OpenCV based JPEG Image to STL File Rendering using Folding of Orthographic View of Shapes

Sughosh P Dixit[1], Prajwal B R[1], Sushrith M G[1], Sumukha Sharma A[1], M Shilpa[2]

[1]Student, Dept. of Information Science and Engineering, Bangalore Institute of Technology, Bengaluru, Karnataka, India
[2]Assistant Professor, Dept. of Information Science and Engineering, Bangalore Institute of Technology, Bengaluru, Karnataka, India

---***---

Abstract – *3D Printing is one of the upcoming and most sought after technologies and STL file extension is the most commonly used file format to 3D print any given design. This paper proposes a solution to render a given 3D shape's basic orthographic view into a 3D STL file format. Based on the contour levels the geometric net will be classified into various views, and these views are folded with each other to produce a 3D shape. The solution is an OpenCV based solution which is implemented in Python, since it provides useful libraries to accomplish the desired outcome. OpenCV requires ambient lighting to be able to interpret the given image input accurately. This solution involves dealing with geometric shapes and their interpretation which is most suitable in python to implement since the libraries provided are most suited for implementation. As a functionality the image path finder uses OpenCV to find contours and fold lines, subsequently the folding process is done, and the corresponding z coordinates are added to the vertices, and the new vertices are grouped to form the STL file. As a useful feature the solution shows a video feed with pathfinding so that user knows when to capture the image of the sketch for processing. As an additional feature the proposed solution also includes a supporting web application through which the user can store, manipulate and render view the 3D STL file obtained.*

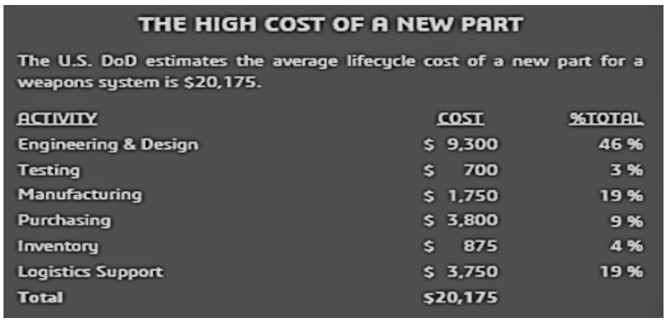
Key Words: **3D File Formats, Orthographic View, OpenCV, STL file, 3D Printing, 3D Geometric Shapes, Shape Interpretation, Image Processing.**

1. INTRODUCTION

JPEG that is expanded as Joint Photographic Expert Group is a 2D image file format, through which one can view the image in 2D dimensions. The concept of the solution proposed is to render the desired 3D STL files that are used from an input 2D image which consists of its 2D geometric net or the orthographic view of the shape. The main aim of the solution is to automate the conversion of the input 2D image that can be in the format of JPEG to the 3D file that consists of the former's geometry rendered to its third dimension. This solution has its motivation in context to 3D data generated in order to optimize supply chain demand reactions. The solution has a supporting web platform to make the solution more viable to use, where the user is given the use case to manipulate the so generated STL file as well as can search user customized 3D storage repository. The solution enables the user to satisfy the software-oriented supply chain and can be regarded as a Product Lifecycle Management (PLM) application.

Modern day large manufacturing industries that have advanced technological capabilities are enabled with software that are enterprise level and have the capability to interpret a 2-Dimensional Drafting that is in JPEG format to a 3-Dimensional file format. To show the significance that 3-Dimensional Designing from 2-Dimensional images have, a statistic conducted from the US DoD that estimates the average lifecycle cost of a new part for a Weapons System is $20,175. The estimate is that under the total cost mentioned, about 46% of the total cost comprises with the Engineering and design, which can be estimated to $9,300. This statistic represents the importance of designing.

The solution proposed is able to render 3D files of basic shapes. All complex parts or shapes can be assumed to be comprising of a mixture of the basic shapes. Thusly, upon assembly a complex shape can be created accurately by using basic shapes with appropriate scaling and placement. Figure-1 represents the statistic. The objective of the proposed solution is to build a system which takes the input in the form of basic shape's geometric net and the end user must input the number of vertices the shape consists of, and the solution will provide with the required STL file. This output obtained from the proposed solution is scalable and is in the appropriate file format, which then can be 3D printed. The solution aims to provide with accurate STL file, to be able to 3D print the desired shape.


