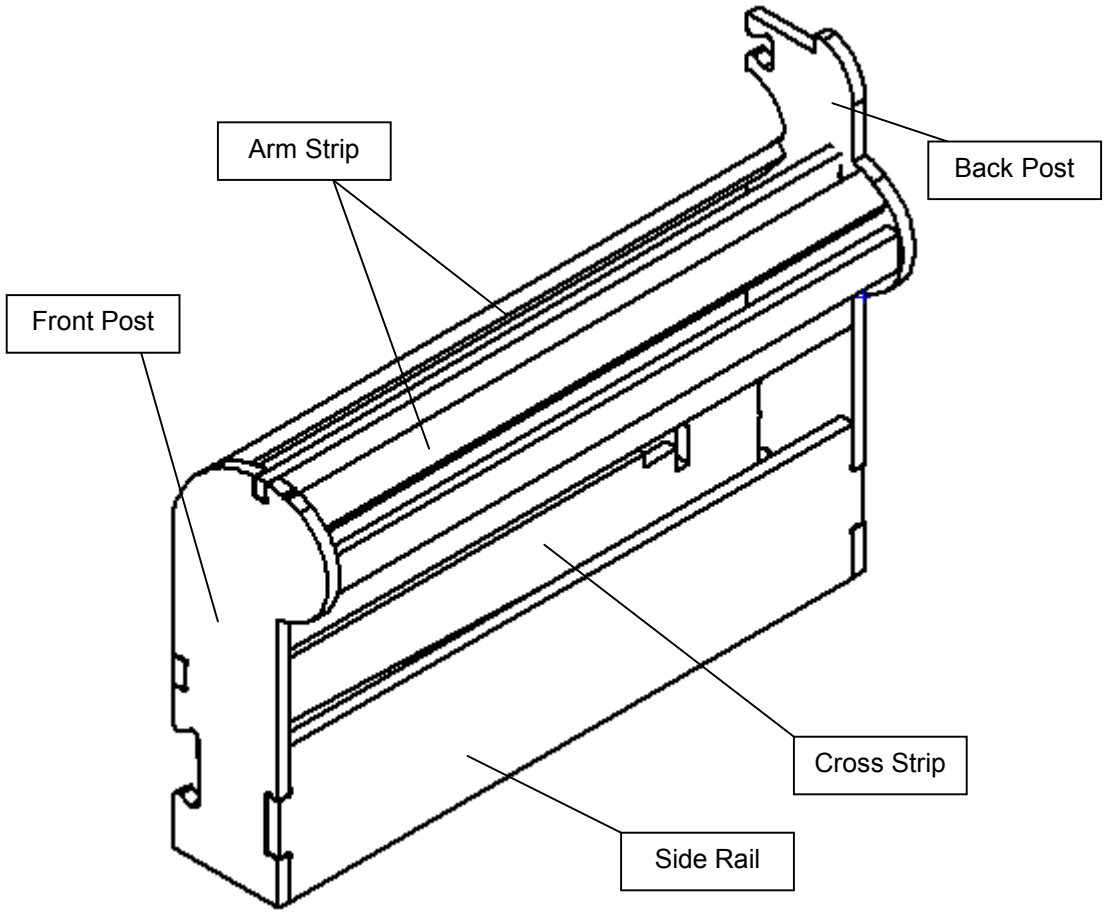


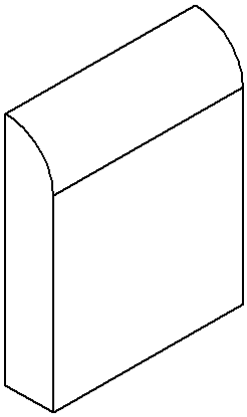
Figure 2.6: End Frame System

2.2 Upholstered Furniture Styles

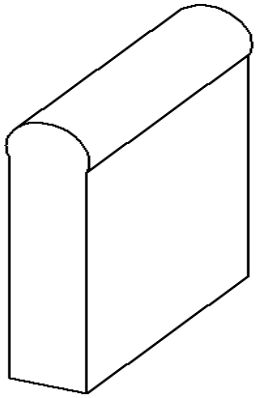
The chair is one of the most basic forms of furniture. Historically, the chair appears to have been the immediate offshoot of the stool and the forerunner of a number of familiar furniture forms. The other forms, such as sofas, loveseats etc., originated from the chair. Wood has always been a favorite building material for chairs and sofas, although other materials (from stone to steel) have been used. The upholstering on the chairs or sofas can cover the entire frame or it can cover certain portions of the frame allowing the arms or legs (or other areas) to remain exposed to view.

Upholstered furniture can be classified in many different ways. One type of classification is according to the types of springs and cushions used. That is it considers whether or not the chair is of spring or non-spring construction. The latter category includes various types of padding (moss, hair, foam etc.) not supported by springs. The padding is supported either by webbing or a piece of wood that covers the frame opening. Spring construction is of two types: (1) spring-supported seats with padded backs, or (2) springs in both the seat and the back. The latter is generally referred to as *overstuffed* furniture and will usually have two sets of springs in the seat. The lower set is the base and is always permanently set in position. The top layer can also be permanent, or it can consist of a removable cushion with springs. A more detailed discussion of various aspects of upholstering is found in the next section.

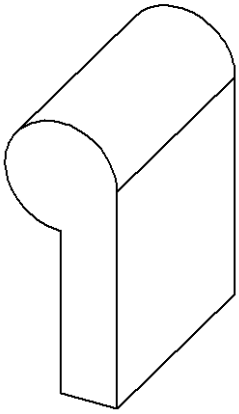
The arms of chairs and sofas can take a great number of forms. They can be fully upholstered, partially upholstered or not upholstered at all. The arms may also be full or set back. A full arm extends all the way to the front edge of the seat. A set-back arm begins at a point somewhat further back from the front edge. Another distinction can be made by the manner in which the arm attaches to the back. It may be a full or set-back arm, it can also be either an independent piece attached to the back (at a point lower than the back top rail) or it may be a continuation of the back top rail itself. Some of the basic styles of fully upholstered arms are shown in Figure. 2.7.



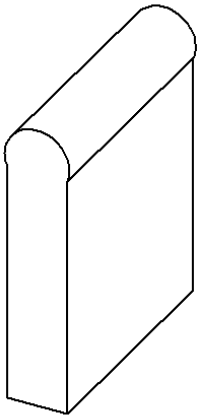
a. Knife- Edge



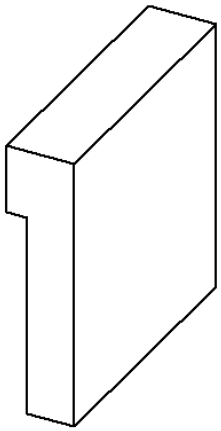
b. Modern Square



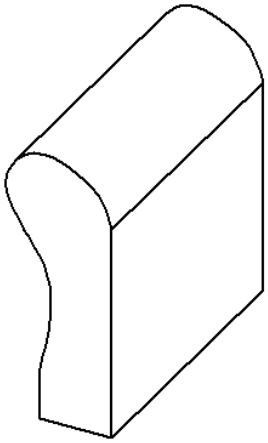
c. Scroll



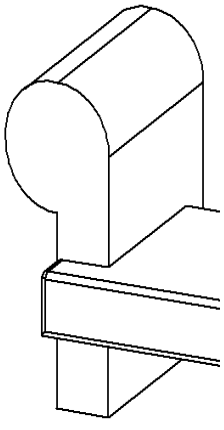
d. Rounded



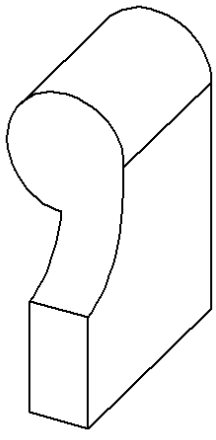
e. Square-scroll



f. Looped



g. T-front



h. Scooped

Figure 2.7: Arm Types

The back of a furniture piece can be all-wood, caned, partially upholstered, or completely upholstered. Considering only the upholstered furniture, following are most typically used types of backs for sofas.

Figure 2.8: Straight Back Sofa

(Source: www.homefurnish.com)

Figure 2.9: Camel Back Sofa

(Source: www.homefurnish.com)

Figure 2.10: Pillow Back Sofa

(Source: www.homefurnish.com)

2.3 Upholstering

There are many types and styles of upholstered furniture. Each type has a frame that supports springs and/or padding. Upholstered furniture is of the three basic types [Zimmerman, 1981]

1. *Padded* construction, which has no springs. Padding can be cotton, batting, kapok, foam rubber or other material.
2. A combination of padded construction and spring construction. The seat will be constructed with springs and the back will be padded.
3. *Overstuffed* construction, which uses springs in both seats and back.

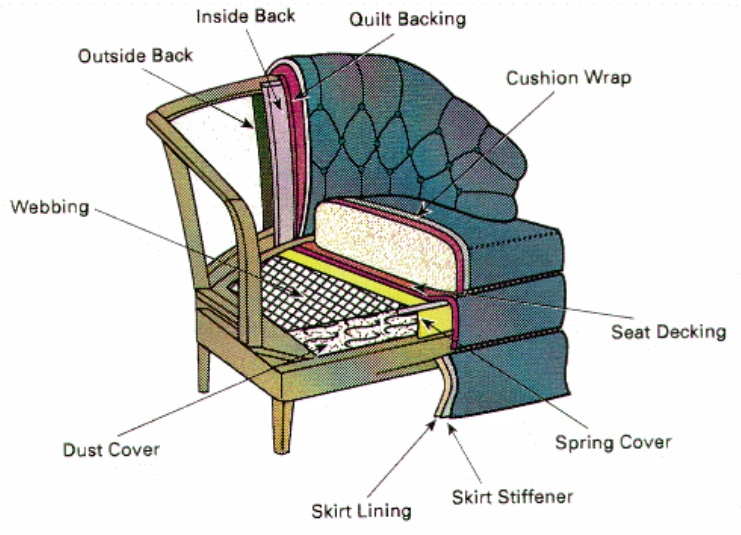


Figure 2.11: Cutout of an Upholstered Chair

(Source: www.kesselsupholstering.com)

2.4 Springs

Springs are one of the best cushioning materials in upholstery. Being resilient and flexible, springs offer the utmost comfort and support when used on upholstery seats, backs, arms and in cushions.

Spring construction is of two general types:

1. Spring supported seats and padded backs
2. Spring supports in both seat and back.

Overstuffed furniture often has two sets of springs in the seat. The lower set is the "Base" and is fastened to the frame. The upper set may be attached to the frame or installed in a removable cushion. Upholstery springs vary in type, size, style and degree of compression to serve a variety of purposes. They are made with steel wire of several gages. The wire is highly tempered and treated to resist corrosion. Seat springs are heavier, stronger and firmer than those used in arms or backs. Cushion springs are the lightest. Durability, flexibility and resilience of springs are controlled by the :

1. Size, gage of wire used and shape of springs
2. Methods of tying springs
3. Quality of spring supports
4. Anchorage

Spring types

Installation procedures for springs vary according to the type of spring used. Types generally used for upholstering include:

1. *Coil compression springs*. The coil spring is used in the backs, seats and cushions

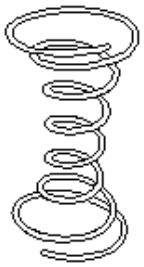


Figure 2.12: Coil Spring

(Source: www.hickorysprings.com)

of upholstered furniture. There are several types of coil springs used according to their placements.

2. *Sinuous* (sagless or zigzag) springs that are used where a low profile is desirable or required. They are faster and easier to install and are very durable. However, their major disadvantage is that they are usually not as comfortable.

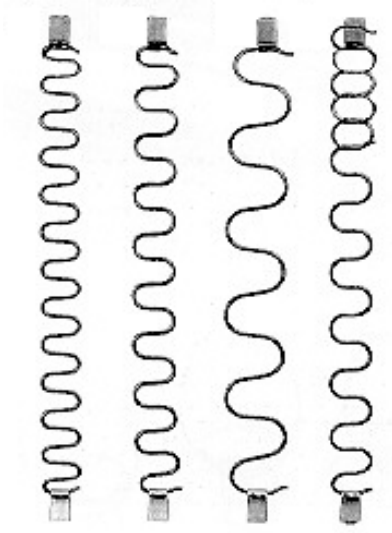


Figure 2.13: Sinuous Springs

(Source: www.hickorysprings.com)

3. *Marshall* units. These are also called “pocket springs” because each individual coil spring is enclosed in a canvas pocket or bag. They are manufactured in single rows or as complete units ready for installation in a seat, back or cushion.

Figure 2.14: Marshall Springs

(Source: www.hickorysprings.com)

2.5 Padding

Padding materials used in upholstering vary from natural fibers to synthetic fibers. Some of the typically used padding materials are [Zimmerman, 1981]:

1. *Fiber Mat:* Fiber mat is a coarse, resilient material made from fibers of the sisal plant. It is used for platform foundation padding and to pad arms and backs of furniture.
2. *White Wadding:* White wadding is a thin cotton padding usually sold in rolls. It provides a thin layer of very soft padding.
3. *Foam Rubber:* Foam rubber is manufactured in flat solid sheets1/4 to 1½ in. thick. Its resistance to compression ranges from medium to firm. Foam plastic is often substituted for foam rubber.
4. *Cotton Mat:* Cotton mat is sold in rolls or by the bale. It is available in several grades and is generally used over other foundation padding for greater softness.
5. *Dacron Polyester*: It is often used over other padding as a substitute for cotton mat. It is sold in rolls and produces a soft, resilient surface.
6. *Kapok (silk floss)*: Kapok is refined from the seed pods of the “ceiba pentandra” tree which grows best in Java. Kapok is a soft, silky, resilient material highly resistant to moisture.

In addition to the springs and the padding materials, there are many other materials used in upholstering such as jute, burlap and muslin for webbing and providing support. Upholstery fasteners such as tacks, nails and staples are used to fasten the padding and supporting materials to the furniture frame. Hog rings, clips and helical springs are used to fasten springs to the frame while metal connecting links are used to distribute weight between sinuous springs. Cardboard is used to shape the padded material according to the furniture style. Twines and sewing thread are used to sew the fabric to the padding material and edging materials like fiber flex welt cord are used to smooth sharp edges around seats or arms.

2.6 CAD-Driven Reverse Engineering

Reverse engineering is the process of developing a computer aided design model and a manufacturing database for an existing part. This process is used in CAD modeling of part prototypes, in designing molds, and in automated inspection of parts with complex surfaces. Reverse engineering is accomplished in three steps:

(1) Part digitization
(2) Feature extraction
(3) CAD modeling

Part digitization can be accomplished using Co-ordinate Measuring Machines (CMMs) or via various types of non-contact techniques laser scanners. Feature extraction is accomplished by recognizing a part surface feature in the digitized data using segmentation. Segmentation is the process of partitioning the 3-D data into non-intersecting homogenous regions. After segmentation, a CAD system is utilized to fit various types of surfaces to the segmented data.

Since long before the advent of CAD systems, working drawings have been derived from physical models. The most common approach today for addressing the need to develop a CAD model from an object is to manually measure pertinent dimensions with calipers, micrometers, rules and any other instrument necessary. When accuracy is not a concern, even the eye of the CAD operator is utilized. Manual methods can work nicely for objects with minimal features, but as the complexity of the part's geometry increases, the effort needed to accomplish the task can become prohibitive.

Feature-based reverse engineering is a relatively new approach to a traditional reverse engineering. CAD-Driven Reverse Engineering is realized by tightly integrating 3D digitizers and coordinate measuring machines with a CAD feature library. The integration is especially powerful when parametric CAD is used. CAD-Driven Reverse Engineering empowers its users to utilize a 3D digitizer as a real-time input device for their CAD system. Physical objects are reverse engineered into a CAD model by applying methods similar to approaches employing traditional hand measurement tools. The end result of the application of this integrated process is a modifiable, history-driven solid model.

2.6.1 Digitizers

Object digitizers fall into two broad categories: *contact* and *non-contact*. The *Contact* digitizers such as coordinate measuring machine (CMM), register the physical dimensions of an object by recording the position of a digitizing probe as it makes contact with the object. Coordinate measuring machines usually consist of a touch probe connected to a mechanical arm. The arm may be mounted to a tabletop system or to a small platform. The probe of a CMM contains a sensor that records the x-y-z coordinates with the press of a button. The operator traces the probe around the target object to create a point-cloud scan of the object. The precision of a CMM system relies not only on the probe but also on the sensors that estimate the position of the digitizer itself. The *noncontact* digitizer typically emits some form of optical light towards the object that is to be scanned. The digitizer then receives the light as it bounces off the object's surface and returns to the digitizer. By measuring this reflection, a noncontact digitizer can extrapolate the object's shape. The contact digitizer shown in Figure 2.15 was used in this research.

Figure 2.15: MicroScribe Digitizing Arm

(Source: www.microscribe.com)

2.7 Statement of Objectives

This thesis represents an effort to analyze current practices in computer-aided designing of upholstered furniture (plywood frames) and to develop new procedures for reducing the lead-time in product development. The thesis is divided into three parts. The first part is aimed at developing and documenting procedures for modeling plywood frame features. Though the furniture industry is beginning to embrace plywood frame construction instead of stick built frames, very little documentation or standards have been published. The aim of developing procedures for modeling plywood frames is to streamline the CAD modeling process and to increase the efficiency and consistency of entire development process.

The second part is aimed at studying the current process of 2-D digitization of plywood upholstery frames and proposing an alternative method of 3-D digitization of the same, using a portable articulating digitizing arm. 3D models play an important role in producing the correct drawing with a minimum amount of design revisions. The traditional process of digitization is cumbersome and time consuming. The new procedure is aimed at reducing the time for reverse engineering plywood upholstery frames and proposing a low cost solution.

During conversations with upholstery product engineers, it was found out that modeling an end-assembly was among the most time consuming processes in developing a 3-D solid model of a furniture frame. The third part of this thesis is therefore aimed at developing a procedure to mirror an end-assembly of a furniture frame in a solid modeling CAD system. A procedure that creates a mirror image of an existing end-assembly has been implemented and tested in Visual Basic using SolidWorks CAD system. This procedure completely eliminates the process of modeling the opposite end-assembly.

3 Plywood Upholstery Frame Feature Creation

3.1 Frame Joints and the Feature Library

A furniture frame is an assemblage of different components joined with each other using various joint features. There are four fundamental types of features used to join plywood parts:

- Notch
- Mortise
- Tenon
- Location Mark

There are variations within each category. The exact geometric shape may vary from one company to another, but the basic categories remain the same. The choice of joint is dictated by strength requirements, component orientations, ease of assembling etc. This chapter describes the procedures and geometry used to model a representative feature in each category. A feature created using these procedures is saved in the feature library of the CAD system, which can be reused for modeling the plywood parts.

A library feature is a frequently used feature, or combination of features that are created once and then saved in a library for future use. Commonly used features, such as holes or slots, can be created and saved as library features to use whenever needed. Several library features can be used as building blocks to construct a single part. This can save time, and it can help ensure consistency in the models. With most solid modelers, the features in the library can be used easily by dragging and dropping them into place, at the location required, on the solid model of the plywood part.

3.2 Types of Plywood Upholstery Frame Joints

3.2.1 In-Plane Joints

In-Plane joints are used for joining plywood parts that lie in the same plane. Different In-Plane joints are discussed below in order of increasing strength of joints

- A *Butt Joint* or *End joint* with staples and glue only: In this joint, two components butt up against each other and they are held together with staples and glue only. As this is not a very strong joint, it is used at locations where the force applied is small.

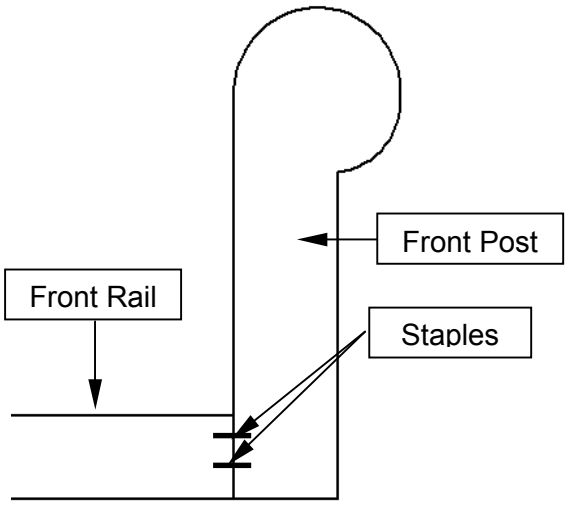


Figure 3.1: Butt Joint with staples and glue

- A *Butt joint* with a reinforcing block on backside: In addition to staples and glue, this butt joint also has a reinforcing block on the backside for greater strength. Figure 3.2 shows a butt joint with reinforcing block on the backside.

