

Development of CAD Platform Independent Software for Automatic Grading of
Technical Drawings

Sanchit Sanjay Ingale

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State
University in partial fulfillment of the requirements for the degree of

Master of Science

In

Mechanical Engineering

Diana Bairaktarova

Christopher B. Williams

David A. Dillard

Robert L. West

August 11, 2017

Blacksburg, VA

Keywords: CAD, Automatic Grading, Design Education, Spatial Visualization,
Image Processing

Development of CAD Platform Independent Software for Automatic Grading of Technical Drawings

Sanchit Sanjay Ingale

ABSTRACT

Spatial visualization is the ability of an individual to visualize an object mentally and understand its spatial orientation. It plays an important role in engineering. There have been multiple works that show that spatial visualization skills can be improved with the right training.

Creating technical drawings requires mental manipulation and visual thinking. Strong spatial visualization abilities are required for such a mental process. To train students in this process and to improve their spatial skills, Professor Diana Bairaktarova conducted a class in spatial visualization for freshmen in engineering at Virginia Polytechnic Institute and State University. The class consisted of 169 students from different engineering disciplines. One of the major pedagogical techniques used in the class was to give students three assignments on drawing sectional views of mechanical objects with a pre-defined cutting plane. All three assignments had the same six mechanical objects to assess their improvement through the class. Students were not given feedback after completion of each assignment. They were asked to do the first assignment at the beginning of the course, the second assignment during the middle of the semester and the final assignment towards the end of the semester. The students were given instructions on how to draw the 2D sectional views. The assignments were then graded by the teaching assistants for the class and the improvement of the students through the semester was recorded. Assignments were graded manually by two different raters using a pre-defined grading rubric. An inter-rater reliability was established between the graders.

There were drawbacks to this process. It was extremely time consuming since there were more than 500 assignments to be graded by the teaching assistants. Also, to establish inter-rater reliability, the assignments had to be graded twice. The process would have been more efficient if there was a software that could automate the grading process. Also, this would eliminate the need to establish an inter-rater reliability. This research aims at developing a software for automatic grading of the technical drawings. The software gives students' feedback on the drawings describing their mistakes. This would give a more complete learning experience as the students

would get a better understanding of the internal details of the object with the help of the feedback they are getting. In addition, the software is independent of the CAD platform used to create the drawings. The instructor can also upload a batch of images that can be processed by the software at once. The grading rubric that was used for manual grading can be implemented in the software. The software uses Image processing and Computer Vision toolboxes in MATLAB which enables the comparison between the submitted technical drawing and the source (solution) drawing. The software is currently developed for simple geometries with less complicated features as it is being employed in a course where students are new to CAD environment. Students can also use this software as an interactive learning tool as they create 3D models and sectional views of mechanical objects.

The proposed software reduces the amount of effort put in by faculty on grading the assignments. It also gives students feedback on the drawings, making it an interactive tool which improves the learning experience. This software can be a powerful pedagogical tool to improve spatial visualization skills.

Development of CAD Platform Independent Software for Automatic Grading of Technical Drawings

Sanchit Sanjay Ingale

GENERAL AUDIENCE ABSTRACT

Every freshman engineering student needs to take a course in engineering graphics and technical drawings. These courses aim to improve the spatial visualization skills of students. Spatial Visualization is defined as 'the ability to mentally manipulate, rotate, twist, or invert a pictorially presented stimulus object' [1]. These skills play an important role in problem solving and learning capabilities of individuals.

Professor Diana Bairaktarova from Virginia Polytechnic Institute and State University conducted a course in Spatial Visualization during Fall 2016. The course aimed at introducing freshman students to Computer Aided Drafting (CAD), which is the use of computer software to create mechanical drawings of mechanical parts in their design stage. They were also taught to create sectional views of mechanical objects. Sectional view is a drawing which describes a hidden area or the interior part of a mechanical object. The class consisted of 169 students from various disciplines of engineering. Students were given assignments to create sectional views and 3D models throughout the semester using the same mechanical objects to assess the improvement in their spatial visualization skills. The teaching assistants had to manually grade more than 500 sets of assignments throughout the semester which was a time consuming process. In order to ease the grading process and improve the learning experience of students, a software for automatic grading of technical drawings was developed.

Programming language MATLAB was used to create the software. Students need to submit their assignment solutions in the form of an image. The software compares the submitted image with the solution images stored in the software and grades them according to a predefined grading scheme. It also provides a description of the mistakes created by students. The software can grade multiple files at once and store the results in an Excel sheet. This helps reducing the time required for grading the assignments and returning them to students. The software can be installed on any computer and does not require the system to have the programming software MATLAB installed. This software is independent of the CAD software used to create the drawings and thus stands out

from other existing softwares. The software has been currently developed for simple geometries with less complicated features. The software can be a powerful pedagogical tool to improve spatial visualization skills.

CONTENTS

Abstract	ii
General Audience Abstract	iv
List of figures	viii
List of Tables	xi
1. Spatial visualization in STEM Education	1
2. Improving spatial visualization skills	3
2.1 Improving spatial visualization skills in Engineering Design Education	4
2.2 Testing spatial visualization skills	5
3. Motivation	8
4. Related work done on Automatic Assessment of Technical Drawings	14
5. Implementation of the software	19
5.1 Difference of Corner, Edge and Blob	20
5.2 Harris Stephen's Algorithm	21
5.3 Development of Algorithm	23
5.4 Input required form the student	25
5.5 Graphic User Interface	25
5.6 Grading Algorithm	26
5.7 Creating source files	30
5.8 Algorithm	31
6. Testing	35
6.1 Testing the software on different CAD package	39
7. Batch Processing and Standalone Application	43

7.1 Batch Processing	43
7.2 Compiling the software into a standalone application	43
8. Applicability and Challenges faced in Development of the Software	45
9. Limitations	47
10. Future Work	48
11. Conclusion	49
References	51
Appendix A: Software Code	55
Appendix B: Instructions for creating 3D model and sectional view	74
Appendix C: Pilot study	92

List of Figures

Figure 1: Purdue spatial visualization Test: Visualization of Rotations (PSVT-R)	6
Figure 2: Mental Rotations Test	6
Figure 3: Mental Cutting Test	7
Figure 4: Differential Aptitude Test	7
Figure 5: Punch Plate	8
Figure 6: Shaft	9
Figure 7: Flange	9
Figure 8: Slide Block	9
Figure 9: Tee Slide	10
Figure 10: Tool Block	10
Figure 11: Correct Sectional View	11
Figure 12: Sectional view without hatching	11
Figure 13: Sectional view without features and hatching	12
Figure 14: Sectional view with incorrect cutting plane	12
Figure 15: Example for Drawing Compare plugin by Autodesk	15
Figure 16: Example of grading using RANSAC algorithm	16
Figure 17: Types of features in an image	19
Figure 18: Difference between Blob, Corner and Edge features	20
Figure 19: Corner feature with window	21
Figure 20: Tee Slide	23
Figure 21: Features detected using Harris Stephen's Algorithm	24

Figure 22: Two Images to be compared	24
Figure 23: Features matched between the two images using detectHarrisFeatures(I)	24
Figure 24: Graphic User Interface for the software	26
Figure 25: Correct sectional view	27
Figure 26: Sectional view without hatching	27
Figure 27: Sectional view without features and hatching	27
Figure 28: Wrong sectional view	28
Figure 29: Completely correct 3D model	28
Figure 30: 3D model with incorrect cutting plane	29
Figure 31: 3D model with incorrect cutting plane	29
Figure 32: 3D model with incorrect cutting plane	29
Figure 33: Source files for the object SlideBlock	30
Figure 34: Cropping the image using Bounding Box function	31
Figure 35: Flowchart for the software algorithm	32
Figure 36: SlideBlock file submitted into the software	33
Figure 37: Cropping the submitted image using Boundingbox function	33
Figure 38: Features detected in the submitted image	33
Figure 39: Comparing extracted features with source files	34
Figure 40: Graphic User Interface for the score of 5	35
Figure 41: Graphic User Interface for the score of 4	36
Figure 42: Graphic User Interface for the score of 3	37
Figure 43: Graphic User Interface for the score of 2	38
Figure 44: Graphic User Interface for the score of 1	39

Figure 45: Graphic User Interface for the score of 5 (Using CATIA)	40
Figure 46: Graphic User Interface for the score of 4 (Using CATIA)	40
Figure 47: Graphic User Interface for the score of 3 (Using CATIA)	41
Figure 48: Graphic User Interface for the score of 2 (Using CATIA)	41
Figure 49: Graphic User Interface for the score of 1 (Using CATIA)	42
Figure 50: Graphic User Interface for grading multiple 2D and 3D CAD files	43
Figure 51: Grades exported to an Excel file in batch processing	44
Figure 52: Process chart for the software	50

List of Tables

Table 1: Grading Rubric for Sectional View (Manual Grading)	11
Table 2: Overview of research done in automatic grading of technical drawings	17
Table 3: Grading Rubric for Sectional View	26
Table 4: Grading Rubric for 3D Drawings	27

Chapter 1: Spatial visualization in STEM Education

Spatial visualization is considered to be an important skillset for students in the STEM (Science, Technology, Engineering and Mathematics) field especially Engineering. It plays an important role in the problem solving ability and leaning capabilities of an individual. Spatial visualization is defined as ‘the ability to mentally manipulate, rotate, twist, or invert a pictorially presented stimulus object’[1]. It is one of the five components of spatial skills, the other four being spatial perception, mental rotations, spatial relations and spatial orientation. There has been some overlap with this classification. According to Tartre spatial visualization and spatial orientation are the two major components of spatial skills [2]. He also argues that spatial visualization is further divided into mental rotation and mental transformation. Mental rotation involves transformation of space to rotate the entire object while mental transformation involves only a part of the object being transformed. Spatial visualization plays an important role in understanding the concept of Engineering Graphics and can be improved with appropriate guidance and training.

In the past few decades, the development of computers and technology have resulted in various CAD software for 3D modeling, thus making spatial visualization skills to be more important for individuals in engineering fields. Studies have shown that these skills play an important role in success with working in a 3D design environment [3] and are also important in various other fields such as mathematics, physics, electrochemistry [4-6].

Spatial visualization skills play an important role in STEM education. Spatial Ability is shown to have an effect on the success of an individual in STEM education. Wai et. al. conducted an extensive research in order to assess the importance of spatial ability in STEM fields and concluded that it has a robust influence on STEM domains [7]. Another study by Wai et. al. accrued the research done over 50 years and data for more than 400,000 participants and showed that the correlation between spatial ability and success in STEM fields is very strong [8]. A chemistry student needs to have spatial visualization skills in order to visualize the structure of molecules for a particular compound. Mechanical Engineering students require spatial visualization skills to manipulate objects and create technical drawings for design and manufacturing. Geology students

use their spatial visualization skills to interpret the interiors of objects. Thus Visualization skills help individuals from various STEM fields and it is important to develop these skills.

Uttal and Cohen provide an excellent explanation of the relationship between spatial visualization ability and success in STEM fields in their work [9]. The work shows the importance of strong spatial abilities in choosing STEM fields for an individual. They also show the importance of spatial abilities decreases in STEM fields as expertise increases. The importance of spatial visualization skills being the motivation, there has been intensive research done on techniques to improve these skills and the next chapter discusses these techniques.

Chapter 2: Improving spatial visualization skills

A growing body of research suggests that spatial visualization skills can be improved with the right type of training. Recent study has shown that between 13-17% of students that are recently enrolled in a technical course require proper training for improving the spatial visualization skills as they scored less than 60% on their PSVT-R tests [10]. These skills can be improved with the right type of training. Mohler and Miller use a technique called mentored sketching to improve the spatial visualization skills of freshman engineering students [11]. The results obtained in the work revealed that these skills can be improved with properly planned and structured training. Work by Rovet claims that 12 minutes of instruction was roughly equivalent to 3 years of untutored development [12]. Many approaches have been used by various authors in Engineering Graphics literature to develop spatial visualization skills. Sorby and Wysocki proposes an approach for developing the spatial visualization skills of an individual through a series of exercise and use of a dedicated software [13]. Uttal and colleagues state that spatial visualization training plays an important role for the success of an individual in the field of mathematics, science and engineering [14].

The development of 3D CAD software in the past few decades have shown their influence on the course structure for spatial visualization courses. Barr et. al. explains the graphics design process using a parametric 3D CAD modelling software environment with examples from freshman students' exercises [15]. Students trained under 3D modelling software show better performance on spatial tests in comparison with the students who were trained using traditional classroom techniques [16]. Studies conducted by Sorby et. al. has shown that the spatial visualization skills of students can be improved with proper training [17]. They developed a hybrid course model where the tasks of rotation of objects and cross section of solids were accomplished using 3D CAD software. The class consisted of 2 hours of lecture and 2 hours of lab work for 3D modelling per week over the course of a semester. Use of 3D CAD modelling software enables an individual to view an object in different orientations and thus get a realistic view of the object. A 3D model has an advantage over the isometric view of an object, as the latter does not give the true pictorial representation of an object. A study has shown that students trained using 3D realistic views performed better in comparison to the students who were trained using isometric views [18].

Christou et. al. developed a library of software applications for enhancing the visualization and spatial thinking skills of students [19].

The use of Rapid Prototyping technology has shown to improve the spatial visualization skills of engineering students. Czapka et. al. conducted a study to assess the effect of physical 3D models on the spatial visualization skills of freshman engineering students. The results showed that students having access to physical 3D models of the drawings tend to easily visualize the object and this helped improve their spatial visualization skills over the period of the course [20]. Barr et. al. also describes the use of Rapid Prototyping in the Graphics design process in his work [15]. Many Engineering Graphics courses are now trying to include the entire design process in a single course. Bi and Mueller modified two courses to incorporate product lifecycle concept, solid modelling certification and rapid prototyping in the course [21]. They developed new exercises to improve the spatial visualization skills of students and also help easy transition from 2D wireframe to parametric 3D environment. Study conducted by Katsioloudis and Jovanovic show that using 3D printed models had a better improvement in the spatial visualization ability of students as compared to using 3D CAD models and 2D drawings [22].

2.1: Improving spatial visualization skills in Engineering Design Education

Every engineering student needs to take a class in Engineering Graphics during freshman year that involves manipulation of objects in a 2D and 3D environment. Developing and understanding technical drawings is an important skill for individuals in fields such as Mechanical Engineering, Civil Engineering and Architecture. Representing a 3-D object in a 2-D space as a perspective or an orthographic projection requires skills that are not naturally developed. Technical drawing introduces conventions and methodologies, which act as a platform for improvement of spatial skills. Scholarly work supports the view that technical drawing can in fact improve spatial skills [23-25]. Advances in technology have enabled the development of interactive CAD software interfaces for engineering students. This allows them to view and transform objects in 3D space as well as create orthographic views from the isometric view of an object. In contemporary engineering design education, the use of computer graphics and the application of web-based drawing approaches and tools help the development and enhancement of engineering students'

visualization abilities [26]. A study conducted by Rafi, Samsudin and Ismail concluded that computer mediated engineering drawing instructions helped students improve their spatial visualization skills [27]. They conducted a multi factorial quasi experimental design study consisting of three groups namely, Interactive Engineering Drawing Trainer (EDwgT), conventional instruction using printed materials enhanced with digital video clips, and conventional instruction using printed materials only. The EDwgT group performed better in terms of spatial visualization in comparison to the other two groups.

2.2: Testing spatial visualization skills:

According to Tartre, spatial visualization is classified into Mental Rotation and Mental Transformation as mentioned in Chapter 1. Various tests have been developed to assess spatial visualization skills in individuals. Two of the most prominent cognitive tests used to assess the Mental Rotation skills in individuals are Purdue Spatial Visualization Test: Visualization of Rotations (PSVT-R) and the Mental Rotations Test. The PSVT-R was developed by Guay in 1977 and consists of 30 questions related to rotation of 3D objects with increasing level of difficulty [28]. The students are given a criterion object and another view of the same object after it is rotated in space. Then they are provided with a second object and asked to indicate what their view of that object would be if the second object were rotated by the same amount in space. An example of PSVT:R is shown in Figure 1.

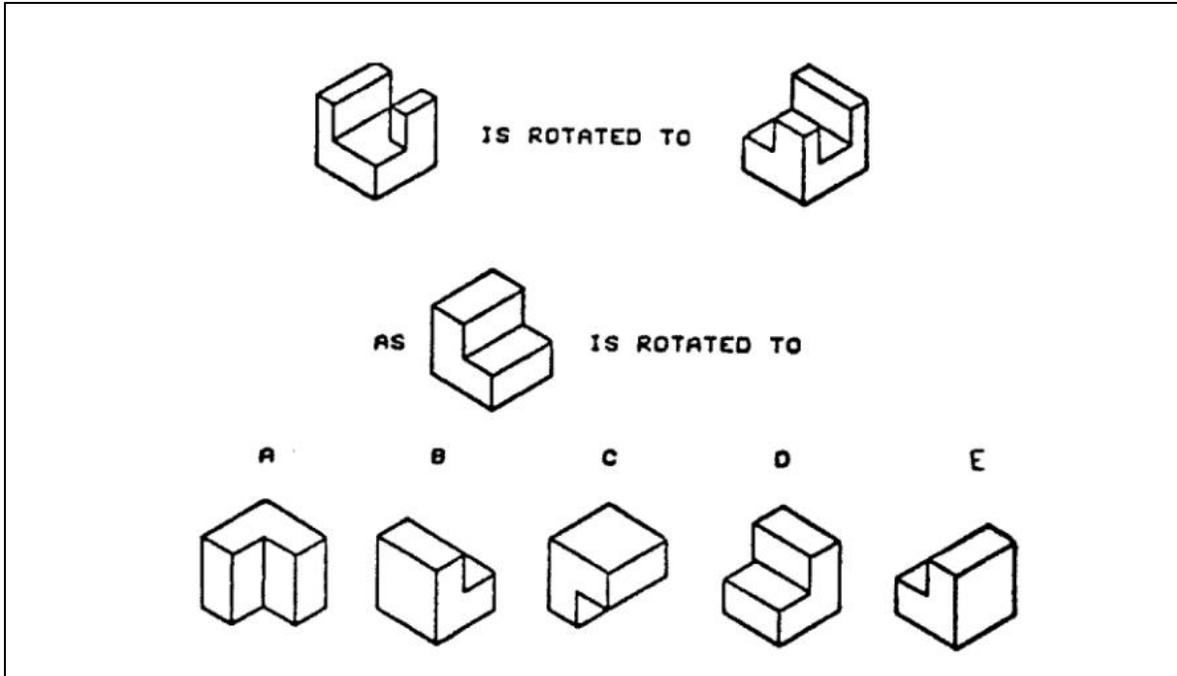


Figure 1. Purdue spatial visualization Test: Visualization of Rotations (PSVT-R) [30]

The Mental Rotation Test (MRT) was developed by Vandenberg and Kuse in 1978 and involves rotation of 3D objects by specific amounts of degrees around the axes [29]. The test consists of 20 items where students are shown a criterion figure on the left and asked to identify which two of four given choices represent the same object after rotation in space. An example of MRT is given in Figure 2.

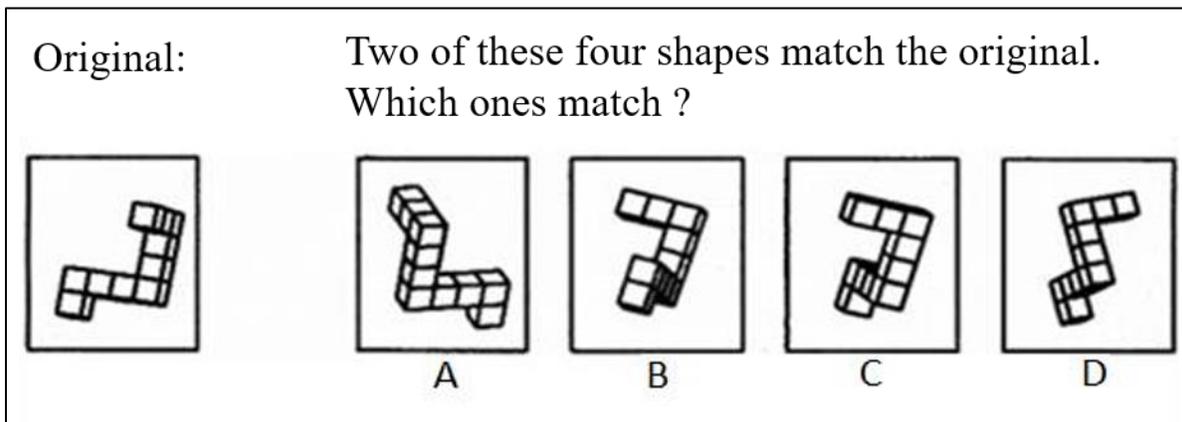


Figure 2. Mental Rotations Test [30]

Mental Transformation skills in individuals are assessed using the Mental Cutting Test (MCT) and the Differential Aptitude Test (DAT:SR). The MCT was developed by College Entrance Examination Board (CEEB) in 1939 and consists of 25 questions [30]. The students are given a criterion object which is to be cut using the given cutting plane. They are then asked which of the given options represents the object after cutting. Figure 3 shows an example of the Mental Cutting Test.

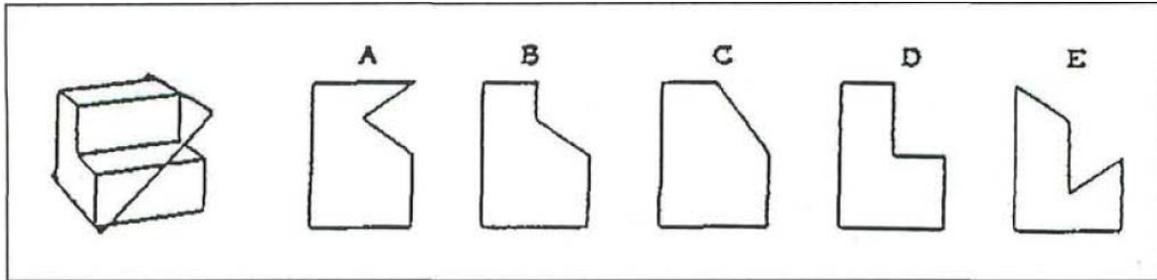


Figure 3. Mental Cutting Test [30]

The DAT:SR consists of 50 questions. In this test the students are given a 2 dimensional pattern of an object and asked which of the given alternatives represent the object after the given pattern is folded [31]. An example of DAT:SR is shown in Figure 4.

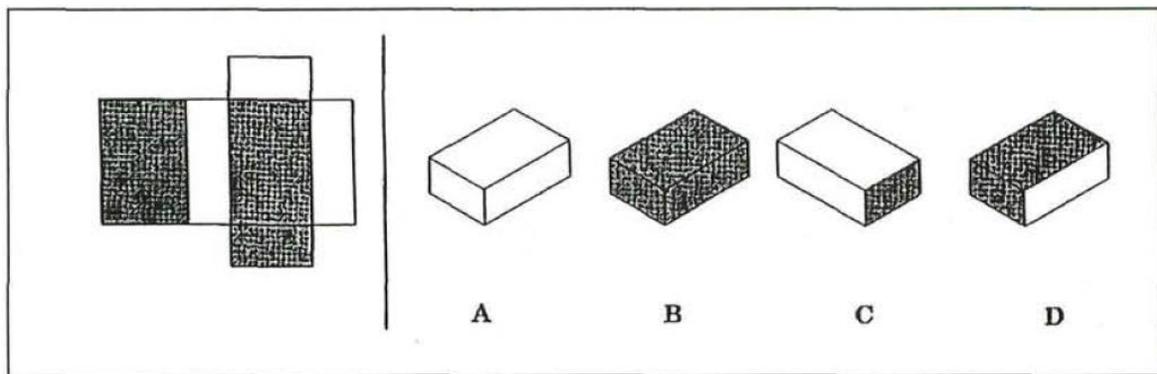


Figure 4. Differential Aptitude Test [30]

Chapter 3: Motivation

To enhance students' spatial reasoning, Professor Diana Bairaktarova from the Department of Engineering Education conducted a class in spatial visualization at Virginia Polytechnic Institute and State University during Fall 2016. The class consisted of 169 freshman engineering students. The course goals aimed at improving first-year engineering students' spatial visualization skills through instruction on creating sectional views of mechanical objects, reading technical drawings, and introduction to CAD environment. During the semester students completed in-class and out-of-class assignments on sectional views of objects. In addition, throughout the semester, students' performance on developing sectional views was assessed using six different mechanical objects with varied complexity. In order to assess improvement in the performance of students, the assignment consisted of same mechanical objects. They were asked to do the first assignment at the beginning of the course, the second assignment during the middle of the semester and the final assignment towards the end of the semester. The six mechanical objects included in the assignments are shown in Figures 5 to 10:

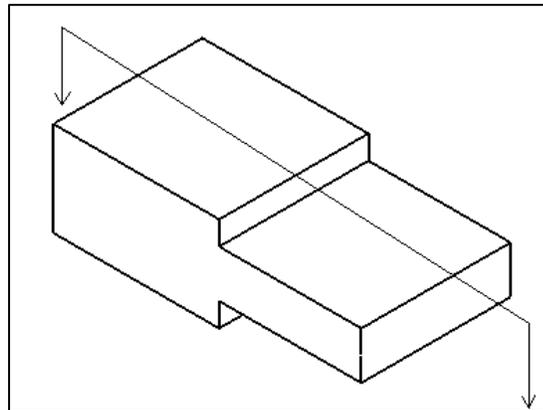


Figure 5. Punch Plate

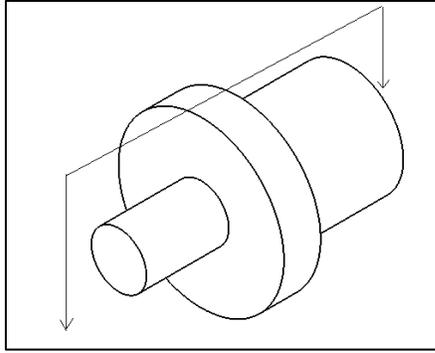


Figure 6. Shaft

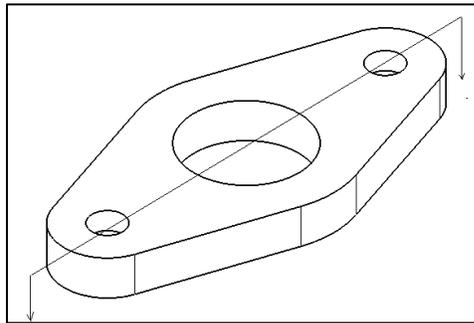


Figure 7. Flange

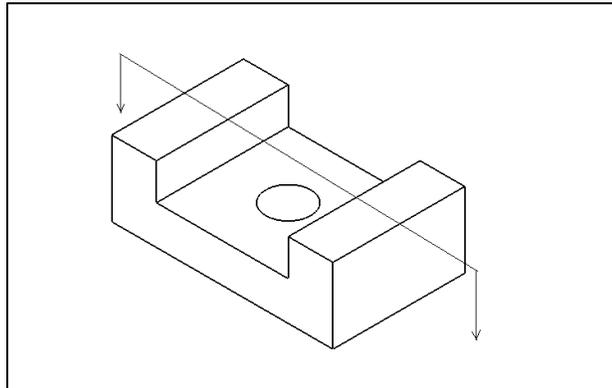


Figure 8. Slide Block

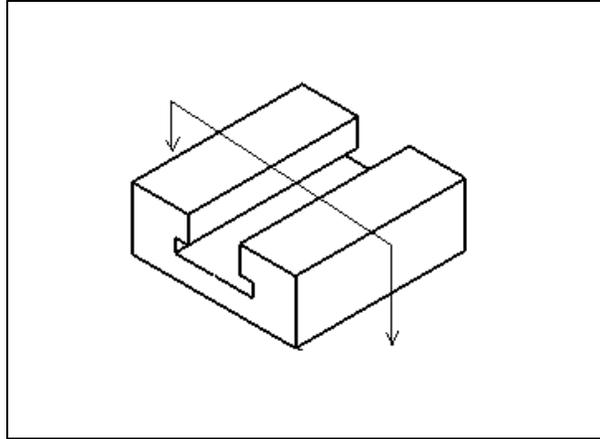


Figure 9. Tee Slide

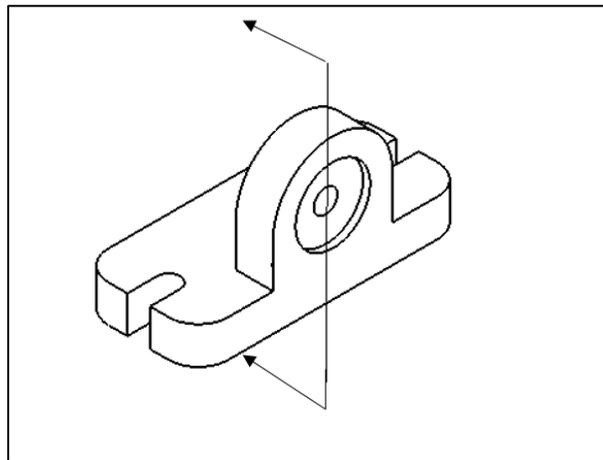


Figure 10. Tool Block

The students were asked to create 3D drawings of the objects and the sectional views using the cutting planes as shown in Figures 5 to 10 above. During the course, students were provided with detailed instructions on how to draw the 3D solids and the 2D sectional views of these solids. Initially, students were asked to create the sectional views manually on paper. The grading rubric used for the assignments is given in Table 1 with examples of grading in Figure 11 to 14.