THE HIGH COST OF A NEW PART

The U.S. DoD estimates the average lifecycle cost of a new part for a weapons system is $20,175.

ACTIVITY	COST	%TOTAL
Engineering & Design	$ 9,300	46 %
Testing	$ 700	3 %
Manufacturing	$ 1,750	19 %
Purchasing	$ 3,800	9 %
Inventory	$ 875	4 %
Logistics Support	$ 3,750	19 %
Total	$20,175	

Figure -1: Statistic showcasing the Importance of Design

1.1 Existing System

There are many software that enable designers to interpret a 2-Dimensional orthographic view of a shape or a 2-D drafting according to the scale and dimensions into a 3-Dimensional file format. There are no web applications that automate this process of rendering 3-Dimensional files from an input image of a shape's geometric net. The solution proposed has the capability to render STL files from input JPEG image of basic shapes such as cube, cuboid, prism, trapezoidal prism, square pyramid etc.

Limitations:

- The existing software do not automate the process of rendering STL files, enabling the designers to interpret the 2D geometry's dimension and carefully design the 3D file.
- A web application that renders STL files from any file format does not exist, although some web applications use protrusion techniques to render 3D models from an input image.

1.2 Problem Statement

To develop a system which provides a solution in context of rendering 3D STL files, from the input provided by the user in the form a JPEG image.

Therefore the aforementioned solution is designed to:

- Accept a JPEG image as input, which recognizes a 2-Dimensional shape's geometric net.
- Interpret different views of the desired 3-Dimensional object, from the input provided by the 2-Dimensional JPEG image.
- Throw an exception if the solution is unable to recognize the input.
- Fold the different views appropriately in order to obtain a 3-Dimensional object.
- Render the 3-Dimensional object that is obtained into a STL file format to be further processed to get the file be 3D printed.

2. METHODOLOGY

The structure of the proposed solution has three separate processes. First, the image pathfinder uses OpenCV to read the contours and fold lines of the input image. The vertices of these lines are passed to the folding process to sort them into faces and then add necessary z coordinates to the vertices. These new 3D vertices are used to create the final STL file. These files are all imported into the main program script, which handles the GUI (Graphical User Interface), user input and output, and integrates all modules.

2.1 Folding Process

The 2-D vertices of the orthographic view of the geometry is folded from the image into a set of 3-D coordinates to represent the folded shape in 3-D space using the Euler-Rodrigues formula. This folding method checks each face against each other and attempts to fold it into place by decreasing the angle between them to check if they will meet. This process continues to fold the sides together until the final shape is assembled and there are no extra sides remaining. Figure- 2 represents the folding process of a given cube's orthographic view.

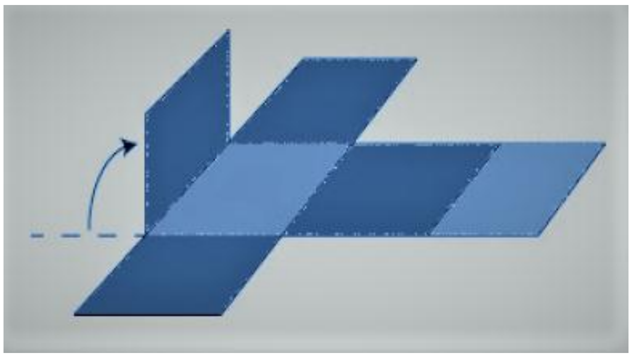


Figure -2: Folding Example of a Cube

2.1.1 Euler-Rodrigues Formula

The Euler-Rodrigues formula describes the rotation of a vector in 3 dimensions. The formula is given the name Euler because of the rotation described by 4 Euler parameters and Rodrigues comes from the Rodrigues formula which describes a way to calculate the position of a rotated point.