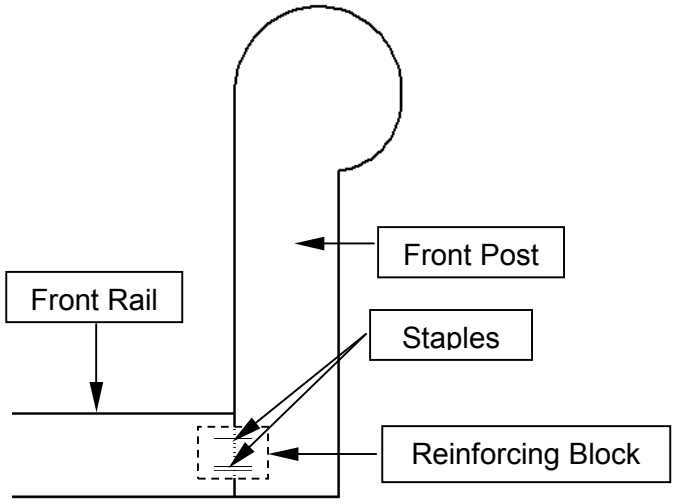


Figure 3.2: Butt Joint with reinforcing block on the backside

- *Butterfly/Dovetail Notch-and-Tenon*: In a mortise-and-tenon joint, the tenon (a rectangular projection) fits into the notch (a rectangular cut). This joint has greater strength than the joints discussed above.

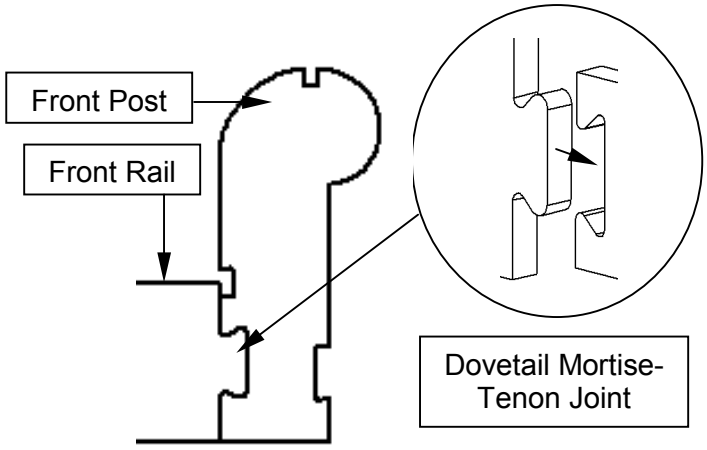


Figure 3.3: Butterfly/Dovetail Notch-and-Tenon Joint

The wedge-shaped projections fit together with matching slots like pieces of a puzzle. This joint is very strong in the direction the wedge flares. This joint is often used to break a single large piece, such as front panel, into several smaller parts in order to improve material utilization and nesting efficiency (Figure 3.4 and 3.5)

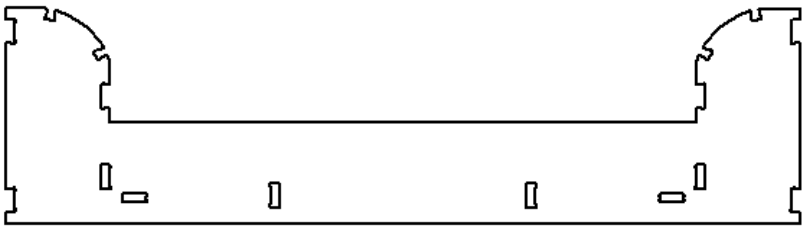


Figure 3.4: One-piece Front Panel

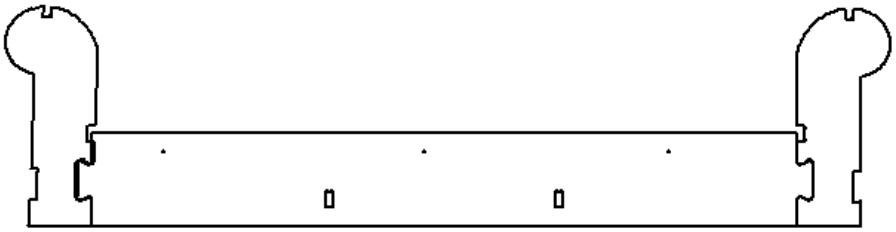


Figure 3.5: Three-piece Panel with Dovetail Joints

3.2.2 Right Angle Joints

Right angle joints, as the name suggests, are used for joining parts that lie in planes, which are at right angles to each other. Right angle joints that are typically used are discussed below.

- *Butt joint* with staple: This joint is used for non-critical strength areas. It is used for joining components for forming the shape, tack strips etc. Figure 3.4 shows a butt joint used for joining parts at right angle to each other. The arm strip, in this case, is used only to form the curved shape of the end assembly.

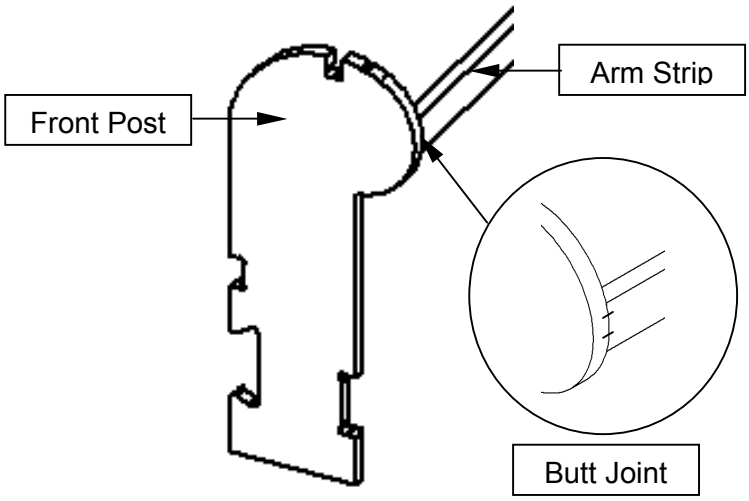


Figure 3.6: Right Angle Butt Joint

- *Notch Joint*: This joint is typically used where a rail joins to a panel somewhere along the panel's perimeter. Also, it is used where the strength is critical. For example, it is used on the top of the end-assembly, since it is a place where people often sit casually (Figure 3.7)

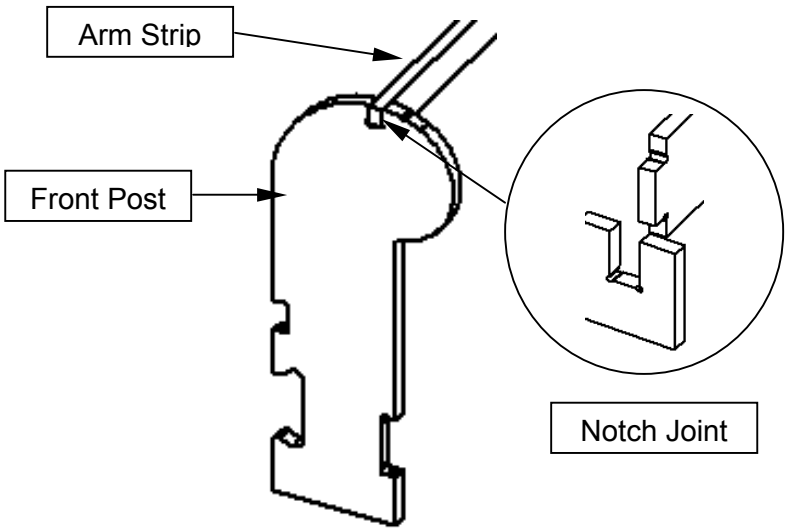


Figure 3.7: Right Angle Notch Joint

- *Mortise Joint*: A mortise joint is used when one part joins to or passes through another part, somewhere in the second part's interior region. A Mortise is basically a rectangular hole in which a tenon is fitted, like a plug in a socket. This joint has very good strength and is used at places where the force is large. For example, the seat braces are joined to the front and back rail with mortise joints.

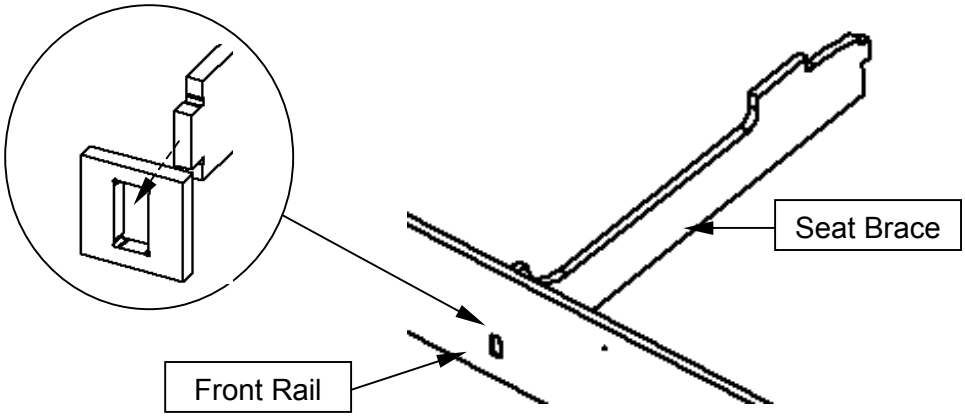


Figure 3.8: Mortise Joint

Various features such as notches, tenons and mortises are used in the joints discussed above. These individual features and their constructions are discussed in section 3.2

3.3 Plywood Upholstery Joint Features

3.3.1 Notches

There are two types of notches,

- Straight Notch
- Butterfly Notch

The straight notch is used for joining right-angled plywood components. A straight notch cannot be used with butterfly tenon for joining parts that lie at a right angle to each other. This would require a sloped notch that cannot be produced on a 3-axis router. The butterfly notch can be used with a butterfly tenon to join panels that lie in the same plane. The butterfly taper provides additional interlocking strength to the joint.

a. Straight Notch **b. Butterfly Notch**

Figure 3.9: Notches

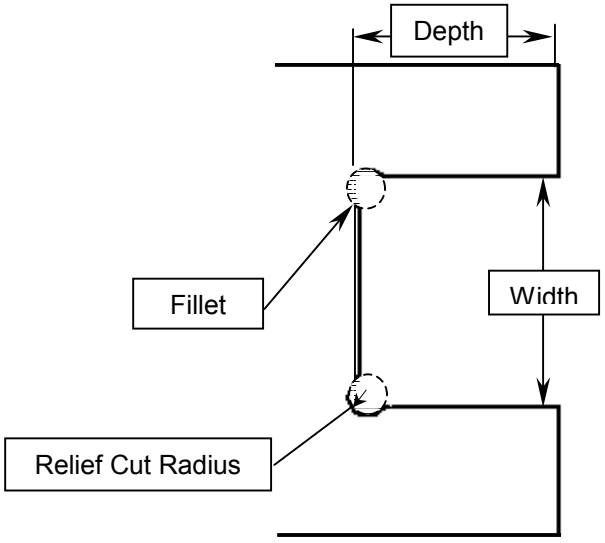


Figure 3.10: Straight Notch Geometry

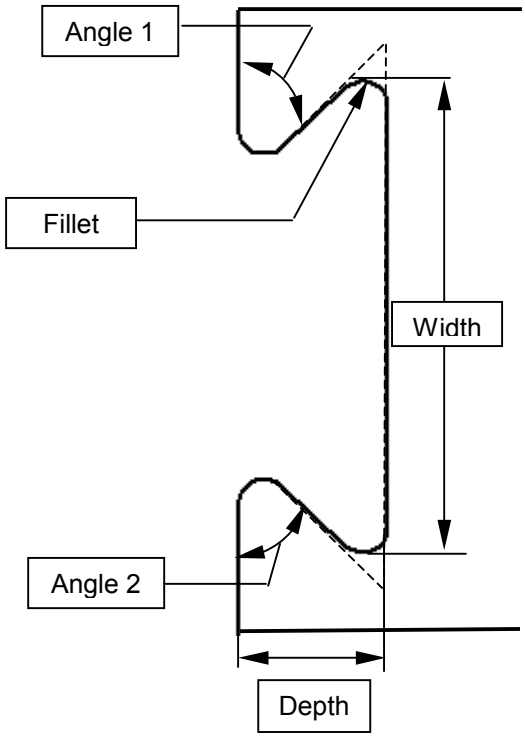


Figure 3.11: Butterfly Notch Geometry

Figure 3.10 shows the straight notch geometry. Typically, a notch has standard depth and variations in the width. Fillet radii blend the faces created by the cutting tool. The fillet radius and relief cut radius is equal for corners and fillets. For a butterfly notch, shown in Figure 3.11, Angle 1 and Angle 2 do not have to be equal. Typically, they differ by approximately 10 degrees, so that it is impossible to assemble components the wrong way. This is a form of error-proofing.

3.3.2 Mortises

A *Mortise* is a rectangular hole cut in plywood. It is interlocked with a tenon to form a mortise-and-tenon joint. A tenon fits in a mortise like a plug in a socket to form the joint. Like notches, the fillet and relief cut radius is equal for all corners. Mortises are typically used for joining parts that lie at right angles to each other.

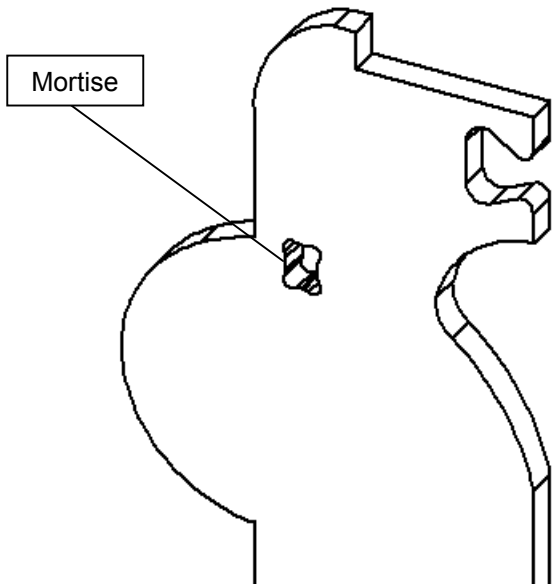


Figure 3.12: Mortise

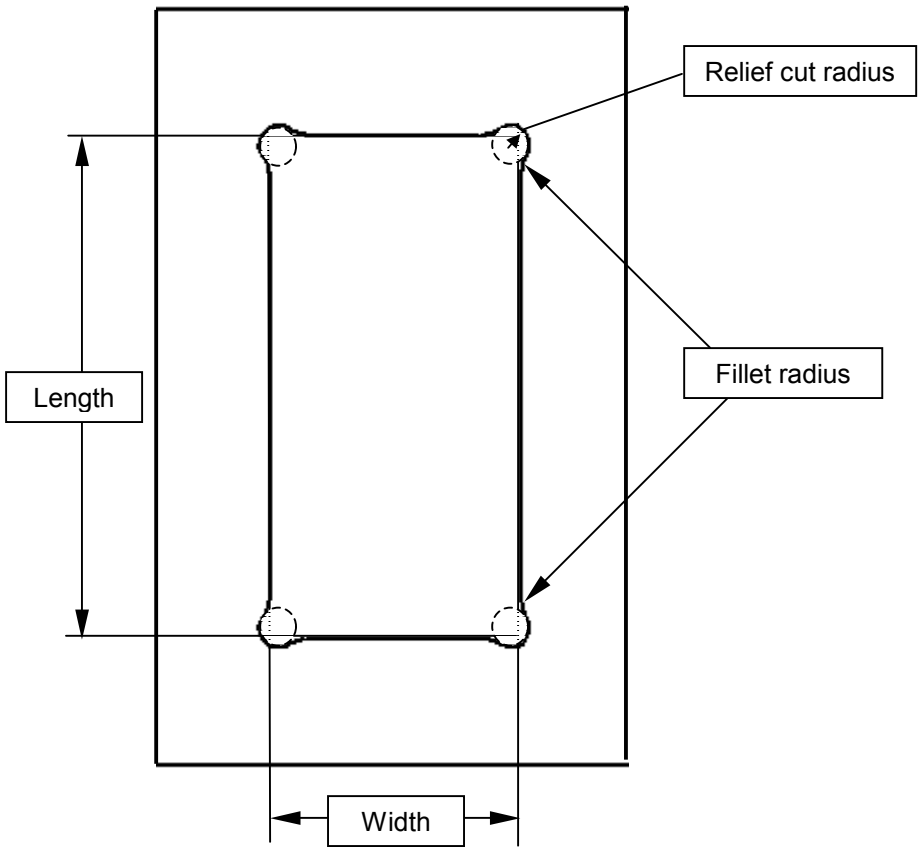


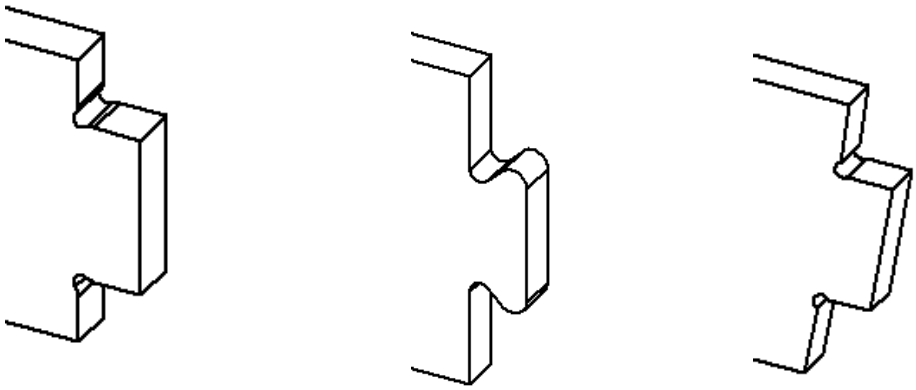
Figure 3.13: Mortise Geometry

3.3.3 Tenon

The tenon is the machined end of the rail that joins with a mortise or a notch. There are three types of tenons that are commonly used,

- Straight Tenon
- Butterfly Tenon
- Mitered Tenon

A straight tenon is used with a notch or mortise to join parts at a right angle. A butterfly tenon is used only with a butterfly notch for joining parts that lie in the same plane. A mitered tenon is used for joining components that are angled with respect to one another. For example, they are commonly used to join side rails to a sloped back panel. (Figure 3.17)



a. Straight Tenon **b. Butterfly Tenon** **c. Mitered Tenon**

Figure 3.14: Tenon

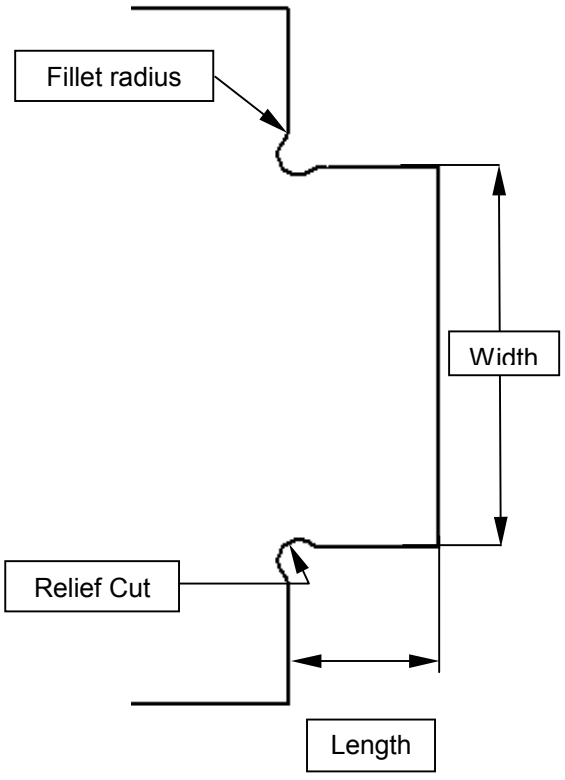


Figure 3.15: Straight Tenon Geometry

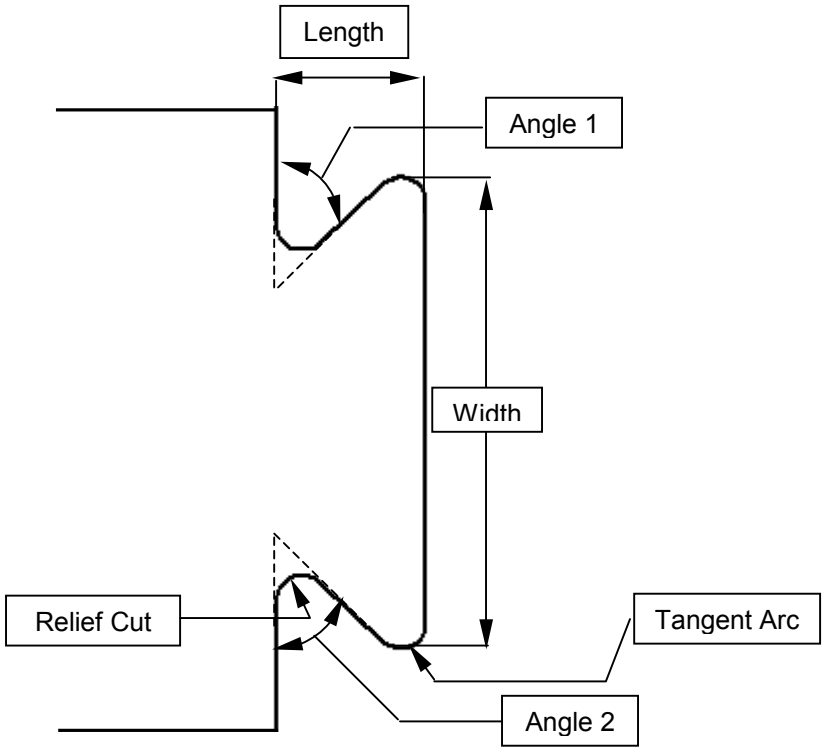


Figure 3.16: Butterfly Tenon Geometry

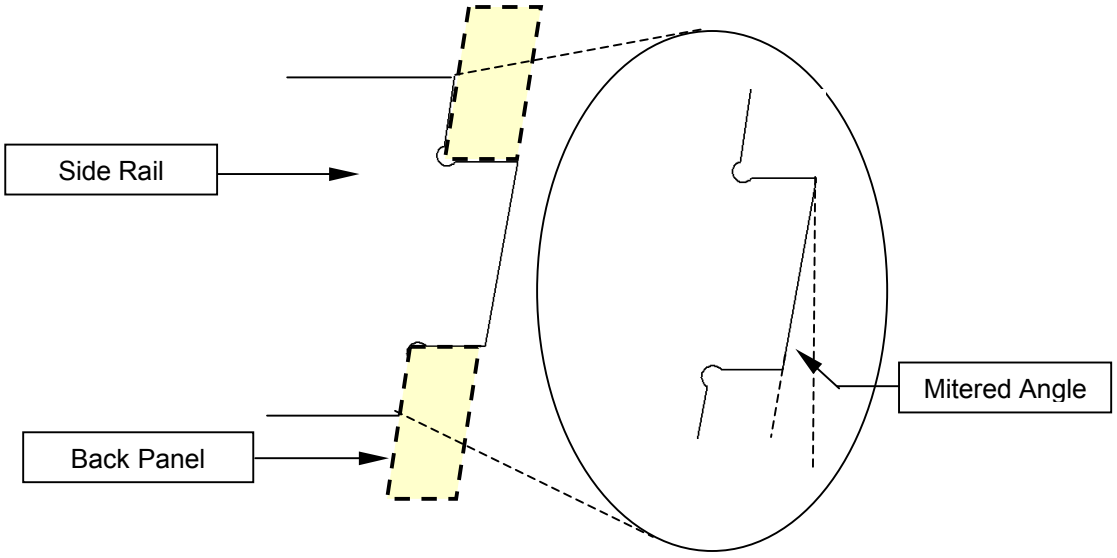


Figure 3.17: Mitered Tenon Geometry

3.3.4 Locating Mark

Location marks are machined “dimples” that provide the frame assemblers with a visual indication of where to place rails or arm strips used in butt joints. Without such marks, the assembler does not know precisely where to locate the rails or arm strips. Also, location marks are used when features such as notches or tenons are not present for assembling. The radii of the circle and fillet are the only two dimensions that are needed to define a location mark (Figure 3.19)

