"[ENPM 673] Project-1"

Perception for Autonomous Robots

PROJECT REPORT

Nevil Patel -116897068

Shesh Mali-116831055

Akshay kurhade-116914529

AR-TAG HOMOGRAPHY

Contents

Abstract

The project focuses on detection and tracking of AR tags. In the first part, the AR tag encoding format was provided and the tag ID were decoded by detecting the AR tags in the video frames . The second part involves superimposition of an image on the AR tag by keeping a track on the tag. The third part involves placing a virtual cube on the detected tag.

Implementation

Preprocess Filtering

All the frames of the video are preprocessed to apply necessary image processing operations.

All the images undergo following image processing operations:

- Converting the image to Grayscale
- Median filtering to remove noise
- Binary thresholding

AR Tag Detection

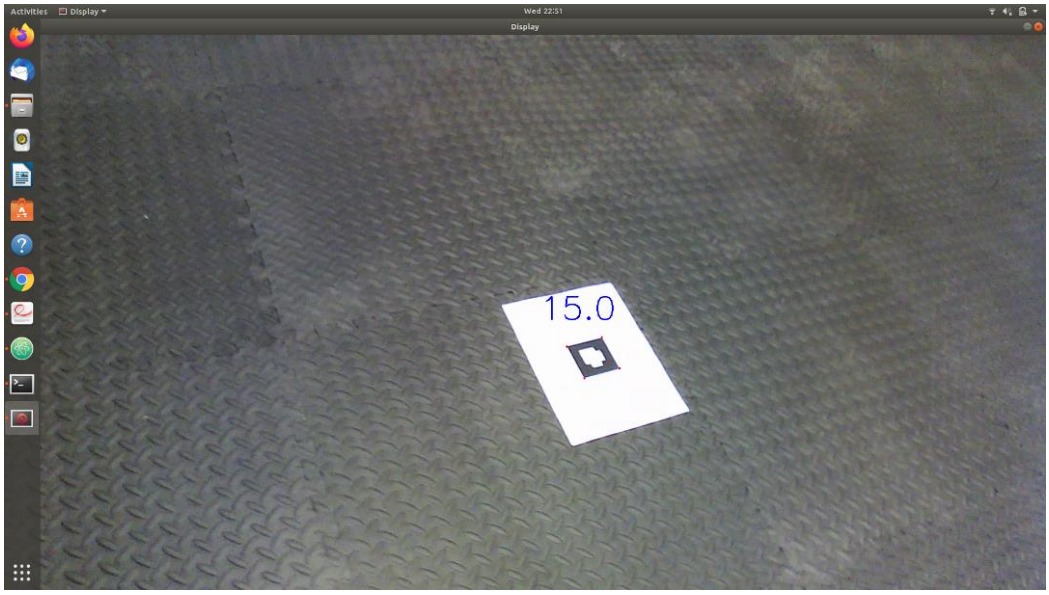
The AR Tag encoding format is already provided to us, hence we need to detect the AR tag from the images and decode them to get the TagID.

To detect the AR tags, the contours are found from the preproccesed images using the inbuilt opencv method with retrieval mode as RETR_TREE which provide the hierarchy information of the detected contours.

In this mode the hierarchy provides the parent-child relationships of the contours. The Now, the required contour i.e the AR tag is selected by comparing the contours w.r.t their areas, hierarchy and number of corners.

Hence, the required AR tag is detected, and we get the four corner points of the Tag in the image cordinate. For each frame, we draw these contours using bounding boxes.

We find the Homography matrix by the “homography” function we created to convert the corners in image frame into the world frame which in our case is (0,0), (0,200), (200,0) and (200,200) to decode the tag. Then the homography matrix is used to transform the contour into the world frame using the “wrap” function we created. The transformed tag in the world frame is then decoded using the function “find_id” to get the TagID.



Super Imposing an Image on the AR Tag

We have been provided with an image of Lena to be superimposed on the detected tag. To achieve this, we need to know the orientation of the AR tag and then rotate the image of Lena accordingly using our function “reorient” before superimposing which considers the orientation of the tag and rotates the image of Lena accordingly. The inverse homgraphy and transformation is done on the rotated image and hence we obtain the image of Lena instead of the tag in the frame on a black background. Also, we generate an another frame using the function “imposeLena” which provides us with the area of tag filled with black and the background as it is in original frame. Now, we perform bitwise_OR operations on the frames so we obtain the required frame with Lena superimposed on the AR tag.

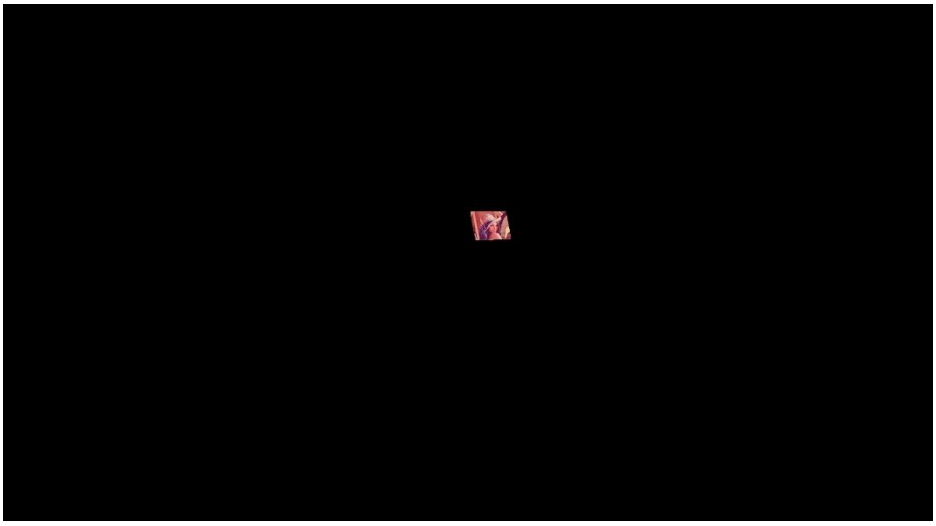


Fig. Lena superimposed on the tag with black background

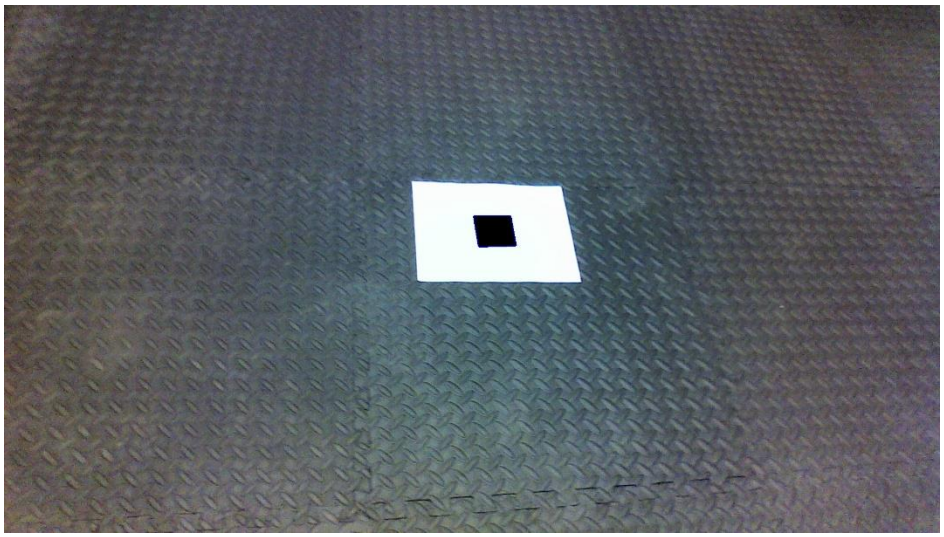


Fig. Tag replaced with black patch

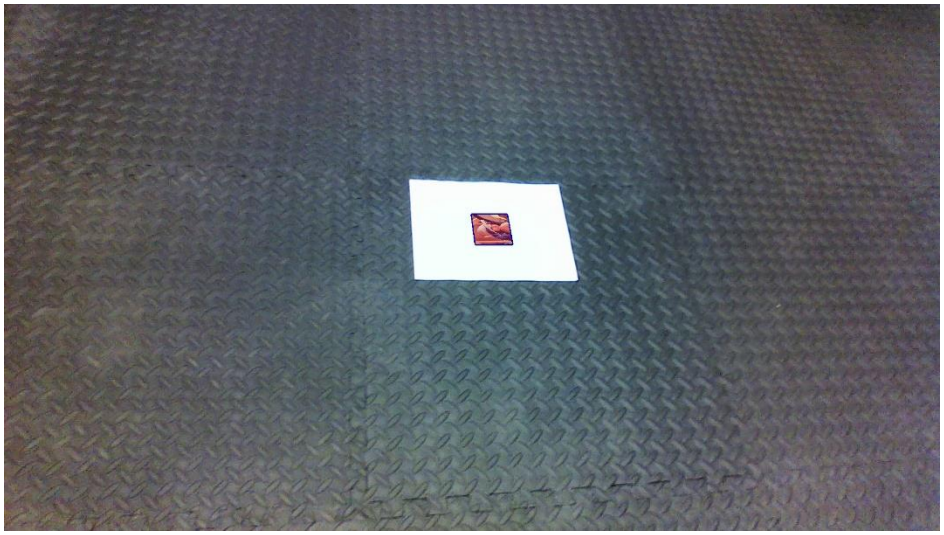


Fig. After BITWISE_OR

Placing a virtual cube on the tag:

To place the virtual cube on the tag, we begin by computing the homography. Then we obtain the projection matrix using the function “projection_mat” which uses the camera calibration matrix and the homography matrix to compute the projection matrix. The obtained projection matrix and homography matrix are used to compute the cube corners using the function “cubePoints”. Then the obtained corners are provided to the function “drawCube” which draws the cube on the tag.

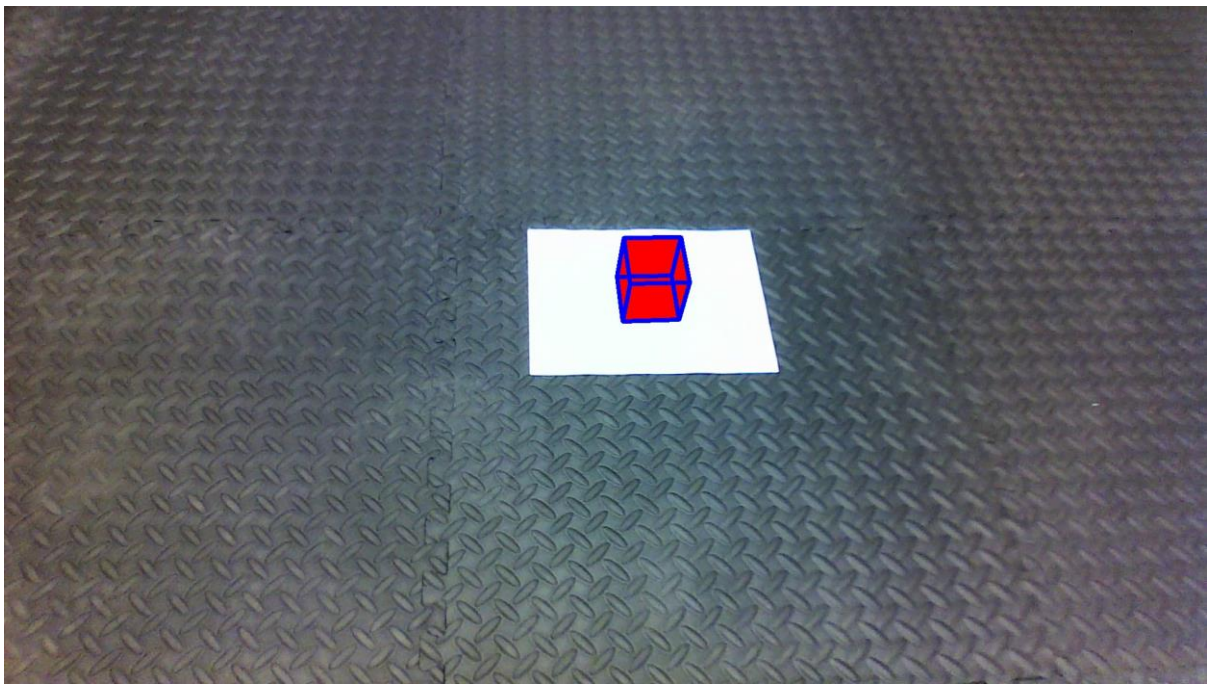


Fig. 3-D Cube projected on the Tag

Challenges faced during implementation

We tried implementing the edge detection techniques like canny and sobel at first but couldn't get satisfactory output. Then we used findContours and area as the deciding parameter but couldn't differentiate between the required contour and others consistently. Then, we studied about the findContours retrieval mode which led us to satisfactorily differentiate between the required contour and others.

Conclusion

By implementing this project, we learnt the concepts as well as practical implementation of image processing operations like filtering, edge detection and projective transformations.

Output Video Link

https://drive.google.com/drive/folders/1S60uAXfZ9dW9W8br1r_xlJDvvcQI_db7

References

1. https://docs.opencv.org/3.2.0/d3/dc0/group__imgproc__shape.html#ga0012a5fdaea70b8a9970165d98722b4c
2. https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html
3. https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html
4. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html