TABLE 1: Grading Rubric for Sectional View

Description	Grade
Completely Correct Sectional View	5
Sectional View with missing Hatching OR Features	4
Sectional View with missing Hatching AND Features	3
Sectional View with incorrect cutting plane	2
Completely incorrect drawing	1

Consider an example of Tee Slide object as shown on Figure (5). 5 points are awarded to the student for a completely correct drawing as shown in Figure 11.

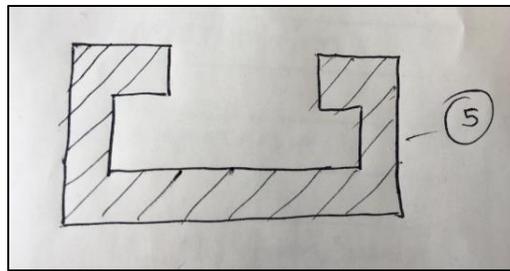


Figure 11: Correct sectional view

A student is awarded 4 points with a penalty of 1 point if the sectional view is without hatching as shown in Figure 12.

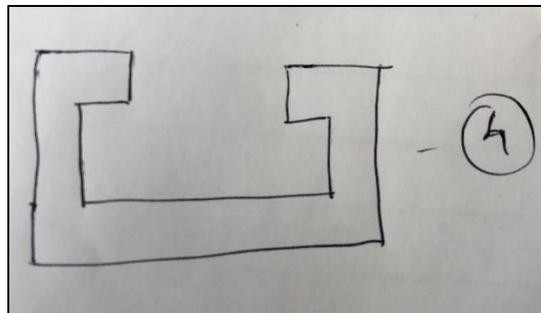


Figure 12: Sectional view without hatching

2 penalty points and a score of 3 is awarded for a sectional view that has missing features and hatching. An example of this is shown in Figure 13.

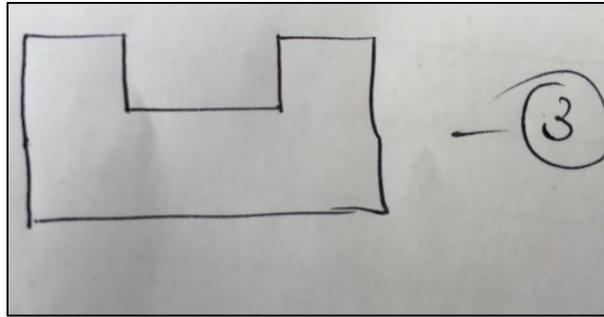


Figure 13: Sectional view without features and hatching

If a student submits a sectional view with an incorrect cutting plane as shown in Figure 14, 2 points are awarded with a penalty of 3 points. If a student submits a completely wrong drawing, a score of 1 point is given for the effort put in to create the drawing.

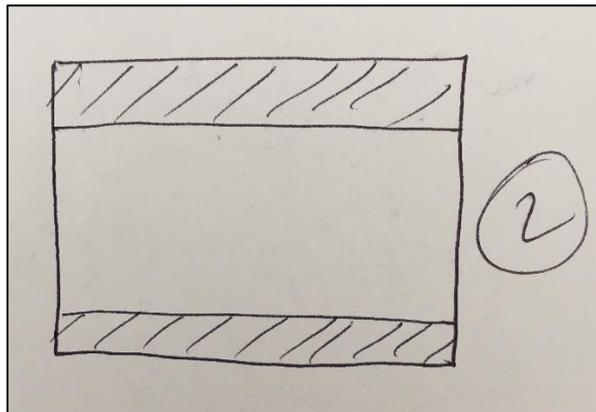


Figure 14: Sectional view with incorrect cutting plane

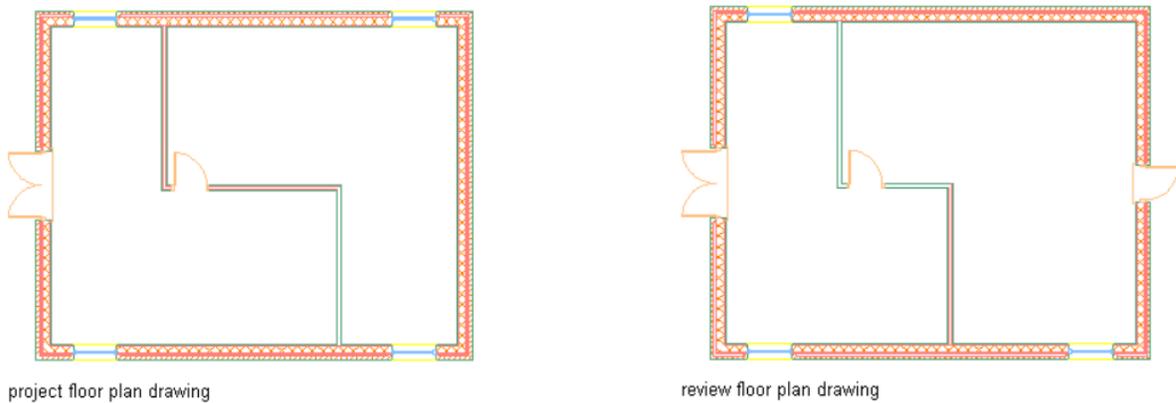
After manual drawing, students were trained on CAD and asked to develop 3D models and sectional views of the same solids using CAD. They were provided with specific instructions to create the models and sectional views. The instructions contain a step-by-step guide on how to develop 3D CAD models. Similar instructions were provided for the sectional views. These instructions have been included in Appendix B. During the course duration, the teaching assistants had to grade more than 500 sets of assignments in total, which was extremely time consuming. In addition, an inter-rater reliability had to be established, which meant that the assignments had to be graded twice according to the grading rubric created (Table 1). The grading rubric was used by both teaching assistants.

Every engineering student needs to take an Engineering Drawing or spatial visualization course during their freshman year. As a result, these classes have large number of students with a large student to teacher ratio. The number of assignments to be graded increases with the number of students. This makes manual grading process time consuming and time required to return graded assignments to the students increases. Motivated to overcome these drawbacks and to enhance students' learning experience, the software was developed for automated grading of technical drawings. In addition, it helps in eliminating the possibility of human error related to manual grading of drawings. The software is currently developed for simple geometries with less complicated features as it is being employed in a course where students are new to CAD environment. Students can also use this software as an interactive learning tool as they create 3D models and sectional views of mechanical objects. Further, the software differentiates itself from the existing softwares as it is completely independent of the CAD platform used by the students to create the drawings. Various online courses in Engineering Graphics are now available online and these courses require an automatic grading software. The work done in this thesis can be modified to employ it in these courses. Finally, it gives feedback to the students on their drawing and the nature of the feedback can be determined by the instructor.

Chapter 4: Related work done on Automatic Assessment of Technical Drawings

Drawings

Various software have been developed to compare technical drawings and highlight the differences. Autodesk have developed a plug-in called 'Drawing Compare' in AutoCAD Architecture to compare two drawings [32]. It was developed using Application Program Interface (API) programming. The plug-in works, in simple terms, as if a drawing on a tracing paper is placed on the original or correct drawing and these two drawings are then compared. The differences in the two drawings are then highlighted using different colors as shown in the example in Figure 1 below. The Drawing Compare plug-in although useful in comparing drawings, has some disadvantages. The plug-in shows the differences between two drawings temporarily, and we cannot save the comparison results. Further, the plug-in can only be used for drawings having the same scale. It does not work for drawings with variable scale and different orientations. An example of Drawing Compare plug-in is shown in Figure 15.



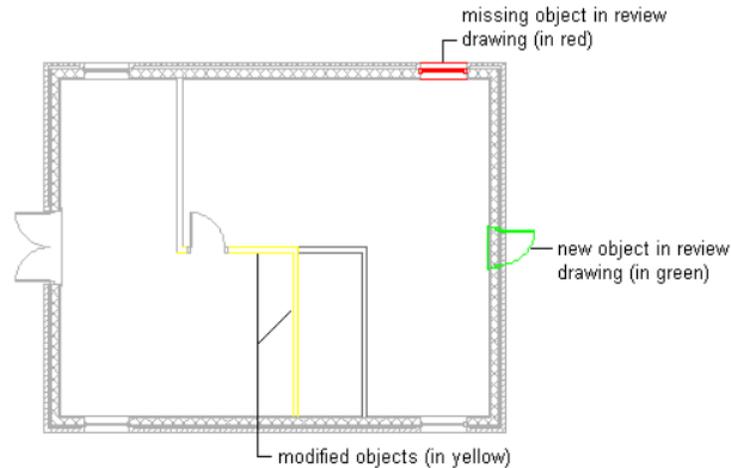


Figure 15: Example for Drawing Compare plugin by Autodesk[32]

Kwon and McMains developed an automatic grading system for multi-view engineering drawings. They built the application using the random sample consensus (RANSAC) algorithm, estimating the affine transformations between the two individual drawings [33]. RANSAC is an algorithm used to estimate parameters of a mathematical model from a set of data. In this work, the submitted file is compared with the solution file by performing various 2D operations on the submitted drawings. These transformations include translating the drawing in the x-y plane, scaling the drawing or rotating the drawing. These transformations are performed until the drawing matches the source drawing. These transformations are then highlighted using different colors. Students' .dxf drawing files from AutoCAD are uploaded into the software using MATLAB and are compared with the solution drawing file. The software performed better in comparison to human grading. An example of the RANSAC algorithm is shown in Figure 16 below:

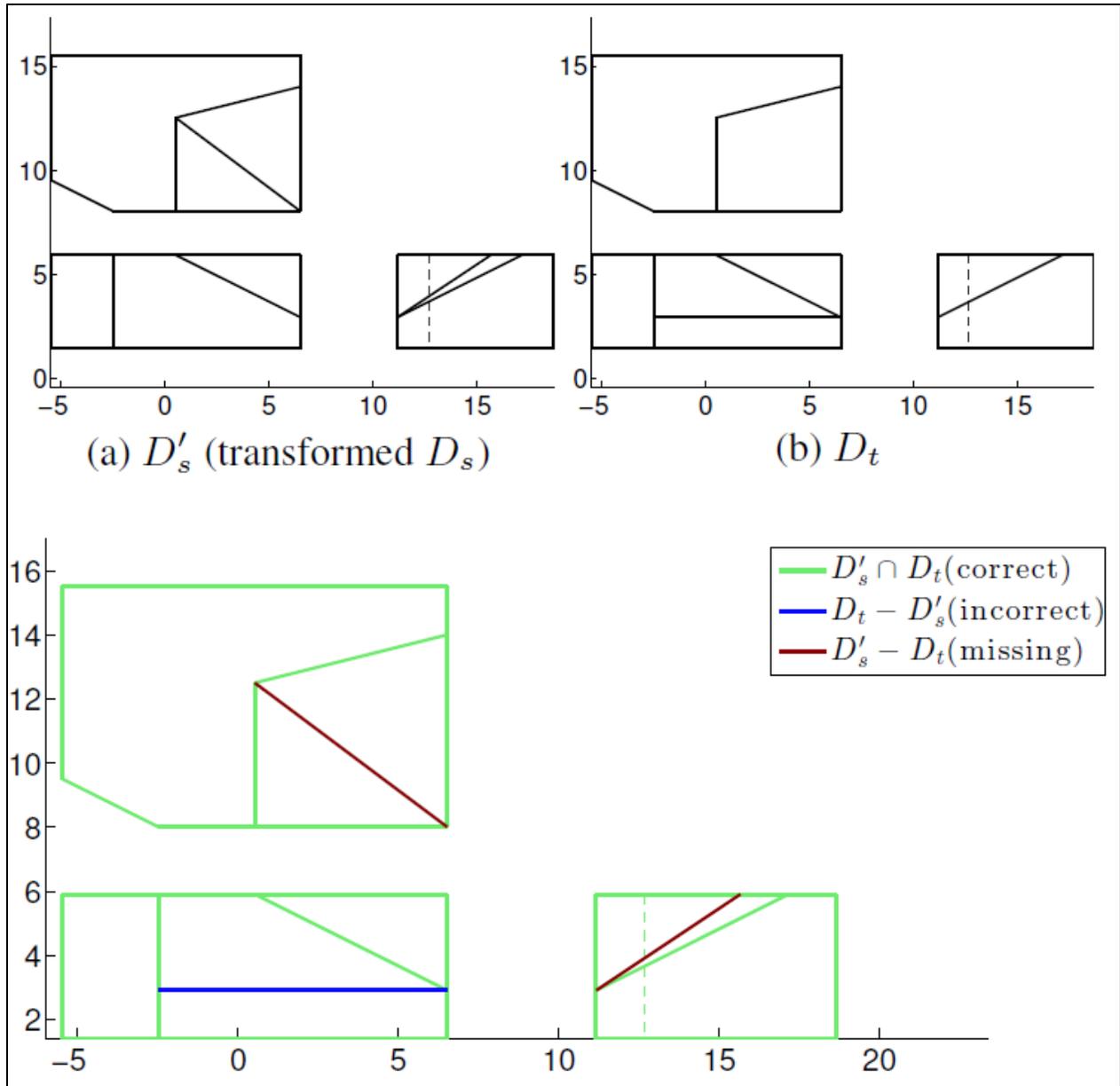


Figure 16: Example of grading using RANSAC algorithm [33]

One of the basic approaches for developing an automated drawing grading software has been converting the drawing into a code and comparing the code parameters to calculate the differences and errors. Goh, Siti and Rofans used the same approach to propose a grading software [34]. The software converts the .dxf files into Scalable Vector Graphics (SVG) format. The software then applies the developed marking algorithm to the SVG format and evaluates the score. Similar work has been done by Shukur et. al, developing a software that compares drawing (.dxf) files from AutoCAD 2000 and grades them automatically [35].

For the CAD software AutoCAD, the .dxf files are in the ASCII format which makes extracting the geometric and object parameters from the files easy. Hekman and Gordon used same approach and developed a software to extract geometric objects from the students' AutoCAD .dwg file, compare them to a master file and send the feedback to the student [36].

San Diego University has developed a proprietary web based application for automatic grading of drawings for Civil Engineering students [37,38]. They used group codes from the .dxf files in AutoCAD to compare the technical drawings. This application enabled students to upload their drawing on the web application and obtain their grade immediately. This web application makes the grading process quick and eliminates human grading errors.

Table 2 gives an overview of research done in automatic grading of technical drawings.

TABLE 2: Overview of research done in automatic grading of technical drawings

Kwon, McMains (2015)	Heckman, Gordon (2013)	Goh, Manao (2013)	Karna (2012)
<ul style="list-style-type: none"> The software used RANSAC algorithm to compare two drawings and highlight differences between them. 	<ul style="list-style-type: none"> Converts students' AUTOCAD DWG files to DXF ASCII format to compare with the solution drawing and grades based on the differences. Feedback is provided to the student via email. 	<ul style="list-style-type: none"> The DXF file format is converted to Scalable Vector Graphics (SVG) file format, which is a graphic vector file format. The difference in drawings is calculated by subtracting their graphic vectors. 	<ul style="list-style-type: none"> Developed a fully functioning application for automated grading of AutoCAD drawings. Compares the object parameters between two drawings using the ASCII code of geometric features in the drawings

<ul style="list-style-type: none"> • File Format: .DWG • CAD Package: AutoCAD 	<ul style="list-style-type: none"> • File Format: .dwg • CAD Package: AutoCAD 	<ul style="list-style-type: none"> • File Format: .dxf • CAD Package: AutoCAD 	<ul style="list-style-type: none"> • File Format: .DWG • CAD Package: AutoCAD Civil
---	---	---	---

As we can see from Table 2, most of the work in automatic grading has been concentrated on a particular CAD platform and file format. In this work we aim to create a grading software which also acts as a pedagogical tool to enhance learner’s experience in improving spatial visualization skills by providing an immediate feedback to the student. At the same time, the grading software is independent of the CAD platform used to create the drawings giving user the flexibility to use a CAD platform available readily. For example if a student has Inventor on his system, he/she can use it create the CAD models and drawings. If another student has CATIA, he/she can use it to create the drawings. The software works independently of the CAD package used to create the drawings.

Chapter 5: Implementation of the software

As seen in the previous chapter, most of the automated grading software applications have been developed for specific CAD packages like AutoCAD using the ASCII code of the .dxf file and the group codes for the .dwg file format. This work aims at developing a software which is independent of the CAD software being used to create the technical drawings or 3D models.

The basic requirement for any grading software is to compare two files, highlight the differences and calculate a score based on these differences. The software developed works by analyzing drawings in the form of an image. There are various tools available for the comparison of two images.

The Computer Vision Toolbox and Image Processing Toolbox in MATLAB is used in this work. MATLAB stores an image as an array where each element of the array represents the corresponding pixel in the image. The array stores information such as the color and intensity in the image. The basic idea used for the comparison of two images is to find matching points in various regions throughout the image by detecting its features. In computer vision and image processing, a feature is a piece of information relevant for completing the computational task related to a certain application [39]. Features in an image are specific structures such as points, edges or objects.

Corner, Blob and Edge are the types of features used in image processing. Example of Patch, Corner and Edge features is shown in Figure 17.

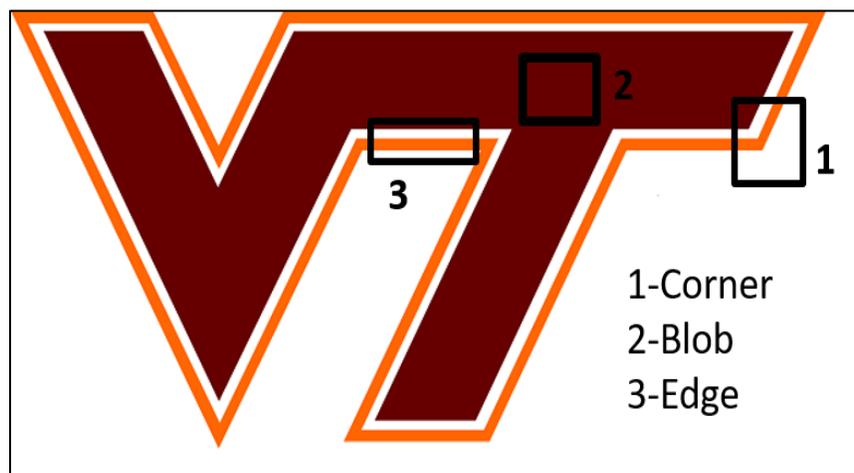


Figure 17: Types of features in an image

A Blob feature is a patch in the image and does not have much variation in any direction in the neighborhood. It does not have point like structures and mainly used to detect smooth regions or objects in an image. An Edge feature is usually a boundary that separates two regions in an image and is a one dimensional structure. A corner is an intersection of two edges and it is a two dimensional structure. The features in an image are stored in the form of a feature vector.

5.1: Difference between Corner, Edge and Blob:

Figure 18 explains the difference between the three types of features used in image processing. Consider a window around a feature that can be moved in any direction around the feature. In case of Blob feature, if we move the window in neighborhood of the blob there is no variation in the intensity of the image as blob feature is a smooth portion or patch in an image. In case of an edge, if you move the window in adjacent regions, there is no variation in intensity along the direction of the edge. For a corner, even a slight movement of the window causes large variations in the intensity of the window. This characteristic makes corner the best feature to compare two images for the current application. The next section explains the algorithm used to detect the corner features in an image.

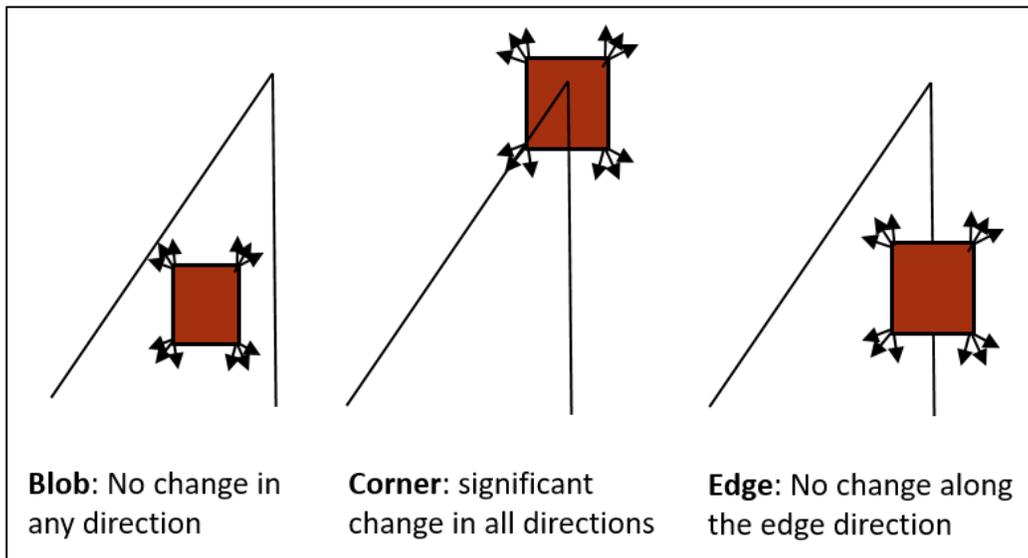


Figure 18: Difference between Blob, Corner and Edge features

5.2: Harris Stephen's Algorithm:

Corner or point like structures are the best features that can be used to extract information from an image as corner features are reliable and are repeatable. The Harris Stephen's algorithm was used to detect features in the image submitted by the student [40]. The algorithm detects corner features by calculating the variations in the gradient of the image. Consider a corner feature with a window as shown in Figure 19.

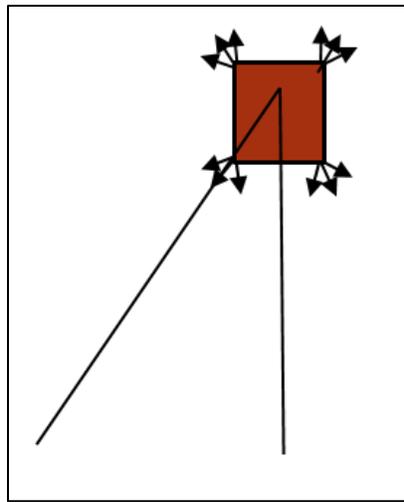


Figure 19: Corner feature with window

Change of intensity in the window after displacement of u in x -direction and v in y -direction is calculated using:

$$E(u, v) = \sum_{x,y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Where $w(x, y)$ is the window function (step function) with the value of 1 inside the window and zero elsewhere:

$I(x + u, y + v)$ is the intensity of the shifted window

$I(x, y)$ is the intensity of the window at the initial position (x, y)

For the feature to be a corner, the variation has to be maximum i.e. the value of $E(u, v)$ has to be maximum. For $E(u, v)$ to be maximum, the term

$$[I(x + u, y + v) - I(x, y)]^2$$

has to be maximum. Performing Taylor series expansion on the first part of above term, we get,

$$E(u, v) \approx w(x, y)[I(x, y) + uI_x + vI_y - I(x, y)]^2$$

Cancelling alike terms and expanding the square term,

$$E(u, v) \approx \sum_{x,y} w(x, y)[u^2 I_x^2 + 2uI_x vI_y + v^2 I_y^2]$$

Writing it in the matrix form, we get,

$$E(u, v) \approx [u \ v] \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$E(u, v) \approx [u \ v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

$$\text{Where } M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

A score is calculated using the eigenvalues of matrix M using the formula,

$$R = \det(M) - k(\text{trace } M)^2$$

Where $\det(M) = \lambda_1 \lambda_2$

$$\text{trace}(M) = \lambda_1 + \lambda_2$$

$\lambda_1, \lambda_2 =$ Eigen values of matrix M

A window with a score R greater than a certain value is considered to be a corner feature. The value of R is negative for an edge feature, small for a blob feature and large for a corner feature. The Harris Stephen's algorithm is used in detecting corner features for images loaded into the software. The 'detectHarrisFeatures' function in MATLAB computes the corner feature in an image.

5.3: Development of Algorithm:

The software uses the feature detection and extraction process in MATLAB. Students need to submit their files in the form of an image. The steps to be executed by the user/student are explained in detail in the next section. When an image is submitted by student/user, the features in the image are detected using the Harris-Stephen's algorithm. The 'detectHarrisFeatures' function in MATLAB uses this algorithm to find the features in an image. The detected features are then extracted using 'extractFeatures' function in MATLAB. This function computes a descriptor that transform the local neighborhood of a feature into a compact vector representation (feature vector). The features extracted are then matched with the source files using 'matchFeatures' function in MATLAB. The process of creating the source (solution) files is explained in section 5.7.