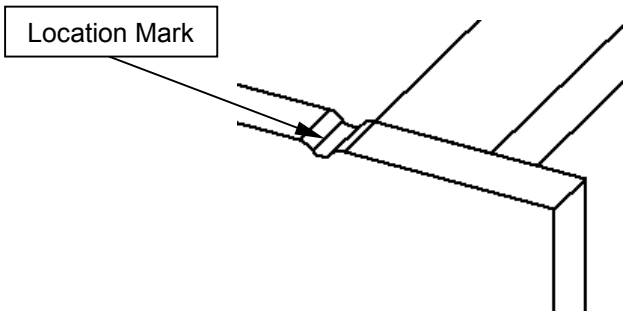


Figure 3.18: Locating Mark

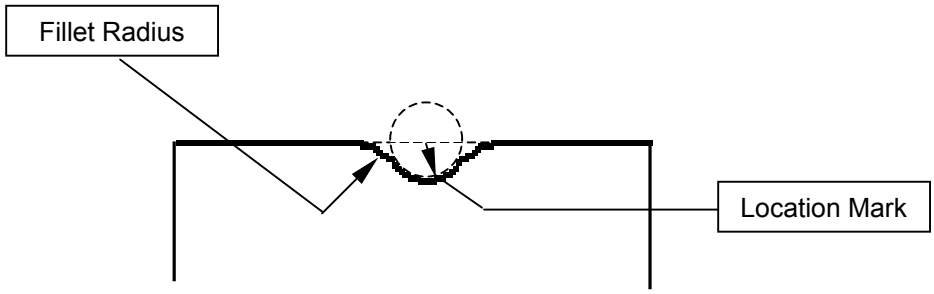


Figure 3.19: Locating Mark Geometry

3.4 Relief Cut Modeling

For notches, tenons and mortises, a relief cut is made by the cutting tool. A relief cut is needed, as a round router bit cannot produce a sharp corner. Without the relief cut, the rail or arm strip will not lie flush against the panel. The following is the procedure for modeling a relief cut in a CAD system.

Step 1: Create a sketch of the feature and dimension it.

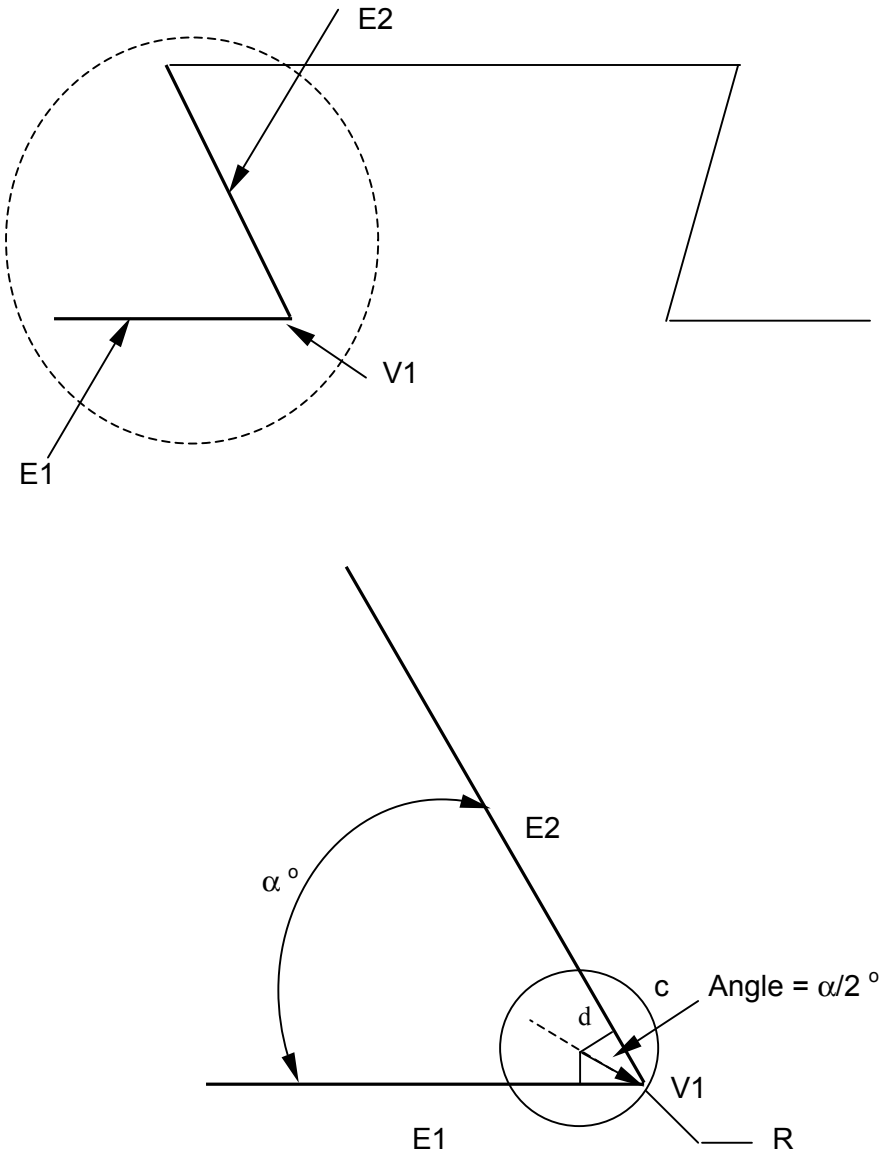


Figure 3.20: (a) Modeling Relief Cut

Step 2: Sketch a circle c and add a coincident relation between the circle and vertex V1.

The perpendicular distance d from the centerpoint of the circle to Line E2 is given by,

$$\text{Sin } (\alpha/2) = d/R$$

$$d = R \text{ Sin } (\alpha/2)$$

The distance of the centerpoint from line E2 is equal to d.

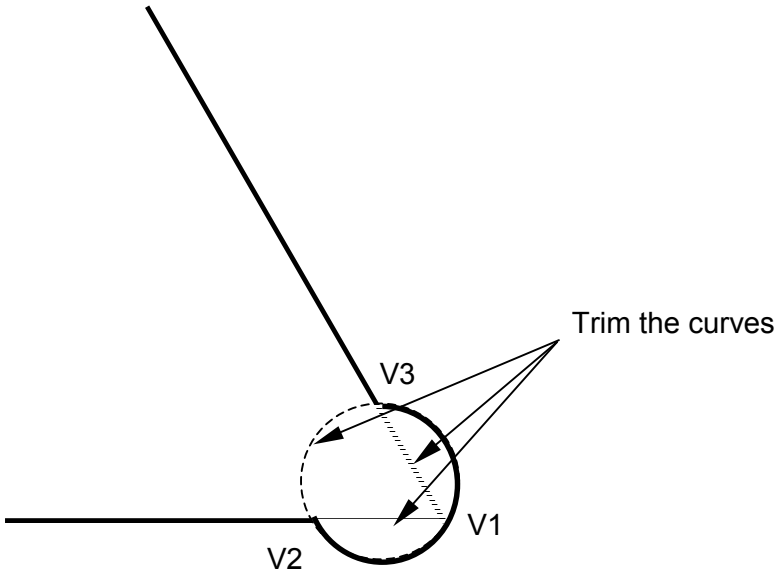


Figure 3.21: (b) Modeling the Relief Cut

Step 3: Create fillets at vertices V2 and V3. The fillet blend the face created by the circular extruded cut and E1 and E2 faces.

Step 4: Repeat steps 1 to 3 for the other side of the feature. The angle α may not be equal on both sides. This asymmetry would prevent incorrect assembly of parts.

The procedures for modeling the joints, mentioned in the previous section and implementation of the feature library are described in section 3.5

3.5 Modeling Plywood Upholstery Joint Features

3.5.1 Modeling a Straight Notch

Step1: Sketch a rectangle (bold lines) and dimension it as per the specified depth and width of the notch required.

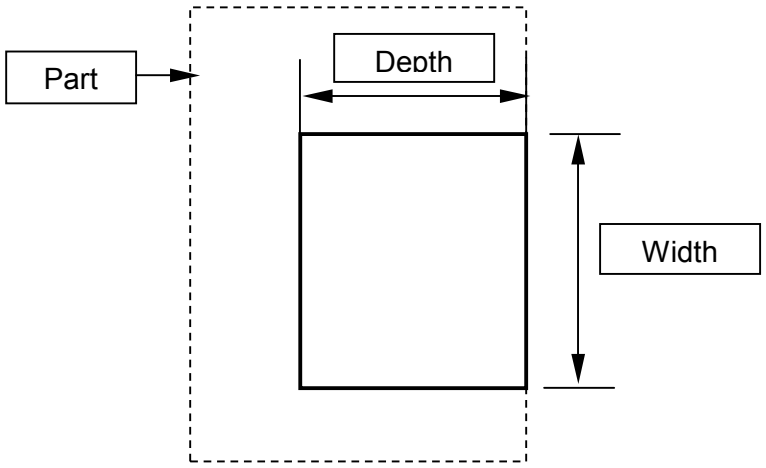


Figure 3.22: (a) Modeling a Notch

Step2: Add a circle at two adjacent corners. Follow the steps mentioned for modeling the relief cut in section 3.4. Cut extrude this sketch

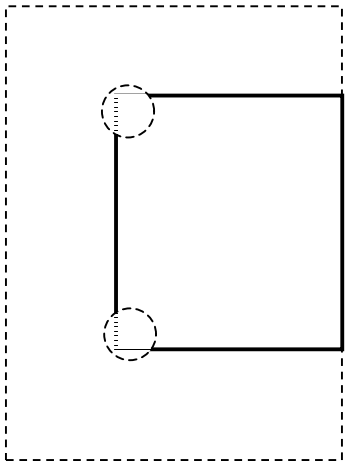


Figure 3.23: (b) Modeling a Notch

Step 3: Cut-extrude the sketch and a notch is formed. Save this feature as a library Feature.

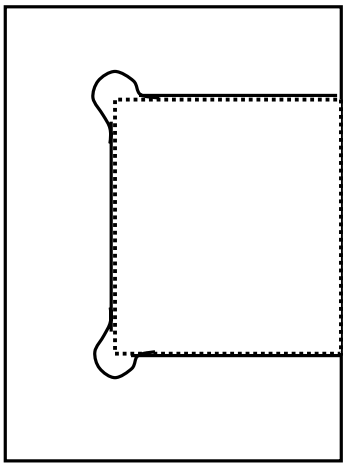


Figure 3.24: (c) Modeling a Notch

3.5.2 Modeling a Butterfly Notch

Step 1: Sketch the figure below (bold lines) and dimension it as per the specifications for required butterfly notch. Angle 1 and angle 2 are not necessarily equal. Cut-extrude this sketch.

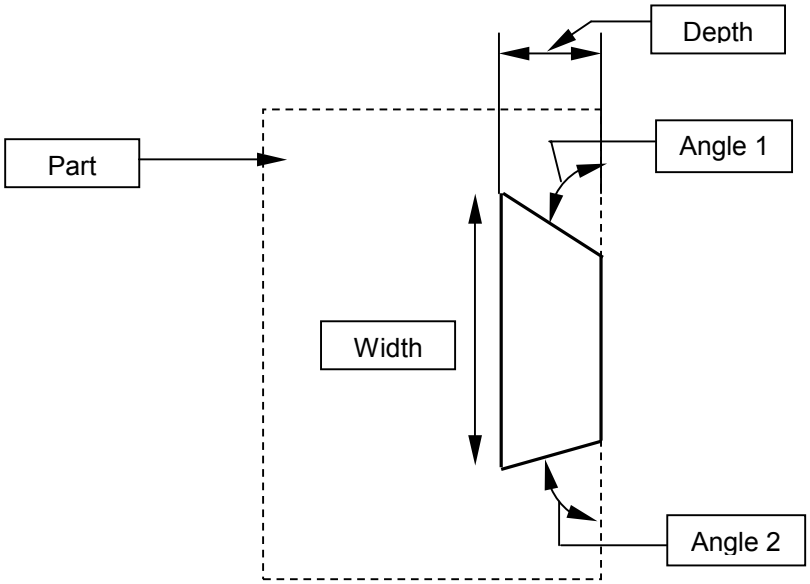


Figure 3.25: (a) Modeling a Butterfly Notch

Step 2: Create fillets at the sharp corners as per the specifications.

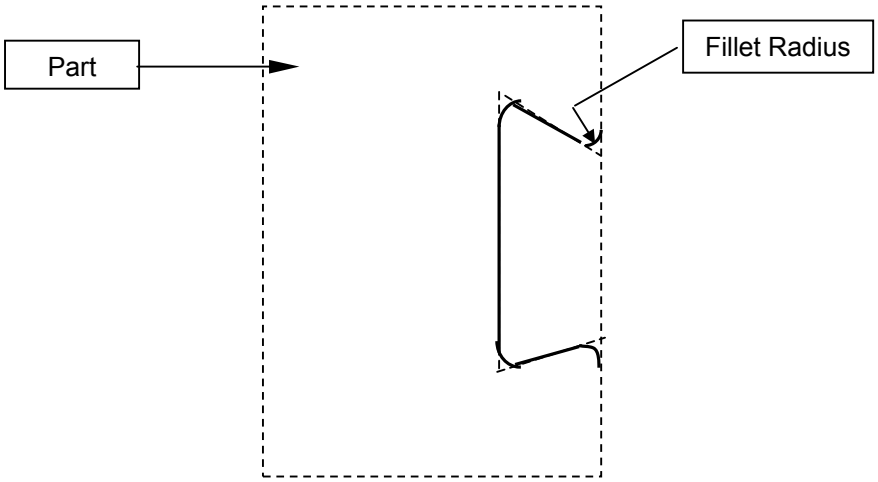


Figure 3.26: (b) Modeling a Butterfly Notch

Step 3: A butterfly notch is formed. Save this feature as a library feature.

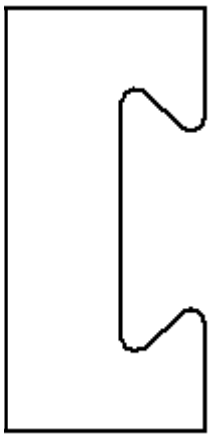


Figure 3.27: (c) Modeling a Butterfly Notch

3.5.3 Modeling a Mortise

Step 1: Sketch a rectangle (bold lines) and dimension it equal to the length and width of the required mortise size.

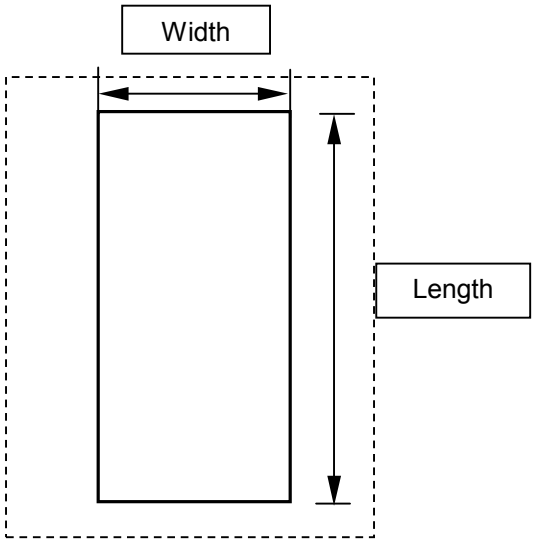


Figure 3.28: (a) Modeling a Mortise

Step 2: Add circles at four corners and follow the steps for modeling the relief cut mentioned in section 3.4 for each corner. Cut-extrude this sketch.

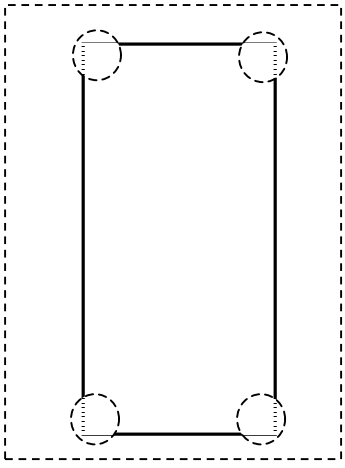


Figure 3.29: (b) Modeling a Mortise

Step 3: A mortise is formed. Save this feature as a library feature.

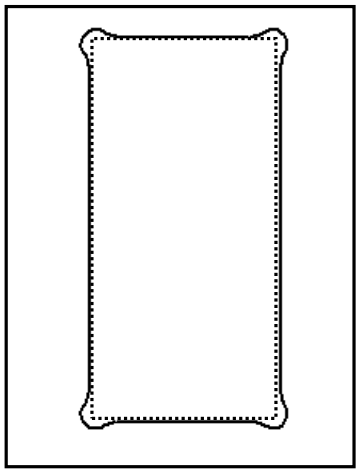


Figure 3.30: (c) Modeling a Mortise

3.5.4 Modeling a Straight Tenon

Step 1: Sketch the figure below (bold lines) and dimension it as per the specification of the required tenon.

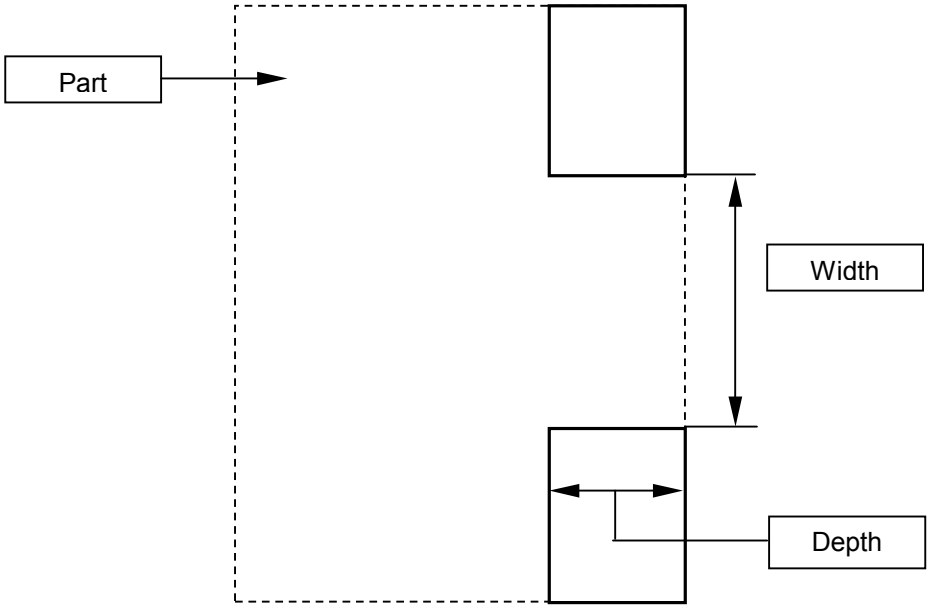


Figure 3.31: (a) Modeling a Straight Tenon

Step 2: Create circles at the corners shown below and follow the steps to model a relief cut as discussed in section 3.4. Cut-extrude this sketch.

