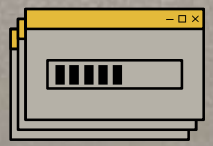
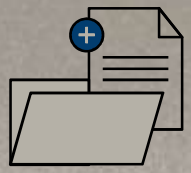


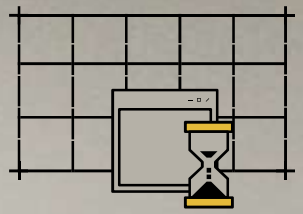
CS299 : INNOVATION DESIGN LABORATORY

Topic - 3D Model from 2D Slices



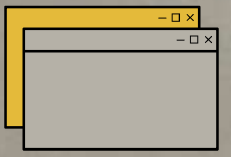
Guided by – Dr. Suman Kumar Maji

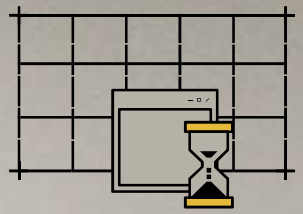
TABLE OF CONTENTS



MOTIVATION

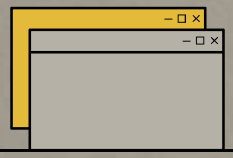
- Over time, there have been various advancements in the field of medical diagnosis.
- The focus has been increasingly shifting to diagnosis via 3D visualization of internal organs.
- The existing techniques are very expensive and hence we develop a cost-efficient model that can create a 3D model out of our 2D dataset.

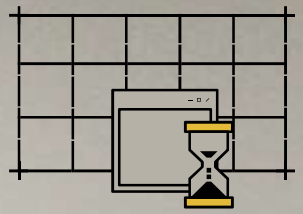




INTRODUCTION

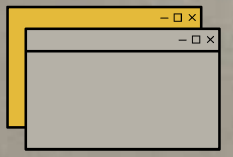
- The objective of this project is to convert 2D slices of ultrasound image to 3D.
- For this purpose we require ultrasound images. Ultrasound images of the body organ is taken using a linear ultrasound probe.
- These techniques are very expensive, time consuming and requires great computational resources . Hence we develop a cost-efficient model that can create a 3D model out of our 2D dataset.



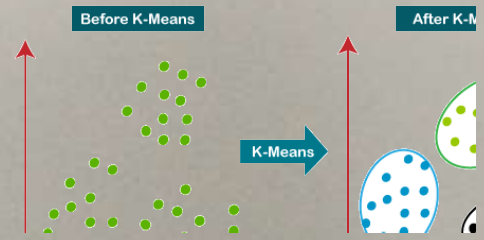


TECHNOLOGIES/ METHODS USED

1. Algorithms
2. Software Used
3. Libraries Used

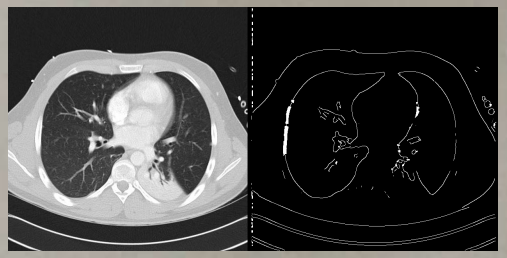


ALGORITHMS



K means Clustering

K means clustering is an unsupervised algorithm used to identify clusters in the data. K means clustering can be used in image data to segment interesting areas from the back- ground. It clusters given data into K clusters using the k centroids.This algorithm is used when we have unlabelled data. The goal is to find certain groups based on some kind of similarity in the data with the number of groups represented by K.



Canny Edge Detection

OpenCV provides cv2.Canny(image, threshold1,threshold2) function for edge detection. The first argument is our input image. Second and third arguments are our min and max threshold respectively. Using the Canny algorithm, the function discovers edges in the input image (8-bit input picture) and marks them in the output map edges. For edge linking, the least value between threshold1 and threshold2 is chosen. The biggest value is used to locate the beginnings of strong edge segments.