Consider an image as shown in Figure 20. The features extracted using the 'detectHarrisFeatures' function are shown in green in Figure 21.

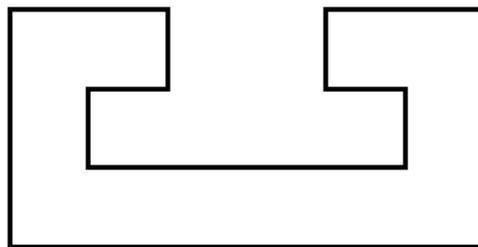


Figure 20: Tee Slide

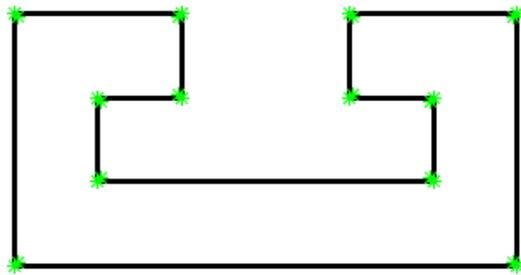


Figure 21: Features detected using Harris Stephen's Algorithm

The detected features are then extracted using 'extractFeatures' function and stored in the form of a feature vector. This extracted feature vector is then matched with the feature vector of source file using 'matchFeatures' function.

An example of 'detectHarrisFeatures(I)' is shown in Figures 22 and 23 below:

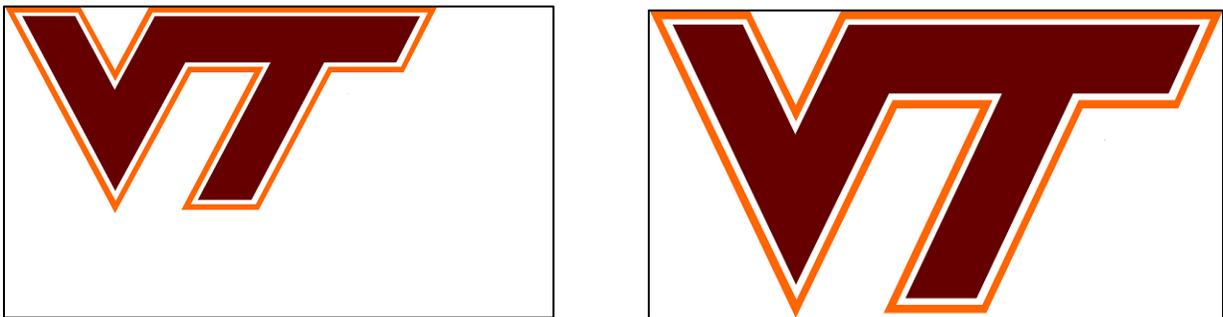


Figure 22: Two Images to be compared

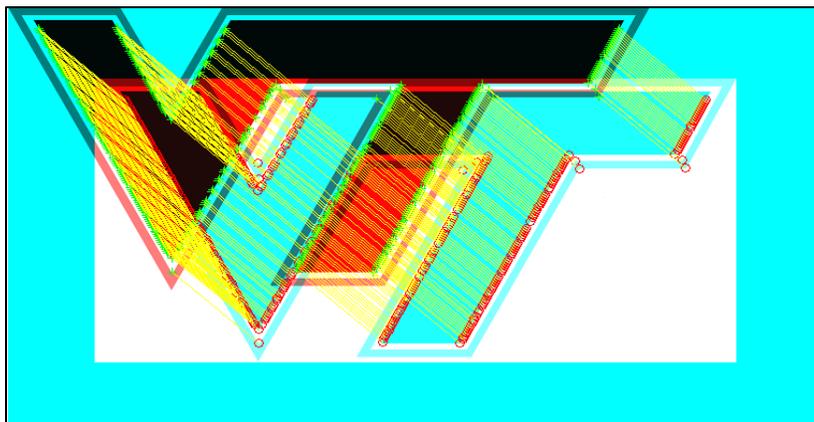


Figure 23: Features matched between the two images using detectHarrisFeatures(I)

5.4: Input required from the student:

As the students are new to CAD environment, they are provided with detailed instructions for creating the 3D models and sectional views. An example of these instructions is provided in Appendix B. A standard drawing sheet needs to be used for the technical drawings. Hatching pattern for the sectional views needs to follow a specific standard. The scale for the technical drawings will also be specified in the instructions and needs to be followed. The drawing planes and dimensions for the 3D models will be specified to the students. Thus the parameters to be followed while developing the 3D models and technical drawings are:

1. Standard Drawing Sheet
2. Scale
3. Hatching pattern
4. Orientation for the 3D models
5. Line weights for 2D drawings

According to the specified instructions, the student needs to create the technical views and save the drawing sheet in .pdf file format. The student then needs to crop the sectional view and store it as an image. The next section explains the graphic user interface of the software.

5.5: Graphic User Interface:

The Graphic User Interface for the software was developed using the GUIDE feature in MATLAB. This feature allows users to create an interface for an application built using MATLAB and add controls such as pop-up-menu, push buttons and text boxes. The user can then link these controls by modifying the algorithm of the interface. The graphic user interface for the software is shown in Figure 24 below. The user needs to select an object to be graded from the pop-up-menu at the top of the interface. Next step is to load the 2D drawing and 3D model using the respective push buttons as shown in Figure 24. The 'Evaluate' push button is used to calculate result for the uploaded files and is displayed in the text box at the bottom.

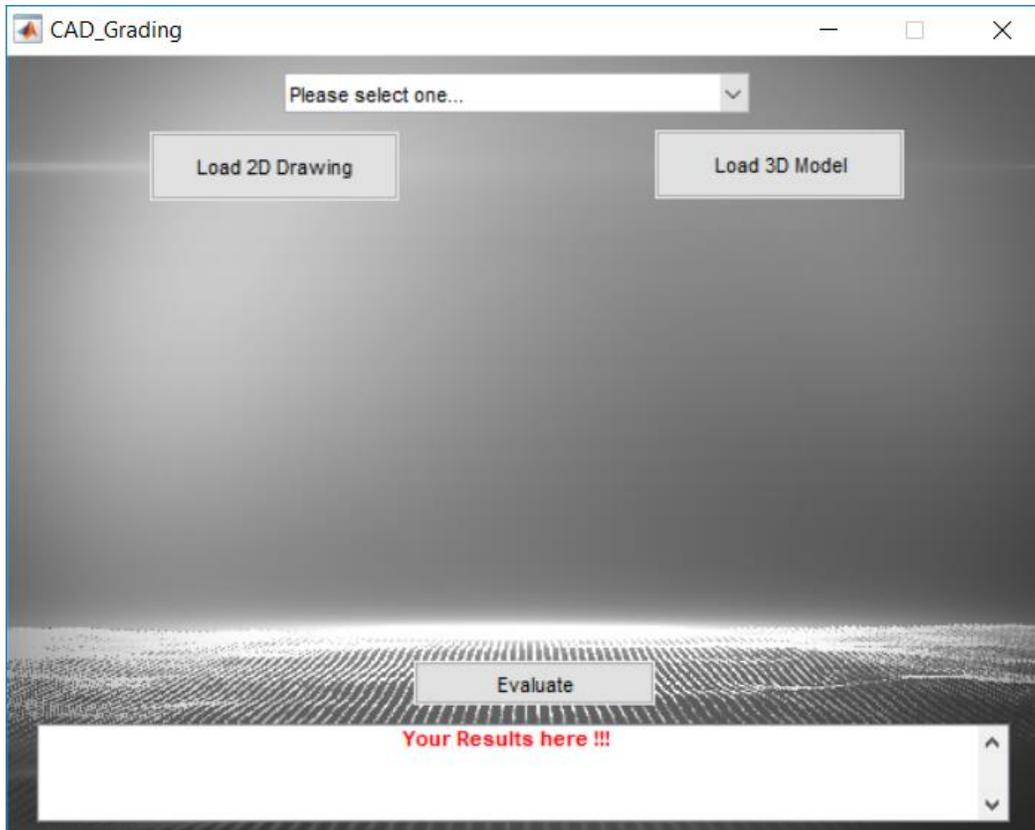


Figure 24: Graphic User Interface for the software

5.6: Grading Algorithm

The grading rubric for the manual grading was implemented in the software to measure the sectional view as shown in Table 3. A second grading rubric was created to score the 3D drawings and is shown in Table 4.

TABLE 3: Grading Rubric for Sectional View

Description	Grade
Completely Correct Sectional View	5
Sectional View with missing Hatching OR Features	4
Sectional View with missing Hatching AND Features	3
Sectional View with incorrect cutting plane	2
Completely incorrect drawing	1

TABLE 4: Grading Rubric for 3D Drawings

Description	Grade
Completely Correct 3D Drawing	5
3D Drawing with Incorrect Cutting Plane	4
3D Drawing with incorrect orientation	3
3D Drawing with missing features	2
Completely Incorrect 3D Drawing	1

Consider an example of Tee Slide object as shown on Figure (5). Student is awarded a full grade of 5 points for the correct sectional view as shown in Figure 25.

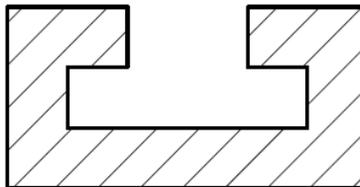


Figure 25: Correct sectional view

If the student creates a sectional view of the object without hatching as shown in Figure 26, a penalty of 1 point is applied and a grade of 4 is submitted.

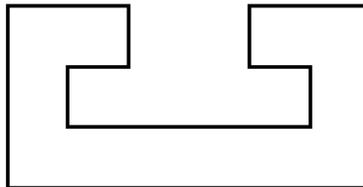


Figure 26: Sectional view without hatching

A penalty of 2 points is applied if a student creates a drawing with both missing feature and no hatching as shown in Figure 27.

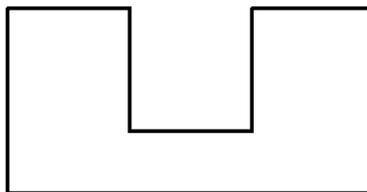


Figure 27: Sectional view without features and hatching

A wrong sectional view as shown in Figure 28 has a penalty of 3 points and the student receives a grade of 2.

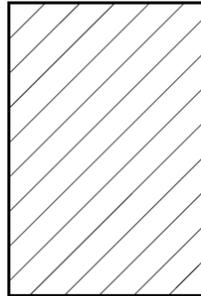


Figure 28: Wrong sectional view

If a student submits a completely irrelevant or incorrect sectional view, a grade of 1 point is awarded for the effort put in to create the sectional view.

For the 3D models, students are awarded a grade of 5 if the object is in correct orientation and has a correct cutting plane as shown in Figure 29.

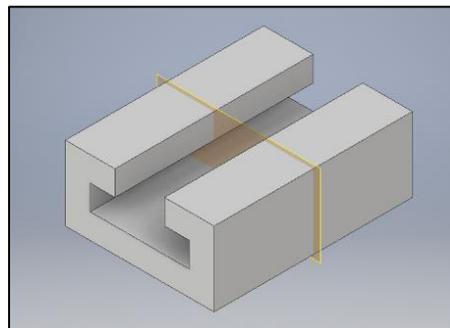


Figure 29: Completely correct 3D model

For a 3D model with an incorrect cutting plane, the student is given 4 points with a penalty of 1 point. An example of this is given in Figure 30.

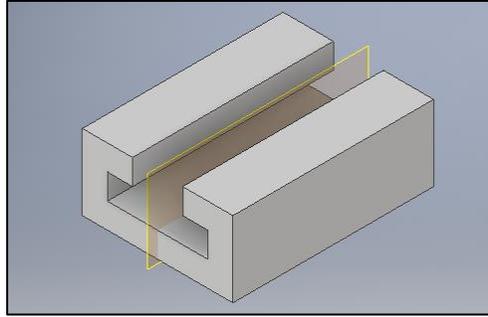


Figure 30: 3D model with incorrect cutting plane

If the student submits a 3D model with wrong orientation, he is awarded 3 points with a penalty of 2 points as shown in Figure 31.

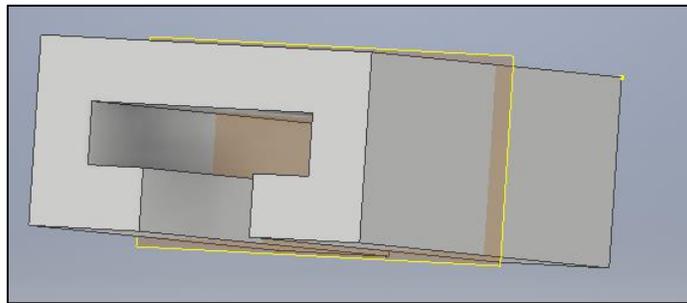


Figure 31: 3D model with incorrect cutting plane

A grade of 2 points with a penalty of 3 points is awarded for a 3D model with missing features. An example of this is shown in Figure 32.

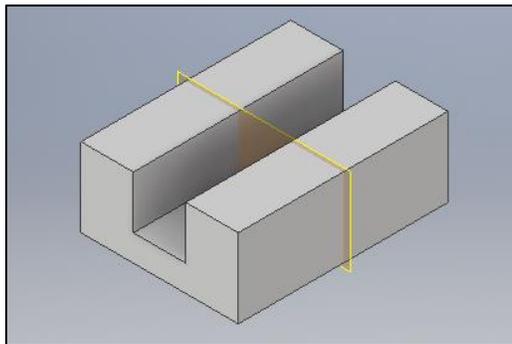


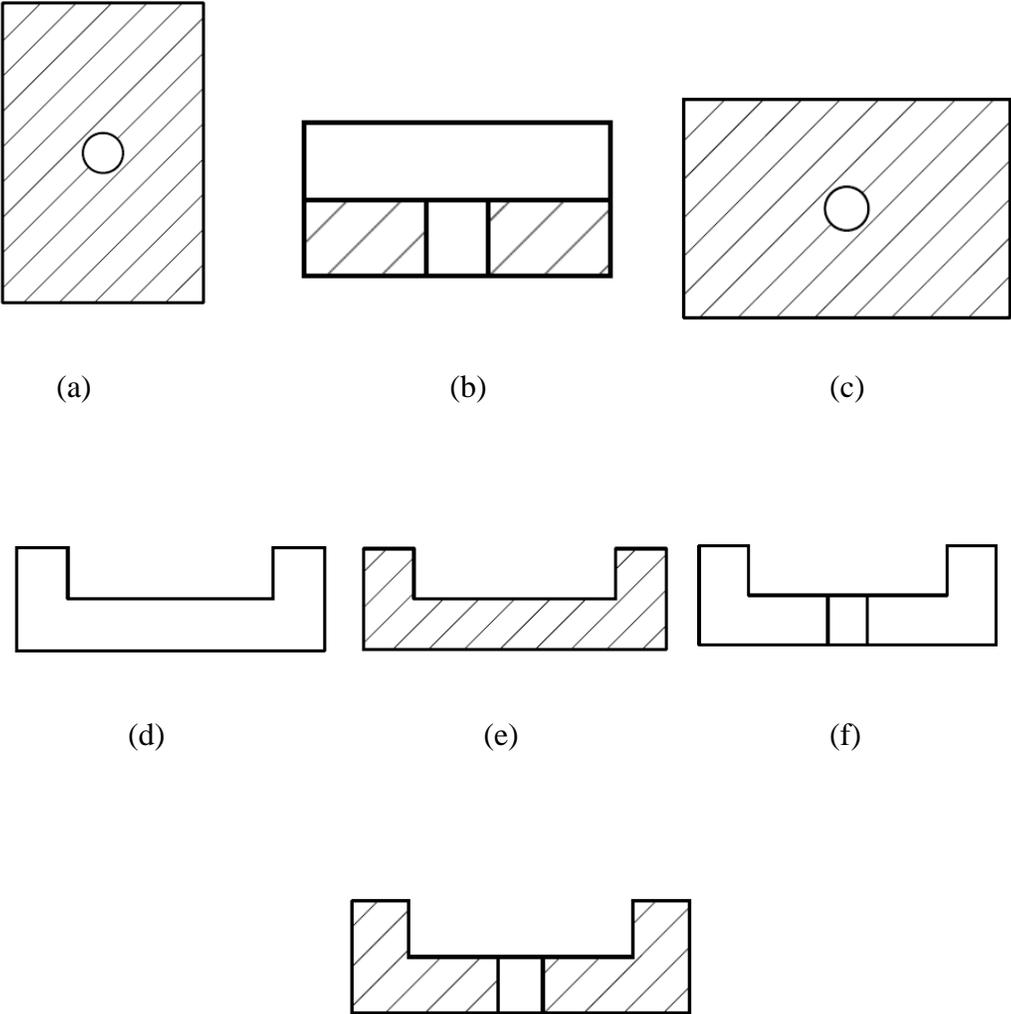
Figure 32: 3D model with incorrect cutting plane

All the possible drawings as defined in the grading rubric are stored in the computer as source files. The software compares the submitted drawing with these source files as discussed in the next

section. The criteria defined in the grading rubric is used to assign a grade after comparing the submitted and source files. If a grade less than five points is assigned, the software will give a description of the mistake(s). This feature enables the software to be an interactive tool and provide an improved learning experience by delivering timely feedback.

5.7: Creating Source Files

Source (solution) files for the six objects are created according to the grading rubric employed in the software. A single grade in the rubric can have one or more solution files associated with it. For Example, grade 3 in the rubric for sectional views can have multiple source files with different features missing in each source file associated with the grade. An example of source files created for the object SlideBlock are shown in Figure 33 below:



(g)

Figure 33: Source files for the object SlideBlock for grade (a) 2 (b) 2 (c) 2 (d) 3 (e) 4 (f) 4 (g) 5

5.8: Algorithm:

This section explains the working of software algorithm. When an image is loaded in the user interface of the software, it is cropped using a function ‘Boundingbox’. This function crops the submitted image along its boundaries and transforms the file into .bmp file format. This helps in reducing the processing time of the software and possibility of any spurious points or features being detected. An example of the Boundingbox function is shown in Figure 34.

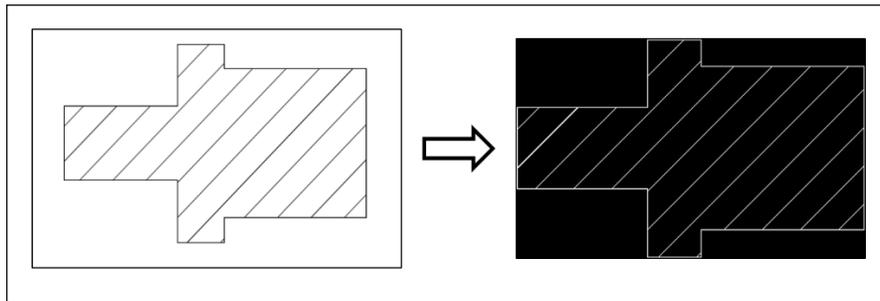


Figure 34: Cropping the image using Bounding Box function

After cropping the image, the next step of algorithm is to detect the features in the image submitted. This is done using the ‘detectHarrisFeatures’ function. The detected features are then extracted using ‘extractFeatures’ function. After extracting the features, these are then compared with each of the source drawings developed and coded for the particular object. If a match is obtained, a grade is given according to the grading rubric employed in the algorithm. Each source file has a particular grade assigned and this grade is displayed on the interface. For a score less than 5, a description of the mistake is provided. The flowchart in Figure 35 explains the algorithm of the software.

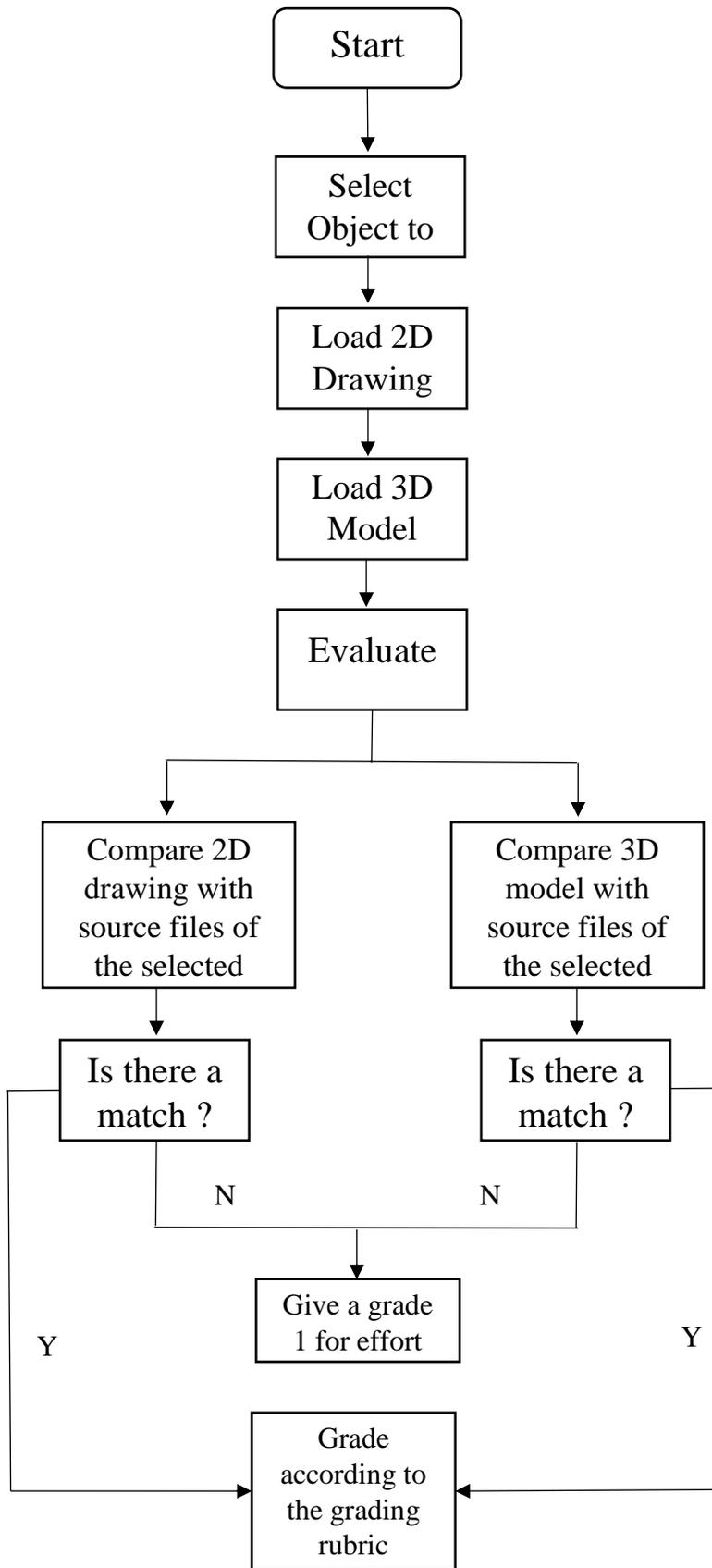


Figure 35: Flow chart for the software algorithm

Consider an example where a file submitted into the algorithm as shown in Figure 36 goes through the grading process.

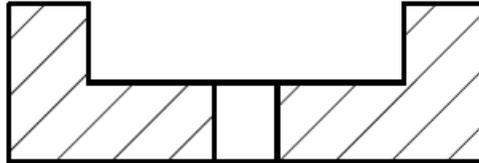


Figure 36: SlideBlock file submitted into the software

The first step in the algorithm is to crop the submitted file along its boundaries using the Boundingbox function. The resulting image is shown in Figure 37



Figure 37: Cropping the submitted image using Boundingbox function

After cropping the image, next step is to detect the features in the submitted image using Harris Stephen's algorithm. This is done using the 'detectHarrisFeatures' function in MATLAB. The detected features are shown in Figure 38

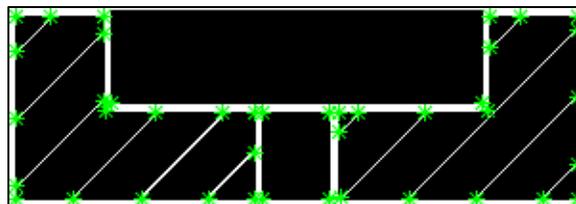


Figure 38: Features detected in the submitted image

The detected features are then extracted using the 'extractFeatures' function and stored in the form of a feature vector. The extracted feature vector is then compared with the feature vector of all source files stored for the SlideBlock object as shown in Figure 39. The Boundingbox function is applied to all the source files when they are coded into the algorithm.

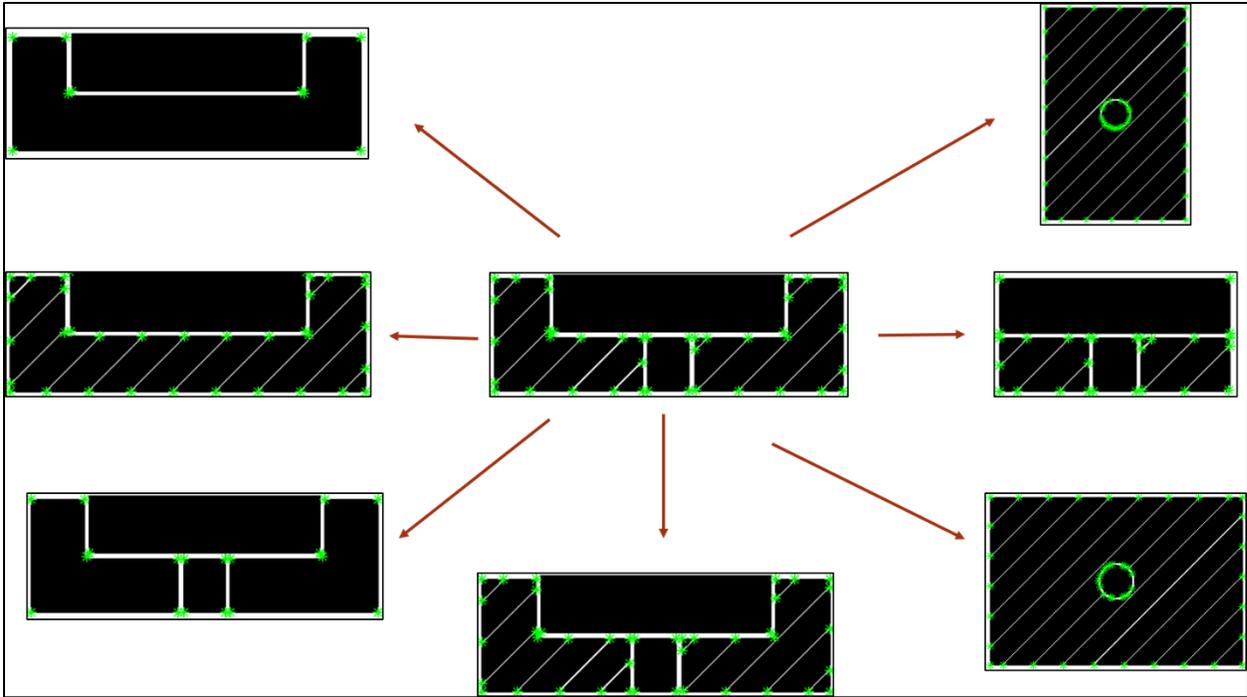


Figure 39: Comparing extracted features with source files

The extracted features are matched using ‘matchFeatures’ function. If a match with the source file is made, a grade associated with that particular source file is given to the submitted drawing and displayed on the interface. If a score of less than 5 is assigned, a description of the mistake is provided.