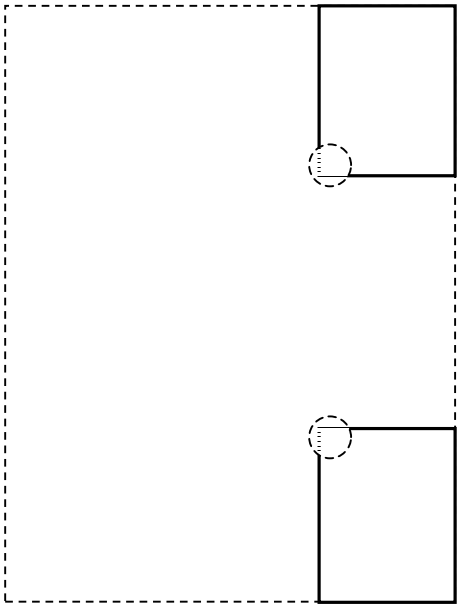


Figure 3.32: (b) Modeling a Straight Tenon

Step 3: A straight tenon is formed. Save this feature (bold lines) as a library feature.

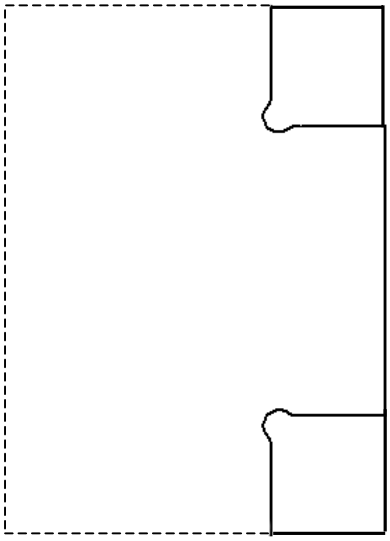


Figure 3.33: (c) Modeling a Straight Tenon

3.5.5 Modeling a Butterfly Tenon

Step 1: Sketch the figure below (bold lines) and dimension it as per the notch specifications. Cut-extrude this sketch.

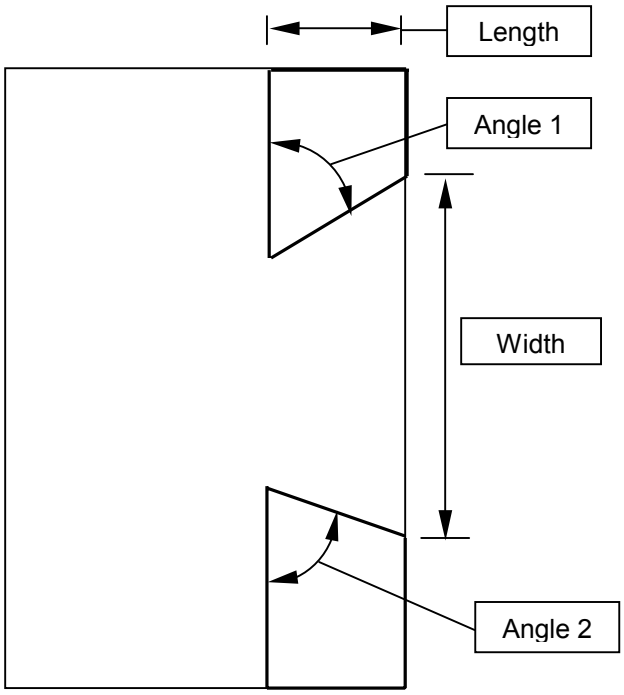


Figure 3.34: (a) Modeling a Butterfly Tenon

Step 2: Create fillets at the sharp corners as per the specifications for fillet radii.

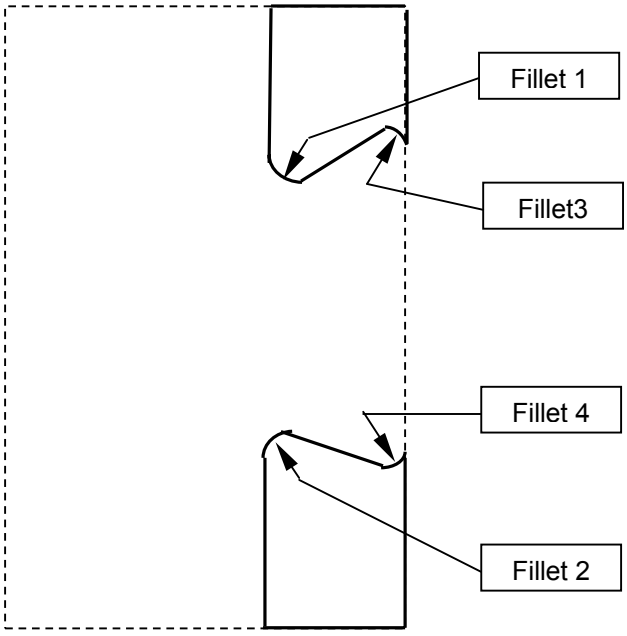


Figure 3.35: (b) Modeling a Butterfly Tenon

Step 3: A butterfly tenon is formed. Save this feature (bold lines) as a library feature.

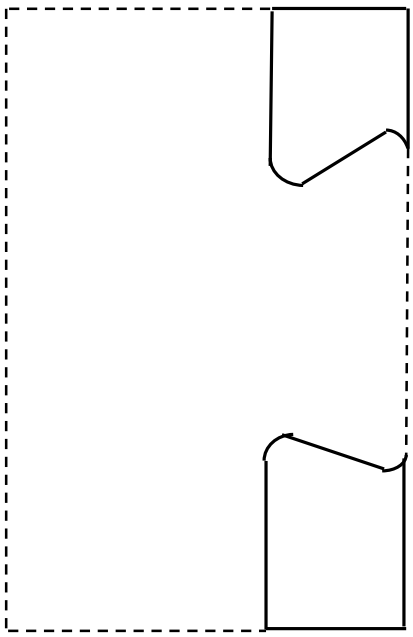


Figure 3.36: (c) Modeling a Butterfly Tenon

3.5.6 Modeling a Locating Mark

Step 1: Create a circle at a desired location and dimension it as per the specifications.

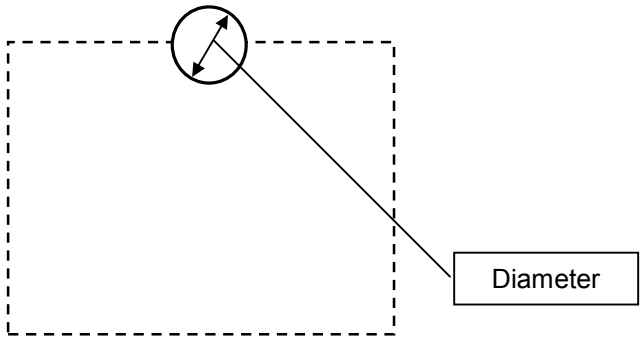


Figure 3.37: (a) Modeling a Locating Mark

Step 2: Add tangent arcs to the circle as shown in Figure 3.38

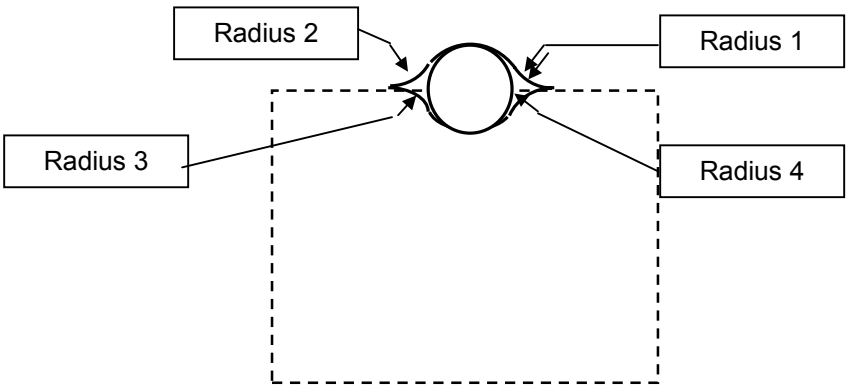


Figure 3.38: (b) Modeling a Locating Mark

Step 3: Trim the curves to form the sketch shown in Figure 3.39.

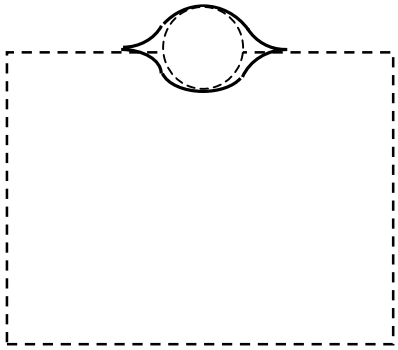


Figure 3.39: (c) Modeling a locating mark

Step 3: Cut-extrude the resulting sketch and save as it as library feature part.

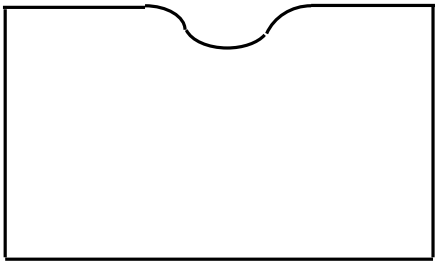


Figure 3.40: (c) Modeling a locating mark

3.6 Implementation of the Feature Library

The different features mentioned in the previous section have been modeled in the SolidWorks CAD system and saved as *Feature Library* parts.

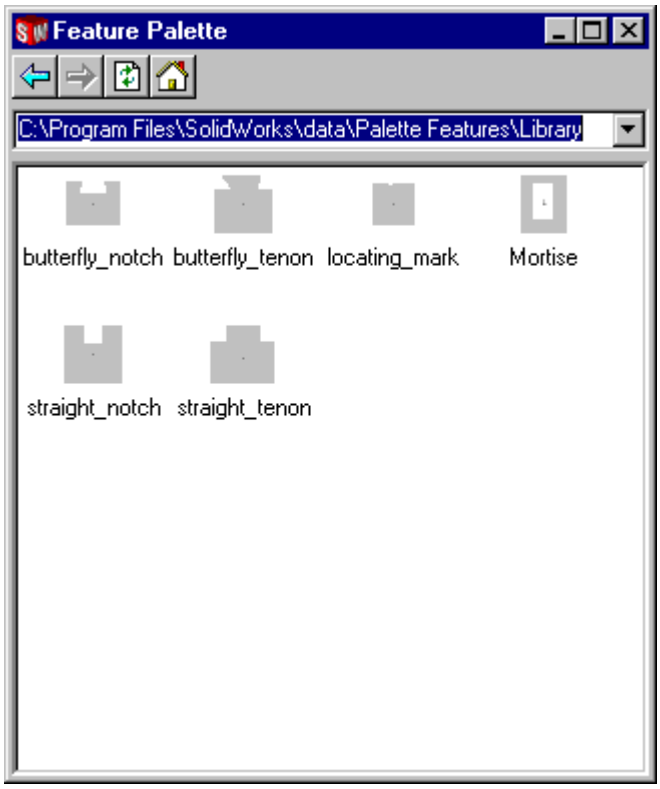


Figure 3.41: Feature Palette Window

The procedure for using the feature library in SolidWorks is as follows:

1. Open the part file and *Feature Palette* window.
2. Select the required feature and drag-and-drop it at an approximate location on the part in part file.
3. Orientation of the feature can be changed by adding relationships with the existing part features.
4. The dimensions of the feature can be changed when the window shown below appears.

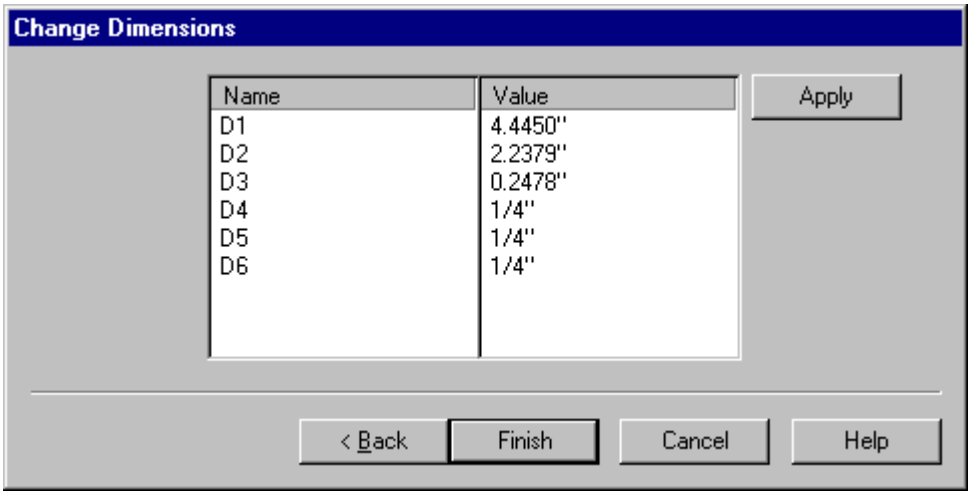


Figure 3.42: 'Change Dimensions' Window for Palette Features

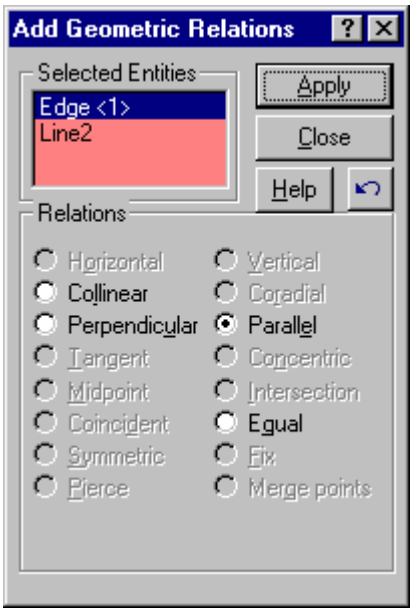


Figure 3.43: Adding Geometric Relations Window

The figure below shows a front post with the library features added by this drag-and-drop method.

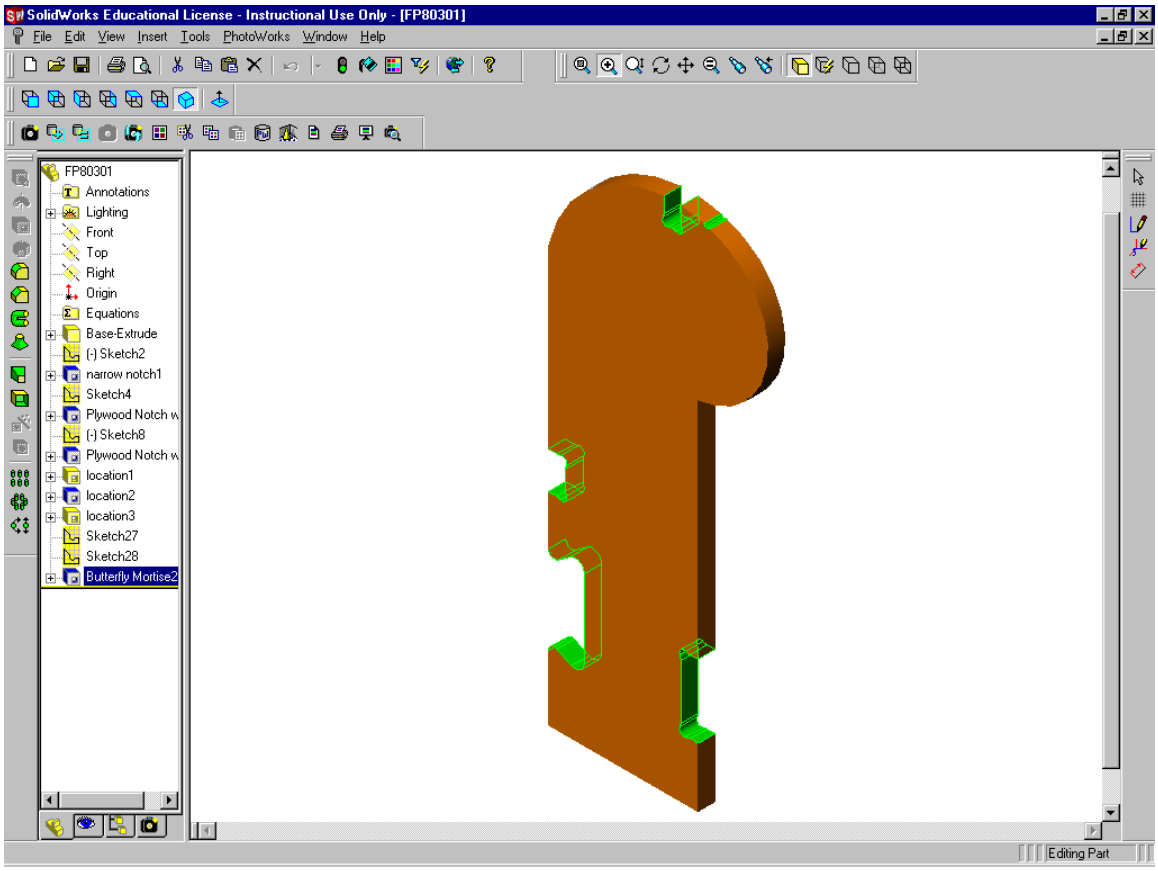


Figure 3.44: Front Post with Library Features Added

The Feature Library serves as a useful tool in modeling plywood frames. Various features can be stored, modified and used consistently. This results in large savings in time required for 3-D modeling of furniture frames and consistent solid models.

4 3-D Digitization of Plywood Upholstery Frames

4.1 Reverse Engineering in Upholstery Product Development

A study of today's CAD end user community uncovers a continuing need for 2D drawings across all industries - despite the visionary promise of a paperless design and manufacturing process. In addition, the modern workforce has been trained to communicate in 2D, employing a drawing's orthographic projections. While a 2D drawing may best represent a final design, 3D plays an important role in producing the correct drawing with a minimum amount of design revisions. Reverse engineering is used in upholstery product development to generate CAD models from physical patterns. The traditional reverse engineering process for creating 2-D models is cumbersome and time consuming. An alternative three-dimensional (3-D) digitizing process that employs an articulating digitizing arm is proposed in this chapter. A prototype system has been developed, and feature-based reverse engineering software has been written to integrate the system with a well-known commercial solid modeling CAD system. Results from testing on a love seat are presented, and recommendations for future applications of the system are provided.

4.2 Plywood Frame Digitization

The product development process starts with the making of complete, assembled frame using the designer’s sketches. In this case, the "design concept" takes the form of an actual upholstered piece of furniture. Once the initial design concept is generated, the more practical issues of frame design and construction are considered. This is primarily the responsibility of the product engineering department. Figure 4.1 shows a product engineering process that is typical of many upholstery manufacturers. Regardless of how the design concept is supplied (i.e. hand drawn sketch, full scale drawing, foam model, or competitor's frame), the product engineering department is charged with the task of designing a frame that produces the look of the item while adhering to the companies' standard frame construction practices.

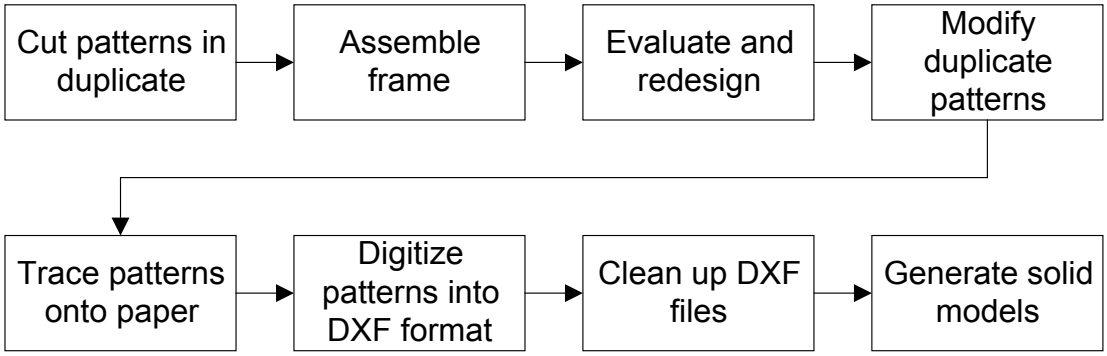


Figure 4.1: Plywood Frame Digitization Process

The designer works with the model shop to iteratively construct a sample piece of upholstered furniture until all parties, including marketing, are satisfied with the design. Since this is an iterative process, it may require several redesigns. Each piece of plywood, foam, cardboard, and fabric is typically hand cut to fit using the appropriate tool (i.e. band saw, jig saw, scissors, etc.). Typically, each piece is cut in duplicate. One copy is used in the assembled frame, while the other copy serves as a digitizing pattern. The duplicate patterns are modified after each redesign.