From the four Euler parameters, which describes the rotation of a point using four real numbers a, b, c, d such that

$$a^2+b^2+c^2+d^2=1$$

From the standard vector notation, the compact form of the Rodrigues formula is

$$\vec{x}' = \vec{x} + 2a(\vec{\omega} \times \vec{x}) + 2\left(\vec{\omega} \times (\vec{\omega} \times \vec{x})\right)$$

Figure -3: Rodrigues Formula

2.2 Creating the Array of Triangulated Faces

The proposed solution uses straightforward OpenCV functions to capture the image from the frames of the video feed. The fold lines and the main contour are thus obtained as mentioned in the methodology.

The further step of creating a numpy array of triangulated facets which is critical to get an STL file must be implemented using a face finder function. The function accepts the main contour and fold lines from the canvas as inputs. This means a list of points of the main contour and a list of point pairs giving fold lines are accepted as input from the geometric net's canvas. A dictionary consisting of links of both the main contour and folded lines is created and all the possible links are stored in the dictionary. Further from each element of the dictionary a list of all possible faces that can be generated is collected using breadth first search algorithm for graphs explained in the latter part.

The duplicate faces that exist in the queue obtained from the algorithm are dropped and the list of triangulated faces are thus obtained. This list is further casted into a numpy array.

2.2.1 Breadth First Search Algorithm

Breadth First Search algorithm is used to search a tree, which as the name suggests traverses or searches at the present depth prior to moving to the next depth. In the proposed solution in order to link the folded lines and contours to form faces the Breadth First Search is used.

Algorithm:
Input: An adjacent list representation of the graph or tree
Output: A queue having an enumeration of all adjacent nodes
Step 1- A queue created and appended with the starting point.
Step 2- Repeat the further steps until the queue is true
Step 3- Get the first path form the queue
Step 4- Get the last node from the path
Step 5- If path is found the path is greater than 3 then return the path
Step 6- Number each adjacent nodes, construct a new path and push it into the queue.

2.3 Working of the System

The proposed solution is an OpenCV based image to STL program that translates JPEG images of orthographic views to STL files. The solution includes OpenCV pathfinding, 3D geometries from 2D points and rendering the STL file.

The solution has an extended GUI which first opens the system's camera to capture the orthographic view of the geometry through a video stream. Figure-4 represents in such orthographic views, shown as the view of a cube. Once the canvas of the geometry is captured, the system accepts the number of vertices from the user.

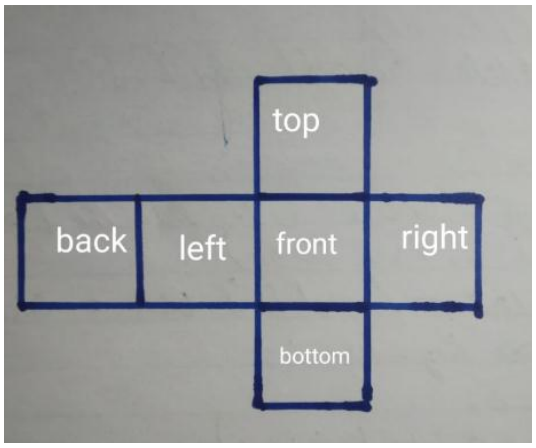


Figure -4: JPEG image of orthographic view of cube

The solution after accepting the number of vertices of the shape, forms triangulated faces, and using the Euler Rodrigues formula, these faces are folded at different angles so as to the different faces meet each while decreasing the angle between the faces. The duplicate faces that do not contribute while forming the 3D shape are dropped.

The solution then using the STL library is converted into STL file format, which is then compatible to be further 3D printed. Figure- 5 represents the flow diagram of the solution.

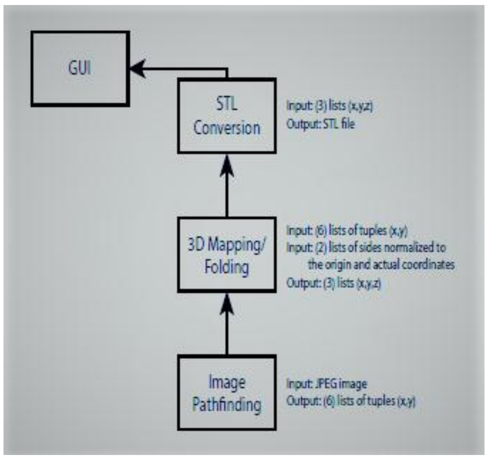

GUI
STL Conversion
Input: (3) lists (x,y,z)
Output: STL file
3D Mapping/ Folding
Input: (6) lists of tuples (x,y)
Input: (2) lists of sides normalized to the origin and actual coordinates
Output: (3) lists (x,y,z)
Image Pathfinding
Input: JPEG image
Output: (6) lists of tuples (x,y)