Chapter 6: Testing

The software was developed and tested for all the grading possibilities as per the grading rubric. 25 possible combinations of the sectional views and 3D drawings were tested according to the six mechanical objects used in the spatial visualization class. One such testing case for the Tool Block object is shown in Figures 40 to 44.

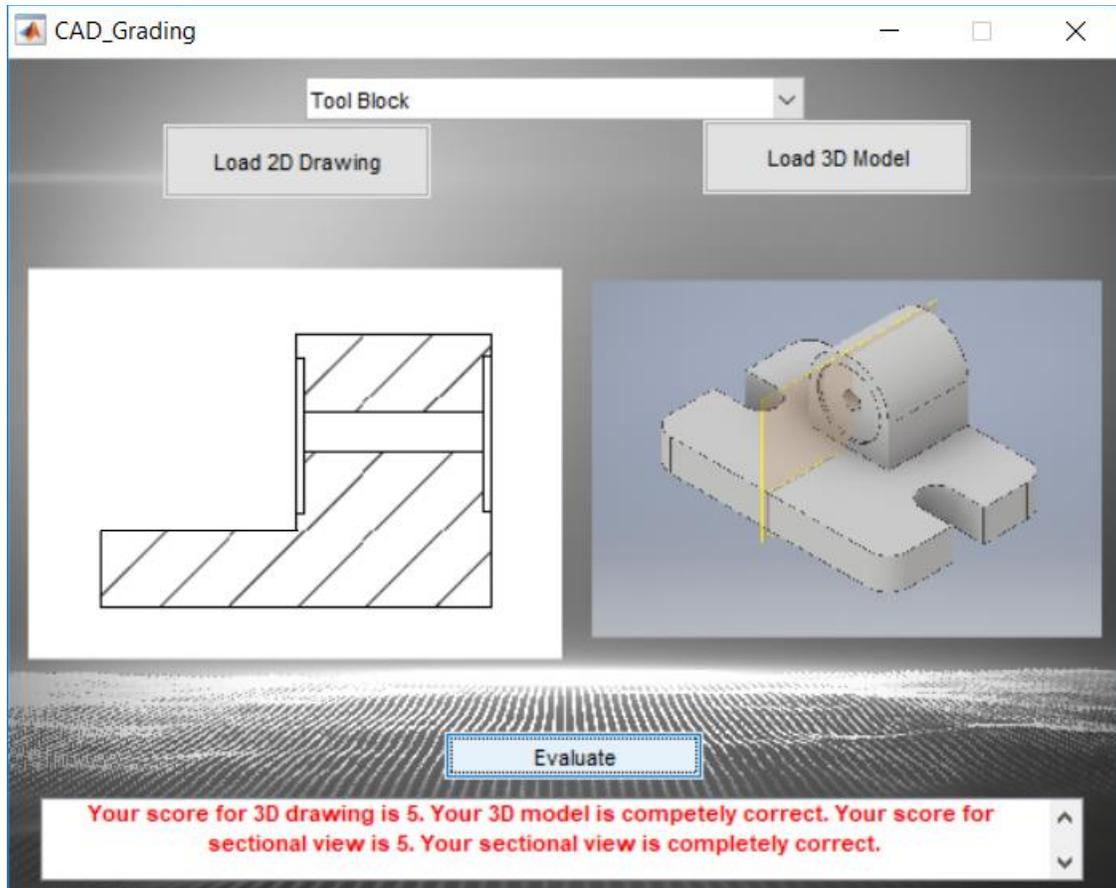


Figure 40: Graphic User Interface for the score of 5

The student has submitted correct sectional view and the 3D model with the right cutting plane and is given a grade of 5.

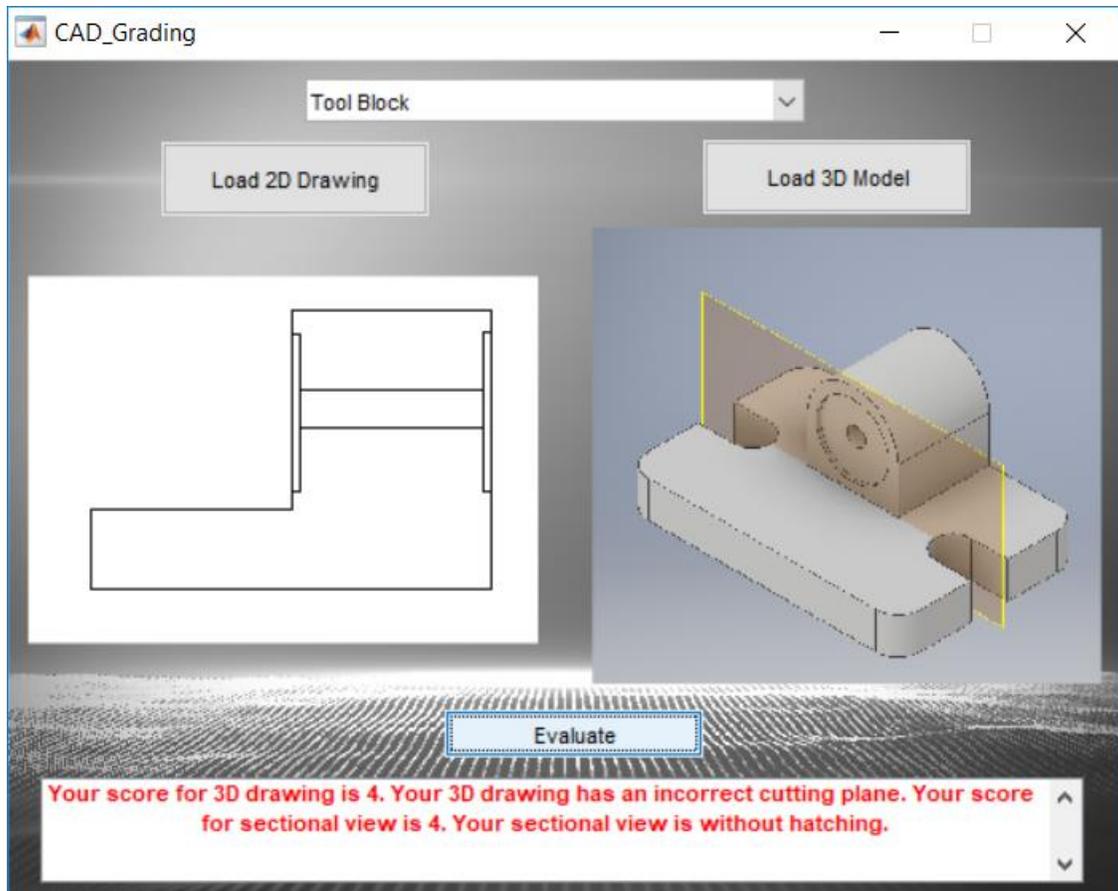


Figure 41: Graphic User Interface for the score of 4

In this case the student has the correct sectional view, but it is without hatching. The 3D model has an incorrect cutting plane. Thus, the student has received a grade of 4 on both sectional view and 3D model with a description of the mistake.

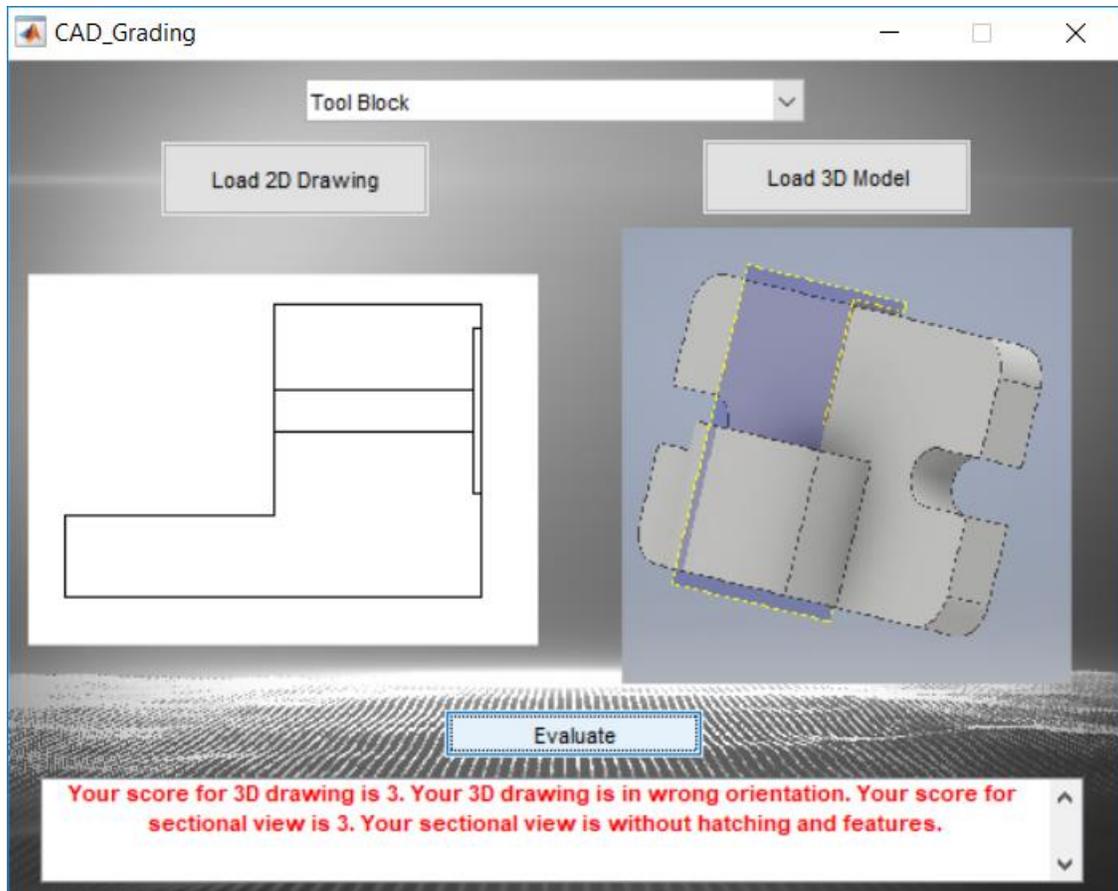


Figure 42: Graphic User Interface for the score of 3

The student has received a grade of 3 for both sectional view and the 3D model. The sectional view has a missing feature and the 3D model is in the wrong orientation.

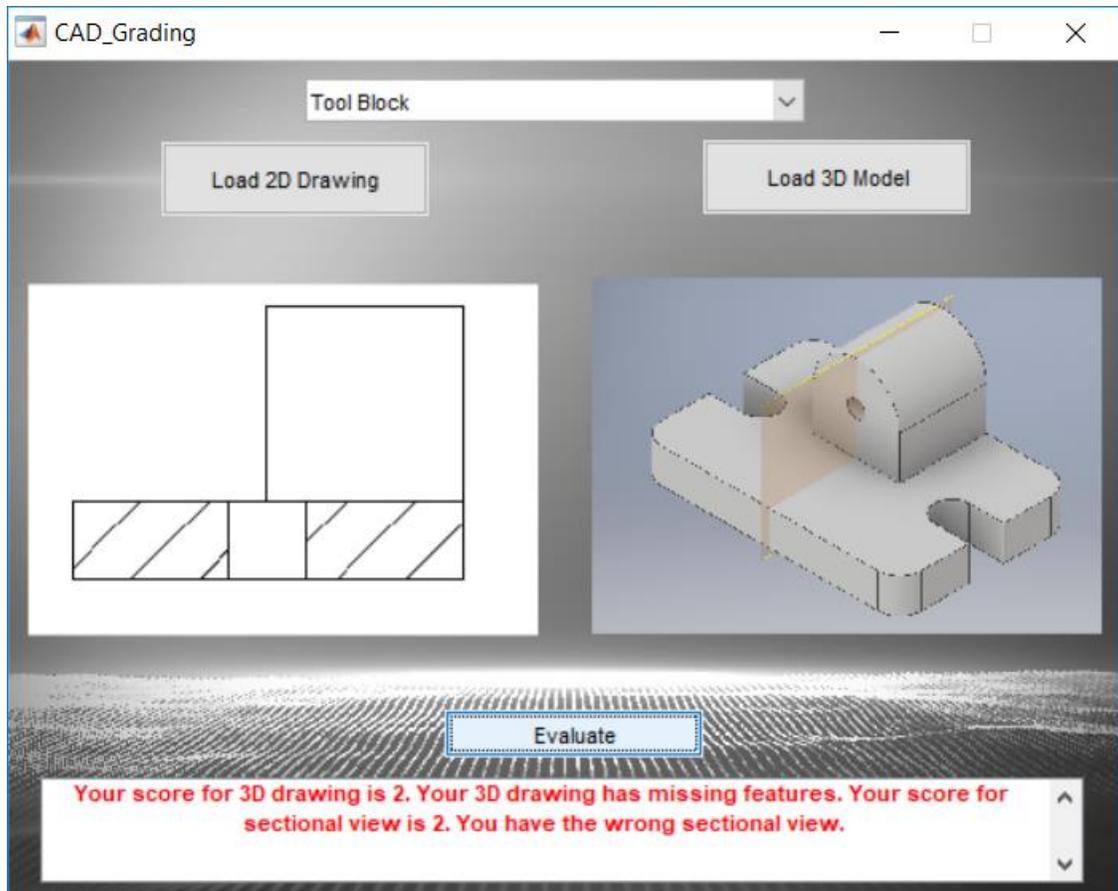


Figure 43: Graphic User Interface for the score of 2

A grade of 2 is awarded in this case as the student has a wrong sectional view and 3D model with missing features.

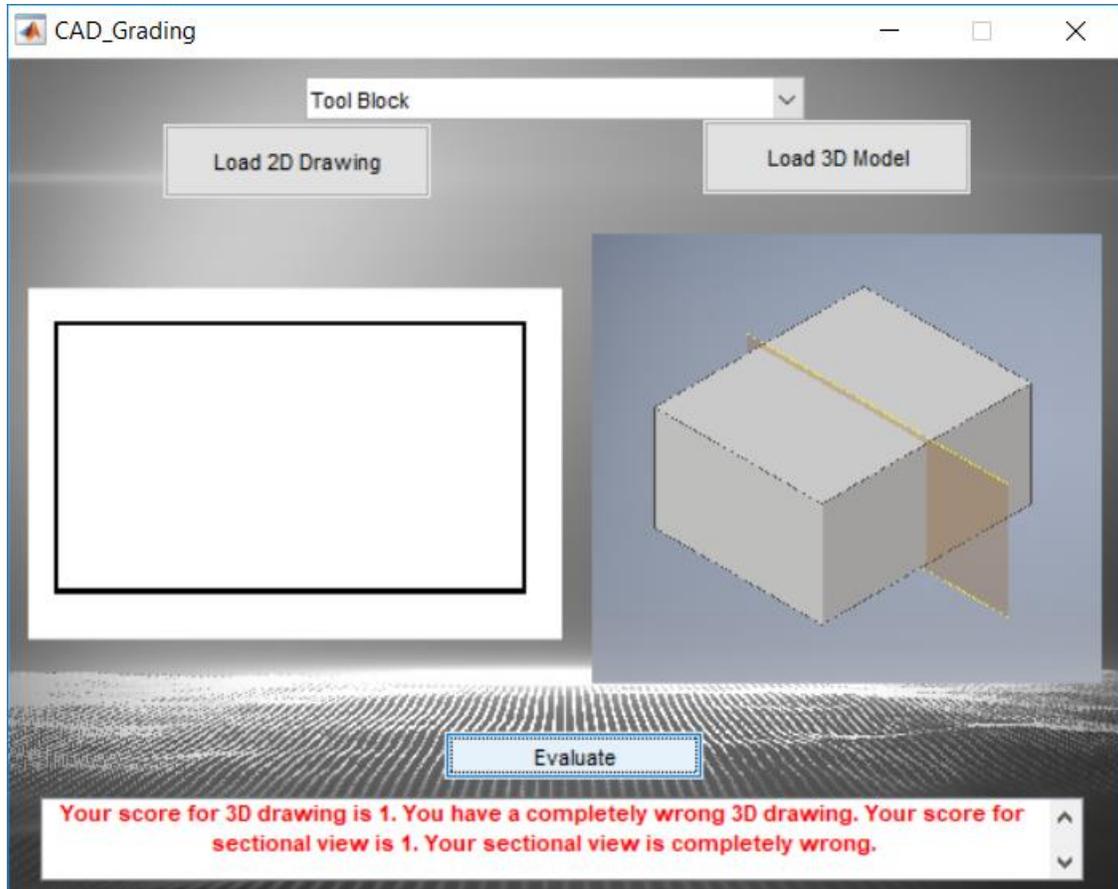


Figure 44: Graphic User Interface for the score of 1

As the student has loaded completely incorrect 2D drawing and 3D model, he has been awarded a grade of 1. The student has been given one point for the effort put in to create the model and the drawing. The testing was also done for other possible combinations of the technical drawings for all six objects.

6.1: Testing the software on different CAD package:

In order to check the effectiveness of the software, it was tested using drawings created with different CAD application, CATIA V5R19 in this case. The models and drawings were created using the same specifications as mentioned before. The results were consistent and are shown in Figure 45 to 49 below:

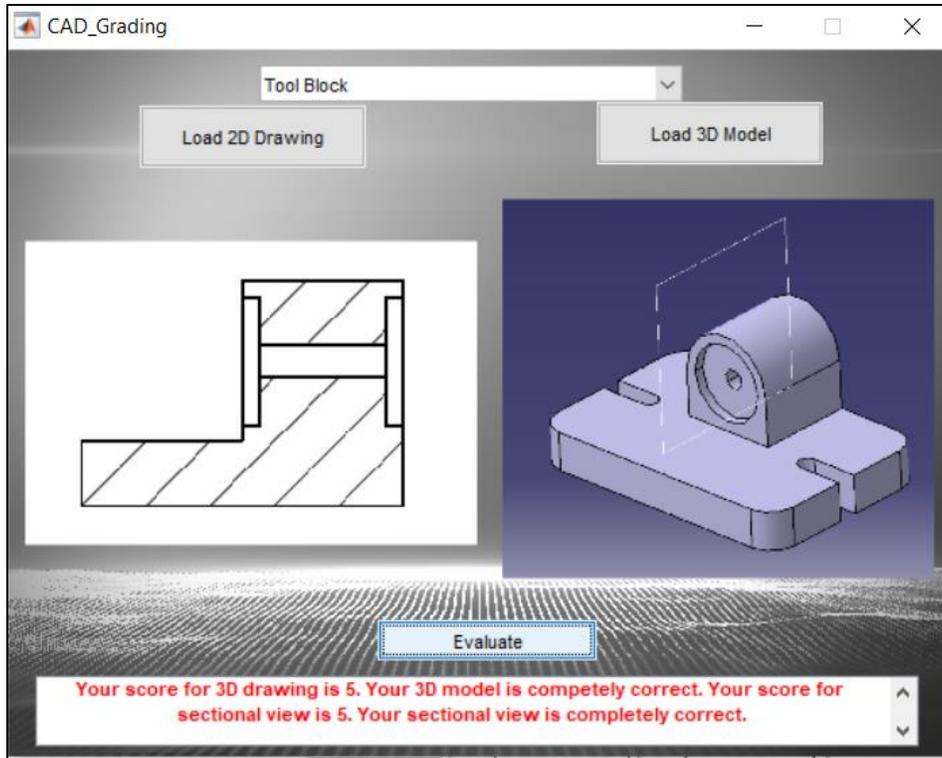


Figure 45: Graphic User Interface for the score of 5 (Using CATIA)

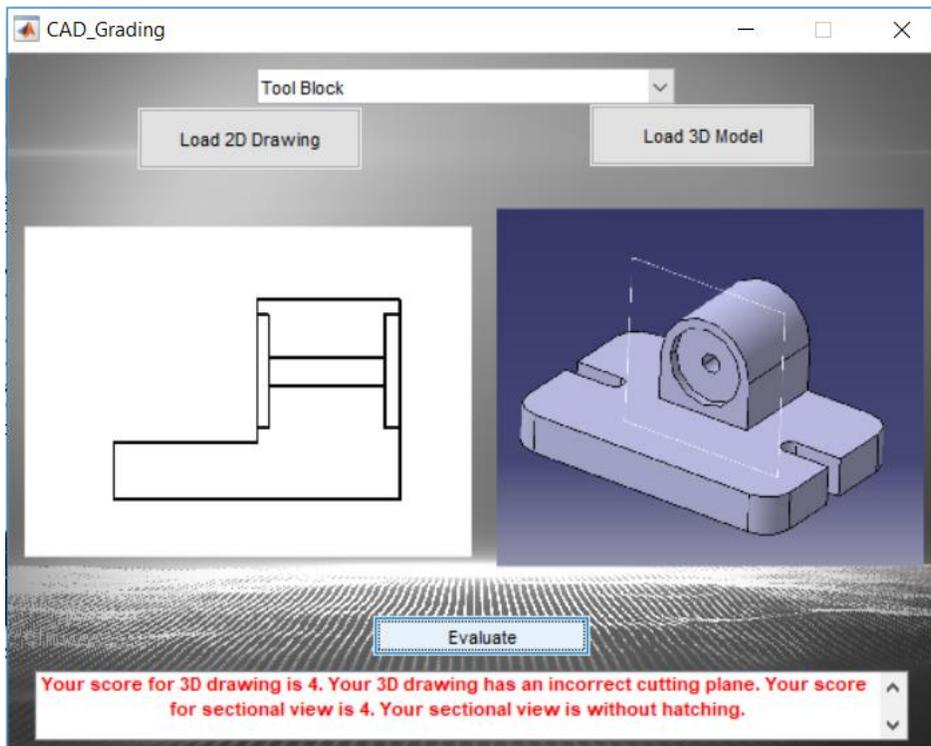


Figure 46: Graphic User Interface for the score of 4 (Using CATIA)

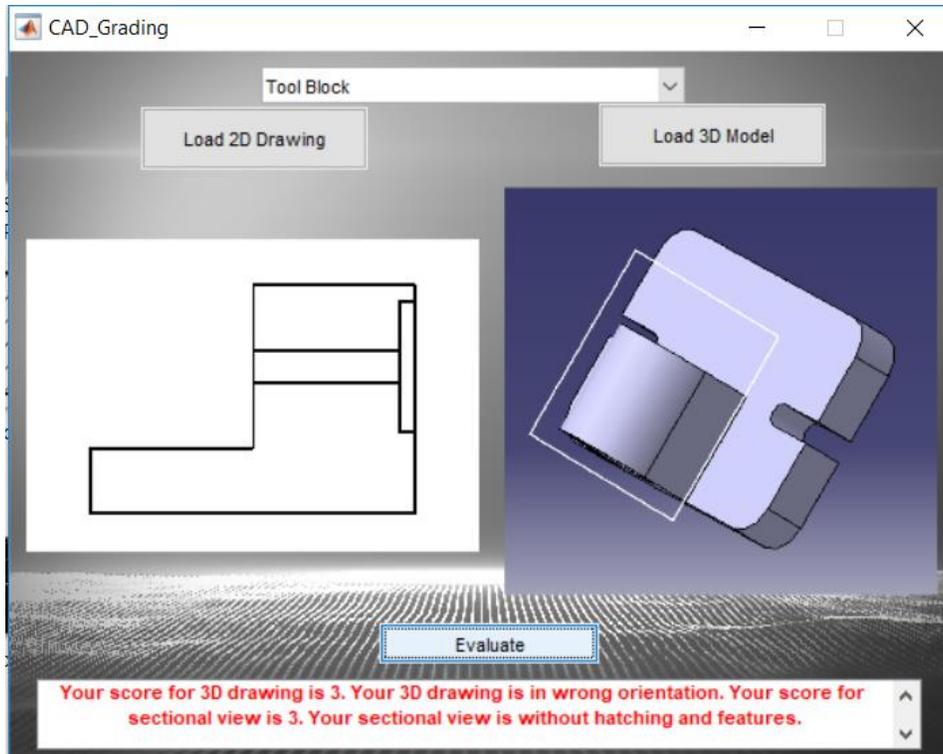


Figure 47: Graphic User Interface for the score of 3 (Using CATIA)

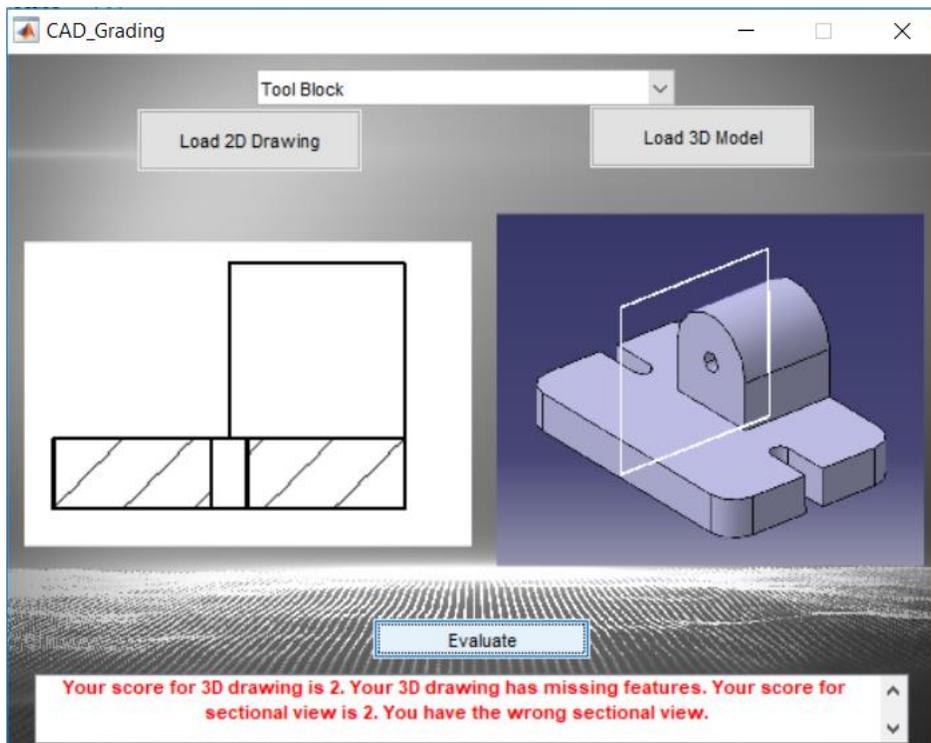


Figure 48: Graphic User Interface for the score of 2 (Using CATIA)

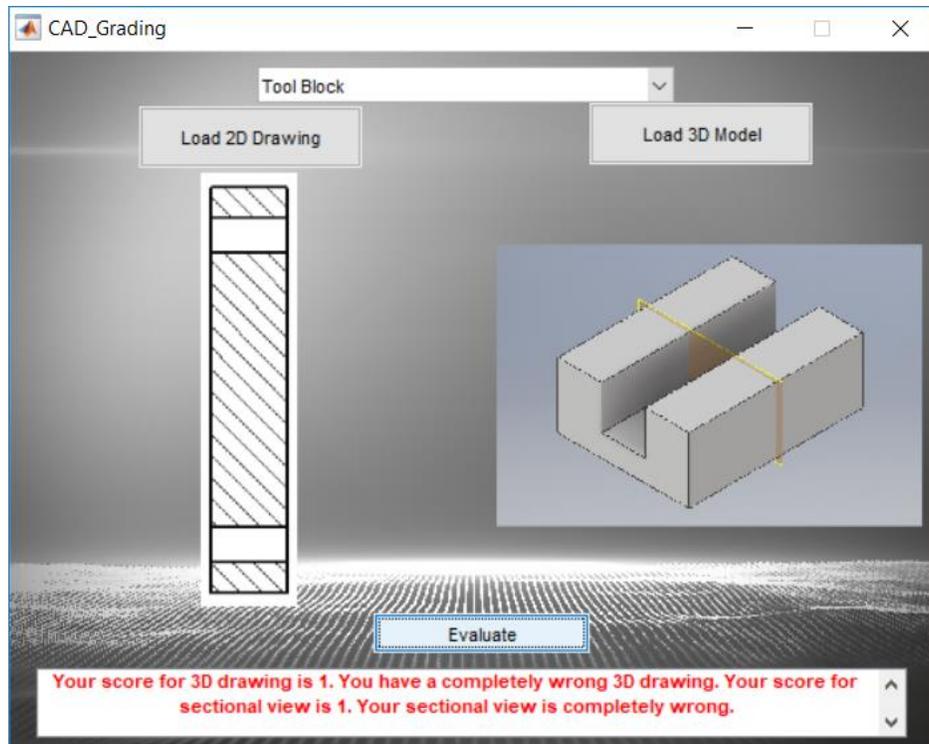


Figure 49: Graphic User Interface for the score of 1 (Using CATIA)

Chapter 7: Batch Processing and Standalone Application

7.1: Batch Processing:

The spatial visualization course conducted by Professor Bairaktarova involves more than 150 students. Implementation of the software in a class requires it to have the ability to grade files in a batch as the process of grading individual files is time consuming. Thus a new interface was developed for grading multiple files at once. Separate graphic user interfaces were developed for 2D sectional views and 3D models as shown in Figure 50 below.