Upon completion of this iterative design process, the outline of each final pattern (plywood, fabric, foam and cardboard) is traced onto paper. The paper patterns are then taped to a large digitizing table, and the patterns are electronically digitized using the industry standard DWG or DXF file formats. The digitization is carried out using a digitizing *Puck,* which is a mouse-like device that collects the X, Y coordinates of different points of the traced part and creates the part profile in a 2-D CAD system. If a three-dimensional model of the item is required, as is increasingly the case, then the last step is to import these two-dimensional profiles into a solid modeling CAD package. Since the digitized patterns frequently contain imprecise geometry, a considerable amount of time may be required to "clean up" the digitized data within the solid modeler before it can be used to create three-dimensional models of each component part. Finally, the digital model is used to drive CNC routers, fabric cutters, and foam cutters.

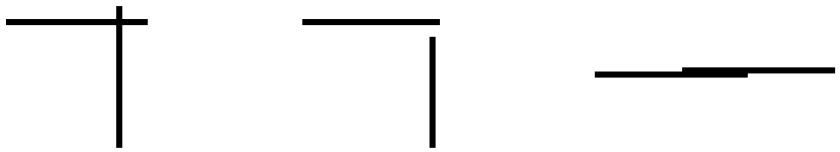


Figure 4.2: Commonly Occurring Digitizing Errors

The reverse engineering process can be quite time consuming. This is mainly due to the number of steps in the process, and the fact that digitized data typically needs to be repaired before it can be used in a solid modeler. Figure 4.2 shows examples of some of the most commonly occurring digitizing errors that lead to problems during the solid modeling process. One error occurs when end points of adjacent segments overlap. Likewise, a pair of adjacent sketch segments can leave an open gap, as shown in the middle illustration. A third common error occurs in situations where a single line segment is digitized as two partially overlapping lines instead of one single line. Any variation of these three situations must be repaired before the digitized curves can be used in the solid modeler.

4.3 Reverse Engineering Using an Articulating Arm

In order to streamline the reverse engineering process, an alternative reverse engineering process is proposed that uses an articulating digitizing arm rather than a digitizing table. The articulating digitizing arm is far more flexible than the flat digitizing table, and it allows for several steps in the conventional reverse engineering process to be eliminated. Articulating digitizing arms such as the one used in this research (Figure 2.15) cost approximately the same amount as digitizing tables, hence the hardware costs of the two approaches are comparable. A flow chart for the proposed process is shown in Figure 4.4. The key difference between the process shown in Figure 4.4 and the traditional process shown in Figure 4.1 is that this alternative process digitizes component shapes directly off of the assembled frame using the articulating digitizing arm. Rather than taping a flat sheet of tracing paper down on a digitizing table, the CAD operator grasps the tip of the digitizing arm and touches the appropriate coordinate points of each given part on the fully assembled frame. The X, Y, Z coordinates are

collected by the articulating arm, and the part profile is generated in the CAD system. This has several important ramifications,

- The individual pieces are no longer cut in duplicate, since all changes are made by digitizing straight off the assembled frame.
- Since duplicate pattern components are not necessary, the process of modifying two sets of parts for every design change is no longer necessary.
- The step of tracing pattern parts onto paper is eliminated.
- By digitizing straight into the solid modeling CAD system via a process of feature-based reverse engineering, the step of cleaning up imported DXF files is also eliminated.

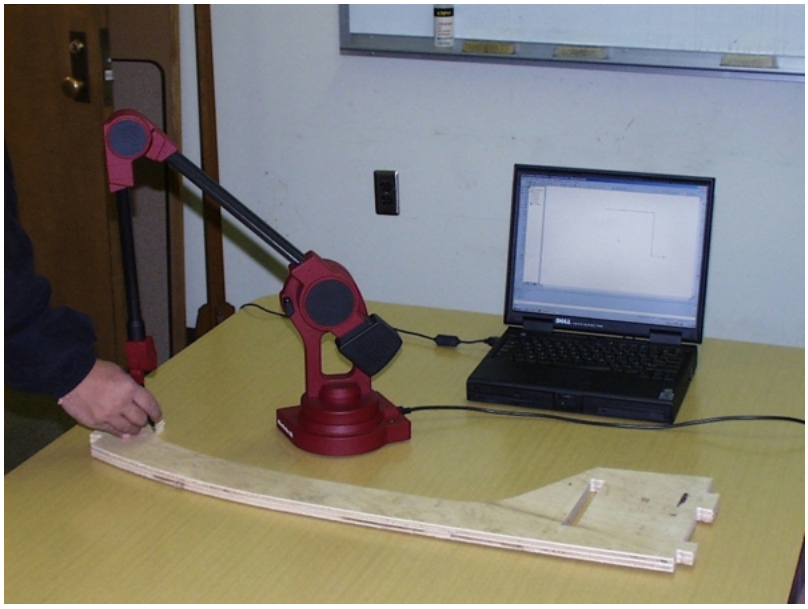


Figure 4.3: Digitization using MicroScribe

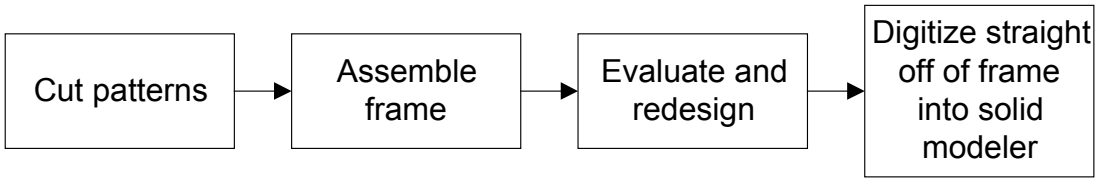


Figure 4.4: Reverse engineering process using articulating arm

The prototype frame parts are typically plain plywood parts without any features such as notches, tenons and mortises that are discussed in Chapter 3. The prototype frame typically uses butt joints, whereas the production frame uses interlocking joints with these features. These features have to be modeled on the parts that are used for building the production frame. As discussed in Chapter 3, various frame features have to be added at appropriate locations on the part. Therefore, the CAD operator needs to identify the locations of these features and then drag and drop the appropriate features from the library. The digitizing arm obtains the coordinates for locations of these features. The features from the feature library are then position and oriented in their appropriate locations.

4.4 Implementation

4.4.1 MicroScribe 3DL Digitizing Arm

In order to evaluate the relative merits of this alternative reverse engineering process, a MicroScribe 3DL digitizing arm was acquired (Figure 4.5). The MicroScribe enables the construction of detailed computer models by simply tracing the surface features of real physical objects. The MicroScribe 3DL has 5 "joints" about which it can rotate, and it has a 66" diameter work envelope when fully extended. The manufacturer claims an accuracy of 0.017”. The MicroScribe comes with an application programming interface (API) that allows the device to communicate with a computer via the serial port. At power-up, the default origin (0,0,0 point) is set directly below the center of the base joint axis, flush with the bottom of the base plate (Figure 4.5).

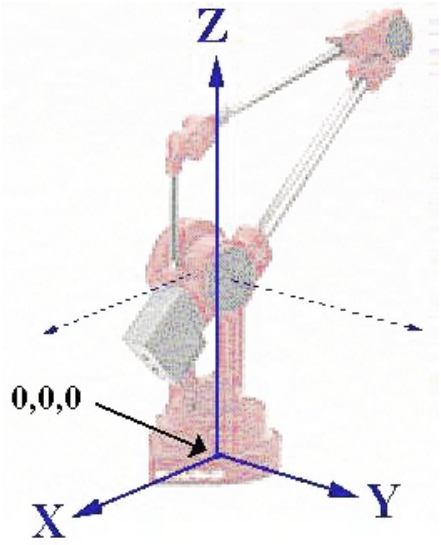


Figure 4.5: Default Coordinate Reference Frame

(Source: www.microscribe.com)

The arm must be configured before it is be used for digitizing. The coordinate system for the part is first defined by touching the probe to three different points on the surface of the part. This defines the X-Y plane. Next, a point is selected on the part that serves as the coordinate system origin. Lastly, two points are selected on the part that define the X and Y-axis directions. From this point forward, all points selected with the digitizing arm are given relative to this work piece coordinate system.

This in turn ensures that the MicroScribe coordinate system matches that of the CAD system.

4.4.2 Integration Software

Using the MicroScribe API, a Visual Basic computer program was written that allows points captured via the MicroScribe to be used directly within the SolidWorks CAD system. Table 4.1 shows the data required for every standard sketch entity that can be created using the MicroScribe and SolidWorks. Figure 4.5 shows the graphical interface that has been developed for selecting different types of entities. Sketch entities that have been implemented in the Visual Basic program include points, lines, arcs, circles, rectangles, parallelograms, and spline curves. The user selects the sketch entity that he/she would like to create. Rather than using a computer mouse or keyboard to pick the sketch points, the user touches the tip of the digitizing arm to the appropriate point(s) on the pattern and then presses a *foot pedal* to send the coordinates of that point to the computer. The Visual Basic program parses the X, Y, and Z coordinate values into the appropriate values for whichever sketch entity is selected. The program encompasses the conversion of all the values into meters as required by SolidWorks.

Sketch Entity	Required Data
Line	Start Point (X_{SP}, Y_{SP}); End Point (X_{EP}, Y_{EP})
Centerpoint Arc	Center Point (X_{CP}, Y_{CP}); Arc Start Point (X_{SP}, Y_{SP}); Arc End Point (X_{EP}, Y_{EP})
Tangent Arc	Arc Start Point (X_{SP}, Y_{SP}); Arc End Point (X_{EP}, Y_{EP})
Three Point Arc	Start Point (X_{SP}, Y_{SP}); End Point (X_{EP}, Y_{EP}); Point On Curve (X_{POC}, Y_{POC})
Circle	Center Point (X_{CP}, Y_{CP}); Point on Circumference (X_{POC}, Y_{POC})
Spline Curve	Point 1 (X_{P1}, Y_{P1}); Point 2 (X_{P2}, Y_{P2}); …; Point n (X_{Pn}, Y_{Pn})
Rectangle	Corner 1 (X_{C1}, Y_{C1}); Corner 2 (X_{C2}, Y_{C2})
Parallelogram	Corner 1 (X_{C1}, Y_{C1}); Corner 2 (X_{C2}, Y_{C2}); Opposing Side Point (X_{OS}, Y_{OS})

Table 4.1: Sketch Entities and Required Data Points

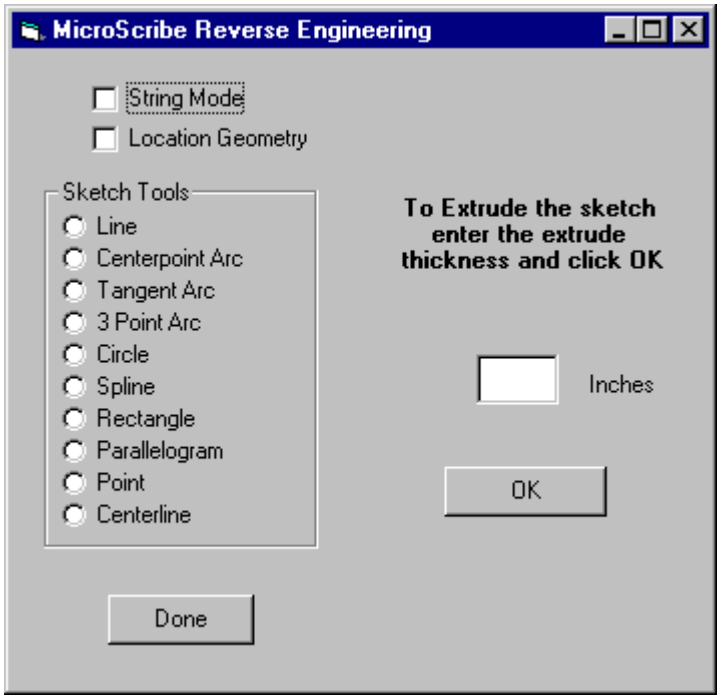


Figure 4.6: MicroScribe Reverse Engineering User Form

The *String Mode* option allows the user to create sketches that are continuous. When the string mode check box is selected, the user can generate a series of sketch entities wherein the starting point of one entity is the ending point of the previous entity. Entering the plywood thickness and clicking the OK button creates the solid object. The *Location Geometry* option sets the mode to generate construction geometry sketches. This is discussed in the following section.

4.4.3 Positioning Frame Features

At present, the SolidWorks API does not allow user defined features to be created using program function calls. They must be created manually. In order to allow the location of these features to be captured via the 3-D digitizing process, the user sketches reference entities that are subsequently used with feature library. Table 4.2 shows the locating entities that are typically used for each of the features described in Chapter 3. The user simply sketches a parallelogram or a point onto the 3-D part surface and then drags-and-drops a feature from the feature library at this position.

Feature	Required Sketch
Notch	Parallelogram
Tenon	Parallelogram
Mortise	Parallelogram
Location Mark	Point

Table 4.2: Positioning frame features

For example (Figure 4.7), if the user wants to insert a notch on the 3-D part model, a parallelogram is created using the MicroScribe. The notch feature is then dropped at this location and coincident relationships are applied to the sides of the notch and the parallelogram.

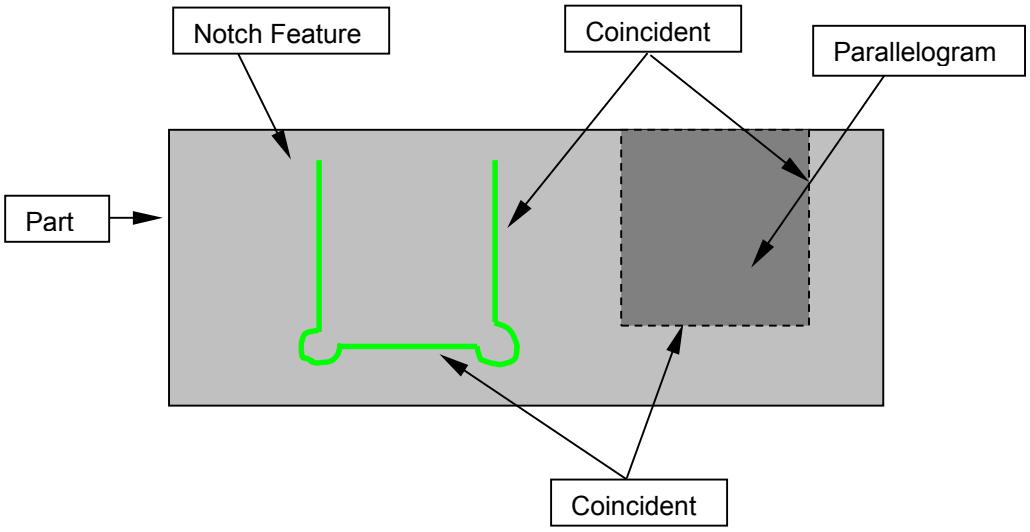


Figure 4.7: Positioning Frame Features

4.5 Testing and Results

In order to test the alternative digitizing approach, a sample love seat along with a complete set of documentation was obtained from a local upholstery manufacturer. The manufacturer also supplied DWG CAD files for each plywood component that they had created using a Summagraphics Summagrid III digitizing table. The love seat's frame contains 17 plywood parts (not counting multiples of the same part). The plywood parts were first digitized using the traditional process of tracing the part outline on paper and then 2-D digitizing using a digitizing table. Time required for tracing and digitizing each part was recorded. The parts were then digitized using the proposed method. The parts were digitized directly, without tracing them on paper, using the MicroScribe. Times for digitizing were recorded. Table 4.3 compares the time taken for digitizing the frame parts, using the traditional reverse engineering approach with time taken by the proposed 3-D digitizing method.

Table 4.3: Recorded Time for Digitizing

Part No.	Recorded time for tracing (minutes)	Recorded time for digitizing (minutes)	Recorded time for digitizing using proposed method (minutes)
1	0.35	1	0.6
2	0.5	1.45	0.7
3	0.2	0.65	0.3
4	0.9	3.5	2.8
5	1.01	3.75	1.5
6	1.95	5.65	2.3
7	0.65	3.35	2.05
8	0.3	4	1.2
9	0.35	0.6	0.6
10	0.75	1	0.5
11	0.35	0.35	0.2
12	0.4	1	0.6
13	0.5	0.75	0.5
14	0.75	1.65	1.3
15	0.45	0.9	0.5
16	0.85	2.8	1.35
17	0.2	0.4	0.2
Total	**10.46**	**32.8**	**17.2**

The total time using the traditional reverse engineering method was 43.26 minutes (10.46 + 32.8 = 43.26). In comparison, total time using the proposed 3-D digitizing method was 17.2 minutes

The experiment showed that the digitizing time was reduced approximately 60% for this real-world example. As described earlier, the operations of tracing part outlines on paper and cleaning up DXF files are completely eliminated. Furthermore, Frame features can be easily added or modified using this method.

Figure 4.8: Partial Solid Model of Love Seat

5 Creating a Mirror Image of an End-Assembly

The two end-assemblies in an upholstered furniture frame account for the majority of the components in that frame. When the frame is assembled in the CAD system, it should come as no surprise that the majority of time is devoted to specifying mating constraints for the end assemblies. Since the end assemblies are almost always symmetric, and the plywood components are symmetric from front to back (i.e. they are two-dimensional shapes that have thickness), the mating conditions that are specified for the left and right end assemblies are nearly identical. Given the time consuming nature of specifying assembly mating conditions, it seems reasonable that one should be able to simply mirror the left end assembly to create a right end assembly automatically. While most CAD packages allow users to mirror individual parts, they generally do not allow users to mirror assemblies. This is due to the fact that most assemblies, upholstery end assemblies notwithstanding, do not have the symmetry needed to enable a general-purpose algorithm to determine the correct mirroring of surfaces and mating conditions. Fortunately, such an algorithm can be developed for upholstery end assemblies. This chapter describes the underlying logic behind an algorithm that is designed to perform this mirroring.

5.1 Assemblies

Typically, an end assembly has 9 to 12 components, consisting of arm strips, side rails, front and back posts and cross strips. These components are joined to each other through either butt joints, notch-tenon joints, or mortise and tenon joints. In a CAD system, complex assemblies consisting of many components can be built. The components of an assembly can include both individual parts and other assemblies, called *sub-assemblies*. For most CAD assembly mating operations, the behavior of components is the same for both types. When a component (either an individual part or a sub-assembly) is placed in an assembly, the component file is linked to the assembly file. The assembly file typically contains the assembly mating conditions along with pointers to the individual component files. If a component is used in multiple assemblies, as is often the case, then any changes made to the component file will automatically

propagate to all of the relevant assembly files by virtue of the fact that the assembly files all contain pointers to the one original file. During the process of creating these assembly files, there are two different approaches that can be used - bottom-up design and top-down design.

5.1.1 Bottom-up Assembly Design

With this method, individual component parts are created in separate files and then inserted into the assembly file. The assembly mating relationships between these parts are then specified and saved in the assembly file. Bottom-up design is the preferred technique when the modeler is using previously constructed parts. An advantage of bottom-up design is that because components are designed independently, their relationships and regeneration behavior are simpler than in top-down design. Working bottom-up allows the modeler to focus on the individual parts. It is a good method to use when references that control the size or shape of parts with respect to each other are not required.

5.1.2 Top-down Assembly Design

With top-down assembly design, the modeler sees all parts in the assembly on the computer screen while he/she creates the geometry for each new part in the assembly. In this way, the geometry (points, lines, curves, etc.) from one part can be used to define the geometry from other parts. For example, a part can be inserted into an assembly, and then a fixture can be built by copying geometric entities from the original part. Working top-down, the fixture is created in the context of the original. Any changes to the original part automatically propagate to the fixture model.

5.1.3 Assembly Mating Relationships

When a component is in an assembly, it can be moved, rotated or fixed in its location. During the CAD assembly process, this allows users to drag, drop, and rotate each component into its approximate location and orientation within the final assembly before he/she begins specifying the assembly mating relationships between components. Generally speaking, assembly mating relationships are used to constrain the relative motion between two parts. Three assembly mating relationships are typically needed to fully constrain any given pair of components. One constraint restricts the side-to-side translation and rotation, one constraint restricts the up-down translation and rotation, and

the third constraint restricts the front-to-back translation and rotation. Each mating constraint creates a geometric relationship, such as coincidence, perpendicularity, or tangency, between faces, edges, planes and vertices in the component parts. Each mating relationship is valid for specific combinations of geometry. Table 5.1 shows the mating relationships that are typically supported between the various types of geometry in a CAD system. Note that each CAD system has its own terminology, and the terminology for a given system may or may not exactly match that which is shown in Table 5.1

	Plane	**Cylinder**	**Line**	**Cone**	**Point**	**Sphere**
Plane (planar face)	Coincident Perpendicular Parallel Distance Angle	--	--	--	--	--
Cylinder (cylindrical face)	Tangent	Concentric Tangent	--	--	--	--
Line (linear edge, axis)	Coincident Perpendicular Parallel Distance	Coincident Concentric Tangent	Coincident Perpendicular Parallel Distance Angle	--	--	--
Cone (conical face)	--	Concentric	Concentric	Coincident Concentric Distance	--	--
Point (origin, vertex)	Coincident Distance	Coincident Concentric	Coincident Distance	Concentric	Coincident Distance	--
Sphere (spherical face)	Distance Tangent	Concentric Tangent	Concentric Distance	Tangent	Concentric Distance	Concentric Distance Tangent

Table 5.1 : Mating Relationships

It is important to note that adding too many mates can cause the position of a component to be over defined. In other words, it is possible to have two mating conditions that attempt to restrict the same relative motion between a pair of components. This causes an error, since it is possible for the two constraints to attempt to place the components in different locations.