Drafting & CAD Technology

Major for college

SOFTWARE USED

Jupyter Notebook

The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. All the code and been written and run in two separate Jupyter notebooks.

Drafting & CAD Technology

Major for college

LIBRARIES USED

Open3D

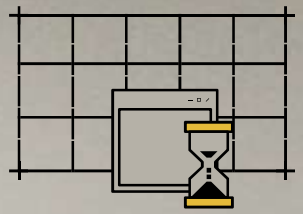
MatPlotLib

PyVista

OS

OpenCV

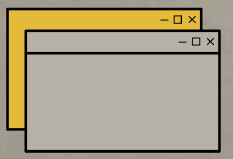
NumPy



CODE AND ITS WORKING

- The whole code is written in Python programming language.
- It has been distributed in two Jupyter notebooks :

1. Pre-processing
2. 3D Visualization



1. Pre-processing

1. Defining a function for pre-processing

In [1]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

def kmeans(original_image):

    crop_img = original_image[0:400, 0:650]  #cropping original image
    img=cv2.cvtColor(crop_img,cv2.COLOR_BGR2RGB)
    vector_form = img.reshape((-1,3))
    vector_form = np.float32(vector_form)
    parameter = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    K = 6   #value of the factor K
    trails=10
    ret,marking,epi_entre=cv2.kmeans(vector_form,K,None,parameter,trails,cv2.KME
    epi_entre = np.uint8(epi_entre)
    res = epi_entre[marking.flatten()]
    final_image_output = res.reshape((img.shape))
    figure_size = 15
    return_edges = cv2.Canny(final_image_output,150,200)

    return return_edges
```

2. Reading images from our dataset

In [2]:
```
import os
cv_img = []
i=1

#specifying the directory where our dataset exists
for img in os.listdir(r"C:\Users\Devesh\Desktop\InnoProjects\Approach_7\3D-recons
    if(img=='desktop.ini'):
        continue
    img_path = os.path.join(r"C:\Users\Devesh\Desktop\InnoProjects\Approach_7\3D-
    n = cv2.imread(img_path)
    cv_img.append(n)
    img=kmeans(n)
    cv2.imwrite(f'tester/crop{i}.png',img)   #specifying the directory where the
    i=i+1
```

2. 3D Visualization

1. Importing Dependencies

```
In [ ]: pip install open3d
```

```
In [ ]: pip install numpy==1.19.5
```

```
In [ ]: pip install opencv_python==4.5.3.56
```

```
In [ ]: pip install matplotlib==3.4.3
```

```
In [ ]: pip install pyvista==0.32.1
```

2. Setting up Co-ordinates for our 3D Model

2. 3D Visualization

In [6]:

```
import cv2
import matplotlib.pyplot as plt
import glob

vis_image = []

#specifying the number of images in our dataset
for i in range(1,151):
  for images in glob.glob(f"tester/crop{i}.png"):
    n= cv2.imread(images)
    vis_image.append(n)


z_cord=[]
x_cord=[]
for i in range(0,len(vis_image)):
  h,w,_=vis_image[i].shape
  zed=[]
  xex=[]
  for l in range(0,h):
    for j in range(0,w):
      b,g,r=vis_image[i][l,j]
      if (b,g,r)!=(0,0,0):
        zed.append(j),
        xex.append(l)
  #z.append(h)
  #x.append(w)
  z_cord.append(zed)
  x_cord.append(xex)



s=1.5
y_axis=[]
for i in range(1,151):
  y_axis.append(s)
  s=s+1.5
print(y_axis)
```

2. 3D Visualization

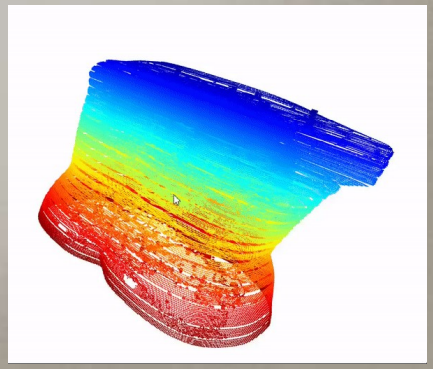
In [ ]:
```
# importing mplot3d toolkits
from mpl_toolkits import mplot3d
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure()