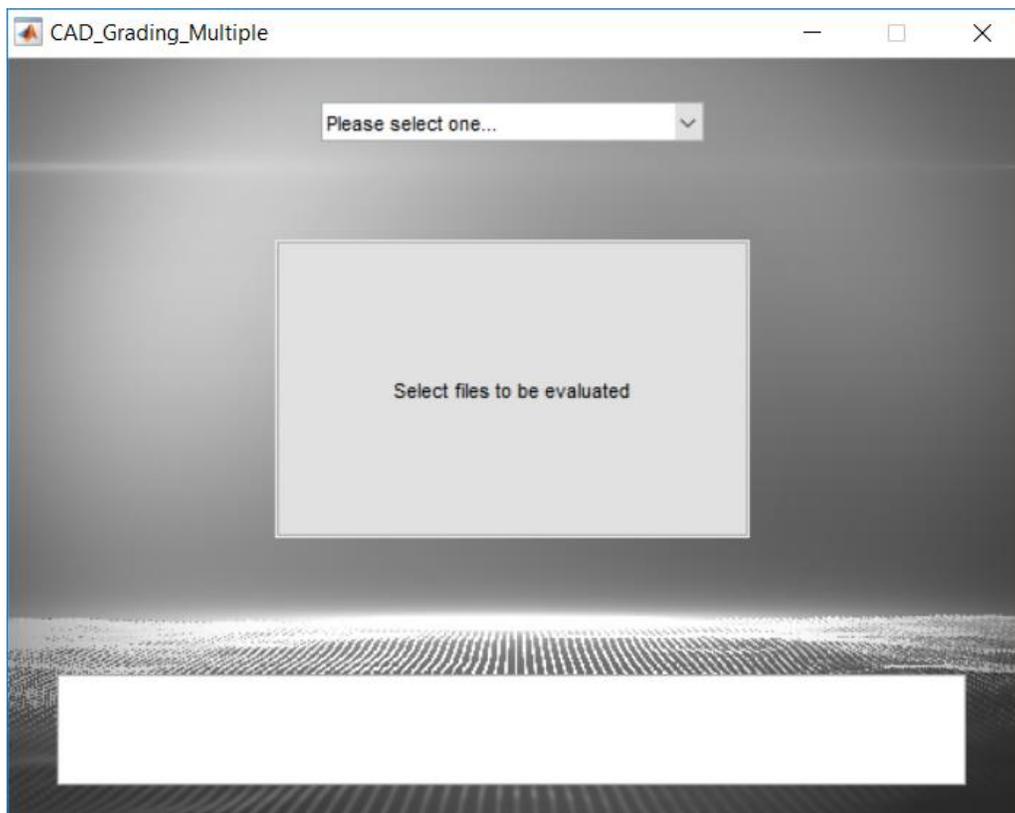


Figure 50: Graphic User Interface for grading multiple 2D and 3D CAD files

The user needs to select the object for which files are to be graded. On clicking the ‘Select files to be evaluated’ push button, the software asks for the specific folder from which the files are to be graded. The user needs to compile all the files for a particular object in a folder. After selecting the specific folder to be graded, the software grades all the files in the folder and exports all the results in an Excel file which has the files name and its corresponding score as shown in Figure 51. The graphic user interface and grading process is the same for both 2D and 3D CAD drawings.

	A	B
1	File Name	Score
2	TeeSlide_1.PNG	4: Your sectional view is without hatching
3	TeeSlide_2.PNG	3: Your sectional view is without hatching and features
4	TeeSlide_3.PNG	4: Your sectional view has missing features
5	TeeSlide_4.PNG	2: You have an incorrect sectional view
6	TeeSlide_5.PNG	5: You have a completely correct sectional view
7		

Figure 51: Grades exported to an Excel file in batch processing

7.2: Compiling the software code into a standalone application:

The software which was coded in MATLAB was then converted to a standalone application using MATLAB Application Compiler. All the source files and codes were packaged into a single installation file. The support packages required to interact with hardware or provide any additional processing capabilities are included in the installation file. This enables the software to run without requiring MATLAB to be installed on the system. While installing, the software installer automatically downloads and installs the runtime libraries required for the functioning of the software. The multiple grading version of the software was also compiled into an application. This can help students new to CAD environment to use the software as an interactive tool. Also, compiling the software into a standalone application helps graders and teaching assistants as they do not have to store individual source files and codes in their system.

Chapter 8: Applicability and Challenges faced in Development of the Software

The software was developed with an aim to aid the instructor and teaching assistant in the grading process and at the same time act as an interactive pedagogical tool for the students. The audience for the software are freshman engineering students. These students are new to mechanical design and CAD environment. As a result, the software has been currently designed for grading of simple CAD geometries with less complicated features. It is not aimed for students in advanced CAD courses and courses which require Engineering Graphics as a prerequisite.

The software developed in this work helps teaching assistants to reduce the time required for grading assignments and returning them back to the students. This aids the learning process of individuals as the concepts are fresh in their minds if they are returned their assignments shortly after submission. On the other hand, in case of manual grading, by the time the assignment is returned to the students, they do not really care about it. Also, due to the development of this software, a variety of assignments can be given to the students that will help develop their spatial visualization skills, as the time required to grade the assignments is reduced.

Using this software, the instructor can gather data on what types of mistakes students generally make. This data can help the instructor to modify the coursework or emphasize on any specific concepts in the class. Students can use this software as an interactive tool to improve their spatial visualization skills. Students can use the single file version of the software to load different drawings and learn from the mistakes they are committing.

As the software can be compiled into a standalone application, the teaching assistants will not require MATLAB and the toolboxes to be installed on their system. Also, they do not need to carry individual solution files as they are already packaged in the compiled application. The code has been developed in a way that makes modifying the grading rubric very easy for the instructor. New objects to be graded can be easily added to the algorithm and incorporated in the software. Thus, the software can be employed in introductory CAD courses aiming to improve the visualization skills of freshman engineering students.

In the early stages of software development, there were some issues that needed to be resolved for proper functioning and validation of the software.

Selecting a proper algorithm for current application was the first hurdle faced in development of this software. First version of the software was developed using the 'detectFASTFeatures' function in MATLAB to detect the features in an image. However, this function was not scale invariable. Thus, slight variation in the scale of a drawing caused the algorithm to fail. The Harris Stephen's algorithm solved this problem as it is scale invariable function.

Initially, the entire image loaded was used to detect features in the submitted image using Harris Stephen's algorithm. This caused spurious features to be detected. This caused the algorithm to fail as these points were not detected in the source files for that particular object. To solve this issue, the 'Boundingbox' function was developed to crop the submitted image along its boundaries and convert it into binary format.

A simple pilot study was conducted in a small group of 4 participants to get the user feedback for the software. The participants were graduate students from Mechanical and Computer Engineering background and they were asked to create a 3D model and sectional view of an object and load it into the software to get their score. They were then given a questionnaire (Appendix C) to rate various aspects of the software on a scale of 1 to 5, 1 being the least score. They were also asked for any feedback or suggestions for improving the user experience of the software. The overall feedback from the participants was satisfactory and their feedback will be implemented in future work. The feedback can be summarized as follows: One of the participants suggested to increase the window size of interface of the software. Another suggestion was to modify the algorithm to grade multi-view drawings and use other techniques to grade 3D models. A full scale pilot study will be conducted later in the next semester to validate the functioning of the software.

The next chapter explains the limitations of the current software.

Chapter 9: Limitations

Although the software was developed and tested for all loaded objects, there are few potential weaknesses or limitations to this work. Firstly, the software cannot predict minute changes in the dimensions of an object. If the user creates a drawing by modifying the dimensions of an object, the software will give this drawing a grade of 1 i.e. completely wrong drawing. If the hatching pattern for sectional view is not followed, it will result in a grade of 1 as the software algorithm will not be able to recognize the modified hatching pattern. This implies that the student needs to strictly follow the instructions in creating the CAD drawings provided during the course.

Secondly, the developer needs to create source files for all the objects loaded into the drawing. This can be a time consuming process for complicated drawings. For example, for grade 3 (missing hatching and features) for a sectional view, the user has to create multiple files with different features missing in each source file. This implies that the developer has to predict the mistakes that students can make while creating the CAD models.

Another limitation of the software is with regards to the grading of 3D model. For the software to grade the 3D model accurately, it needs to be placed in a perfect orientation in the 3D space. Slight variation in the orientation of the 3D model will result in a wrong grade. This problem can be solved by employing other efficient techniques in comparing the 3D models. The software cannot grade the 3D models in all orientations. Thus, an error which is not seen in isometric view image of the 3D model might go unnoticed.

The software validation needs to be carried out. In the previous chapter, the results from the initial validation were presented, however, a pilot study to determine the effectiveness of the software for the students, instructors, and graders has not been conducted yet. It will be done in the next semester when the software will be implemented and the feedback will be studied to perform the modifications needed to improve user experience. New questionnaires will be created to assess the applicability and effectiveness of the developed software in classroom environment and capture the perspective from the three user groups (Instructor, grader and students).

Chapter 10: Future Work

In order to enhance the performance of the software and experience for the user, there are various areas which can be improved. The software algorithm currently works for individual sectional views of objects. This can be modified in order to grade an entire drawing sheet. This will require the algorithm to calculate the distance between individual views as this distance will be different for each case.

The grading of 3D models can be made more accurate by considering new techniques in analyzing the models like comparing the mesh of two models. There are softwares such as MeshLab that can easily compare two 3D CAD models.

A pilot study will be conducted in order to validate the software and the feedback from the study will be used to address the issues in the algorithm. Data about the types of errors made by the student can be collected throughout the semester. This data will help us understanding the problems faced by the student while creating the CAD models. At the same time, this data can be used to create source files more efficiently observing the errors of student.

The software currently grades technical drawings created using a CAD software. The algorithm can be updated to grade sectional views drawn manually on paper by the students. In case of manual drawings, the geometry is not guaranteed to be accurate. The lines and curves drawn manually may not be accurate. The algorithm needs to be updated to solve these challenges.

Grade for the submitted file is shown on the interface of the software along with the description of the mistake if a score of less than 5 is obtained. The software interface can be made more interactive by displaying the correct answer after the grade has been evaluated. This will improve learning experience of the student using the software.

The software can be embedded into the university course portal thus making the grading process easy and grades readily available to the students. Students can upload their assignments directly on the portal and get them graded easily and in a short time. A discussion portal can be added to the software for students to ask their queries related to coursework.

Chapter 11: Conclusion

The work described in this thesis presents the development of an automated grading software for technical drawings which is independent of the CAD platform/software used to create the drawings. The proposed software can grade technical drawings and CAD models created using any CAD package. The software is currently developed for simple geometries with less complicated features as it is being employed in a course where students are new to CAD environment. Students can also use this software as an interactive learning tool as they create 3D models and sectional views of mechanical objects. The implementation of this software in class will reduce the grading time for the teaching assistant and, at the same time, reduce grading errors, and provide timely feedback to students.

In summary, the salient features of the software are:

- It is independent of the CAD platform.
- It can be used as an interactive tool as it gives live feedback to students.
- Drawings can be loaded and graded collectively.
- It has a simple and user friendly Graphic User Interface.

The software can be used at various levels by individuals. Students can use it as an interactive tool to learn what mistakes are being made while learning engineering design using a CAD platform. The software is currently developed for simple geometries with less complicated features as it is being employed in a course where students are new to CAD environment. As large number of freshman students need to take spatial visualization or engineering drawing courses, this software will help in reducing the time required for grading process and returning graded assignments to students. Adding new CAD models to the algorithm is an easy process. Any changes in the grading rubric can be easily incorporated in the algorithm. The software can also be used with a simple binary grading rubric where only the correct file will be graded. In recent years there have been many online courses in Engineering Graphics and spatial visualization for which an automated grading software is critical. The software developed in this research can be implemented in such online courses. Compiling the software into standalone application eliminates the requirement of individual source files and codes on the grader's or teaching assistant's system.

Figure 53 shows the simple grading process made feasible by the software:

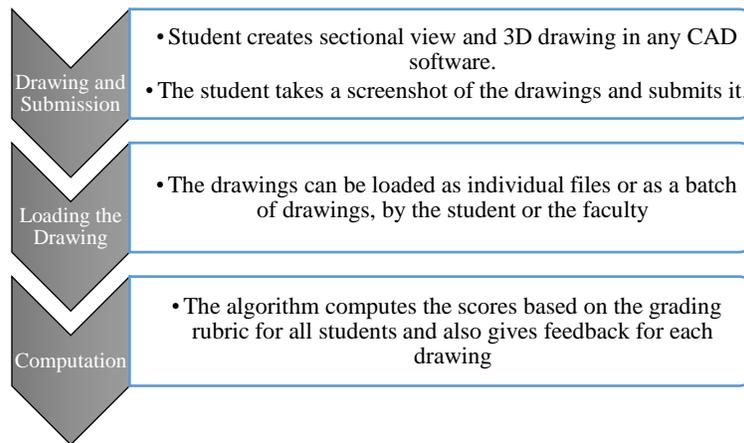


Figure 52: Process Chart for the software

The software will be tested with students in the upcoming Fall semester in the same spatial visualization class and the results will be compared with manual grading. The data collected during the semester will be used to modify the software algorithm to enhance user experience and the results will be shared in a journal publication.

References

- [1] McGee, M G. 1979. "Human spatial Abilities: Psychometric Studies and Environmental, Genetic, Hormonal, and Neurological Influences." *Psychological Bulletin* 86 (5): 889–918. doi:10.1017/CBO9781107415324.004.
- [2] Fennema, Elizabeth, and Gilah C Leder. 1990. *Mathematics and Gender*. ERIC.
- [3] Hamlin, Amy, Norma Boersma, and Sheryl Sorby. 2006. "Do spatial Abilities Impact the Learning of 3-D Solid Modeling Software?" American Society for Engineering Education. http://search.asee.org/search/fetch;jsessionid=2s9wah17ct1cd?url=file:///localhost/E:/search/conference/12/2006Full2233.pdf&index=conference_papers&space=129746797203605791716676178&type=application/pdf&charset
- [4] Kozhevnikov, Maria, Michael a Motes, and Mary Hegarty. 2007. "spatial visualization in Physics Problem Solving." *Cognitive Science* 31 (4): 549–79. doi:10.1080/15326900701399897.
- [5] Kali, Yael, and Nir Orion. 1996. "spatial Abilities of High School Students in the Perception of Geologic Structures." *Journal of Research in Science Teaching* 33 (4): 369–91.
- [6] YANG, Eun-Mi, Thomas ANDRE, and Thomas J. GREENBOWE. 1987. *International Journal of Science Education*. International Journal of Science Education. Vol. 25. Taylor & Francis. <http://cat.inist.fr/?aModele=afficheN&cpsidt=14647914>.
- [7] Wai, Jonathan, David Lubinski, Camilla P Benbow, and James H Steiger. 2010. "Accomplishment in Science, Technology, Engineering, and Mathematics (STEM) and Its Relation to STEM Educational Dose A 25-Year Longitudinal Study." *Journal of Educational Psychology* 102 (4): 860–71. doi:10.1037/a0019454.
- [8] Wai, Jonathan, David Lubinski, and Camilla P. Benbow. 2009. "spatial Ability for STEM Domains: Aligning over 50 Years of Cumulative Psychological Knowledge Solidifies Its Importance." *Journal of Educational Psychology* 101 (4): 817–35. doi:10.1037/a0016127.
- [9] Uttal, David H., and Cheryl A. Cohen. 2012. *spatial Thinking and STEM Education. When, Why, and How? Psychology of Learning and Motivation - Advances in Research and Theory*. Vol. 57. Elsevier Inc. doi:10.1016/B978-0-12-394293-7.00004-2.

- [10] Knott, T W. 2009. “EnViSIONS at Virginia Tech.”
- [11] Mohler, James L., and Craig L. Miller. 2008. “Improving spatial Ability with Mentored Sketching.” *Engineering Design Graphics Journal* 72 (1): 19–27.
- [12] Olson, David R, and Ellen Bialystok. 2014. *spatial Cognition: The Structure and Development of Mental Representations of spatial Relations*. Psychology Press.
- [13] Sorby, Sheryl A, and Anne Francis Wysocki. 2003. *Introduction to 3D spatial visualization: An Active Approach*. Cengage Learning.
- [14] Uttal, David H., Nathaniel G. Meadow, Elizabeth Tipton, Linda L. Hand, Alison R. Alden, Christopher Warren, and Nora S. Newcombe. 2012. “The Malleability of spatial Skills: A Meta-Analysis of Training Studies.” *Psychological Bulletin* 139 (2): 352–402. doi:10.1037/a0028446.
- [15] Barr, R E, T J Krueger, and T A Aanstoos. 2002. “The New Digital Engineering Design and Graphics Process.” *Engineering Design Graphics Journal* 66 (3). ERIC: 6–11.
- [16] Gerson, Helena B P, Sheryl A. Sorby, Anne Wysocki, and Beverly J. Baartmans. 2001. “The Development and Assessment of Multimedia Software for Improving 3-D spatial visualization Skills.” *Computer Applications in Engineering Education* 9 (2): 105–13. doi:10.1002/cae.1012.
- [17] Sorby, S A, and B Baartmans. 2000. “The Development and Assessment of a Course for Enhancing the 3-D spatial visualization Skills of First-Year Engineering Students.” *Journal of Engineering Education* 89 (3): 301–7. doi:10.1002/j.2168-9830.2000.tb00529.x.
- [18] "Jianping Yue, Essex County College", and *Engineering Design Graphics*. 1977. “spatial visualization By Realistic 3 D Views.” “2007 Annual Conference & Exposition, Honolulu, Hawaii,” 28–38. <https://peer.asee.org/1619>.
- [19] Christou, Constantinos, Keith Jones, Demetra Pitta-Pantazi, Marios Pittalis, Nicholas Mousoulides, João Filipe Matos, Evgenia Sendova, Theodossios Zachariades, and Pavel Boytchev. 2007. “Developing Student spatial Ability with 3D Software Applications.”
- [20] Czapka, Jason T, Manssour H Moeinzadeh, and James M Leake. 2002. “Application of Rapid Prototyping Technology to Improve spatial visualization.” *American Society for Engineering Education Annual Conference and Exposition*.

- [21] Bi, Zhuming, Donald W Mueller, Fort Wayne, and Fort Wayne. 1977. "Incorporating Product Lifecycle Concept , Rapid Prototyping , and Certification in a Solid Modeling Course — Final Report," 1–16.
- [22] Katsioloudis, Petros J, and Vukica Jovanovic. 2014. "spatial visualization Ability and Impact of Drafting Models: A Quasi Experimental Study." *Engineering Design Graphics Journal* 78 (2).
- [23] Olkun, Sinan. 2003. "Making Connections : Improving spatial Abilities with Engineering Drawing Activities." *International Journal of Mathematics Teaching and Learning*, no. April: 1–10. <http://www.cimt.plymouth.ac.uk/journal/default.html>
- [24] Adanez, GP, and AD Velasco. 2004. "Training Visualization Ability by Technical Drawing." *Journal for Geometry and ...* 8 (1): 107–15. <http://www.heldermann-verlag.de/jgg/jgg08/j8h1prie.pdf>.
- [25] Srinivasan, Anirudh, Jeremy D Smith, Diana Bairaktarova, and Virginia Tech. 2016. "Identifying Freehand Sectional View Technical Drawing Activities in Engineering Design Graphics Course to Enhance spatial Skills of Engineering Students," 1–7.
- [26] Contero, Manuel, Ferran Naya, Pedro Company, José Luís Saorín, and Julián Conesa. 2005. "Improving Visualization Skills in Engineering Education." *IEEE Computer Graphics and Applications* 25 (5): 24–31. doi:10.1109/MCG.2005.107.
- [27] Rafi, Ahmad, Khairul Anuar Samsudin, and Azniah Ismail. 2006. "On Improving spatial Ability through Computer-Mediated Engineering Drawing Instruction." *Educational Technology and Society* 9 (3): 149–59.
- [28] Bodner, George M, and Roland B Guay. 1997. "The Purdue Visualization of Rotations Test." *The Chemical Educator* 2 (4). Springer: 1–17.
- [29] Vandenberg, S G, and A R Kuse. 1978. "Mental Rotations, a Group Test of Three-Dimensional spatial visualization." *Perceptual and Motor Skills* 47 (2): 599–604. doi:10.2466/pms.1978.47.2.599.
- [30] Sorby, Sheryl A. "Developing 3-D spatial visualization skills." *Engineering Design Graphics Journal* 63, no. 2 (2009).

- [31] Bennett, G. K., H. G. Seashore, and A. G. Wesman. 1973. Differential aptitude tests: forms S and T.
- [31] Ekstrom, Ruth B Rb, Jw John W French, Harry Hh Harman, and D Dermen. 1976. "Manual for Kit of Factor-Referenced Cognitive Tests." Princeton NJ Educational Testing Service 102 (41): 117. doi:10.1073/pnas.0506897102.
- [32] <https://knowledge.autodesk.com/support/autocad-Architecture/learn-explore/caas/CloudHelp/cloudhelp/2017/ENU/AutoCAD-Architecture/files/GUID-20918F56-EDE6-42B6-AE13-E06CB0C66D37-Htm.html>."
- [33] Kwon, Youngwook Paul, and Sara McMains. 2015. "An Automated Grading / Feedback System for 3-View Engineering Drawings Using RANSAC." ACM Conference on Learning at Scale, 157–66. doi:10.1145/2724660.2724682.
- [34] Goh, Kim Nee, Siti Rohkmah, Mohd Shukri, Rofans Belem, and Hilisebua Manao. 2013. "Automatic Assessment for Engineering Drawing," 497–507.
- [35] Shukur, Zarina, Yuwaldi Away, and Mohammad Asmady Dawari. 2004. "Computer-Aided Marking System for Engineering Drawing." In Proc. of Society for Information Technology & Teacher Education International Conference, 1852–57.
- [36] Hekman, Keith Alan. 2013. "Automated Grading of First Year Student CAD Work Automated Grading of First Year Student CAD Work."
- [37] Sri Datta Teja Karna. 2012. "ENHANCED AUTOCAD GRADING APPLICATION: BACK-END COMPONENT." San Diego State University.
- [38] Sri Krishna Kashyap Ivaturi. 2013. "ENHANCED AUTOCAD GRADING APPLICATION: FRONT-END." San Diego State University.
- [39] Wikipedia. n.d. "No Title." [https://en.wikipedia.org/wiki/Feature_\(computer_vision\)](https://en.wikipedia.org/wiki/Feature_(computer_vision)).
- [40] Harris, Chris, and Mike Stephens. 1988. "A Combined Corner and Edge Detector." Proceedings of the Alvey Vision Conference 1988, 147–51. doi:10.5244/C.2.23.