Assembling end assemblies is a very time consuming process. Each of the 9 to 12 components in a typical end assembly requires three assembly mating conditions. The resulting 27 – 36 mating conditions take an average modeler 30 or more minutes to specify depending on whether or not complex angles are involved. The procedure proposed in this chapter is designed to automatically generate a mirror image of an

existing end assembly as a separate file (i.e. if the right end assembly is loaded in the system, a left end-assembly is automatically created). This procedure has the potential to significantly reduce the time required to model a complete assembly.

The algorithm proposed and developed generates a mirror image of the side frame assembly in the CAD system as a separate file i.e. if the right end assembly is loaded in the system, left end-assembly will be created automatically when the program is executed and vice versa. This procedure reduces the time required to model the complete assembly. The process of assembling the opposite end-assembly is completely eliminated.

5.1.4 Assembling Parts in SolidWorks

In order to facilitate discussion of the mirroring algorithm, it is helpful to see an example of how a mirror would be accomplished manually. Figure 5.1 shows a two-piece assembly with a front post and arm strip. These two components have been assembled using mating relationships discussed earlier. The close-up figure shows the different faces and edges of these two parts that are used for mating. Figure 5.2 shows the user interface for adding mating relationships for assembling parts in SolidWorks.

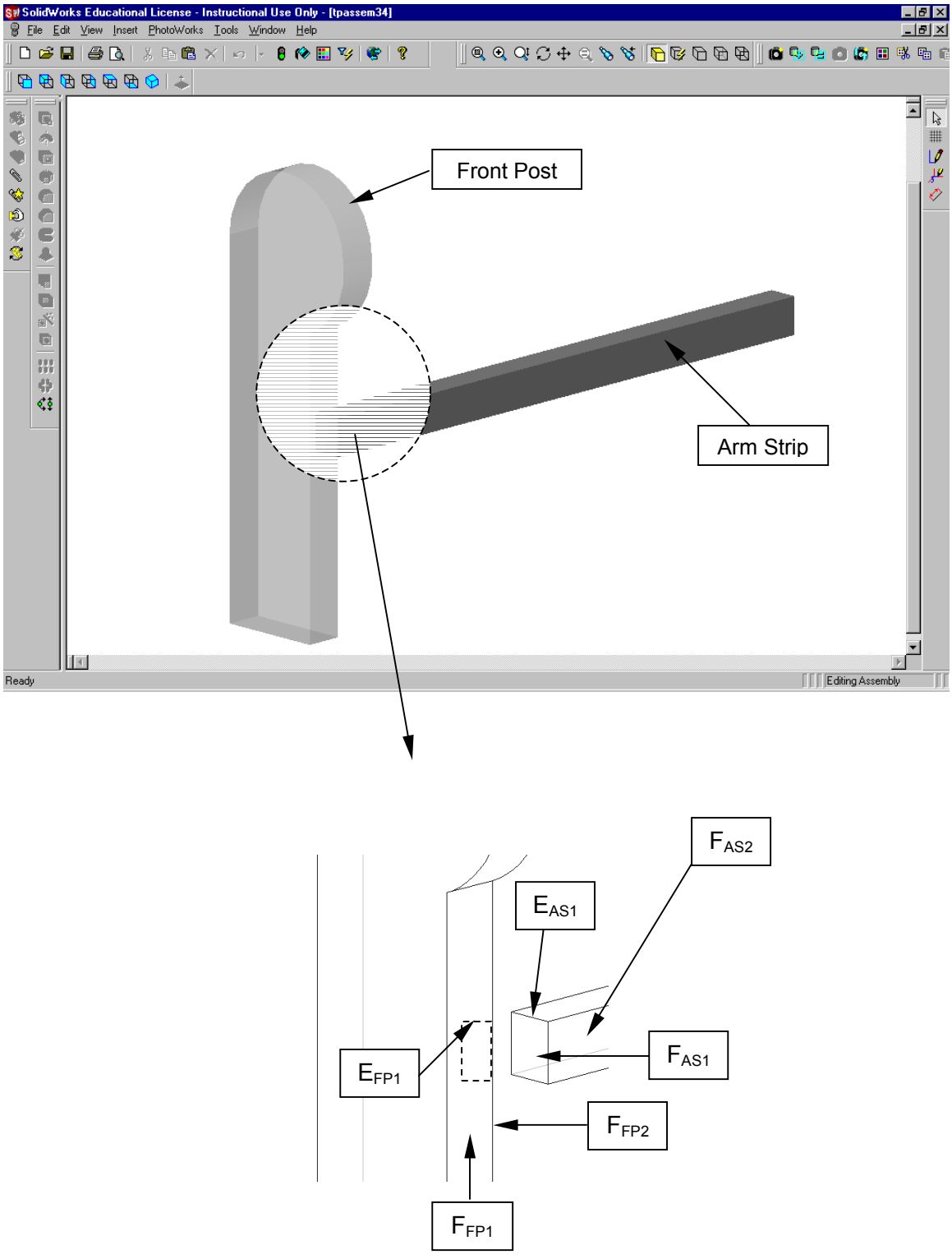


Figure 5.1: Mating Parts in SolidWorks

1. The arm strip face F_{AS1} is made coincident with the rear face F_{FP2} of the front post. This restricts front to back relative motion between these parts.
2. The arm strip face F_{AS2} is made coincident with the front post face F_{FP2}. This restricts side-to-side relative motion between these parts.
3. Location sketch edge E_{FP1} is made coincident with the arm strip edge E_{AS1}. This restricts top-to-bottom relative motion between the parts.

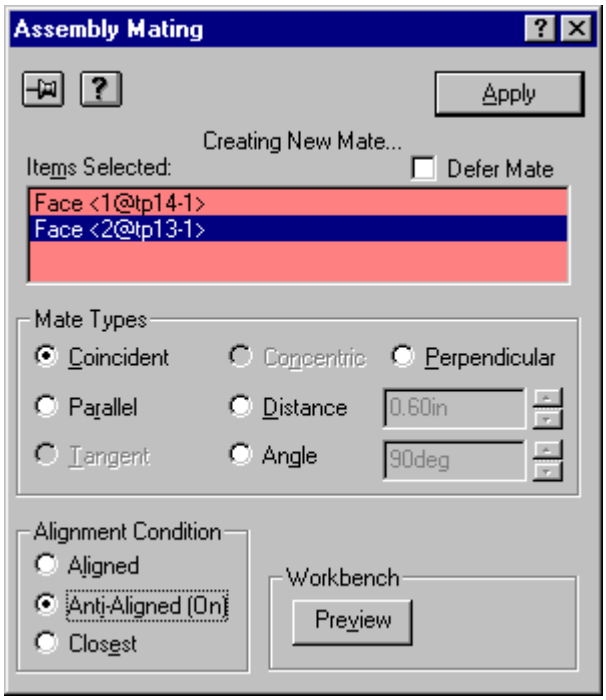


Figure 5.2: Assembly Mating Window of SolidWorks

Now, suppose that we wish to mirror the assembly. Each assembly mating constraint is examined,

- The same arm strip face F_{AS1} is used in the mirror, however, the opposite face on the front post is used.
- The same front post face F_{FP1} is used in the mirror, however, the face opposite to F_{AS2} of the arm strip is used.
- The edges used for mating in the mirror remain the same.

Figure 5.3 shows the original assembly and the mirror assembly. Based on this simple example, one can see that mirroring requires logic to be developed that identifies how the assembly mates are to be rewritten.

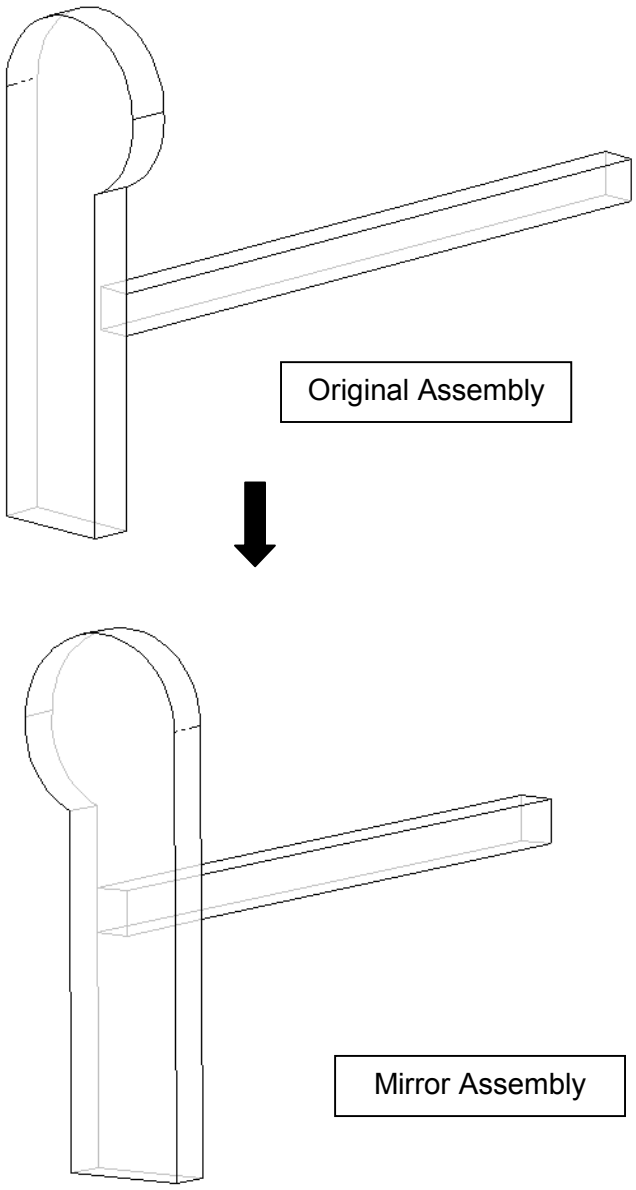


Figure 5.3: Mirror Image of an Assembly

5.2 Procedure for Mirroring Assembly Mating Conditions

Figure 5.4 gives a high level overview of the proposed procedure for mirroring an end assembly. The fundamental logic for creating a mirror image of the right end assembly involves making a copy of a right end assembly and then rewriting the assembly mating conditions for the copy so that a mirrored assembly results. The logic for how to rewrite the mirrored assembly mating conditions is detailed in the following sections.

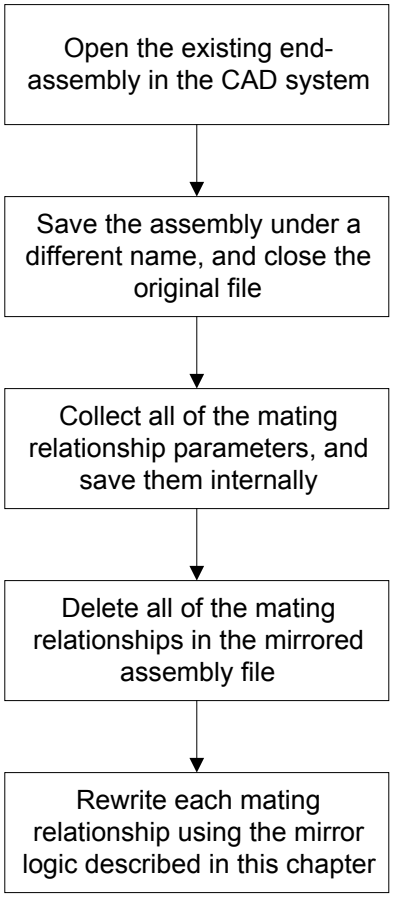


Figure 5.4: Overview of the algorithm for mirroring an end-assembly

5.2.1 Mating conditions involving front post entities

Figure 5.5 shows the algorithm for rewriting the assembly mating conditions in the mirrored assembly. Every mating condition constrains the relative motion between a pair of geometric entities. Since the mirroring logic depends on whether a particular geometric entity belongs to a front post or one of the front-to-back connecting members,

the algorithm checks to see which type of part a given geometric entity belongs to. For all mating conditions that involve the back face of the front post, the mirror mating condition should use the opposite (i.e. front) face of the front post. Likewise, mating conditions that involve edges belonging the back face of the front post should use the corresponding edge on the opposing (i.e. front) face in the mirror assembly. Lastly, any mating condition involving a vertex on the front post should use the corresponding vertex on the opposing (i.e. front) face in the mirror assembly.

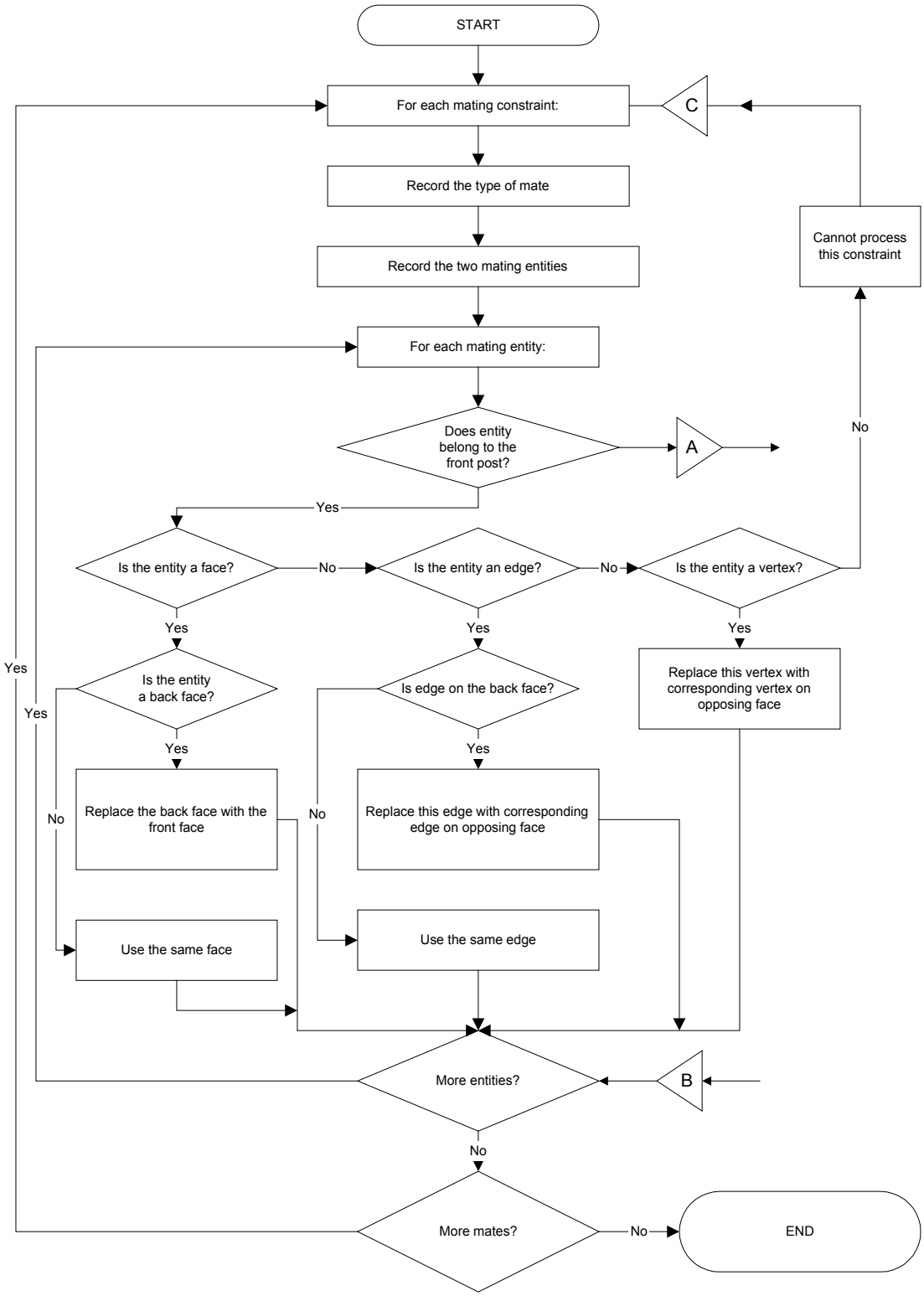


Figure 5.5: Mating conditions for front post entities

5.2.2 Mating conditions involving non-front post entities

Figure 5.6 shows the algorithm that determines how to mirror geometric entities involving all other components in the assembly. A component face that is not mated with the back or bottom face is to be replaced with the corresponding opposite face. All the edges and vertices that are mated are replaced by the corresponding opposite edges and vertices respectively.

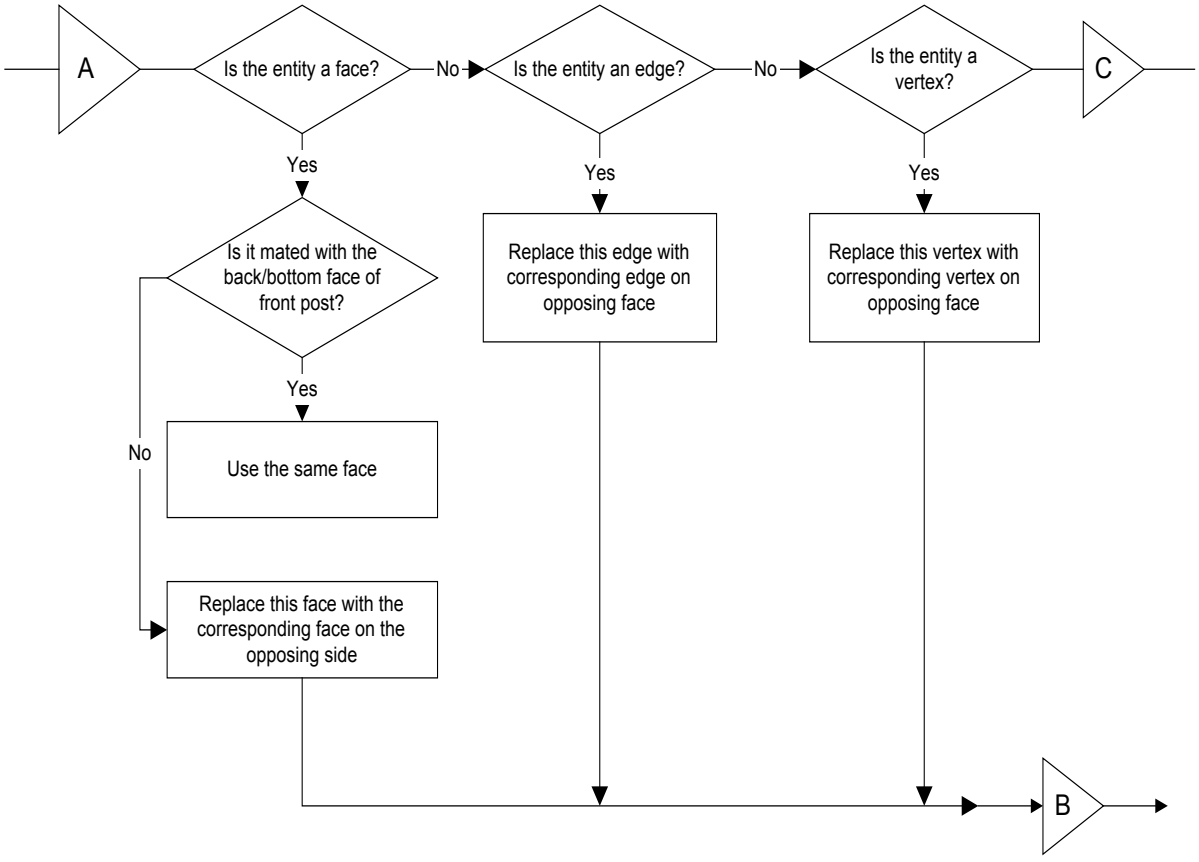


Figure 5.6: Mating conditions for component entities

5.3 Identifying opposing geometry

The algorithms presented in the previous section call for the opposing faces, edges and vertices to be identified. Sections 5.3.1 through 5.3.3 detail how this is done.

5.3.1 Selecting an opposing edge

Figure 5.7 shows the algorithm developed for selecting opposite edge. The algorithm starts by checking if the edge belongs to the back or front face of the front post. If it does not belong to the front post then it is checked for association with any other member. If it does not belong to any of the other members then the algorithm ends, as this edge does not required to be mirrored. The algorithm traverses the adjacent faces to this edge in order to find an edge with same length, equal X, Y coordinates and different Z coordinates for its start and end vertices. The edge that satisfies these conditions is the required corresponding opposite edge. A unique identity number is set for the current edge so that it is not selected while traversing the faces.