# syntax for 3-D projection
axes = plt.axes(projection ='3d')

# defining axes
test=[]
for i in range(0,len(y_axis)):
    axes.scatter(x_cord[i], y_axis[i], z_cord[i], color='red',s=1)
    for k in range(len(x_cord[i])):
        test.append([x_cord[i][k],z_cord[i][k],y_axis[i]])


# syntax for plotting
axes.set_title('3d Model')
axes.set_xlabel('X Label')
axes.set_ylabel('Y Label')
axes.set_zlabel('Z Label')
axes.view_init(30, -60)
plt.show()
```

3. Generating and visualizing 3D Model

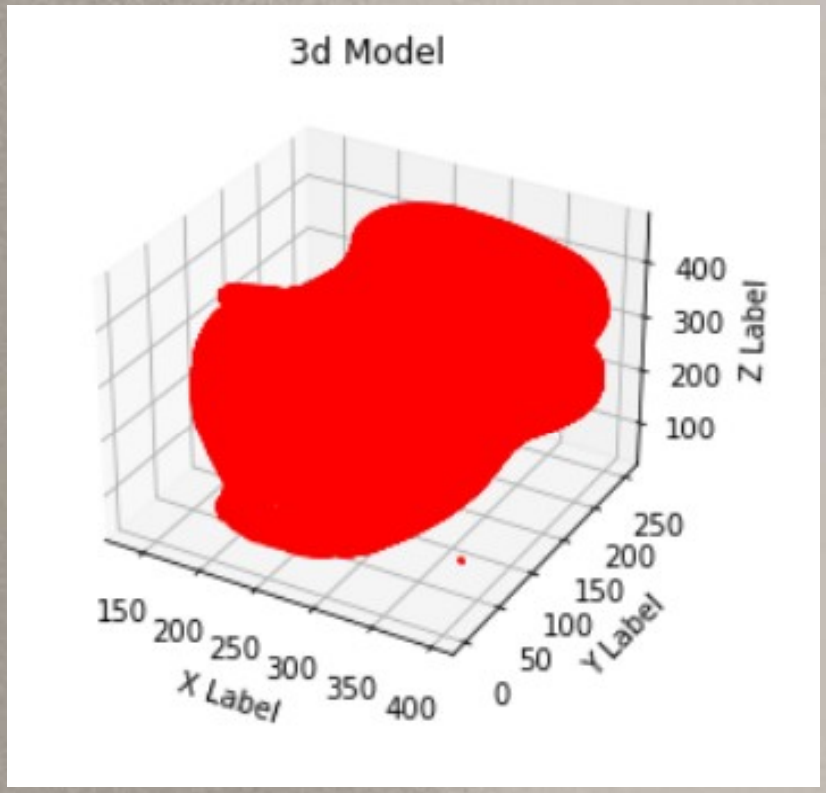
In [ ]:
```
# Passing numpy array to Open3D.o3d.geometry.PointCloud and visualize
import open3d as o3d

placed = o3d.geometry.PointCloud()
placed.points = o3d.utility.Vector3dVector(test)
o3d.io.write_point_cloud('data2.ply', placed)
o3d.visualization.draw_geometries([placed])

import numpy as np
import pyvista as pv

# points is a 3D numpy array (n_points, 3) coordinates of a sphere
cloud = pv.PolyData(test)


generated_volume = cloud.delaunay_3d(alpha=3.)
shell_created = generated_volume.extract_geometry()
shell_created.plot(color='white')
```

OUTPUT

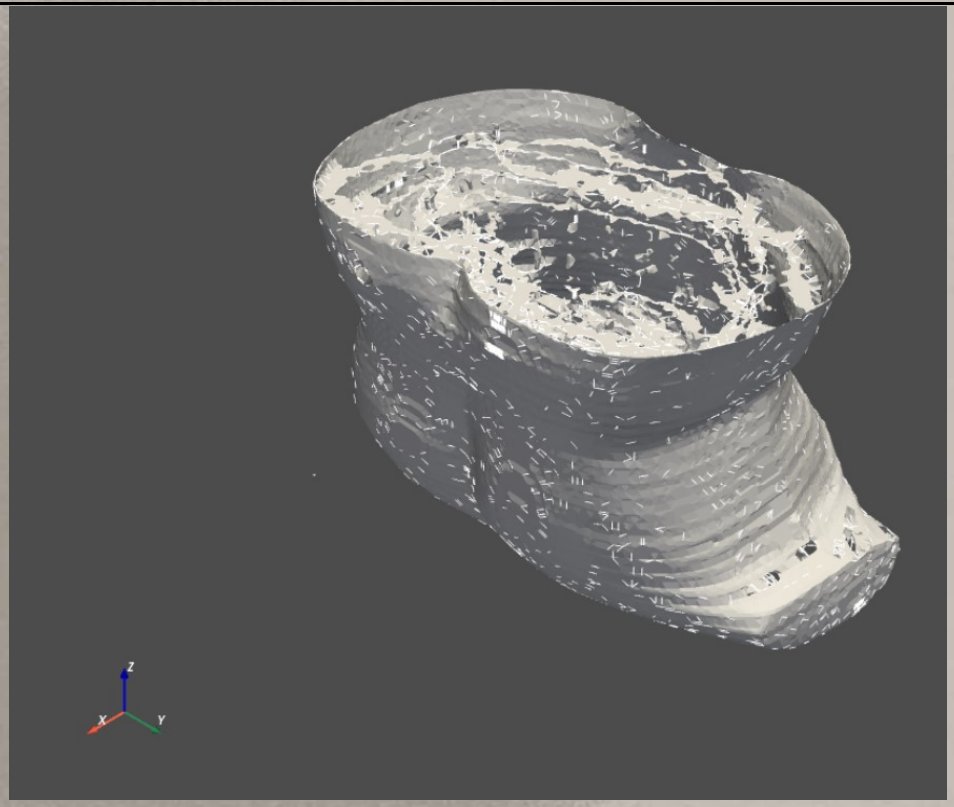
Here is a small simulation of our 3D model. We have used a chest slice dataset containing 150 images.



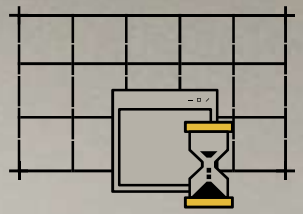
Different regions of chest can be viewed using different colors and textures.


3d Model
400
300
200
100
Z Label
250
200
150
100
50
0
Y Label
150
200
250
300
350
400
X Label

3D Plot

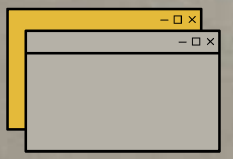


3D Vizualization using PyVista



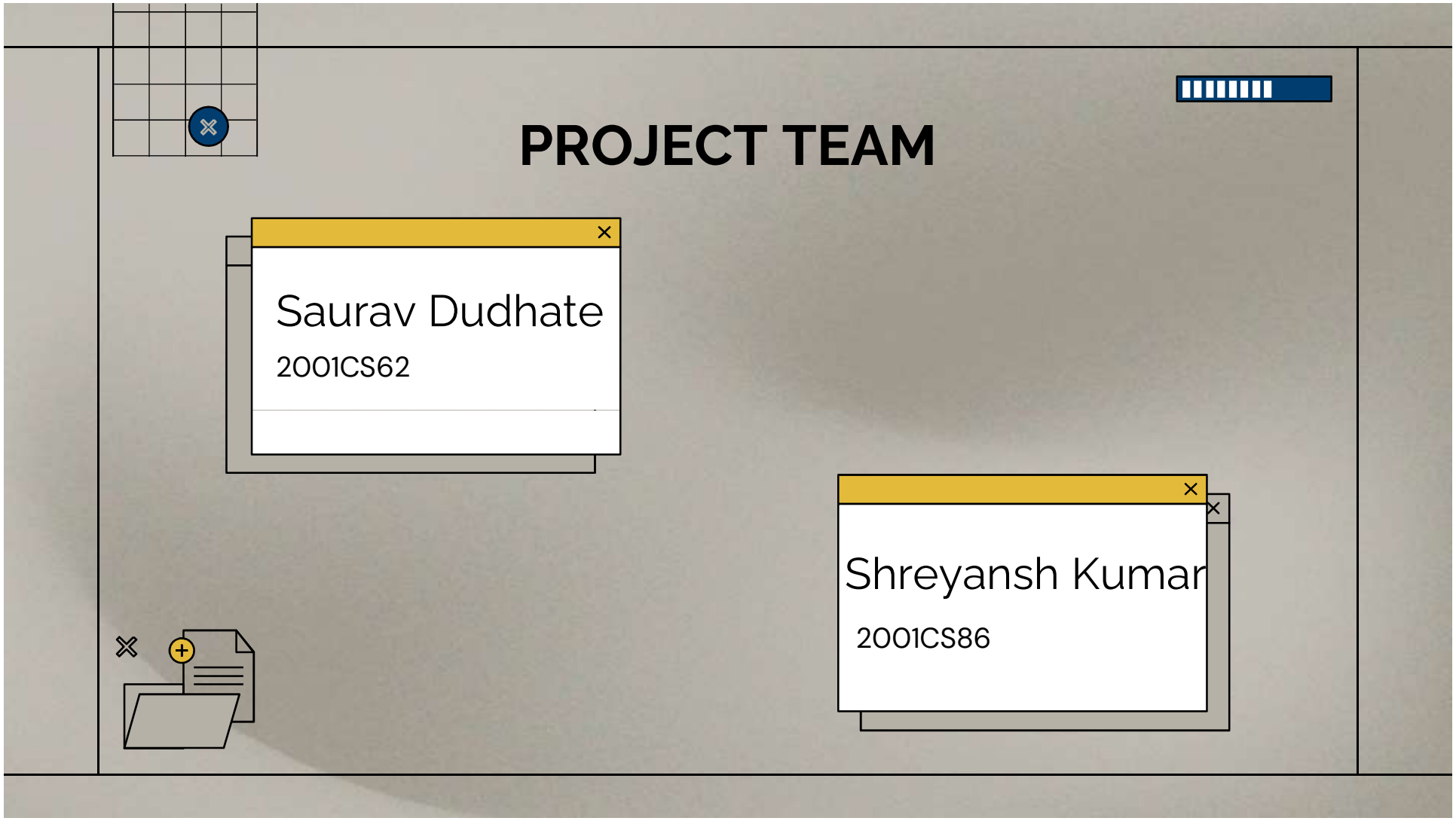
ANALYSIS AND FURTHER SCOPE

Through this project we explored different methods of image processing techniques like binary thresholding, canny edge detection and K Means clustering to segment our region of interest.



Our goal was to visualize 2D images of any body organ in three-dimension. We were able to visualize our given dataset which consisted of ultrasound chest images using two approaches :
VTK's marching cubes for isosurfaces and point cloud using pyvista .

We can move forward to improve the results by applying machine learning techniques for generating intermediate slices, for obtaining smoother 3D volume and better visualization. Preprocessing of ultrasound images can also be focused on to produce better results.


PROJECT TEAM
Saurav Dudhate
2001CS62
Shreyansh Kumar
2001CS86