Appendix A: Software Code

MATLAB code for developing the Graphic User Interface:

```
function varargout = CAD_Grading(varargin)
% CAD_GRADING MATLAB code for CAD_Grading.fig
%   CAD_GRADING, by itself, creates a new CAD_GRADING or raises the
existing
%   singleton*.
%
%   H = CAD_GRADING returns the handle to a new CAD_GRADING or the handle
to
%   the existing singleton*.
%
%   CAD_GRADING('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in CAD_GRADING.M with the given input
arguments.
%
%   CAD_GRADING('Property','Value',...) creates a new CAD_GRADING or
raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before CAD_Grading_OpeningFcn gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to CAD_Grading_OpeningFcn via varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CAD_Grading

% Last Modified by GUIDE v2.5 21-Jul-2017 12:59:01

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
```

```

        'gui_OpeningFcn', @CAD_Grading_OpeningFcn, ...
        'gui_OutputFcn', @CAD_Grading_OutputFcn, ...
        'gui_LayoutFcn', [] , ...
        'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before CAD_Grading is made visible.
function CAD_Grading_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to CAD_Grading (see VARARGIN)

% Choose default command line output for CAD_Grading
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CAD_Grading wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = CAD_Grading_OutputFcn(hObject, eventdata, handles)

```

```

% varargout    cell array for returning output args (see VARARGOUT);
% hObject     handle to figure
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function axes2D_CreateFcn(hObject, eventdata, handles)
% hObject     handle to axes2D (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes2D

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject     handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
handles.pop=get(handles.popupmenu1,'value'); % chosen value
guidata(hObject,handles);
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1
contents as cell array
%         contents{get(hObject,'Value')} returns selected item from popupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject     handle to popupmenu1 (see GCBO)
% eventdata   reserved - to be defined in a future version of MATLAB
% handles     empty - handles not created until after all CreateFcns called

```

```

% Hint: popupmenu controls usually have a white background on Windows.
%     See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUiControlBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in load2D.
function load2D_Callback(hObject, eventdata, handles)
% hObject    handle to load2D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global im im2
[path,user_cancel]=imgetfile();
if user_cancel
    msgbox(sprintf('Error'),'Error','Error');
    return
end
im=imread(path);
% im=im2double(im); %converts to double
handles.im2=im; %for backup process :)
guidata(hObject,handles);
axes(handles.axes2D);
imshow(im);

% --- Executes on button press in load3D.
function load3D_Callback(hObject, eventdata, handles)
% hObject    handle to load3D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global img img2
[path,user_cancel]=imgetfile();
if user_cancel
    msgbox(sprintf('Error'),'Error','Error');
    return
end

```

```

img=imread(path);
% img=im2double(img); %converts to double
handles.img2=img; %for backup process
guidata(hObject,handles);
axes(handles.axes3D);
imshow(img);

% --- Executes during object creation, after setting all properties.
function axes3D_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes3D (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate axes3D

% --- Executes on button press in evaluate.
function evaluate_Callback(hObject, eventdata, handles)
% hObject    handle to evaluate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.pop==2
    c=Shaft(handles.im2);
elseif handles.pop==3
    c=Flange(handles.im2);
elseif handles.pop==4
    c=TeeSlide(handles.im2);
elseif handles.pop==5
    c=ToolBlock(handles.im2);
elseif handles.pop==6
    c=PunchPlate(handles.im2);
elseif handles.pop==7
    c=SlideBlock(handles.im2);
end

if handles.pop==2
    d=Shaft_3D(handles.img2);

```

```

elseif handles.pop==3
    d=Flange_3D(handles.img2);
elseif handles.pop==4
    d=TeeSlide_3D(handles.img2);
elseif handles.pop==5
    d=ToolBlock_3D(handles.img2);
elseif handles.pop==6
    d=PunchPlate_3D(handles.img2);
elseif handles.pop==7
    d=SlideBlock_3D(handles.img2);
end

if c==1
    if d==1
        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 3. Your sectional view is
without hatching and features.');
```

```

        elseif d==2
            a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 3. Your sectional
view is without hatching and features.');
```

```

        elseif d==3
            a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 3. Your sectional view is
without hatching and features.');
```

```

        elseif d==4
            a=sprintf('Your score for 3D drawing is 5. Your 3D model is
completely correct. Your score for sectional view is 3. Your sectional view is
without hatching and features.');
```

```

        else
            a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 3. Your sectional view is
without hatching and features.');
```

```

    end
elseif c==2
    if d==1
```

```

        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 4. Your sectional view is
without hatching.');
```

```

    elseif d==2
        a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 4. Your sectional
view is without hatching.');
```

```

    elseif d==3
        a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 4. Your sectional view is
without hatching.');
```

```

    elseif d==4
        a=sprintf('Your score for 3D drawing is 5. Your 3D model is
completely correct. Your score for sectional view is 4. Your sectional view is
without hatching.');
```

```

    else
        a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 4. Your sectional view is
without hatching.');
```

```

    end
elseif c==3
    if d==1
        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 4. Your sectional view is
without features.');
```

```

    elseif d==2
        a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 4. Your sectional
view is without features.');
```

```

    elseif d==3
        a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 4. Your sectional view is
without features.');
```

```

    elseif d==4
        a=sprintf('Your score for 3D drawing is 5. Your 3D model is
completely correct. Your score for sectional view is 4. Your sectional view is
without features.');
```

```

    else
```

```

        a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 4. Your sectional view is
without features.');
```

end

```
elseif c==4
    if d==1
        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 5. Your sectional view is
completely correct.');
```

elseif d==2

```
        a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 5. Your sectional
view is completely correct.');
```

elseif d==3

```
        a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 5. Your sectional view is
completely correct.');
```

elseif d==4

```
        a=sprintf('Your score for 3D drawing is 5. Your 3D model is
completely correct. Your score for sectional view is 5. Your sectional view is
completely correct.');
```

else

```
        a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 5. Your sectional view is
completely correct.');
```

end

```
elseif c==5
    if d==1
        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 1. Your sectional view is
completely wrong.');
```

elseif d==2

```
        a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 1. Your sectional
view is completely wrong.');
```

elseif d==3

```

        a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 1. Your sectional view is
completely wrong.');
```

```

        elseif d==4
            a=sprintf('Your score for 3D drawing is 5. Your 3D model is
competely correct. Your score for sectional view is 1. Your sectional view is
completely wrong.');
```

```

        else
            a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 1. Your sectional view is
completely wrong.');
```

```

        end
elseif c==6
    if d==1
        a=sprintf('Your score for 3D drawing is 3. Your 3D drawing is in
wrong orientation. Your score for sectional view is 2. You have the wrong
sectional view.');
```

```

        elseif d==2
            a=sprintf('Your score for 3D drawing is 4. Your 3D drawing has an
incorrect cutting plane. Your score for sectional view is 2. You have the
wrong sectional view.');
```

```

        elseif d==3
            a=sprintf('Your score for 3D drawing is 2. Your 3D drawing has
missing features. Your score for sectional view is 2. You have the wrong
sectional view.');
```

```

        elseif d==4
            a=sprintf('Your score for 3D drawing is 5. Your 3D model is
competely correct. Your score for sectional view is 2. You have the wrong
sectional view.');
```

```

        else
            a=sprintf('Your score for 3D drawing is 1. You have a completely
wrong 3D drawing. Your score for sectional view is 2. You have the wrong
sectional view.');
```

```

        end
end

set(handles.edit1, 'string', a);

```

```

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function axes8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
axes(hObject);
imshow('774372.jpg');

% Hint: place code in OpeningFcn to populate axes8

```

MATLAB code for functions of CAD models:

Tool Block (2D):

```
function c=ToolBlock(I1_gray)

%% Load all possible cases
I2=imread('3_ToolBlock_bb.bmp'); % Without hatching and features
I3=imread('4_ToolBlock_bb.bmp'); % Without hatching
I4=imread('4_ToolBlock_2_bb.bmp'); % Without features
I5=imread('5_ToolBlock_bb.bmp'); % Correct drawing
I6=imread('2_ToolBlock_bb.bmp'); % Wrong View
I7=imread('2_ToolBlock_2_bb.bmp'); % Wrong View
I8=imread('2_ToolBlock_3_bb.bmp'); % Wrong View
I9=imread('2_ToolBlock_4_bb.bmp'); % Wrong View

%% Applying Bounding Box function to loaded image
I1_gray=im2bw(I1_gray);
I1_gray=imcomplement(I1_gray);
[Xdim Ydim]=size(I1_gray);
j=1;
for i=1:Xdim
    k=find(I1_gray(i, :)==1);
    if size(k)~=0
        a(j,1)=[i];
        a(j,2)=[k(1)];
        a(j,3)=[k(end)];
        j=j+1;
    end
end
startX=a(1,1);
endX=a(end,1);
startY=min(a(:,2));
endY=max(a(:,3));
I1_gray=I1_gray(startX:endX,startY:endY);
```

```

%% Grayscale conversion
I2_gray= rgb2gray (I2);
I3_gray= rgb2gray (I3);
I4_gray= rgb2gray (I4);
I5_gray= rgb2gray (I5);
I6_gray= rgb2gray (I6);
I7_gray= rgb2gray (I7);
I8_gray= rgb2gray (I8);
I9_gray= rgb2gray (I9);

%% Detect Features
points1 = detectHarrisFeatures (I1_gray);
points2 = detectHarrisFeatures (I2_gray);
points3 = detectHarrisFeatures (I3_gray);
points4 = detectHarrisFeatures (I4_gray);
points5 = detectHarrisFeatures (I5_gray);
points6 = detectHarrisFeatures (I6_gray);
points7 = detectHarrisFeatures (I7_gray);
points8 = detectHarrisFeatures (I8_gray);
points9 = detectHarrisFeatures (I9_gray);

% Extract neighbourhood features
[features1,valid_points1] = extractFeatures (I1_gray,points1);
[features2,valid_points2] = extractFeatures (I2_gray,points2);
[features3,valid_points3] = extractFeatures (I3_gray,points3);
[features4,valid_points4] = extractFeatures (I4_gray,points4);
[features5,valid_points5] = extractFeatures (I5_gray,points5);
[features6,valid_points6] = extractFeatures (I6_gray,points6);
[features7,valid_points7] = extractFeatures (I7_gray,points7);
[features8,valid_points8] = extractFeatures (I8_gray,points8);
[features9,valid_points9] = extractFeatures (I9_gray,points9);

%% Comparing the loaded image with various cases in the grading rubric

if valid_points1.Count==valid_points2.Count
    c=1;

```

```

elseif valid_points1.Count==valid_points3.Count
    c=2;

elseif valid_points1.Count==valid_points4.Count
    c=3;

elseif valid_points1.Count==valid_points5.Count
    c=4;

elseif valid_points1.Count==valid_points6.Count ||
valid_points1.Count==valid_points7.Count ||
valid_points1.Count==valid_points8.Count ||
valid_points1.Count==valid_points9.Count ||
    c=6;

else
    c=5;
end

return

```

Tool Block (3D):

```

function d=ToolBlock_3D(I1)

%% Load all possible cases
I2=imread('3_3D_Tool Block.png'); % Without hatching and features
I3=imread('4_3D_Tool Block.png'); % Without hatching
I4=imread('2_3D_Tool Block.png'); % Without features
I5=imread('5_3D_Tool Block.png'); % Correct drawing

%% Applying Bounding Box function to loaded image
I1_gray=im2bw(I1);
I1_gray=imcomplement(I1_gray);
[Xdim Ydim]=size(I1_gray);
j=1;

```

```

for i=1:Xdim
    k=find(I1_gray(i,:)==1);
    if size(k)~=0
        a(j,1)=[i];
        a(j,2)=[k(1)];
        a(j,3)=[k(end)];
        j=j+1;
    end
end

startX=a(1,1);
endX=a(end,1);
startY=min(a(:,2));
endY=max(a(:,3));
I1_gray=I1_gray(startX:endX,startY:endY);

%% Grayscale conversion
I2_gray=rgb2gray(I2);
I3_gray=rgb2gray(I3);
I4_gray=rgb2gray(I4);
I5_gray=rgb2gray(I5);

%% Harris Features
points1 = detectHarrisFeatures(I1_gray);
points2 = detectHarrisFeatures(I2_gray);
points3 = detectHarrisFeatures(I3_gray);
points4 = detectHarrisFeatures(I4_gray);
points5 = detectHarrisFeatures(I5_gray);

% Extract neighbourhood features
[features1,valid_points1] = extractFeatures(I1_gray,points1);
[features2,valid_points2] = extractFeatures(I2_gray,points2);
[features3,valid_points3] = extractFeatures(I3_gray,points3);
[features4,valid_points4] = extractFeatures(I4_gray,points4);
[features5,valid_points5] = extractFeatures(I5_gray,points5);

if valid_points1.Count==valid_points2.Count

```

```

        d=1;
elseif valid_points1.Count==valid_points3.Count
    d=2;
elseif valid_points1.Count==valid_points4.Count
    d=3;
elseif valid_points1.Count==valid_points5.Count
    d=4;
else
    d=5;
end

return

```

MATLAB code for developing the Graphic User Interface for grading multiple files:

```

function varargout = CAD_Grading_Multiple(varargin)
% CAD_GRADING_MULTIPLE MATLAB code for CAD_Grading_Multiple.fig
%     CAD_GRADING_MULTIPLE, by itself, creates a new CAD_GRADING_MULTIPLE or
raises the existing
%     singleton*.
%
%     H = CAD_GRADING_MULTIPLE returns the handle to a new
CAD_GRADING_MULTIPLE or the handle to
%     the existing singleton*.
%
%     CAD_GRADING_MULTIPLE('CALLBACK',hObject,eventData,handles,...) calls
the local
%     function named CALLBACK in CAD_GRADING_MULTIPLE.M with the given input
arguments.
%
%     CAD_GRADING_MULTIPLE('Property','Value',...) creates a new
CAD_GRADING_MULTIPLE or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before CAD_Grading_Multiple_OpeningFcn gets called.
An
%     unrecognized property name or invalid value makes property application

```

```

%      stop. All inputs are passed to CAD_Grading_Multiple_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help CAD_Grading_Multiple

% Last Modified by GUIDE v2.5 21-Jul-2017 23:46:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',  gui_Singleton, ...
                  'gui_OpeningFcn', @CAD_Grading_Multiple_OpeningFcn, ...
                  'gui_OutputFcn',  @CAD_Grading_Multiple_OutputFcn, ...
                  'gui_LayoutFcn',  [] , ...
                  'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before CAD_Grading_Multiple is made visible.
function CAD_Grading_Multiple_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure

```

```

% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to CAD_Grading_Multiple (see VARARGIN)

% Choose default command line output for CAD_Grading_Multiple
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes CAD_Grading_Multiple wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = CAD_Grading_Multiple_OutputFcn(hObject, eventdata,
handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject handle to popupmenu2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
handles.pop=get(handles.popupmenu2,'value'); % chosen value
guidata(hObject,handles);
% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2
contents as cell array
% contents{get(hObject,'Value')} returns selected item from popupmenu2

```

```

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%         See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.pop==2
    c=ShaftMultiple();
elseif handles.pop==3
    c=FlangeMultiple();
elseif handles.pop==4
    c=TeeSlideMultiple();
elseif handles.pop==5
    c=ToolBlockMultiple();
elseif handles.pop==6
    c=PunchPlateMultiple();
elseif handles.pop==7
    c=SlideBlockMultiple();
end
f='The results are stored in an Excel File';
if get(hObject,'Value')

```

```

        set(handles.edit2,'string',f)%Display text only if a folder is selected
else
        set(handles.edit2,'Visible','Off')
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%        See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
axes(hObject);
imshow('774372.jpg');% Hint: place code in OpeningFcn to populate axes2

```

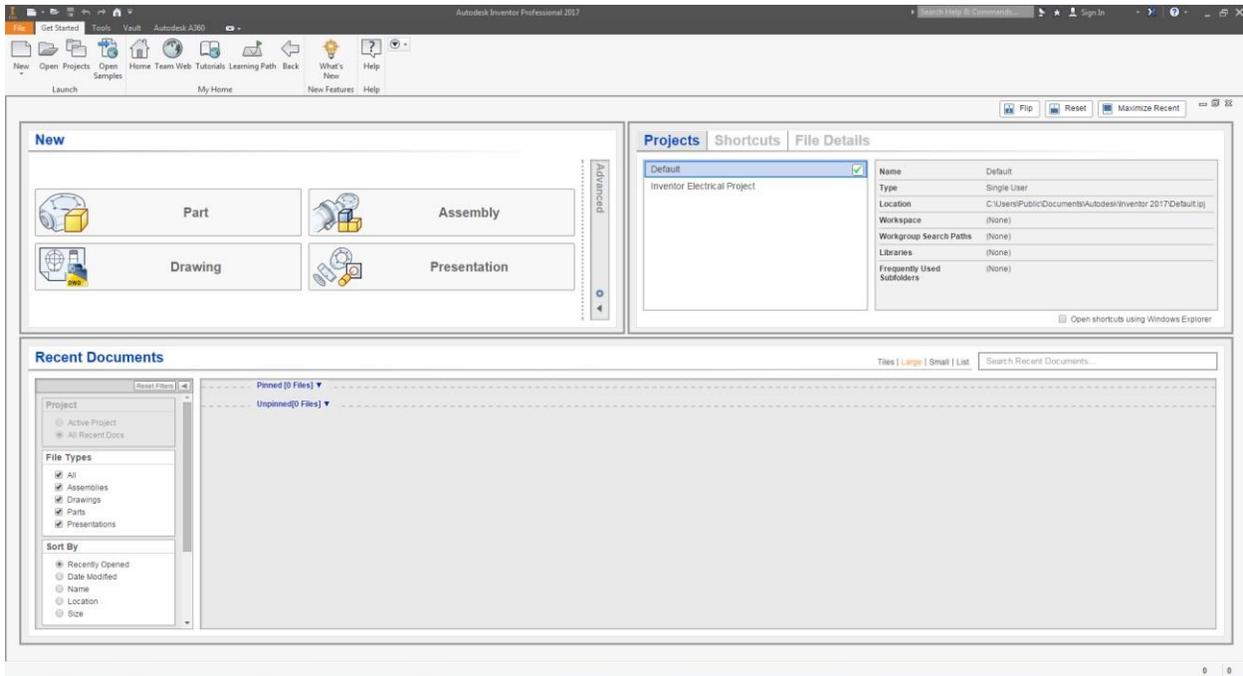
Appendix B: Instructions to create 3D model and sectional view

Instructions for Flange:

Follow the instructions given below to develop a 3D model for Flange object. The instructions are created for Autodesk Inventor CAD software, but can be followed for any similar CAD software.

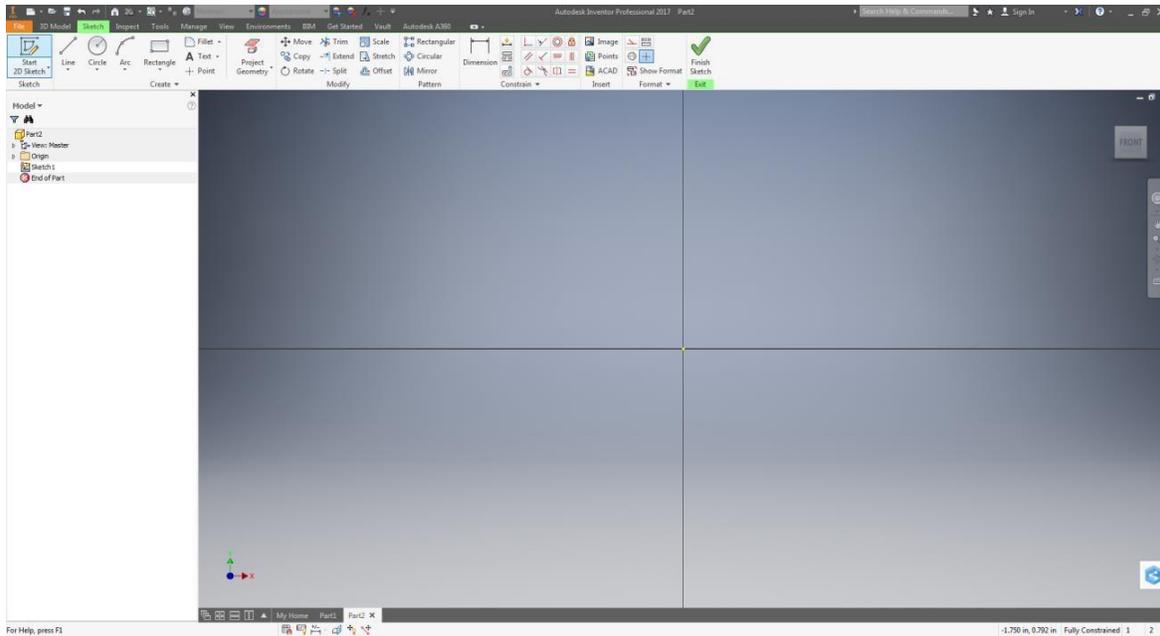
Step 1:

Open Autodesk Inventor Professional. You should see the following screen:

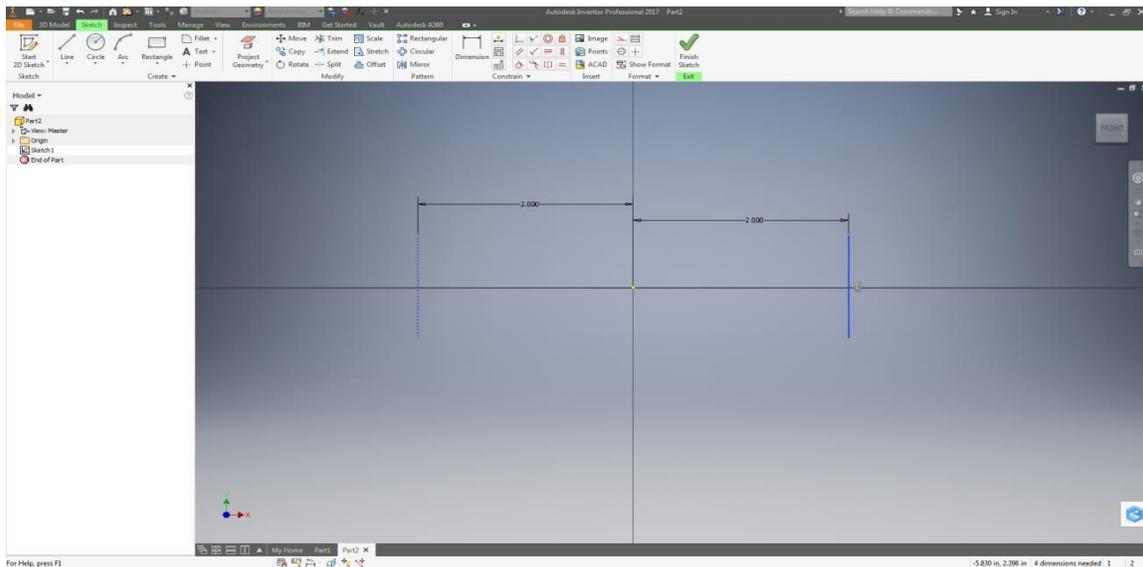


Step 4:

Click on the XY plane. You will see the following screen once you do that:

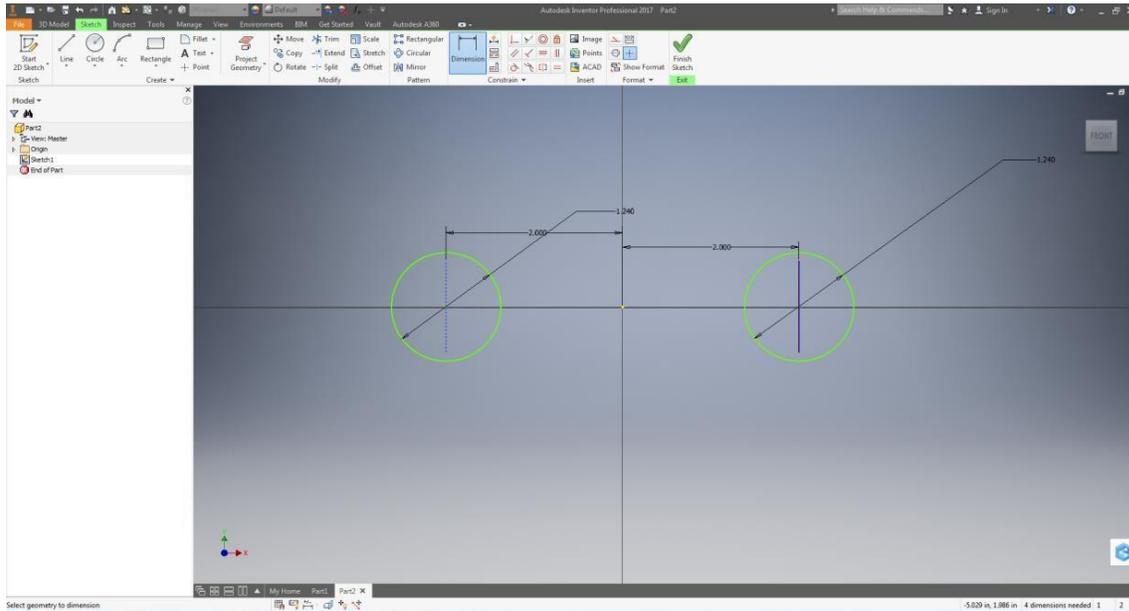


Step 5: Draw two vertical lines 2 of random length, each of which is 2” away from the center point as shown in the figure below. You can click ‘Dimension’ to enter the dimensions. Further right click on the line and choose ‘Construction’ to transform these lines into construction lines.



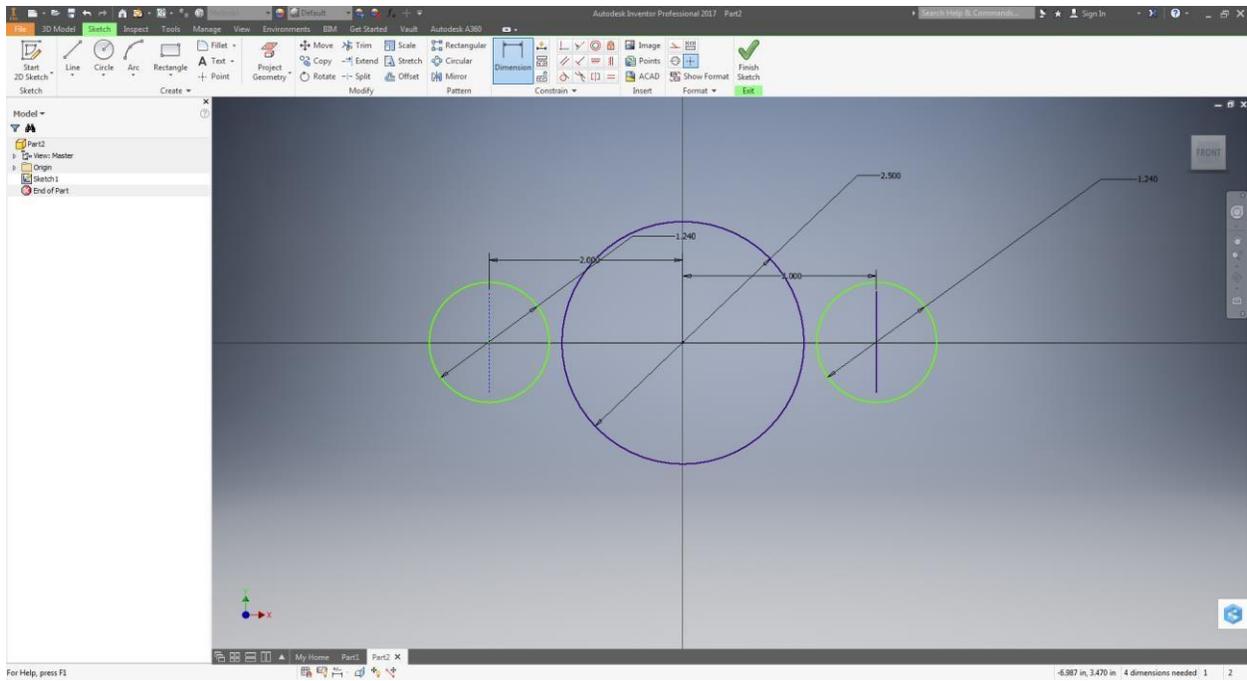
Step 6:

Draw two circles on the intersection point of the drawn vertical lines and the horizontal axis. The diameter of the circles is 1.24”.