The three conditions that are satisfied by the opposing edge are:

1. The length is equal to the length of the input edge.
2. The X, Y coordinates of the start and end vertex of the opposing edge are equal to the X, Y coordinates of start and end vertex of the input edge.
3. The Z coordinate of the start and end vertex of opposing edge is different from the Z coordinate of the start and end vertices on the input edge.

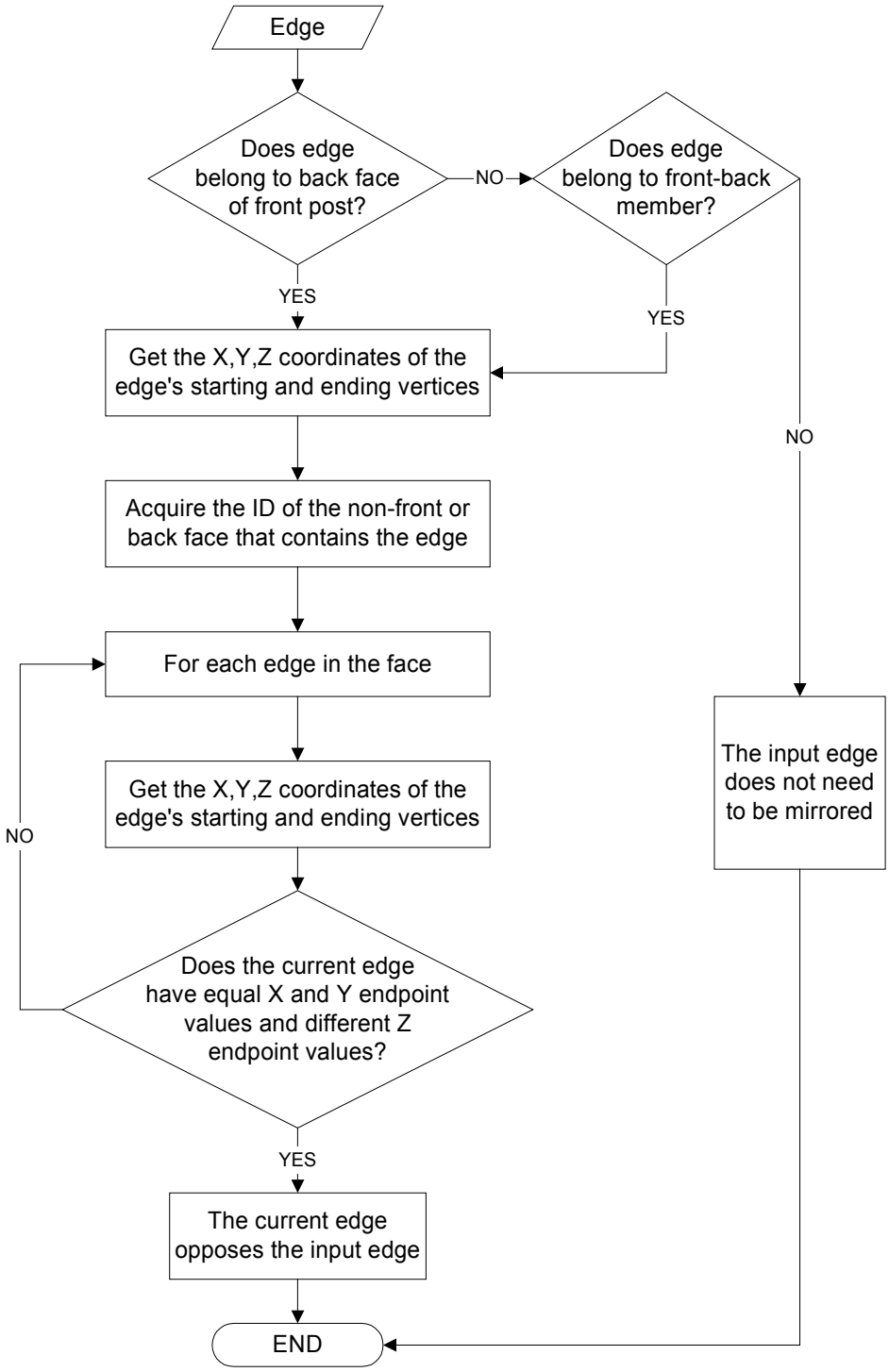


Figure 5.7: Algorithm for selecting the opposite edge

Figure 5.8 shows graphically how the algorithm works and how the opposite edge is selected.

1. The process of determining the opposite edge starts by selecting the two adjacent faces to the current edge. Let us consider that we want to find the opposite edge to edge E1 (blue). This opposite edge is located on the opposite face F2. Faces F1 and F3 are selected, as they are adjacent to edge E1.

2. Assume that face F1 (shaded) is selected and traversed first. Let use assume that the traversing starts with edge E2. Since it does not fulfill all the three requirements mentioned earlier, next edge E3 is selected.

3. All the edges of F1 are traversed and checked for required conditions. None of the edges on F1 satisfy the conditions. (E1 satisfies the conditions, but it is the input edge that edge we are considering to find the opposing edge. It is given a unique ID to avoid its selection)

4. Now, the next adjacent face F3 is traversed in a similar way. Edge E5 (red) satisfies all the three condition and it is selected as the opposite edge.

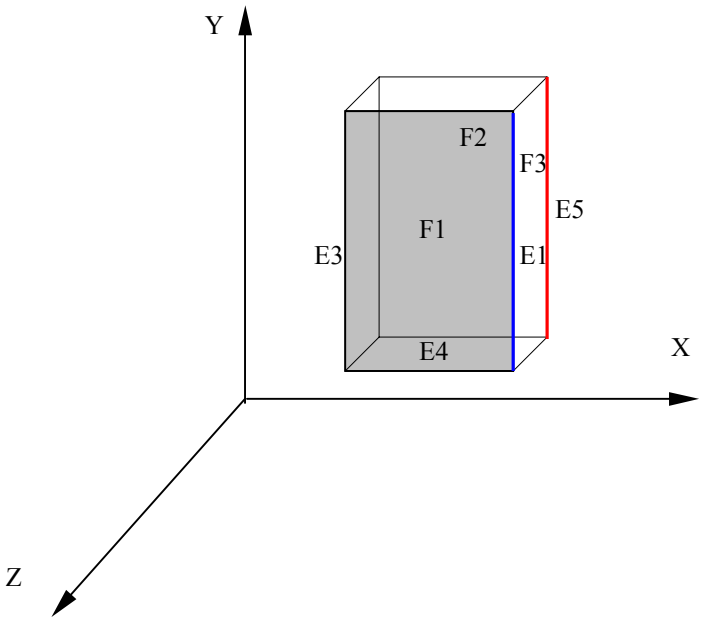


Figure 5.8: Selecting the Opposite Edge

5.3.2 Selecting the opposite face

Figure 5.9 gives an overview of the algorithm developed for selecting the opposite face for mirroring. The algorithm starts by checking if the input face is the back face of front post or any face of other members that is not mated with the back or bottom face of the front post. The face that had the same area and surface vector normal opposite to that of current face is the opposite face that is required for mirroring. For the front post, the front face is selected if the current face is its back face. For other components, the algorithm returns the corresponding opposite face.

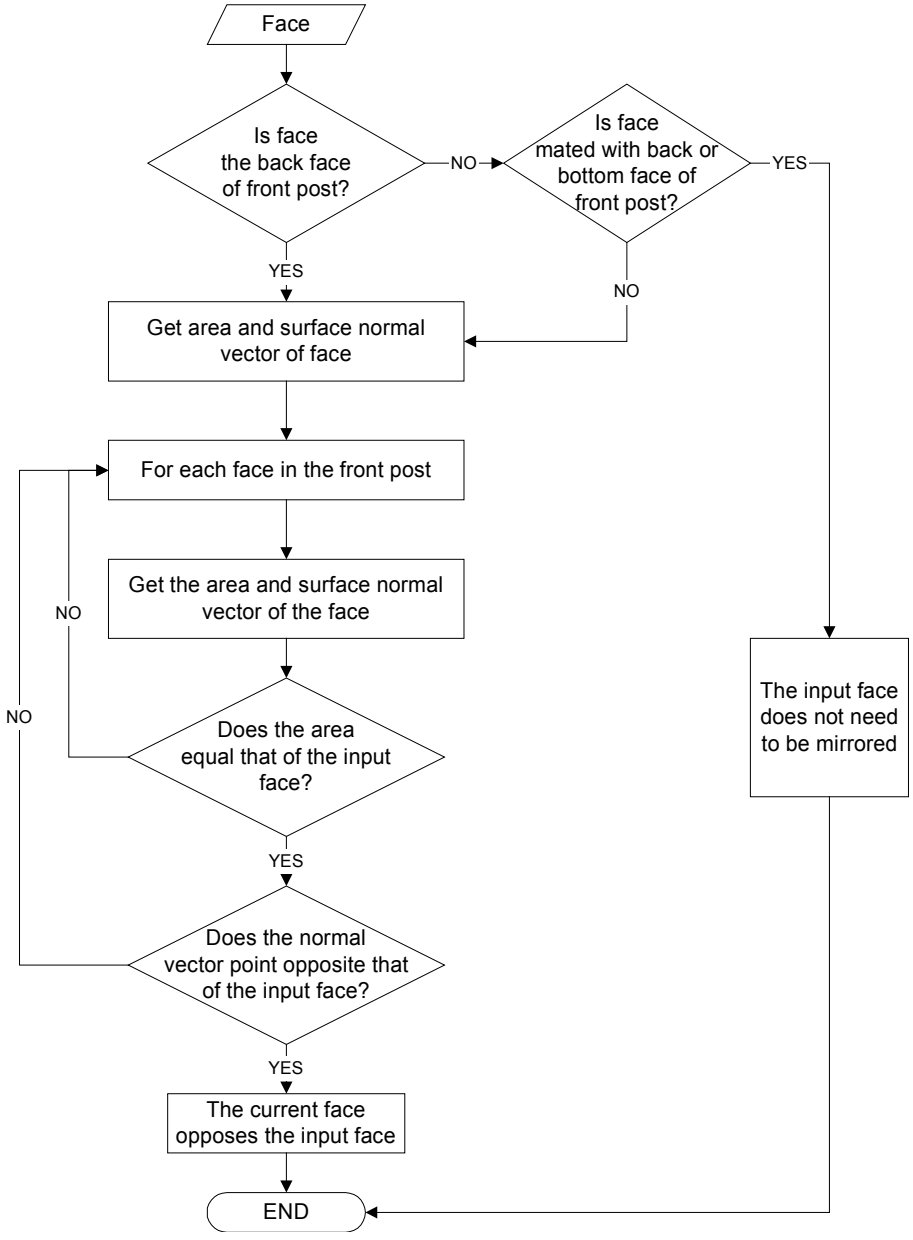


Figure 5.9: Algorithm for selecting opposite faces

Figure 5.10 describes the algorithm graphically.

To find the opposing face, two conditions need to be satisfied:

1. The area of that face should be equal to the area of input face.
2. The surface normal vector should be opposite to the surface normal vector of the input face.

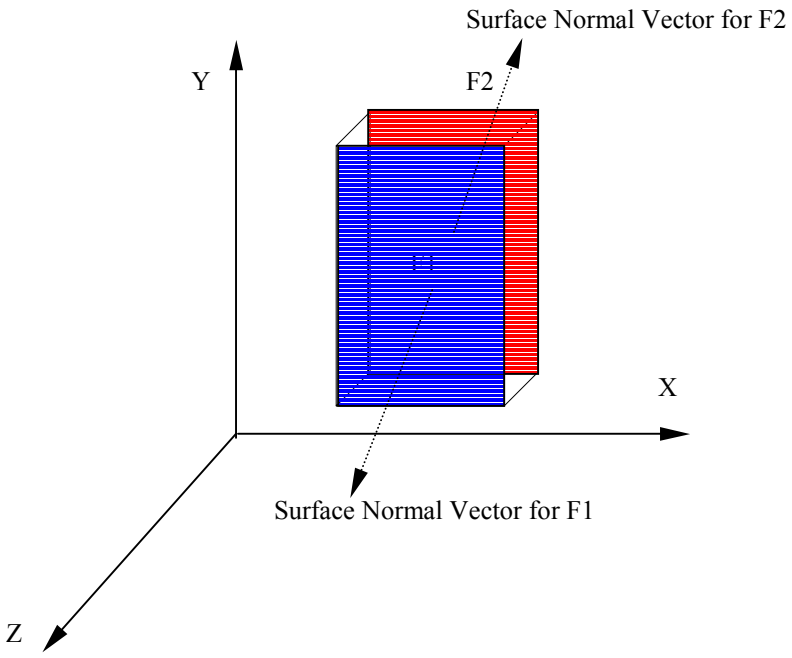


Figure 5.10: Selecting the Opposite Face

1. The algorithm traverses all the faces of the body. Let us assume that we need to find the opposing face to F1.

2. The algorithm starts traversing body for faces that satisfy the above two conditions. If a face does not satisfy both the conditions then next available face is selected.

3. The face F2 satisfies both the conditions. F2 has area equal to F1 and its surface normal vector is opposite to the surface normal vector of F1. Thus, face F2 is selected as the opposing face to F1.

5.3.3 Selecting the opposite vertex

Figure 5.11 is the algorithm for selecting opposite vertices. It starts out by determining the X, Y, Z coordinates of the current vertex. Then it searches all the faces of the component for a vertex that has equal X, Y coordinates and different Z coordinate. The vertex satisfying this condition is the mirror vertex.

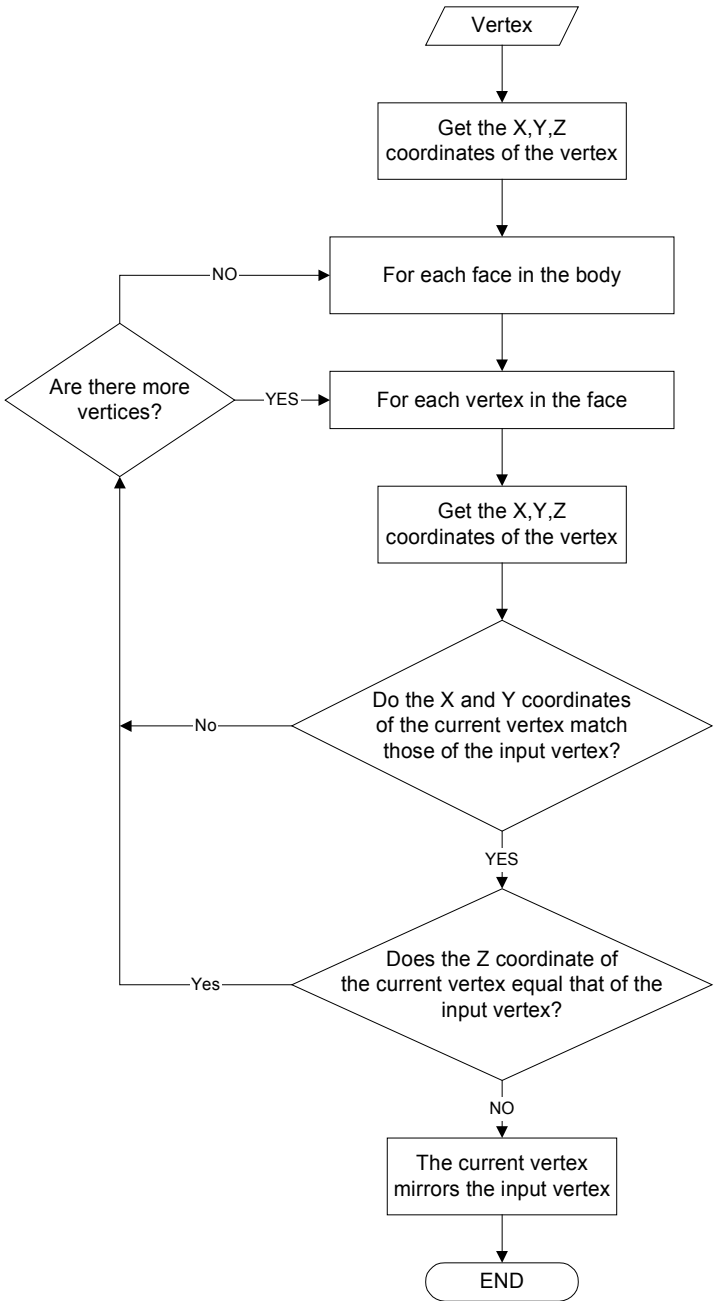


Figure 5.11: Algorithm for selecting opposite vertex

The following conditions need to be satisfied to find the opposite vertex:

1. The X and Y coordinates of the vertex should be equal to X and Y coordinates of input vertex
2. The Z coordinate of the vertex should not be equal to the Z coordinate of the input vertex.

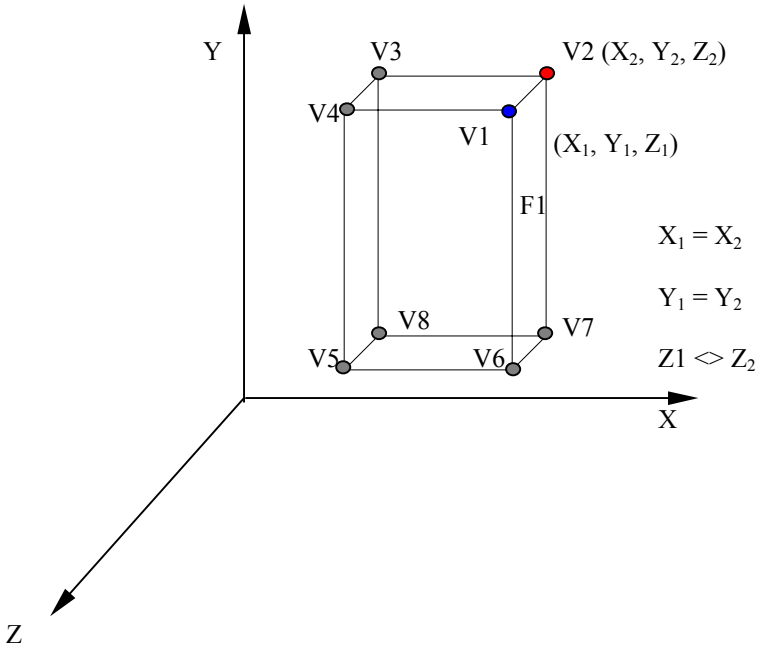


Figure 5.12: Selecting the Opposite Vertex

1. Let us assume that we need to find the vertex opposite to vertex V1. The algorithm starts by traversing all the faces of the body and their vertices.

2. Assuming that face F1 is traversed first, all its vertices are traversed and each vertex is checked for the conditions mentioned above. If a vertex does not satisfy both the conditions then the next vertex is selected. If the opposite vertex is not found after traversing all the vertices of one face then the next face is selected.

3. The only vertex that satisfies the two conditions mentioned above is vertex V2. The X, Y coordinates of V2 are equal to X, Y coordinates of V1, but the Z coordinate of V2 is not equal to Z coordinate of V1. Thus V2 is selected as the opposite vertex.

5.4 Testing and Results

The complete set of algorithm was developed, implemented and tested using SolidWorks CAD system and Visual Basic. SolidWorks API information and the Visual Basic code for selecting opposite face, edge and vertex are included in Appendices. The program was tested on different assemblies with different mating conditions. Figure 5.13 shows the user interface for this program. The user follows the steps given below to mirror the assembly,

1. The user has to select the front post of the end-assembly.
2. The user types the file name that he/she would like to save the mirror assembly file.
3. The user clicks OK to start the program.

Figure 5.13: User Interface to Mirroring Program

The program creates the mirror image of the assembly, and it is saved as per the file name entered by the user. The program saves the current file, as a copy, by the name entered by the user and then collects all the information about all the mating

relationships. These mating parameters are stored in a data structure for easy retrieval. The final step is to use the algorithms discussed in section 5.3 to add the mirrored mating relationships. Figure 5.14 shows the original assembly along with a mirror assembly of an end-assembly that was tested. Several end assemblies have been tested, and implementation properly mirrors each component.