Figure -5: Flow Diagram of the Proposed Solution

3. IMPLEMENTATION

3.1 Image Capturing of Sketch and Processing Paths

For capturing of the JPEG orthographic view of the required shape, the proposed solution uses the OpenCV library, through which the height and width of the video feed is set. Through the video feed the solution enables the user to input the orthographic view of the shape in the form of an image. The Height is set using CV_CAP_PROP_FRAME_HEIGHT and

Width of the video feed console is set using and CV_CAP_PROP_FRAME_WIDTH which are a couple of enumeration arguments of the camera properties in the OpenCV library. Ideally the values are set to 300 and 200 for width and height respectively, which can be altered as per user requirements. Table-1 lists the various enumeration arguments of the camera properties that can be set.

For further processing, the solution reads the captured geometric net. To make the captured data suitable for processing the image from the video feed is flipped horizontally using the flip function. The cv2.flip function enables the frame captured from the video feed to be flipped horizontally when an argument is passed along with it. Once the image is captured from the frame and is flipped accordingly, the image must be written or saved. Using the cv2.imwrite function the image of the geometric net captured is saved.

3.2 Folding the Geometric Net

The proposed solution next prompts to input the number of vertices the fed geometric net consists of within. Folding the geometry includes finding and normalizing the outside contour. Further the input geometric net's image is folded.

Table -1: List of camera properties used for video stream

Enumeration	Argument	Description
0	CV_CAP_PROP_POS_MSEC	Current position of the video file in ms
1	CV_CAP_PROP_POS_FRAMES	Based index of the frame to be captured next
3	CV_CAP_PROP_FRAME_WIDTH	Width of the frames in the video feed
4	CV_CAP_PROP_FRAME_HEIGHT	Height of the frames in the video feed
7	CV_CAP_PROP_FRAME_COUNT	Number of frames in the video feed
8	CV_CAP_PROP_FORMAT	Format of the mat() objects
10	CV_CAP_PROP_BRIGHTNESS	Brightness of the image
11	CV_CAP_PROP_CONTRAST	Contrast of the image
12	CV_CAP_PROP_SATURATION	Saturation of the image

The solution processed converts image to grayscale. The so obtained grayscale image is converted to binary format. This way the image can also be adjusted to differed lighting conditions. The so converted binary image file will have canny edges that are detected. The contours of these edges are identified. This method is reiterated as through outside contour to reorganize the data as a list of tuples. The solution then finds only the inside points of the contour that defines fold line points for folding the geometric net edges. At this point the solution provides with two lists, inside points and corner points, to use for normalization and finding folds. Therefore, the final fold lines are set through this process.

3.3 Rendering Interactive Plot of the Geometry

The proposed solution once has the fold lines figured out, creates a canvas from the geometry. This can be implemented through tkagg from the matplotlib library. Further, the obtained canvas is set to plot in 3-D, using the suitable function having the argument value as 3D. This is an additional feature which is implemented in the solution to showcase the 3-Dimensional geometry in the form of a set of plot points. Figure -6 depicts the plot of a cube rendered from the proposed solution.

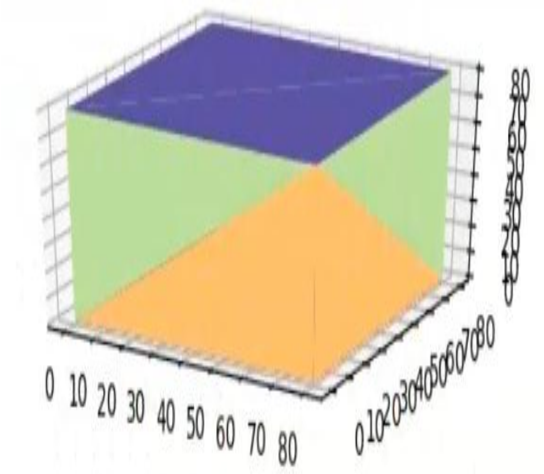


Figure -6: 3-D Plot of a Cube

3.4 Creation of STL File

In order to create the STL file, the proposed solution requires a triangulated facets of the folded geometric net. Thus, the necessity of a numpy array of triangulated faces arises.

To get the set of triangulated facets to form the STL file, the solution takes in a list of points of the main contour, and a list of point pairs giving fold lines. A dictionary of links of both contour and folded lines is created which collects a list of faces which includes duplicate faces. In order get the triangulated facets that links the various vertices of the orthographic view of the JPEG image, Breadth First Search algorithm for a tree or graph is implemented.

These triangulated faces forms a numpy array, which are rotated and folded by decreasing the angle between them using the Euler Rodrigues formula to get an enclosed solid 3-D shape. This shape is further converted into an STL file using the STL library.

RESULTS

The result given by the proposed solution is in the form of an STL file, which can be scaled further or directly 3-D Printed. The purpose of this undertaking is to create a system to create STL files of basic shapes, as all the complex shapes are a composite of simple shapes. The proposed system is able to convert various basic shapes such as cubes, cuboids, square prisms, rectangular prisms and so on to their respective STL files. In order to make the proposed solution more user friendly, a button that triggers the system to save the STL file is implemented. Figure- 7 represents the visualization of a cube's STL file.

Thusly, the proposed solution helps in visualizing folded sheet metal parts more easily.

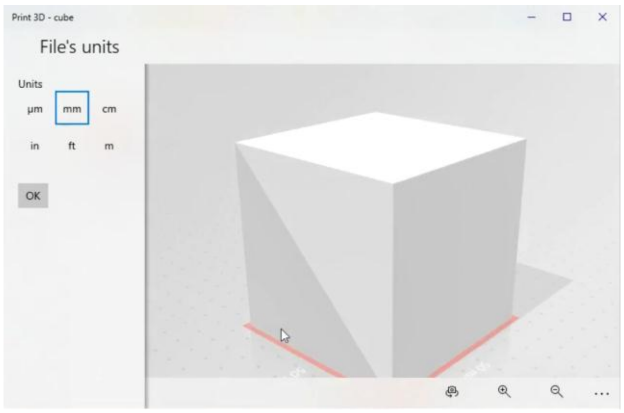


Figure -7: The STL File of a Cube

CONCLUSIONS

The proposed solution is aimed to create a system that would allow users to convert JPEG orthographic views of basic shapes to STL files. The output STL files can then be assembled to form more complex 3-D shapes. Appropriate GUI is implemented in the solution to make it easy to the users. The proposed solution has the motivation to help out the designers in visualizing the parts and rendering the STL file from the input JPEG image.

The solution is mainly implemented in Python, since it provides appropriate libraries that help in getting the required output. The conclusion that can be drawn from the solution is that STL files can be rendered from the system when appropriate orthographic views of basic shapes are input to the system. The solution cannot render 3-D shapes having curved edges or curved vertices. Although this solution can be extended to shapes of non-rectangular prisms. The solution can be extended to be supported as a web application to provide the proposed solution as a service.

ACKNOWLEDGEMENT

The team are most humbled to acknowledge the enthusiastic influence that was provided by the guide Prof. M Shilpa for the ideas, insights that were provided, frequent advises and expertise that was given and the cooperation that was exhibited during the implementation of the proposed solution resulting in its success.

REFERENCES

[1] Paul Murrell, "Vector Image Processing", University Of Auckland, Aug. 2011
[2] Jian S Dai, "Euler Rodrigues Formula variations, quaternion conjugation and intrinsic connections", Science Direct, Volume 92, Oct 2015
[3] O Rinsiati, M Ihsan, Dedi Suhaimi, "Connectivity alogorithm with depth first search on simple graphs", Journal of Physics Conference Series, Jan 2018
[4] M. Szilvasi Nagy, G Y Matyasi, "Analysis of STL files", Budapest University of Technology, 2003
[5] Janusz Konrad, Meng Wang, Prakash Ishwar,"2D to 3D Image Conversion by learning depth from examples", 16 July 2012