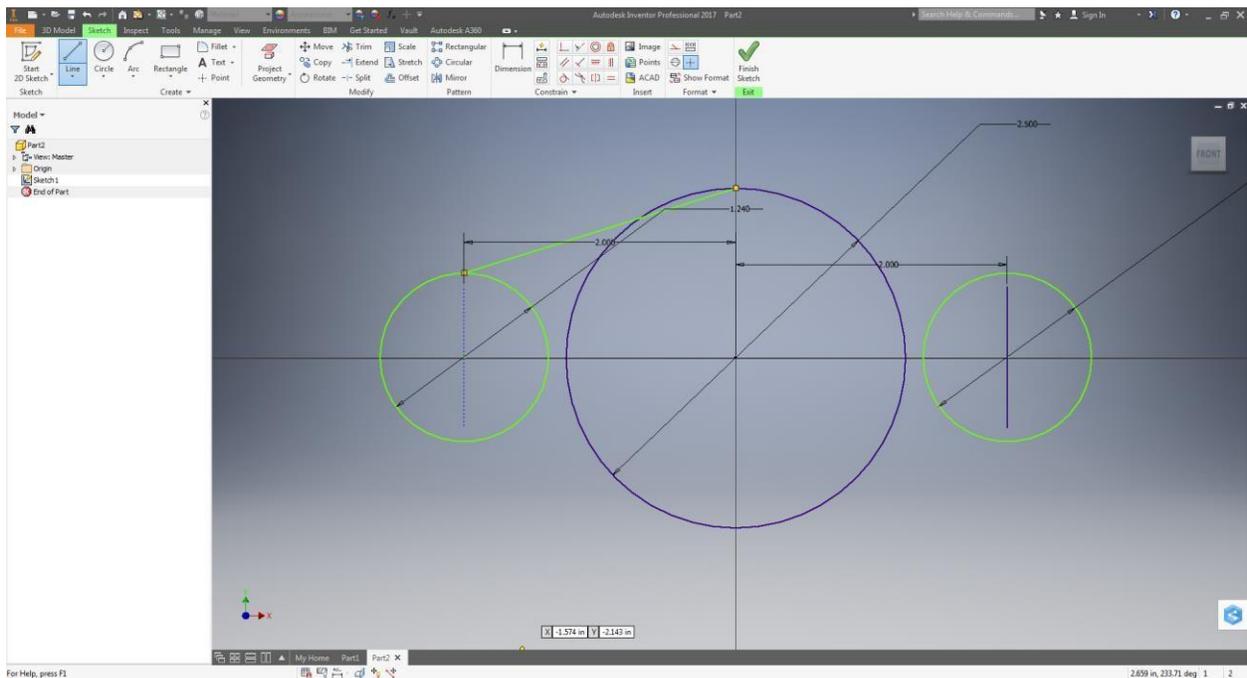
Step 7:

Draw a circle with the origin as the center and diameter 2.5” as shown:



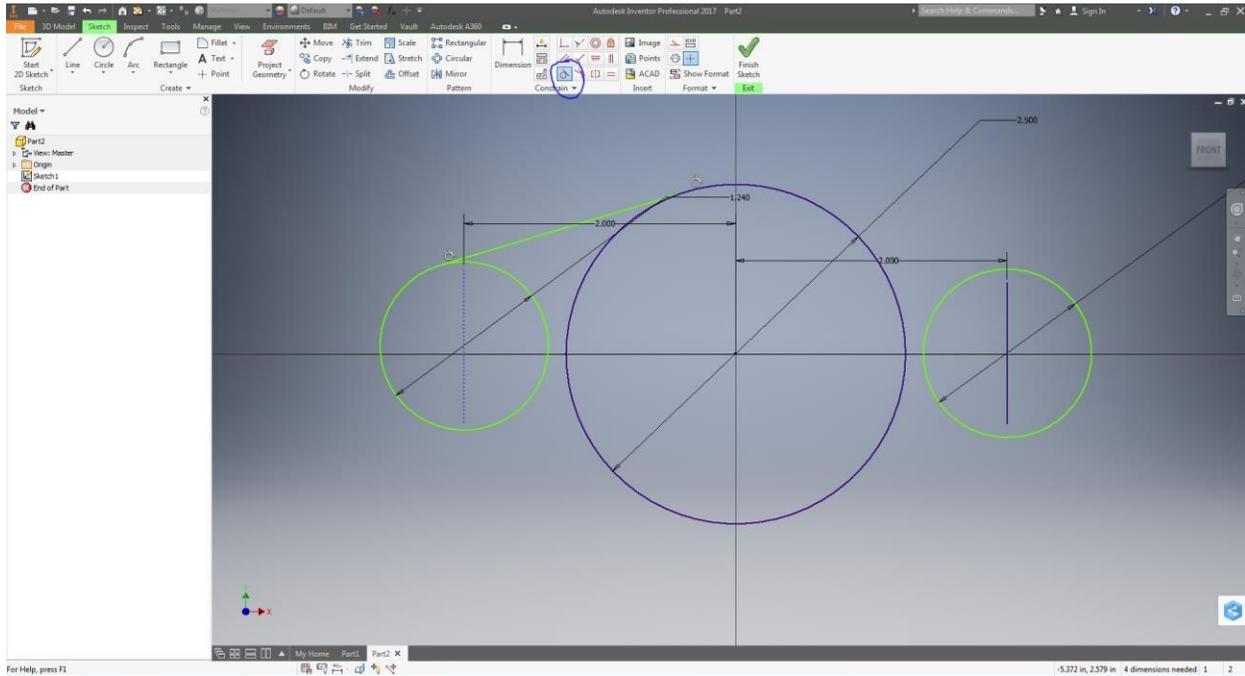
Step 8:

Draw a line joining the two topmost points on the left and middle circles as shown in figure below:



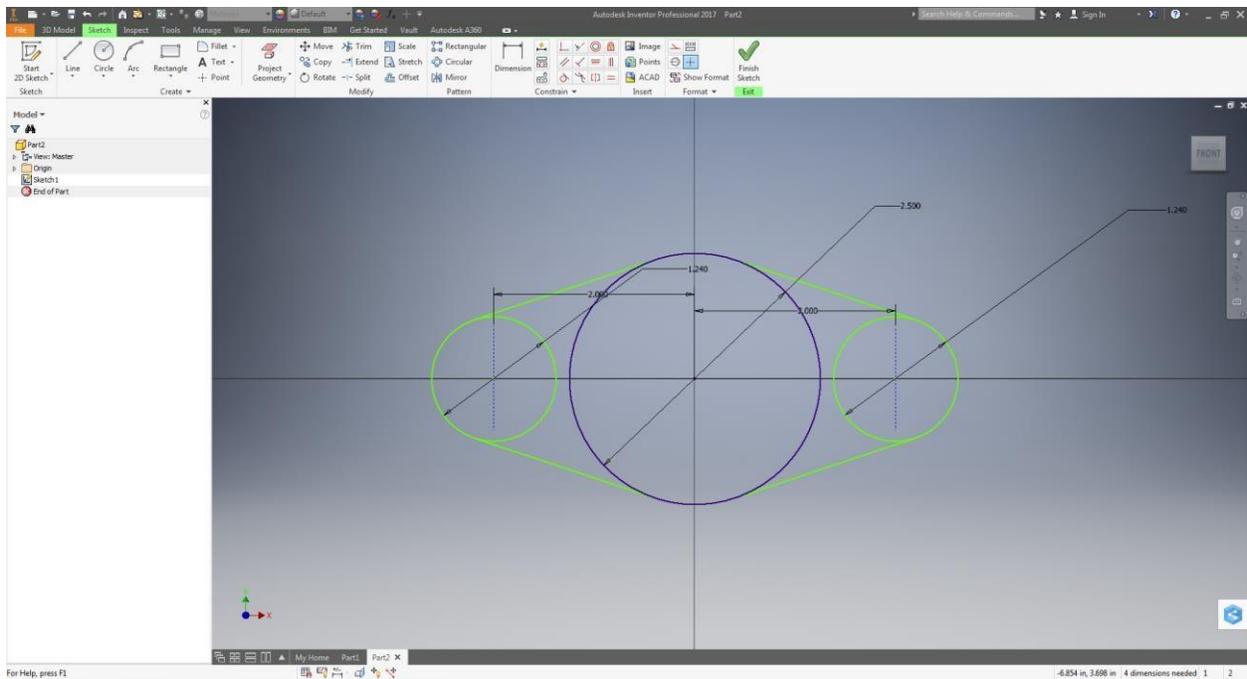
Step 9:

In the 'Constrain' toolbox, select 'Tangent' constraint (circled in the figure below). Now, select the line and the left circle to make it tangent to the left circle. Follow the same procedure for the line and larger circle and your drawing will look as shown in figure below:



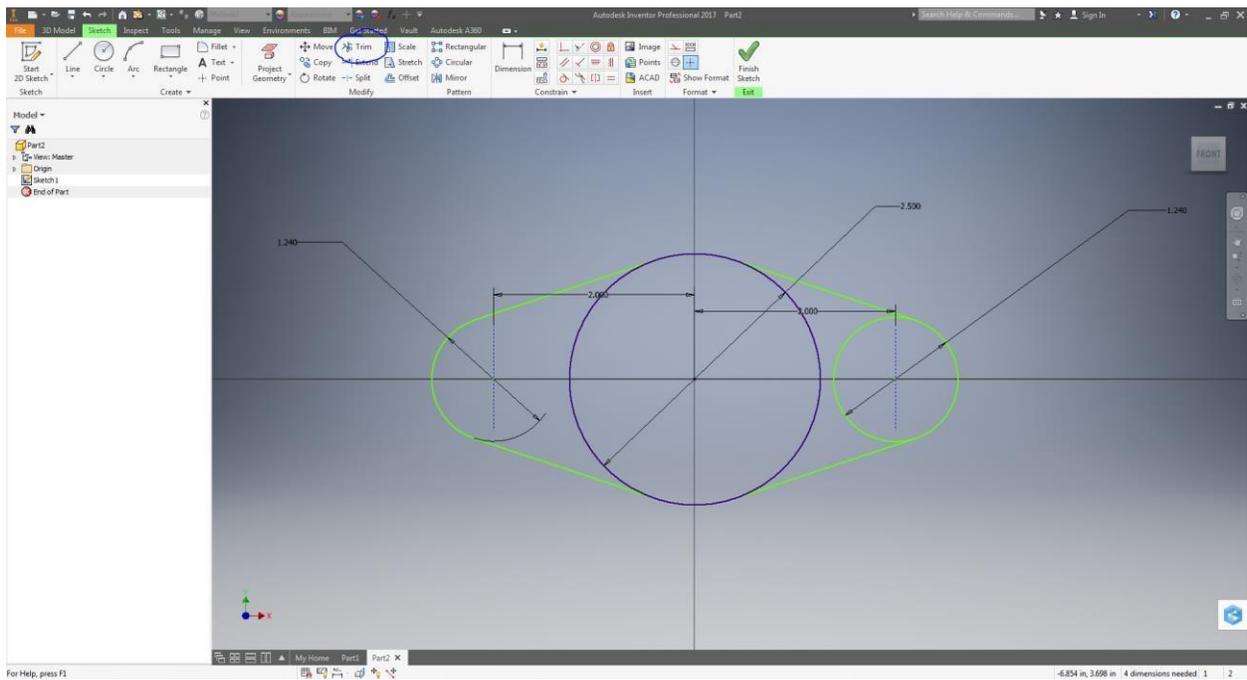
Step 10:

Do the same procedure for all the four sides of the sketch and the drawing will look as shown:



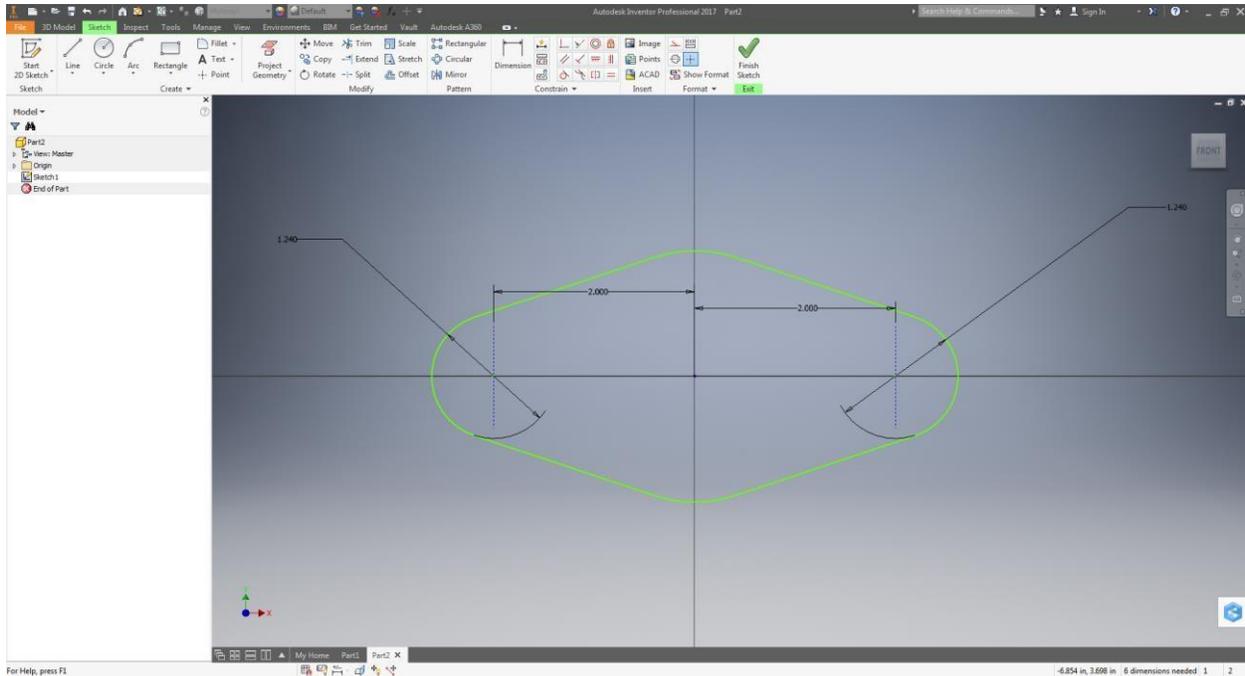
Step 11:

To get an outline of the object, select 'Trim' feature from the 'Modify' toolbox and select the inner part of left circle to trim it. The resulting sketch is as shown:



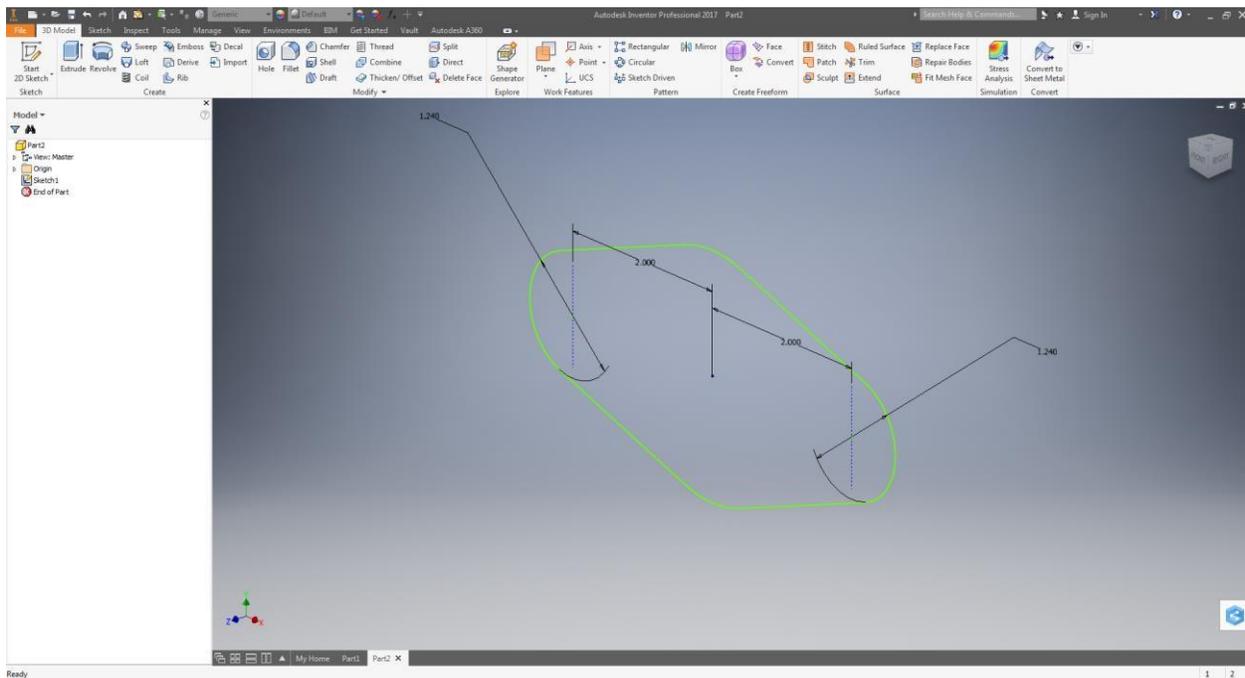
Step 12:

Perform the same operation for all interior curves to get an outline of the model as shown:



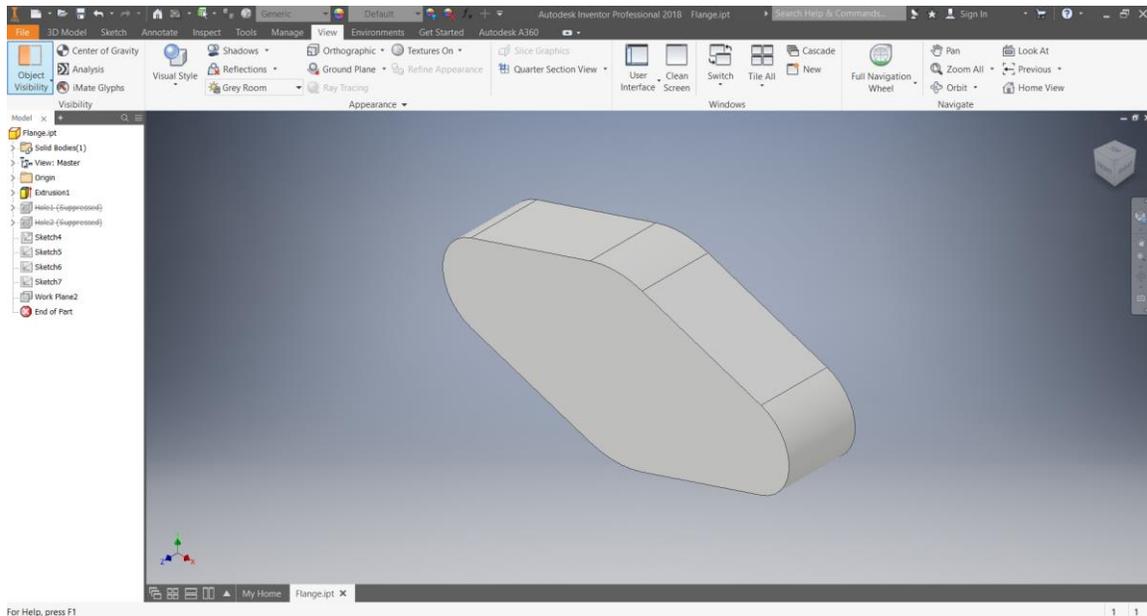
Step 13:

Now click on “Finish Sketch” and you will see the following screen:



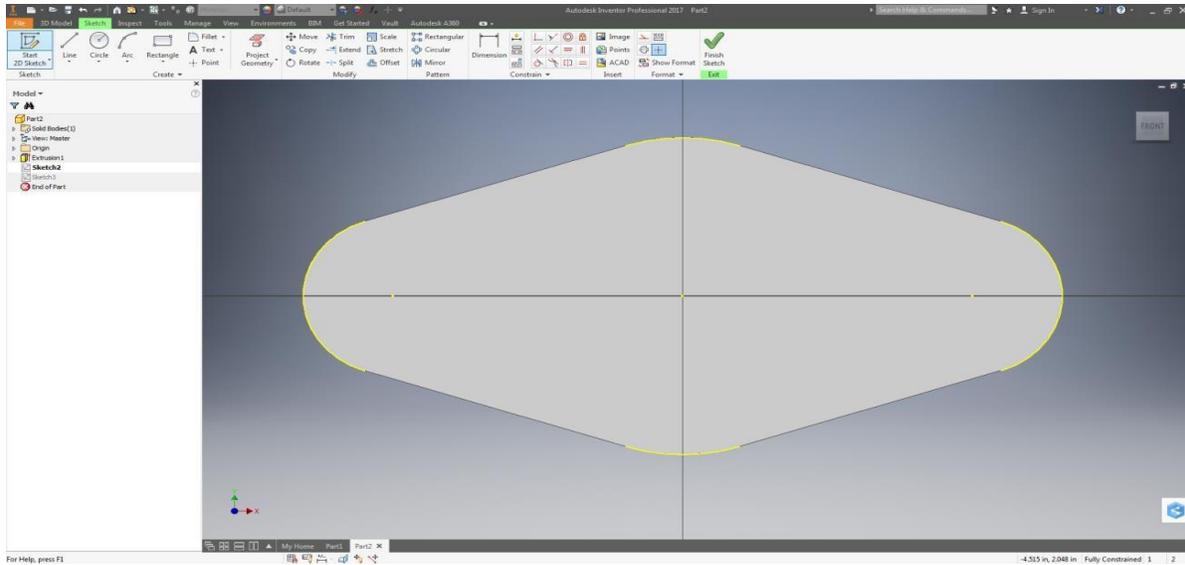
Step 14:

Click on “Extrude”. Then select the profile (model outline). Give the length you need for the extrude (1”) and the press enter. You will see the following

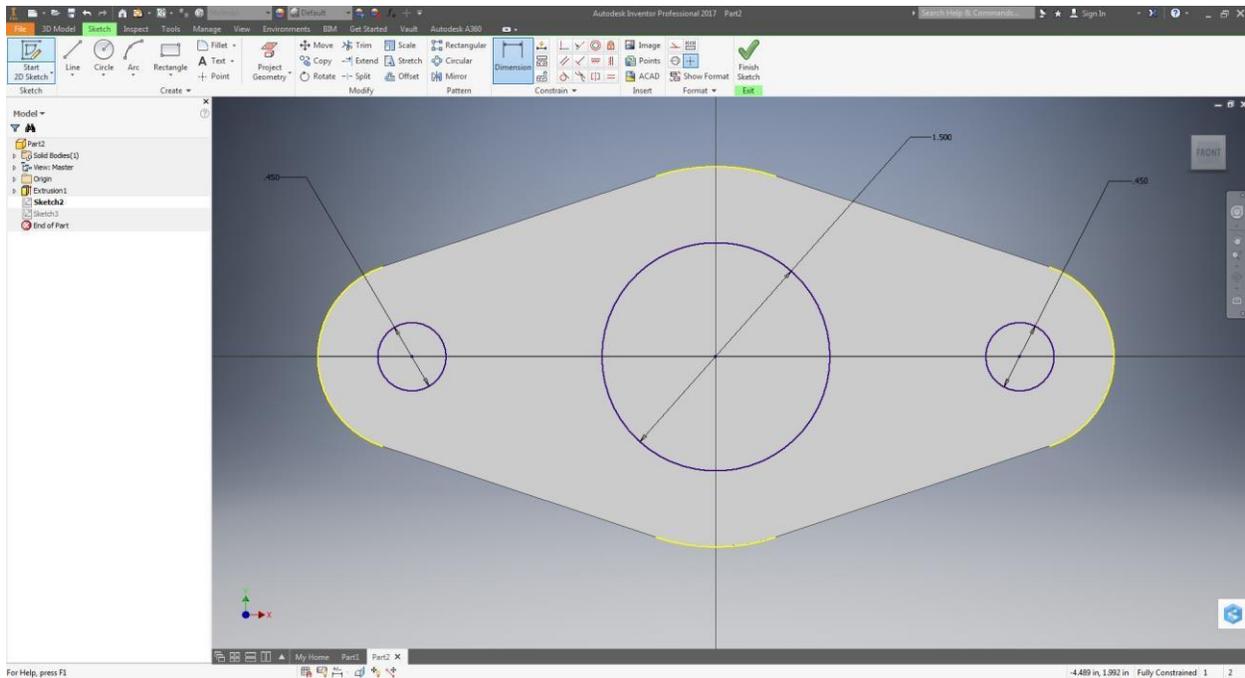


Step 15:

Select the face of the model and click ‘Sketch’. Now click ‘Project Geometry’ and select the left and right circular sections as well as the middle circular section. This will highlight the center of the circles as shown.