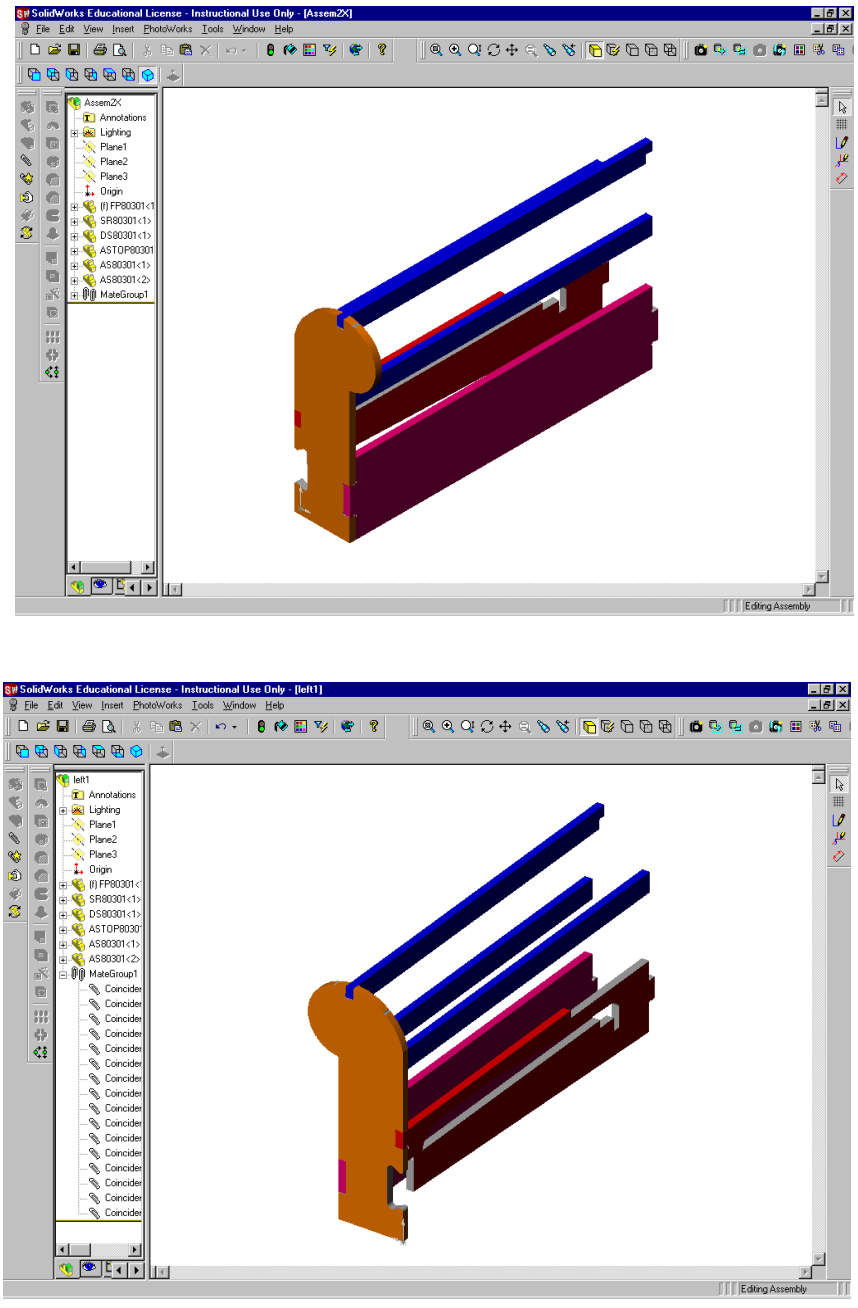


Figure 5.14: Original Assembly (Right-End) and Mirrored Assembly (Left-End)

5.5 Limitations

Although the algorithms appear to work as intended, there are certain limitations to this implementation in SolidWorks. They are as follows.

- The entities that are used for mating should be a face, edge or vertex of a component in the assembly. A sketched entity in the model space or on faces of the components is not considered as an entity for mating. The SolidWorks API does not support the function of returning an object handle for such entities.

- The base sketch for all the components should be on Plane 1 i.e. the default X-Y plane in SolidWorks. This is required for selecting opposite edges or vertices for mirroring. While searching for the corresponding opposite edges the algorithm checks for edges with X and Y coordinates which are equal to the X, Y coordinates of the start and end vertex of the original edge. Only the Z coordinates should be different to find the opposite edge. This condition is true for vertex coordinates as well. This is required as the algorithm searches for edges on its two adjacent faces or vertices on all faces and could return the opposite edge or vertex that is not corresponding to input edge or vertex respectively.

6 Conclusions and Recommendations

6.1 Conclusions

Computer-aided engineering holds tremendous potential for the furniture industry. This research has specifically addressed computer-aided engineering of upholstered furniture. The first part of thesis addressed the creation of a feature library for plywood upholstery frames. The use of a feature library considerably reduces modeling time as commonly used features are stored in the library and are used as required. This also increases consistency in the solid models. The feature geometry and procedure for modeling different features is documented. As the furniture industry shifts its focus from stick-built frames to plywood frames, such CAD applications are required for fast and accurate product development.

In Chapter 4, a 3-D digitizing method was proposed. The proposed method is faster and offers greater flexibility than conventional 2-D digitizing. This thesis has described the results of a novel 3D reverse engineering system that was developed. Software was written to interface a 3D-digitizing arm directly within a solid modeling CAD system. With this system it is possible to digitize straight off of the assembled frame. This eliminates many steps followed in the traditional reverse engineering method. The result is faster and more streamlined upholstery product development. In addition to reducing the time needed to reverse engineer plywood upholstery frames, it is anticipated that the 3D digitizing system will offer several additional benefits beyond those explored thus far as part of this research. These advantages are discussed in the section 6.2

In the third part of this thesis, an algorithm to create a mirror image of an end-assembly is developed and implemented. This program completely eliminates the cumbersome process of assembling the other end-assembly if one end has been assembled, i.e. it creates a mirror image of the assembly. This considerably reduces the total time to assemble a complete frame. The program was tested for its robustness on different assemblies with various mating relationships. Looking at the current competitive state of the furniture industry, companies that successfully implement 3-D CAD technologies for designing and manufacturing their products will have an edge over their competitors.

6.2 Future Work

One of the most exciting extensions of the work started with this thesis involves the possibility of automatic frame design based on designer's artistic surface models. More specifically:

- Foam scale models can be scanned using the 3-D digitizing approach. There is also the prospect of scanning sculptured surfaces from foam models with the ultimate goal of automatic frame design. The scan data would be used to produce a digital surface model as shown in Figure 6.15

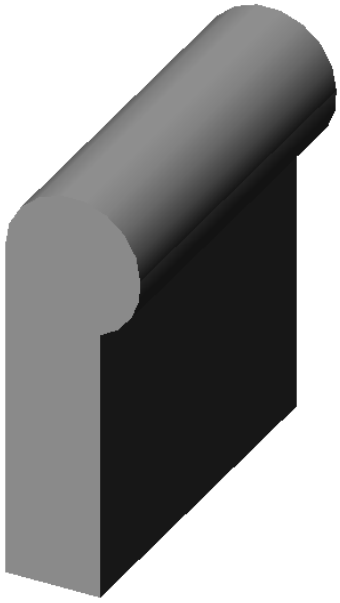


Figure 6.15: 3-D Solid Model of an End-Assembly

- Following scanning, the surface model features would be mapped to frame features. For example, the front post for this upholstered piece is given in Figure 6.2. It could be possible for a CAD system to generate the frame features by scanning the surface solid model. This paves way for generating the frame automatically if the surface model is generated using 3-D digitizing method.

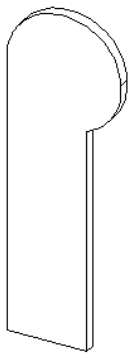


Figure 6.2: Front Post mapped from the surface model

- This also raises the possibility of generating surface models for in-store kiosks in which customers swap fabrics on virtual frames. While the technology to do this exists with packages such as ModaCAD, they still require the surface model to be created. This is quite difficult to do; hence the digitizing arm is a promising low-cost solution to this problem.

- *Photo-realistic computer renderings* – rather than asking a customer to imagine what a sofa will look like using a small fabric swatch, a retailer can now create a photo-realistic computer rendering of that sofa using systems such as ModaCAD.

Thus CAD technologies have very good potential in the furniture industry for design and manufacture of upholstered furniture. As furniture companies run on lean profits, they need such technologies for rapid and efficient product development and smaller factory-to-market time.

7 References

Carl A. Eckelman, “Product Engineering and Strength Design Manual for Furniture”, Purdue University

CSIL - Centro Studi Industria Leggera, Furniture Industry Research, 1999

Fred W. Zimmerman, “Upholstering Methods”, 1981, The Goodheart-Willcox Company Inc.

Hickory Springs, Springs, www.hickorysprings.com

Home Furnishings, www.homefurnish.com, Sofa Styles

James E. Brumbaugh, “Upholstering”, 1991, Third Edition, Maxwell Macmillan International

James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, “Computer Graphics, Principles and Practice”, Second Edition, 1996, Addison-Wesley Publishing Company, Inc.

Klaussner, “Realistic Furniture Catalog”, 1999, Klaussner Furniture Industries

Mahlon R. Saibel, “CIM & Its Applications to Upholstered Furniture”, 1992, CIFAC ’92 Proceedings

Marc Halpern, “Mapping CAD modeling to process”, Computer-Aided Engineering, Jan 1999, Vol. 18

MicroScribe Digitizing Arm, MicroScribe, www.microscribe.com

Prak Anco L and Thomas W. Myers, “Furniture Manufacturing Processes, 1997, Raleigh: Dept. of Industrial Engineering, North Carolina State University.

SigmaTEK, “SigmaNest”, slide 15, 1998, SigmaTEK Corporation.

SolidWorks Corporation, SolidWorks 99: User Guide, 1999, SolidWorks Corporation, Concord, MA

William F. Tierney, “Modern Upholstering Methods”, 1965, First Edition, McKnight & McKnight Publishing Company

8 Appendices

8.1 SolidWorks Application Programming Interface (API)

The SolidWorks API is an OLE programming interface to SolidWorks. The API contains hundreds of functions that can be called from Visual Basic, VBA (Excel, Access etc.), C, C++, or SolidWorks macro files. These functions give direct access to SolidWorks functionality, such as creating a line, extruding a boss, or verifying the parameters of a surface.

8.1.1 SolidWorks API Objects

The SolidWorks API interface uses an object-oriented approach. These functions are methods or properties that apply to an object. To call any of the methods or properties in API, the object needs to be obtained first. For example, to use *Curve:GetLength* method the *Curve* object is required.

There are many different ways to access objects within the API. For example, to obtain the *Curve* mentioned above,

- Obtain the Body object using *PartDoc:Body* and then get the *Face* object using *Body:GetFirstFace* and *Face:GetNextFace* methods. This *Face* object is used to get edges using *Face:GetEdges* method. This edge object is then used to get the underlying curve by calling *Edge:GetCurve*

- The edge object can be directly obtained from the current set of selected items using *SeletionMgr:GetSelectedObject3*

Figure 8.1shows the SolidWorks API objects. There are three main document types: Parts, Drawings and Assemblies. Each document type has its own API object (PartDoc, DrawingDoc and AssemblyDoc) with its own set of related functions. The API also has functions that are common to all document types. For example, printing, saving or determining the file name associated with a document are common operations.

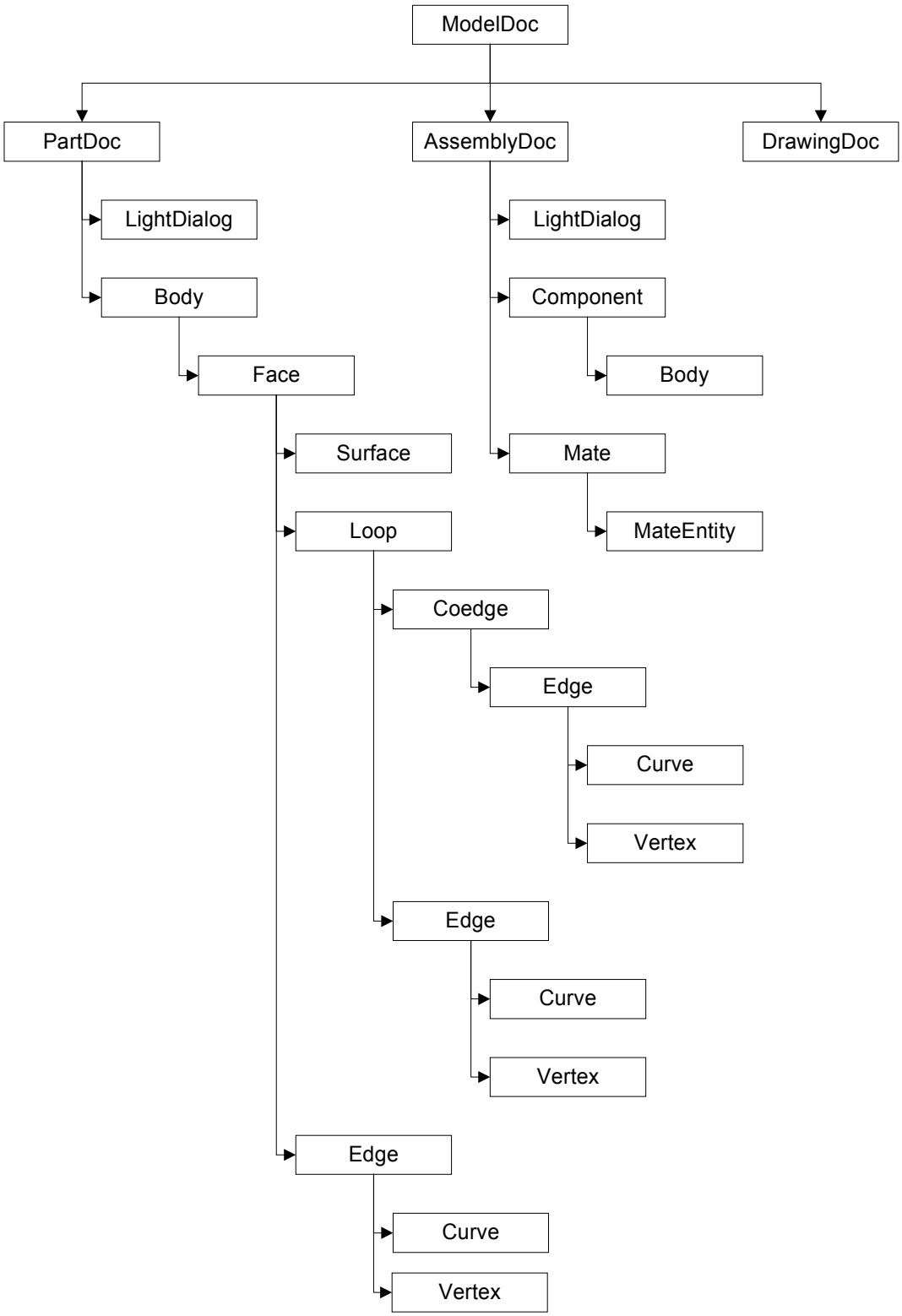


Figure 8.1: SolidWorks API Objects

8.2 Code for Selecting the Opposite Edge

```
'To check whether the edge is on front or back face
        If compname = frontpost_name Then
           adjfaces2 = entipointer.GetTwoAdjacentFaces()
           Set adjfaceobj = adjfaces2(0)
           Set adjfaceobj2 = adjfaces2(1)
           adjfaceID = adjfaceobj.GetFaceId()
           adjfaceID2 = adjfaceobj2.GetFaceId()
           If adjfaceID <> face1ID Or adjfaceID <> face2ID Then
              If adjfaceID2 <> face1ID Or adjfaceID2 <> face2ID Then
              GoTo Line3
              End If
           Else
           End If
        End If

         entipointer.SetId (27)          'Set ID to identify the edge
         edgeID = entipointer.GetId()
         Set curveobj = entipointer.GetCurve()
         curveparams = entipointer.GetCurveParams2()
         SX = Round(curveparams(0), 5)       'All coordinates rounded off to 5 decimal places
         SY = Round(curveparams(1), 5)
         SZ = Round(curveparams(2), 5)
         EX = Round(curveparams(3), 5)
         EY = Round(curveparams(4), 5)
         EZ = Round(curveparams(5), 5)

        'Get Edge Length
         edgelength = Round(Math.Sqr(((SX - EX) ^ 2) + ((SY - EY) ^ 2) + ((SZ - EZ) ^ 2)), 5)
         adjfaces = entipointer.GetTwoAdjacentFaces()
         For V = 0 To UBound(adjfaces)
            Set adjface = adjfaces(V)
            adjface_edges = adjface.GetEdges()
            For d = 0 To UBound(adjface_edges)
               Set edgepointer = adjface_edges(d)
               Set edgeobj = edgepointer.GetCurve
               newcurveparams = edgepointer.GetCurveParams2()
               newSX = Round(newcurveparams(0), 5)    'All coordinates rounded off to 5 decimal places
               newSY = Round(newcurveparams(1), 5)
               newSZ = Round(newcurveparams(2), 5)
               newEX = Round(newcurveparams(3), 5)
               newEY = Round(newcurveparams(4), 5)
               newEZ = Round(newcurveparams(5), 5)
               newedgelength = Round(Math.Sqr(((newSX - newEX) ^ 2) + ((newSY - newEY) ^ 2) +_
                                ((newSZ - newEZ) ^ 2)), 5)
               newedgeid = edgepointer.GetId()
               If newedgelength = edgelength Then
                  If newedgeid <> edgeID Then    'TO PREVENT SELECTING THE SAME EDGE
                     If ((newSX = SX) And (newSY = SY) And (newEX = EX) And (newEY = EY)) Then
                        Set entipointer = edgepointer
                        Exit For
                        Exit For
                        GoTo Line5
```

```
                    Else
                    End If
                Else
                End If
            Else
            End If
            Next d
        Next V
        GoTo Line5
```

8.3 Code for Selecting the Opposite Face

```
If compname = frontpost_name And entiname = "FACE" Then GoTo Line1 Else GoTo Line2

 Line1: ‘ Select the front face
            faceid = entipointer.GetFaceId()
            If faceid = face1ID Then
              Set entipointer = face2
              GoTo Line5
            ElseIf faceid = face2ID Then
               Set entipointer = face1
               GoTo Line5
            Else
            GoTo Line3
            End If

 Line2:
            ‘ Check whether the mating relationship is with back or bottom face or some other face
           If entiname = "EDGE" Then GoTo Line6
           If entiname = "VERTEX" Then GoTo Line8
           Set selectedobj = SelMgr.GetSelectedObject3(1)
           selectedtype = SelMgr.GetSelectedObjectType2(1)
           If selectedtype = 2 Then
             selectedId = selectedobj.GetFaceId()
             If selectedId = face1ID Or selectedId = face2ID Or selectedId = BottomFaceID Then
             GoTo Line3
             Else
             GoTo Line7
             End If
           Else
           End If
           GoTo Line3

Line7:
           ‘ Get the face area and normal vector
          facearea = entipointer.GetArea()
          roundval = Round(facearea, 3)
          normals = entipointer.Normal
          GoTo Line4

Line4:
```

```
            'Get face with same area and opposite normal vector
            newnormal(0, 0) = normals(0) * -1
            newnormal(1, 0) = normals(1) * -1
            newnormal(2, 0) = normals(2) * -1
            Set compbody = entipointer.GetBody()
            Set newface = compbody.GetFirstFace()
            Do While Not newface Is Nothing
               newarea = newface.GetArea()
               roundvalnew = Round(newarea, 3)
               facenormal = newface.Normal
               If roundvalnew = roundval Then

If facenormal(0) = newnormal(0, 0) Then
                     If facenormal(1) = newnormal(1, 0) Then
                        If facenormal(2) = newnormal(2, 0) Then
                        Set entipointer = newface
                        Exit Do
                        GoTo Line5
                        End If
                     End If
                  End If
               End If
            Set newface = newface.GetNextFace()
            Set facenormal = Nothing
            Loop
            GoTo Line5
```

8.4 Code for Selecting the Opposite Vertex

```
         vertexpoint = entipointer.GetPoint() 'Get the x,y,z coordinates of the current vertex
         VX = Round(vertexpoint(0), 5)
         VY = Round(vertexpoint(1), 5)
         VZ = Round(vertexpoint(2), 5)
         Set vertex_compobj = entipointer.GetComponent
         vertexcompname = vertex_compobj.Name
         Set vertexbody = vertex_compobj.GetBody() 'Get Component Body
         Set vertexface = vertexbody.GetFirstFace()
         Do While Not vertexface Is Nothing
           vertexedges = vertexface.GetEdges()
           For edgecount = 0 To UBound(vertexedges)
              Set Vedge = vertexedges(edgecount)
              Set edgecurveobj = Vedge.GetCurve()
              Vcurveparams = Vedge.GetCurveParams2() ' Get Start and End Vertex for the edge
              VSX = Round(Vcurveparams(0), 5) 'Start Vertex
              VSY = Round(Vcurveparams(1), 5)
              VSZ = Round(Vcurveparams(2), 5)
              VEX = Round(Vcurveparams(3), 5) 'End Vertex
              VEY = Round(Vcurveparams(4), 5)
              VEZ = Round(Vcurveparams(5), 5)
              If ((VX = VSX) And (VY = VSY) And (VZ <> VSZ)) Then
                 Set Vstartvertex = Vedge.GetStartVertex
                 Set entipointer = Vstartvertex
                 GoTo Line5
```

```
            ElseIf ((VX = VEX) And (VY = VEY) And (VZ <> VEZ)) Then
              Set Vendvertex = Vedge.GetEndVertex
              Set entipointer = Vendvertex
              GoTo Line5
            Else
            End If
          Next edgecount
        Set vertexface = vertexface.GetNextFace()
        Loop
      GoTo Line5
```