PATTERN RECOGNITIION AND COMPUTER VISION
CS 5330

PROJECT 4

Name: Kumar Selvakumaran
Collaborators: None

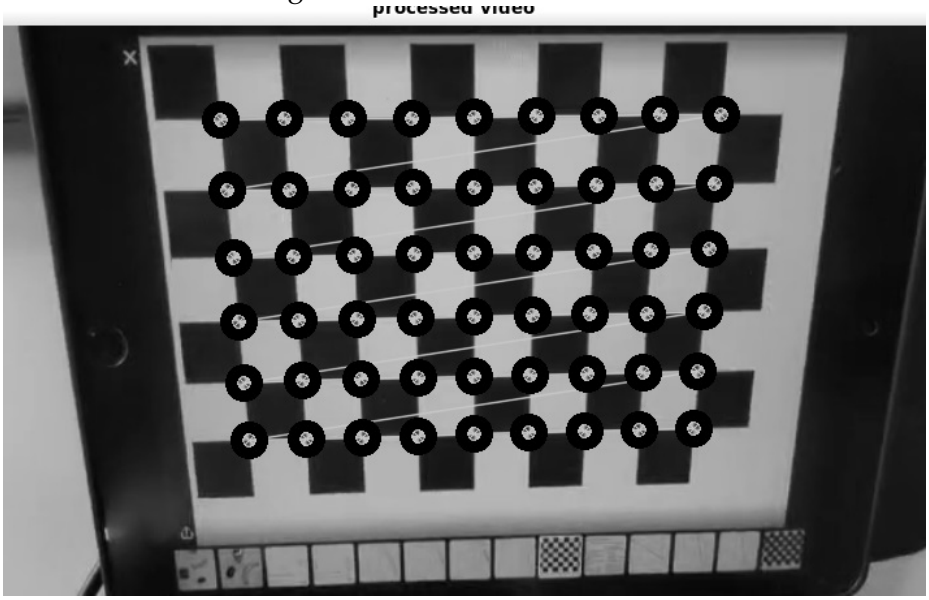
---

1. A short description of the overall project in your own words. (200 words or less)

This project focused on exploring how 3d world objects are represented in a 2d image space. This exploration lead to an understanding of the underlying concepts like perspective projection, affine transforms, re-projection into new bases. Subsequent parts of the project requires us to apply this concepts to place objects in the image, in way which seems like it is somehow coordinated with the geometry of our 3d world, which is famously called Augmented Reality. More specifically, marker based AR is explored in this project, where certain key, easily detectable points are used as pivots to configure our 3d world coordinates in the image space. One main mathematical concept that is the literal foundation of this whole project is Linear Algebra, and It's flexibility, prowess and efficiency is explored in the context 3d-world to image re projection by the mandatory requirements, and graphics manipulation by extension.

TASK 1:

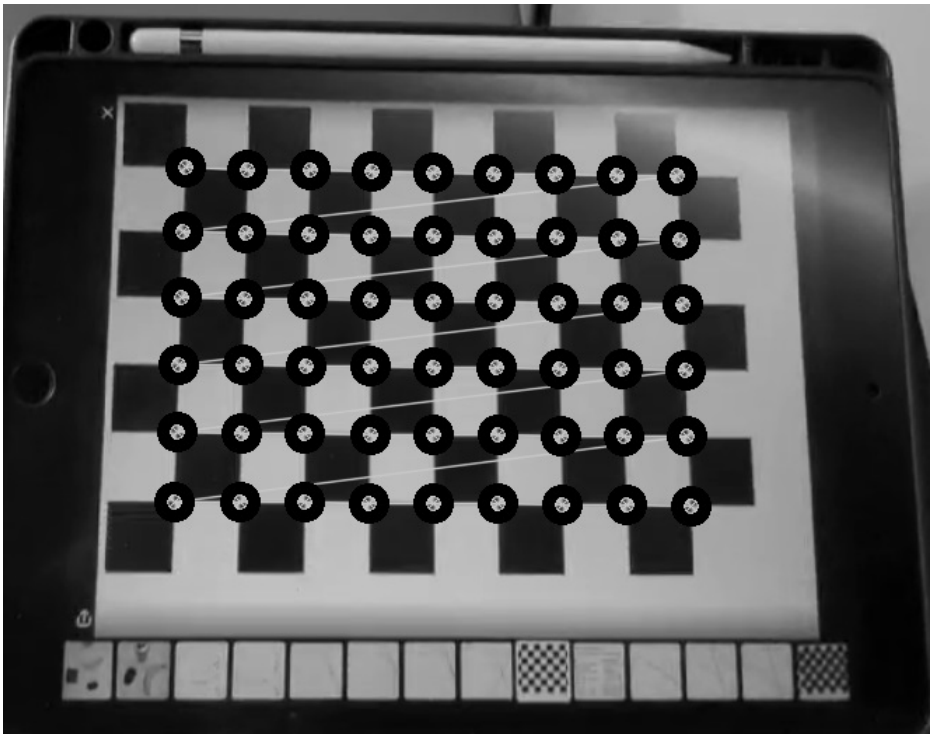
Detect and Extract Target Corners



For this Project, I have chosen the Chessboard as my marker image for the following reasons:

- Starting out with the checkerboard seemed simpler to understand, and a better starting point for a beginner. It also got me thinking of a lot of ideas, and so i was drawn to using it as a marker, even though I suspected it may not work as well as ArUco Markers. Moreover, I wanted to encounter this issues first hand, as i tend to learn more by making mistakes, and coming across issues.
- The First issue i encounter, like you mentioned it was mentioned in the instructions was the fact that the marker was symmetric. This wold occasionally confuse the re projection mechanism which results in **flipping of axes**, hence there are some "abhorrent teleportations" of the graphic objects that are re projected, that are caused by the alternate positioning due to the change of axes.

TASK 2: Select Calibration Images



The above image is one of the images used for camera calibration.

TASK 3: Calibrate the Camera

Below, The resulting parameters from camera calibration is shown at different stages of calibration namely after using 5 images, 7 images, and 10 images. The parameters shown are the resulting camera matrix, and re-projection errors. The resolution of image used for calibration is [1000px x 600px].

calibration paramters after 5 calibration images

```
camera Matrix
matrix chunk :
array([[1634.962305384893, 0, 1026.089274804191],
       [0, 1634.962305384893, 79.38878429545547],
       [0, 0, 1]], dtype='float64')
channels : 1 type : 64FC1 Min val : 0 Max val : 1634.96 rows : 3 cols : 3

 Reprojection Error 0.486146
```

calibration paramters after 7 calibration images

```
camera Matrix
matrix chunk :
array([[1541.150401927324, 0, 979.6494087966951],
       [0, 1541.150401927324, 116.9600269737417],
       [0, 0, 1]], dtype='float64')
channels : 1 type : 64FC1 Min val : 0 Max val : 1541.15 rows : 3 cols : 3

 Reprojection Error 0.661058

 Reprojection Error 0.977671
```

calibration paramters after 10 calibration images

```
camera Matrix
matrix chunk :
array([[1086.103928428084, 0, 824.425690650586],
       [0, 1086.103928428084, 257.6254316012696],
       [0, 0, 1]], dtype='float64')
channels : 1 type : 64FC1 Min val : 0 Max val : 1086.1 rows : 3 cols : 3

 Reprojection Error 0.977671
```

TASK 4: Calculate Current Position of the Camera

As camera moves, side to side, we expect the rotation and translation matrix to adjust accordingly. Using examples, we analyze how the rotation and translation matrices adapt to change in camera position.

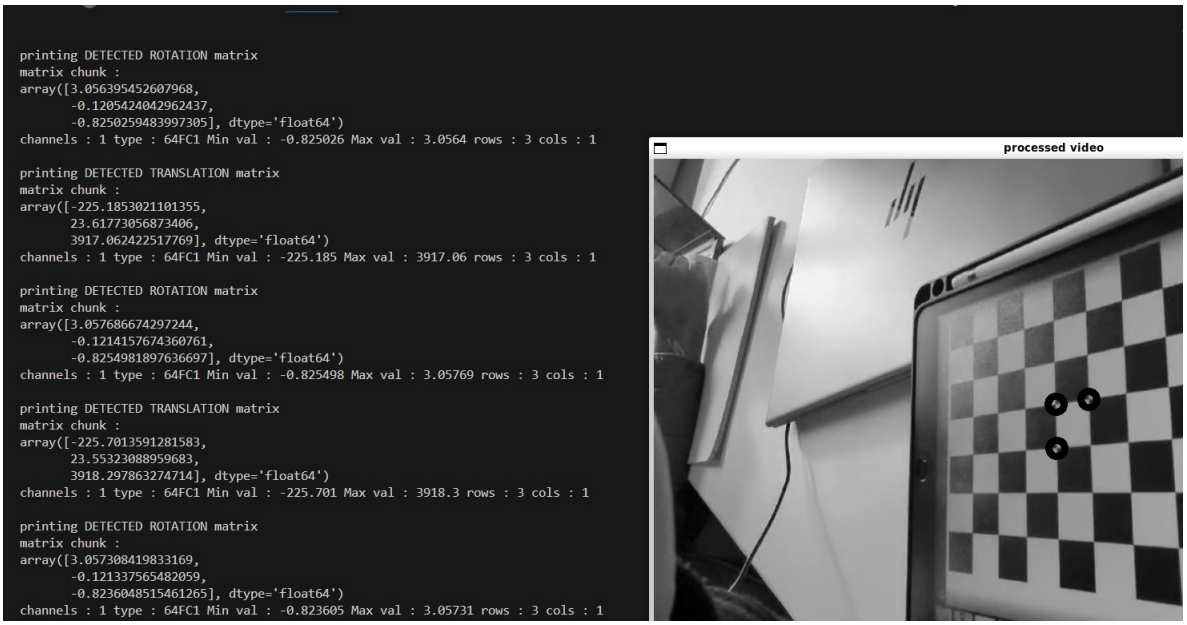
overview of changes:

Consider the below rotation and tranlation matrices and the following image orientation. Let us consider this as approxiamtely, the central angle, and tranlation.

- Change in angle of view changes the rotation matrix drastically. This behavious can be seen in the too images below:

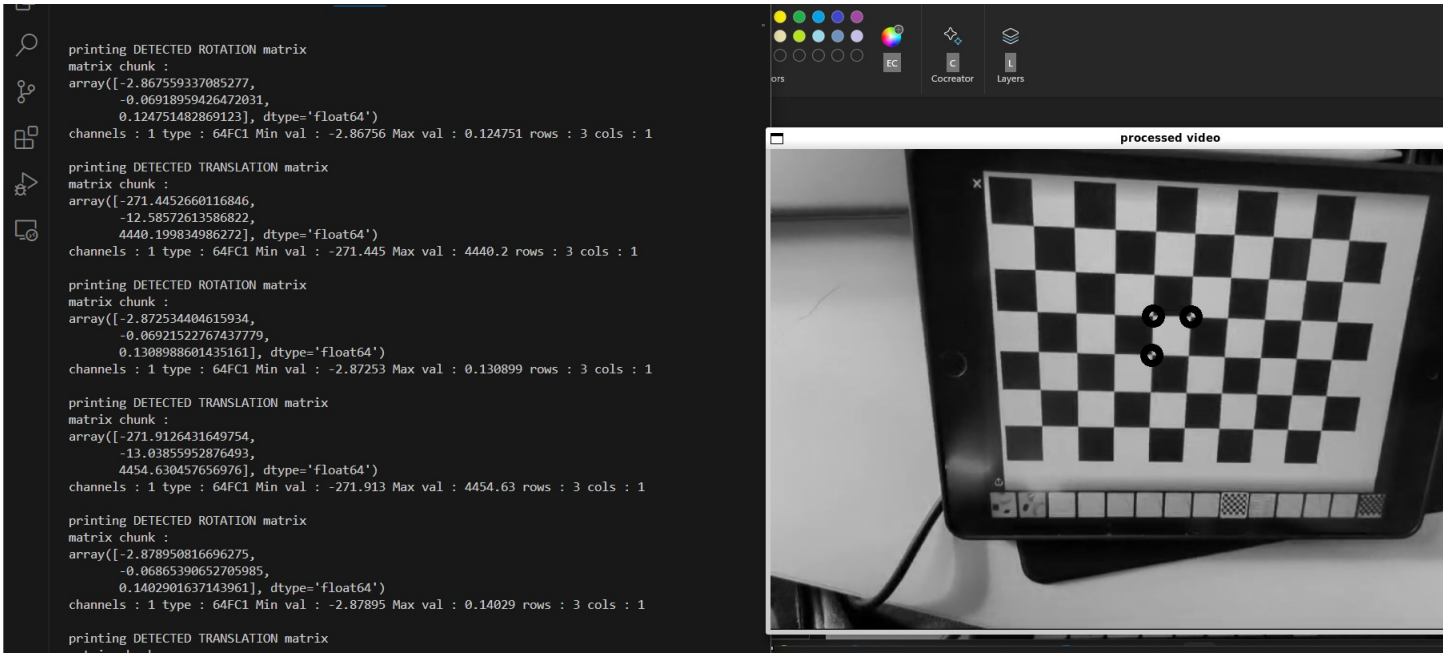
left oriented image

right oriented image



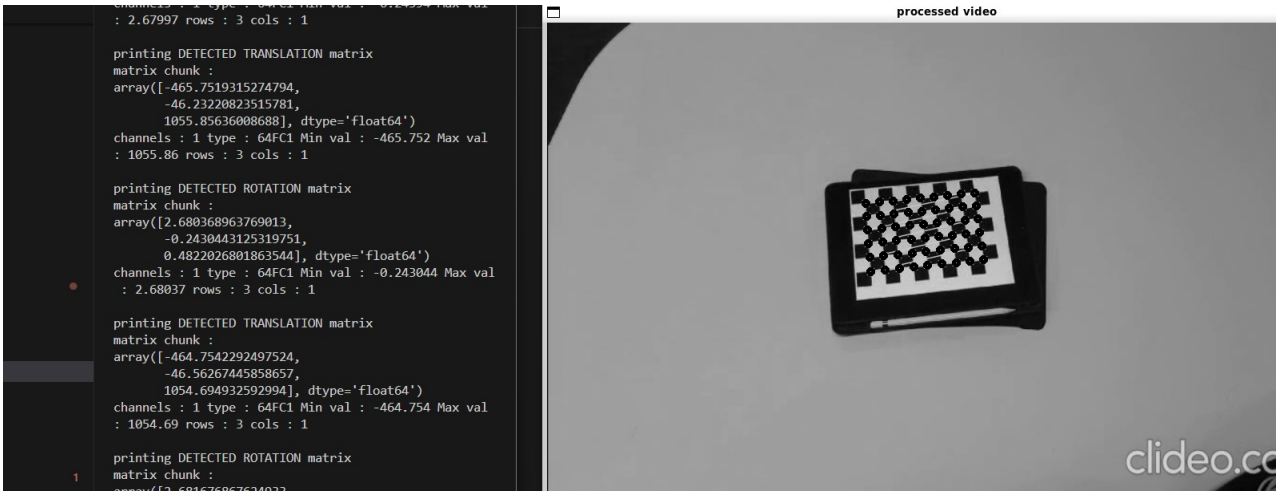
we see that the main change in angle is along the x axis, and the same is reflected in the rotation matrix. the sign has changed as if the x rotation value for left oriented, and the respecive value for right oriented are polar opposites. as expected.

From the below image, where the camera is tilted vertically where the axis of rotation is towards the left in a righthanded system. We notice that the Y axis value has changed significantly from the earlier shown centrally positiononed image's paramters.



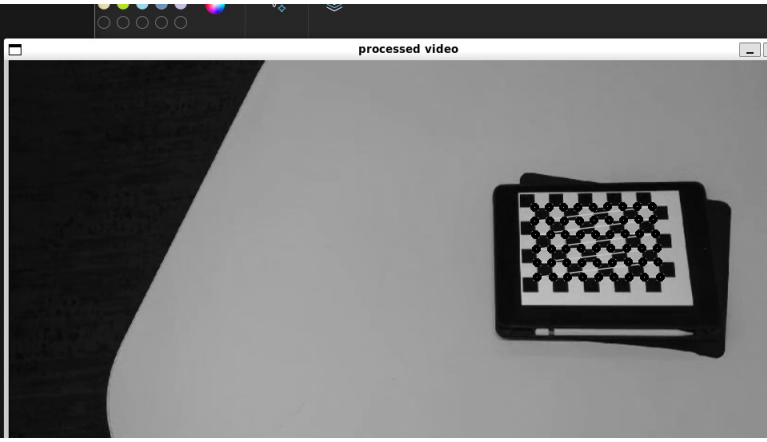
overview of parallel translation of the camera :

when the camera is approximately parallel to the world coordinate z axis , and is translated , The change in translation matrix can be seen evidently. The magnitude of the corresponding axis in the rotation matrix increases, as it moves further away from the origin along that axis. This change can be seen in the below images.

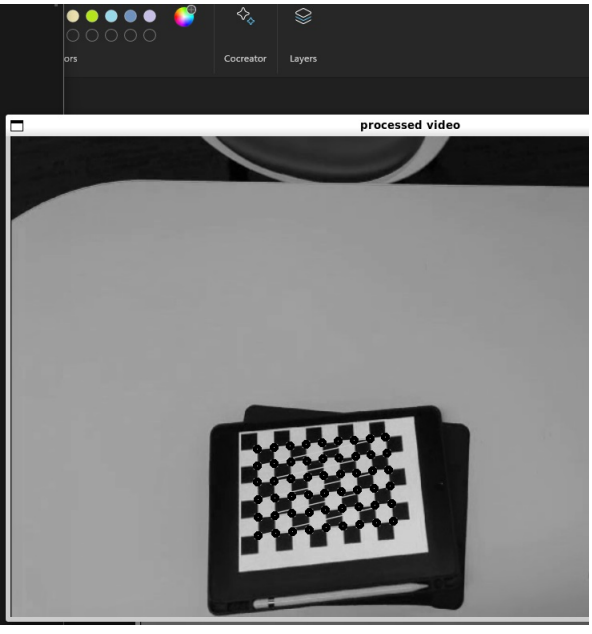


```
printing DETECTED TRANSLATION matrix
matrix chunk :
array([-893.7468000338209,
       -56.45507492824277,
       1270.231744285309], dtype='float64')
channels : 1 type : 64FC1 Min val : -893.747 Max val
: 1270.23 rows : 3 cols : 1

printing DETECTED ROTATION matrix
matrix chunk :
array([-1.666897604350498,
       0.3634175123744946,
       0.9156175327263623], dtype='float64')
channels : 1 type : 64FC1 Min val : -1.6669 Max val :
 0.915618 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
matrix chunk :
array([-105.4665662703149,
       -34.82256547963811,
       115.1461422975893], dtype='float64')
channels : 1 type : 64FC1 Min val : -105.467 Max val
: 115.146 rows : 3 cols : 1
```

```
printing DETECTED ROTATION matrix
matrix chunk :
array([2.703230897287479,
       -0.08101333963469902,
       0.3331280165152306], dtype='float64')
channels : 1 type : 64FC1 Min val : -0.0810133 Max val : 2.70323 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
matrix chunk :
array([-163.506130264747,
       -88.74317243577958,
       1077.740541784173], dtype='float64')
channels : 1 type : 64FC1 Min val : -163.506 Max val : 1077.74 rows : 3 cols : 1

printing DETECTED ROTATION matrix
matrix chunk :
array([2.705110519339518,
       -0.08108489341328735,
       0.3332703390222218], dtype='float64')
channels : 1 type : 64FC1 Min val : -0.0810849 Max val : 2.70511 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
matrix chunk :
array([-163.506794107672,
       -89.06872294427355,
```



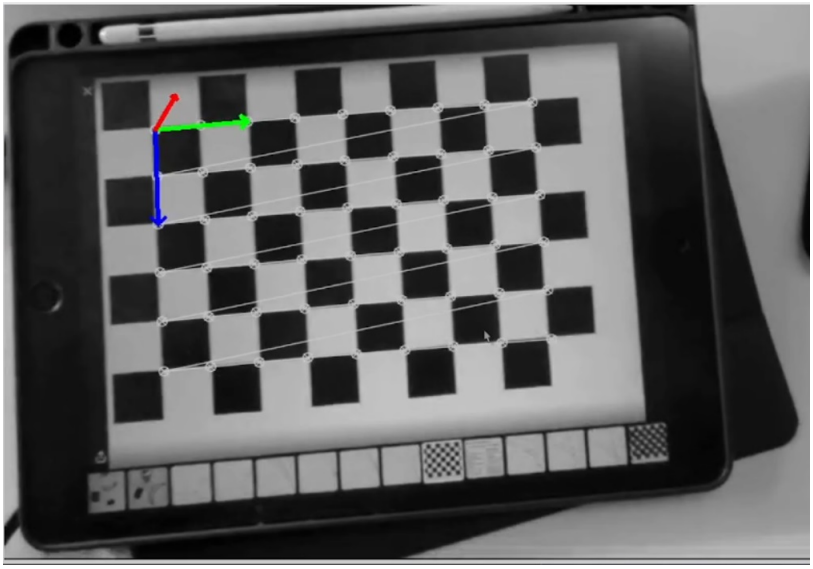

```
printing DETECTED ROTATION matrix
matrix chunk :
array([2.686718978989045,
       -0.1943323027786292,
       0.4645969864320025], dtype='float64')
channels : 1 type : 64FC1 Min val : -0.194332 Max val : 2.68672 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
matrix chunk :
array([-491.8209584848723,
       115.9811383223217,
       894.971562586884], dtype='float64')
channels : 1 type : 64FC1 Min val : -491.821 Max val : 894.972 rows : 3 cols : 1

printing DETECTED ROTATION matrix
matrix chunk :
array([2.685913158496589,
       -0.1915269993294671,
       0.4661980000149988], dtype='float64')
channels : 1 type : 64FC1 Min val : -0.191527 Max val : 2.68591 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
matrix chunk :
array([-491.0994343959665,
       115.2321537778936,
       894.2777154792607], dtype='float64')
channels : 1 type : 64FC1 Min val : -491.099 Max val : 894.278 rows : 3 cols : 1

printing DETECTED ROTATION matrix
matrix chunk :
array([2.68608069110842,
       -0.1883117190958992,
       0.4631826379775618], dtype='float64')
channels : 1 type : 64FC1 Min val : -0.188312 Max val : 2.68608 rows : 3 cols : 1

printing DETECTED TRANSLATION matrix
```



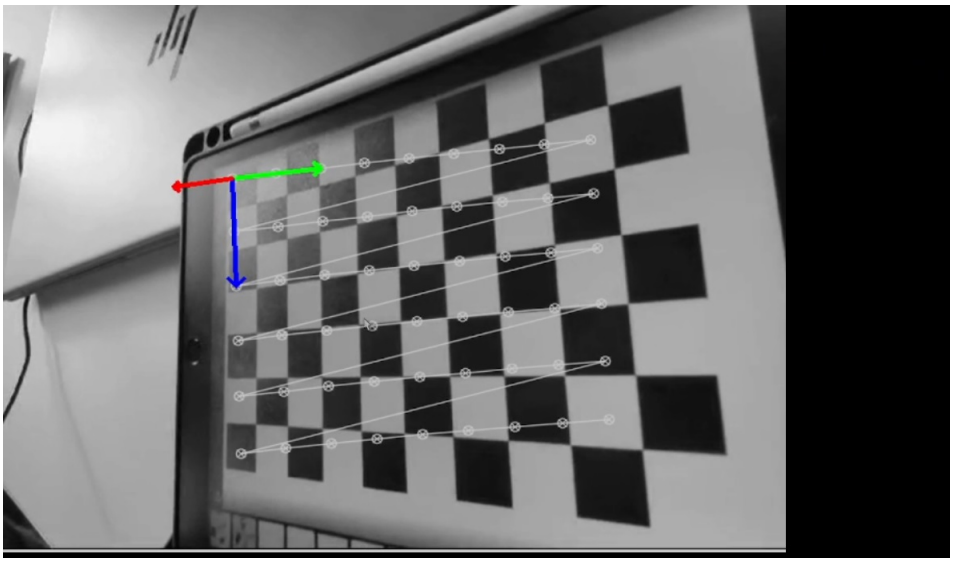
TASK 5: Project Outside Corners or 3D Axes

For this task, it was noticed that the projection was off when the marker is rotated by a large amount, and the shape of the marker is changed extensively from the camera's perspective, Whereas, when the shape of the target in the camera's perspective is the same as it's shape in the real world, re-projection happpens more precicely.

good re-prejection



less precise re-projection



TASK 6: Create a Virtual Object + EXTENSIONS

I approached this task with OpenCV's drawing functionality. My idea was to write a class that draws 3d shapes, that resemble 3d polygon which with sufficient resolution should resemble a sphere. The polygonal object is defined by :

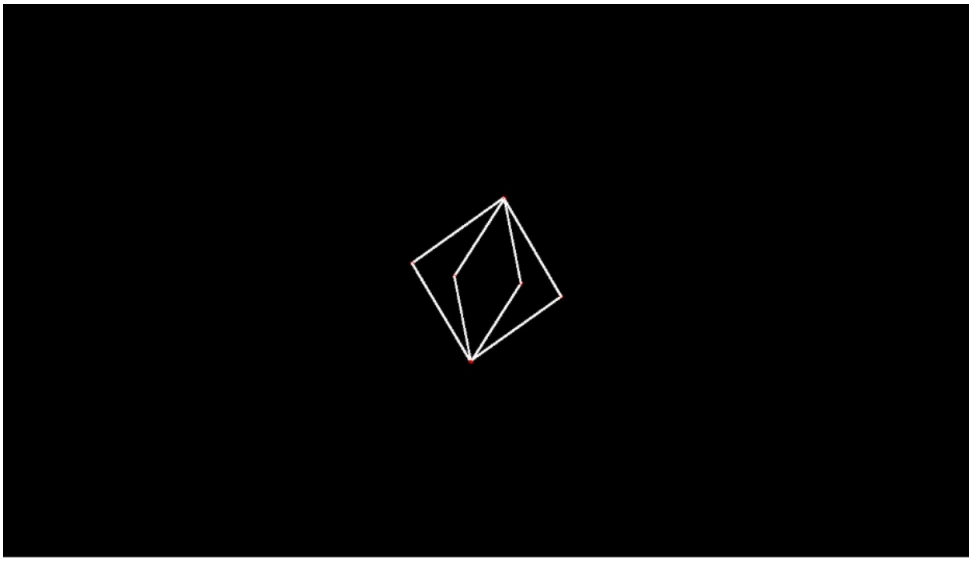
- radius : the distance from the origin to each of the vertices.
- number of vertices per stitch : a stitch here is a a single slice of the 3d object that consists of the center, like a circle slice of a sphere where both have the same radius. The number of stitches correspond the to the number of vertices in this "stitch" (slice). These "stitches" are used as atomic units while making the 3d shape. THe name stitches was inspired by the black lines in a basketball, although, in the shape, all stitches are exactly the same with the differences in rotations along the common axis.
- numStitches : this corresponds to how many stitches the polygon comrpises of.

given these parameters, we have a list of object vertices which is initialized with the first stitch. Once the first stitch is made, the stitch is rotated using a rotation matrix (matrix multiplication) and this new product is appended in the list object vertices. THis process is repeated for "numStitches" , which will finally give us our 3d object.

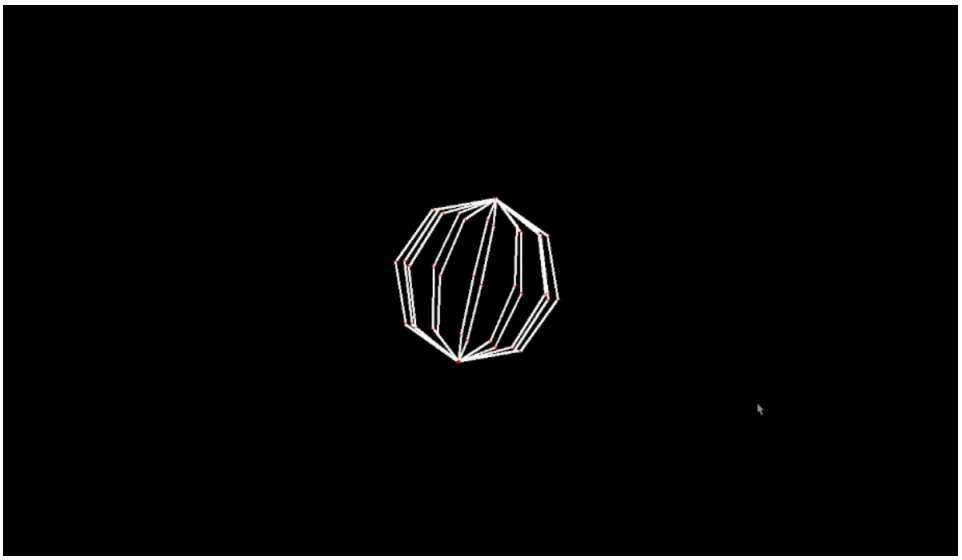
small EXTENSION 1 : making an extendable 3d object class based on user driven parameters and efficient generation using matrix multiplcaton and rotation matrices

Objects corresponding to different values of the 3 paramters are as given below:

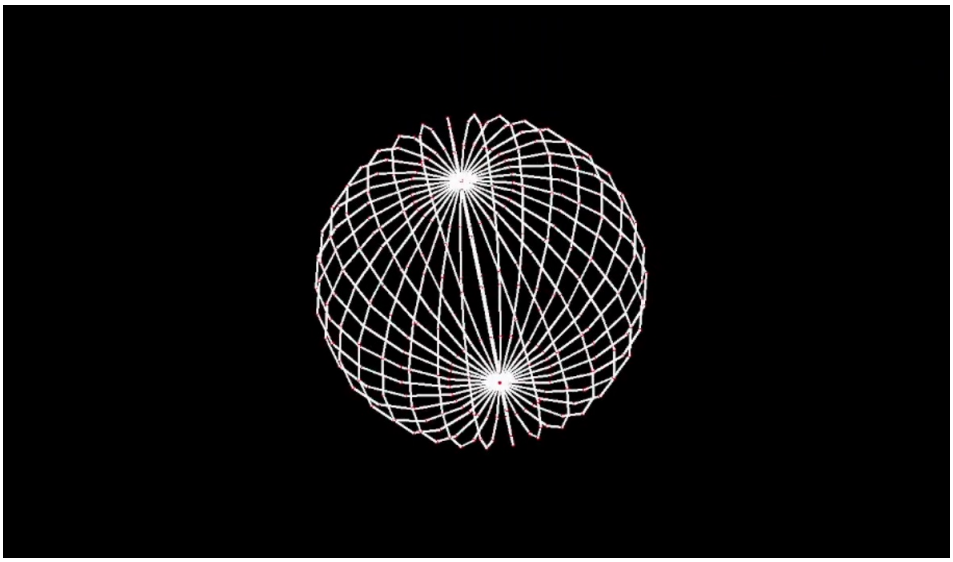
numStitches = 2, verticesPerStitch = 4, radius = 120



numStitches = 6, verticesPerStitch = 8, radius = 120



numStitches = 16, verticesPerStitch = 20,radius = 240



EXTENSION : 2

As an extension, I Tried to animate the 3d object in a slightly sofisticated, yet somewhat real-

istic manner. The goal was to make the object bounce from the 1 coordinate to another coordinate in a parabolic fashion, while spinning in the opposite direction, To avoid following it's inertial direction and jumping out of the board.

The rotation done using matrix multiplication and the angles are computed by taking the video frames index as time, and implementing angular velocity of rotation as quadratic function of time.

Similarly, The tranlation of the object is also governed by the same time construct. For tranlation, the velocity along X axis is uniform and the velocity along z axis is a parabolic function of the displacement along X axis. each bounce interval is predefined, and odd bounces and even bounces are opposite in direction using a simple "bounceNumber % 2 == 0" based mechanism.

Efficiently using linear algebra , and optimization flags during compilation, I was able to make the rendering pretty smooth considering that the complexity of the object (numStitches = 10, numVerticesPerStitch = 12 , radius = 20 : 120 vertices rotated, transformed, reprojected, and (the drawing was done using a for loop to draw the object line by line which is probably the bottleneck of the program)).

DEMO LINK : EXTENSION VIDEO DEMO

TASK 7: Detect Robust Features

For this task i have chosen the Harris Corners to detect features, and upon analysis and testing, The following observations were made :

- It is a thresholding based technique: because it is a thresholding based technique, It is not generalizable. It may work well for some occasions and poorly in others. Some techniques like OTSU thresholding may be used to make this detection techinque more robust.
- It dosen't detect the same features often. For camera calibration, we need a technique that detects the same features at every occurance, but This detection mechanism, often misses occurance, even in successive frames, This can be quite problematic if continuous uninterupted and accurate reprojection and precise calibration is needed

Link to the demo of Harris Corner Features : HARRIS CORNERS DEMO

REFLECTION :

As I have been doing this project, I have been reading the material from Dr.Linda Shapiro's Computer vision text book, while watching the course on linear algebra by Gilbert Strang. Both in Unison has greatly heightened my appreciation of linear algebra, and this project was extremely rewarding to complete, as to me, there is no greater feeling to writing a simple yet optimized program that renders something visual, seemless, and realistic.

Acknowledgement.

This concepts required for this project, I feel, were especially difficult to grasp, but through the process, I have delved deep, explored, experimented and learned a lot more than I could have hoped to. I am grateful for the opportunity as it pushed me a bit out of my comfort zone, but I come out of it feeling closer to my goal of becoming a perception engineer, Thank you. I am also greatful for the the aforementioned resource namely the computer vision text book by Dr. Linda Shapiro, and the course on linear Algebra by Dr. Gilbert Strang.

[1] Linear Algebra - Gilbert Strang MITOpenCourseWare

[2] Computer Vision, Stockman and Shapiro