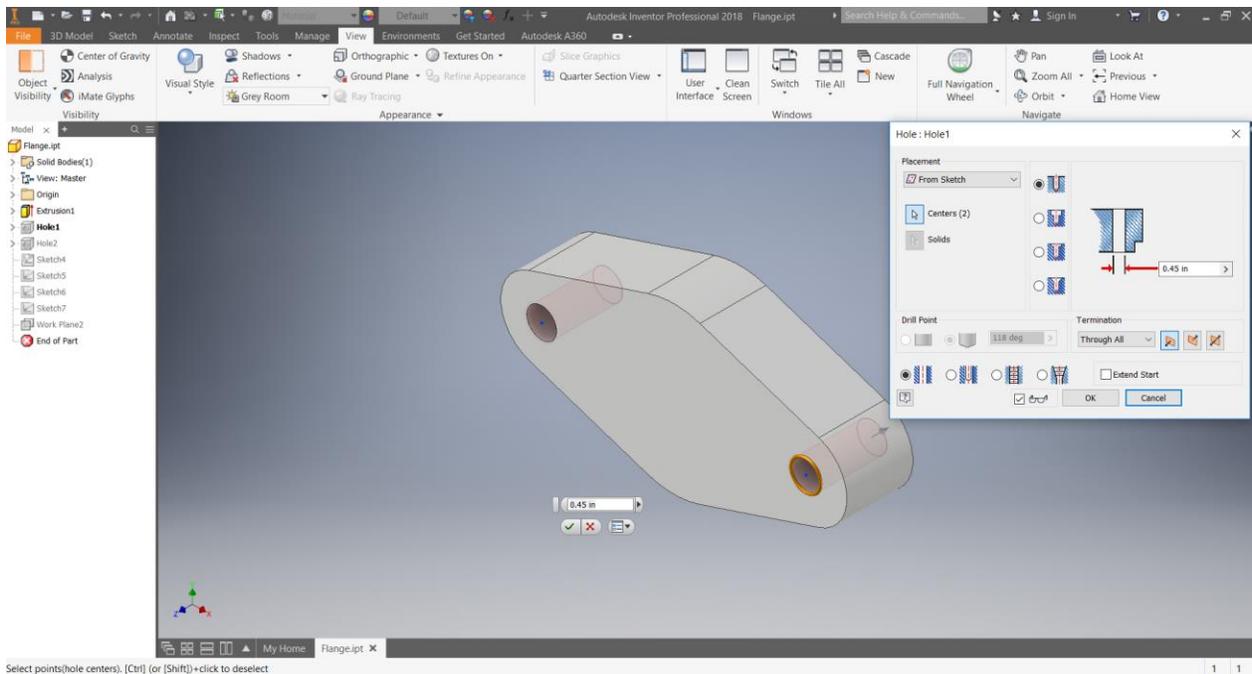


Step 16: Draw a circle of 0.45” diameter on the left and right highlighted centers and a 1.5” diameter circle on the middle center and click finish



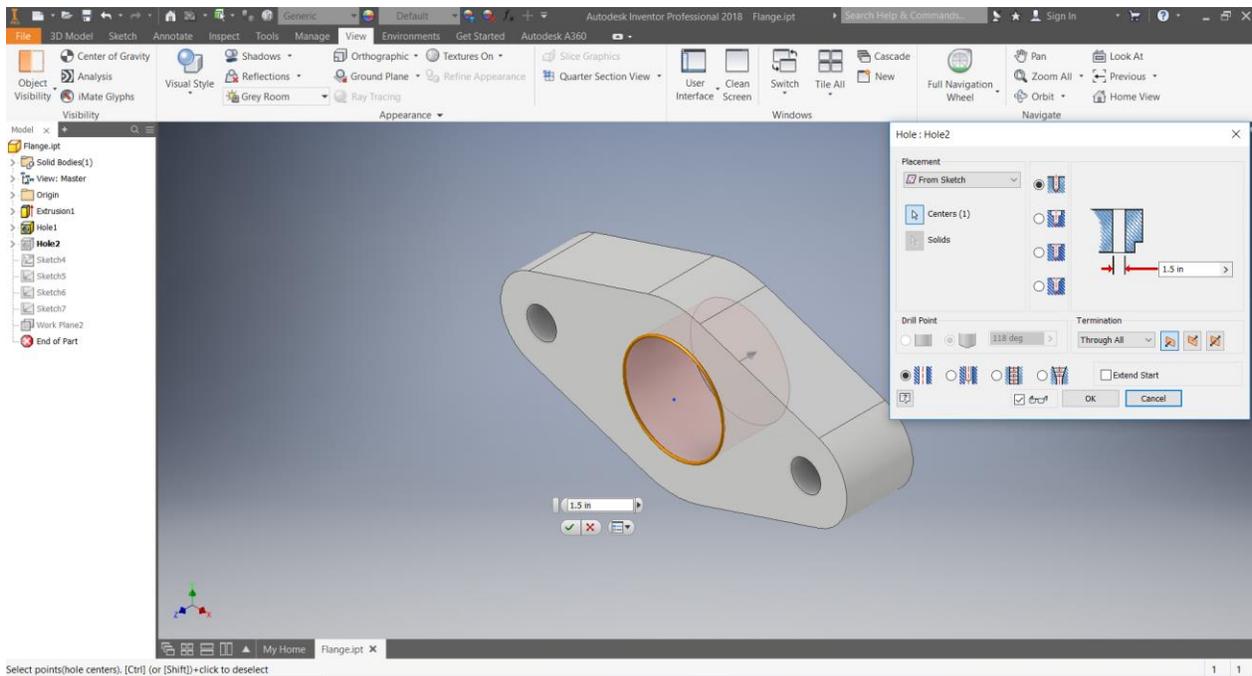
Step 17:

Click the ‘Hole’ feature and the dialog box appears as shown. Select the center of circles on the left and right side (0.45” Diameter Circle). Make sure that the hole diameter is 0.45”.



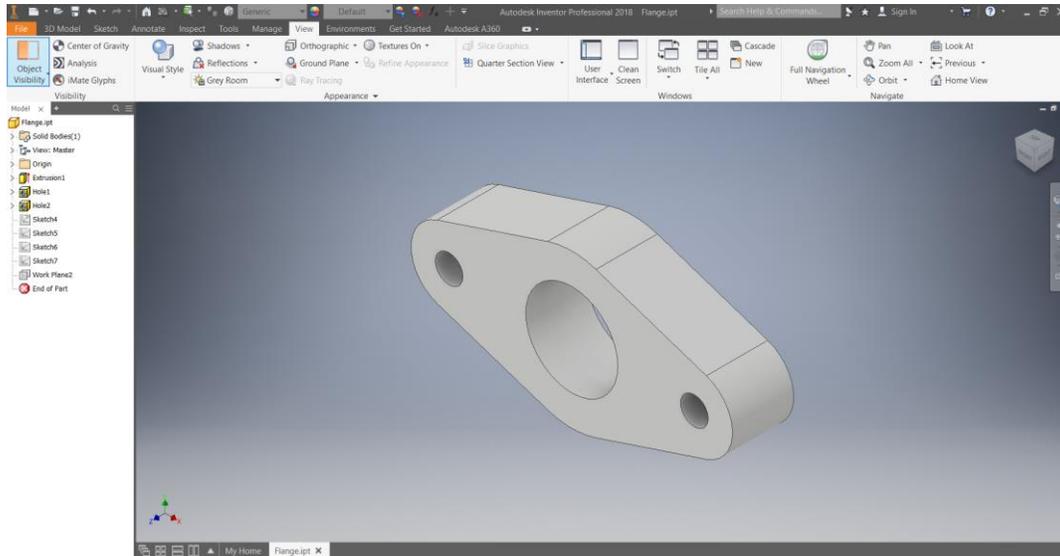
Step 18:

Repeat the above procedure for Middle circle and make sure diameter of hole is 1.5”.



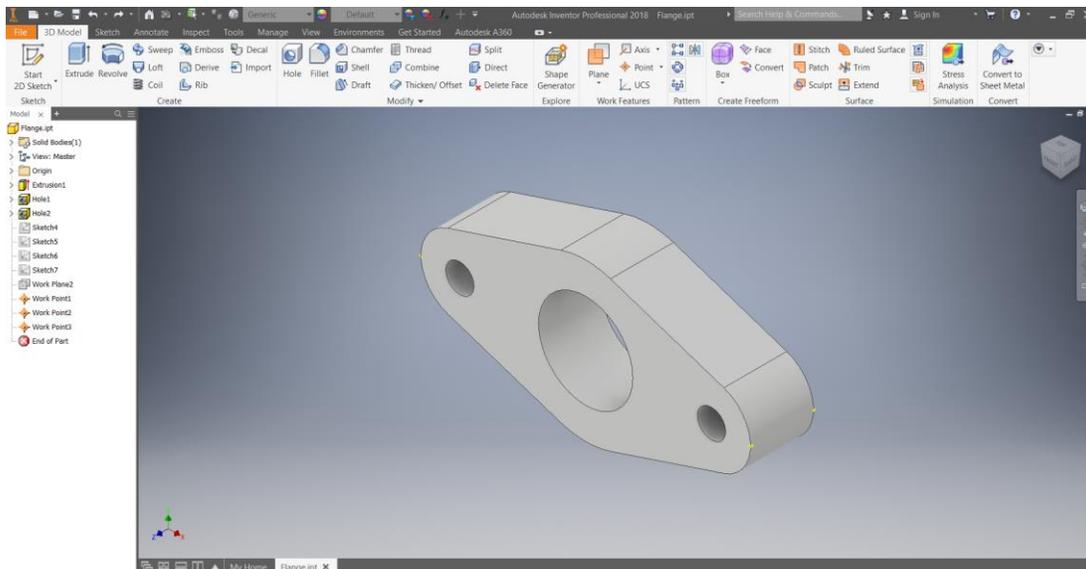
Step 19:

The final object looks as shown below:



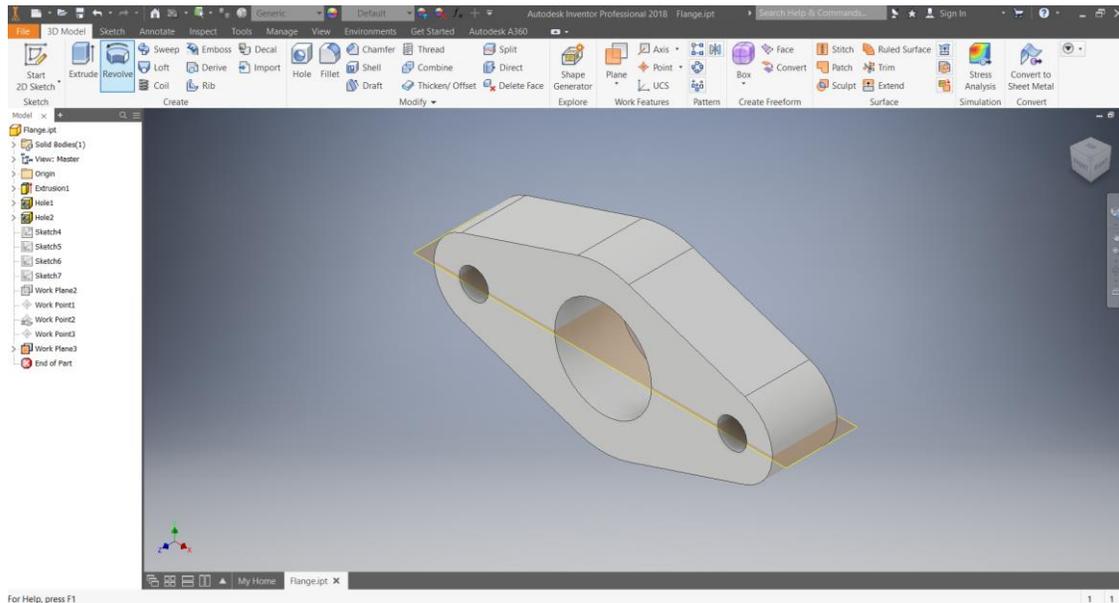
Step 20:

We will now create a plane using which we need to cut the object and create a sectional view. In 3D Model tab, select point option and create 3 points as shown in figure below (points marked in yellow)



Step 21:

In order to create a plane using the three points created in the previous step, select Plane option in the 3D Model tab. From the various options available, select 'through three points' option. Now select the three points and a plane passing through these points is created as shown in figure below.



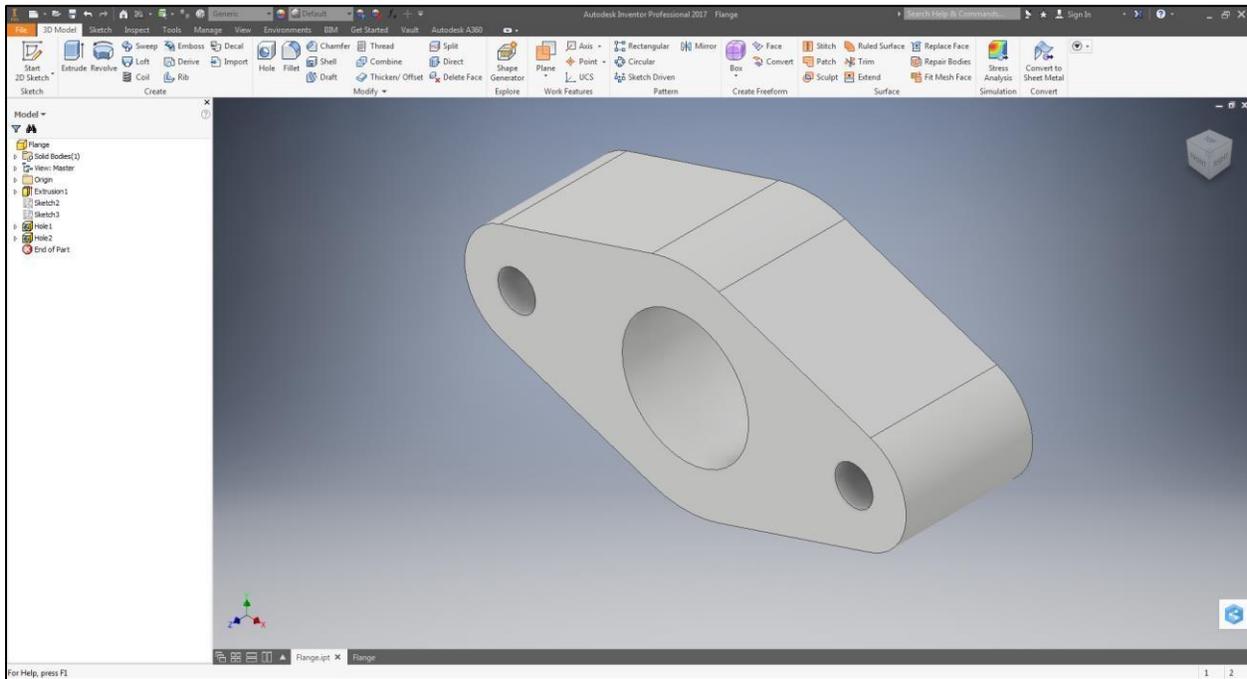
Step 22:

Click the house icon on top left corner of the orientation cube to load object in default isometric orientation

Instructions for Sectional View:

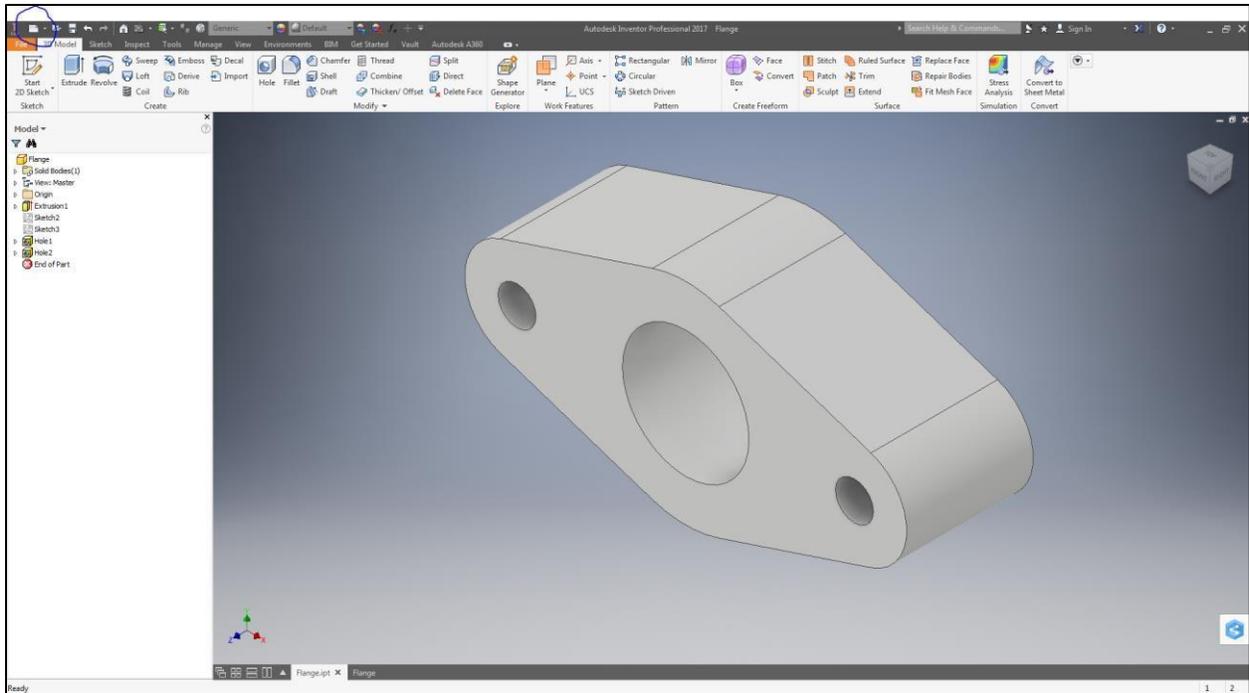
Step 1:

Open the Inventor .ipt file for Flange.



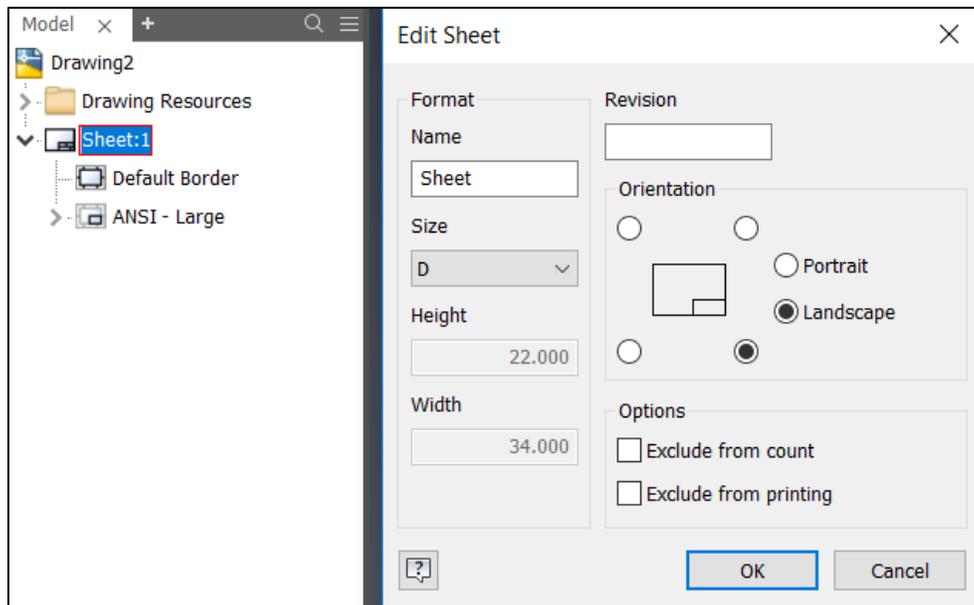
Step 2:

Click on 'New' in the topmost left corner of the screen as shown and select 'Drawing'. A new window shall appear



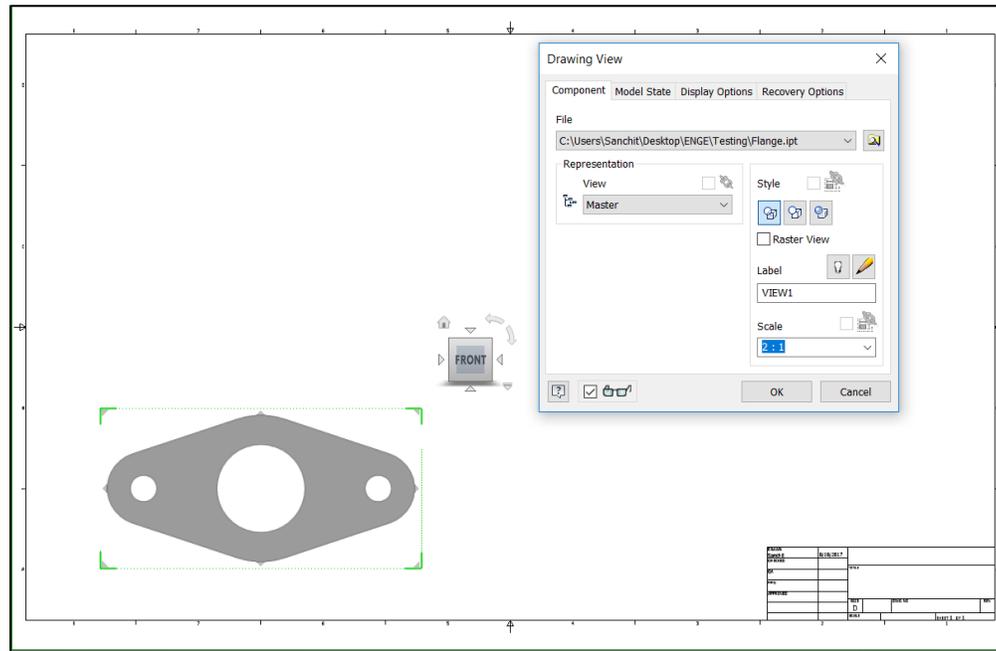
Step 3:

To create sectional view, you need to use ANSI D sheet. To edit the size of the sheet, right click the Sheet 1 tab in the tree on left side of the screen and select 'Edit Sheet'. In the edit window, select the size of the sheet as 'D' and Landscape orientation.



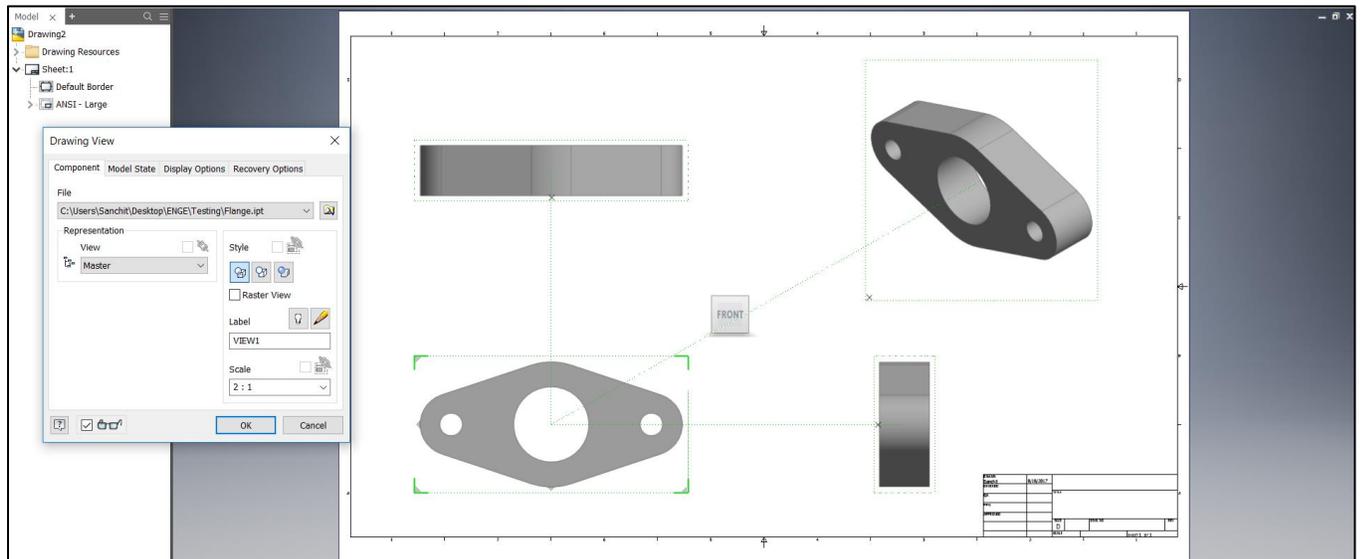
Step 4:

In the new drawing window, click 'Base' in the top left corner and a dialog box will appear as shown



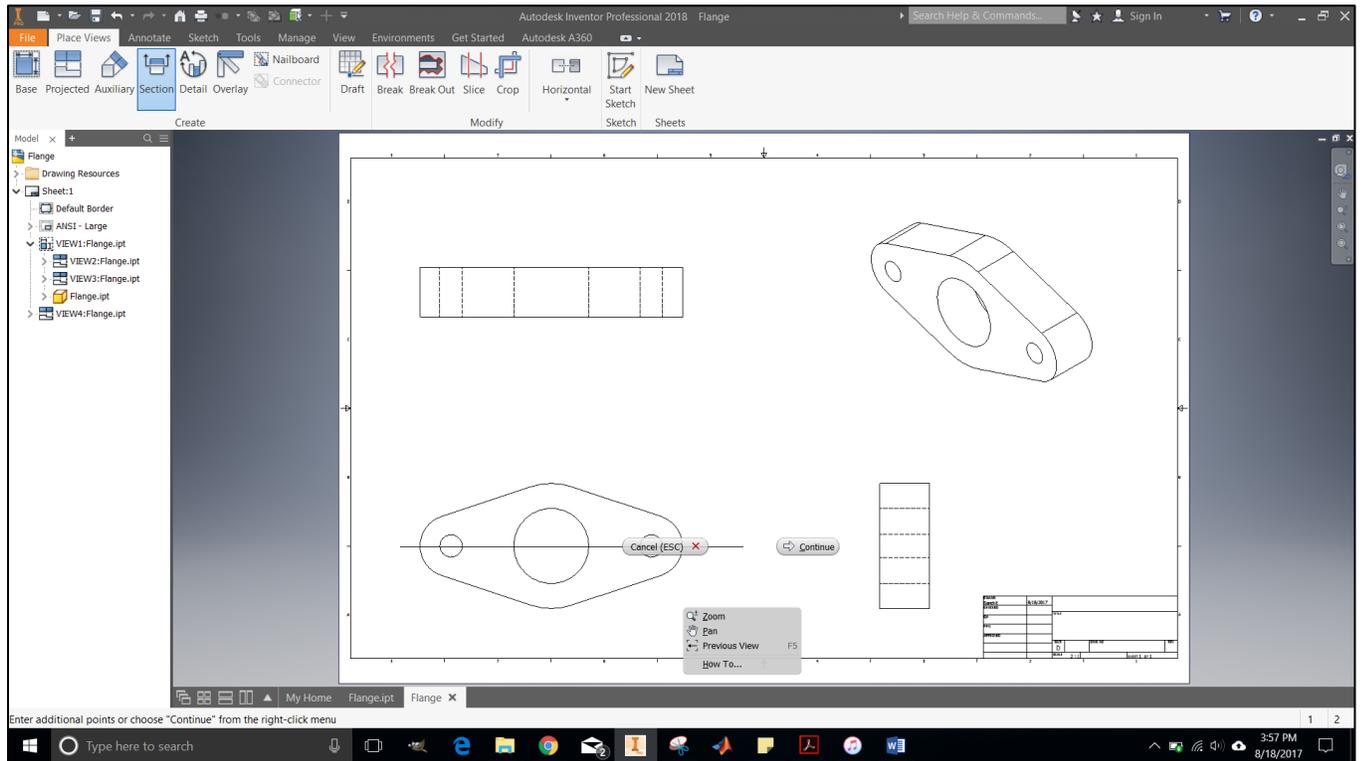
Step 5:

You will be able to see your part on the drawing sheet. Set the scale to 2:1. Now move the cursor in the top and left/right directions to create the Top and Side views respectively and in a angled direction to create the Isometric View, although this view is not required. Click 'Ok'



Step 6:

Click 'Section' and select the Top view (as shown) and draw a line through the center, right click at the end of the line and select continue



Appendix C: Pilot Study

Create the drawings and load them into the software as specified in the instructions provided. Answer the following questionnaire after you are done.

Rate the following aspects of the software on a scale of 1 to 5, 1 being the least and 5 being the highest. You can provide any feedback at the end of questionnaire.

1. How simple or user friendly was the graphic user interface for the software ?

Extremely Difficult	Somewhat difficult	Neutral	Easy	Very easy
<input type="radio"/>				
1	2	3	4	5

2. How easy was the installation of software on your computer ?

Extremely Difficult	Somewhat difficult	Neutral	Easy	Very easy
<input type="radio"/>				
1	2	3	4	5

3. How important is this software as an interactive learning tool for freshman students new to CAD environment ?

Not at all important	Somewhat important	Neutral	Important	Very Important
<input type="radio"/>				
1	2	3	4	5

4. How effective is this software as a grader for a course in Spatial Visualization ?

Extremely Ineffective	Somewhat Ineffective	Neutral	Effective	Very Effective
<input type="radio"/>				
1	2	3	4	5

5. How would you rate the applicability of this software in a classroom as a beneficial tool for the instructor ?

Not beneficial

Neutral

Extremely Beneficial

1

2

3

4

5

Feedback/Suggestions: