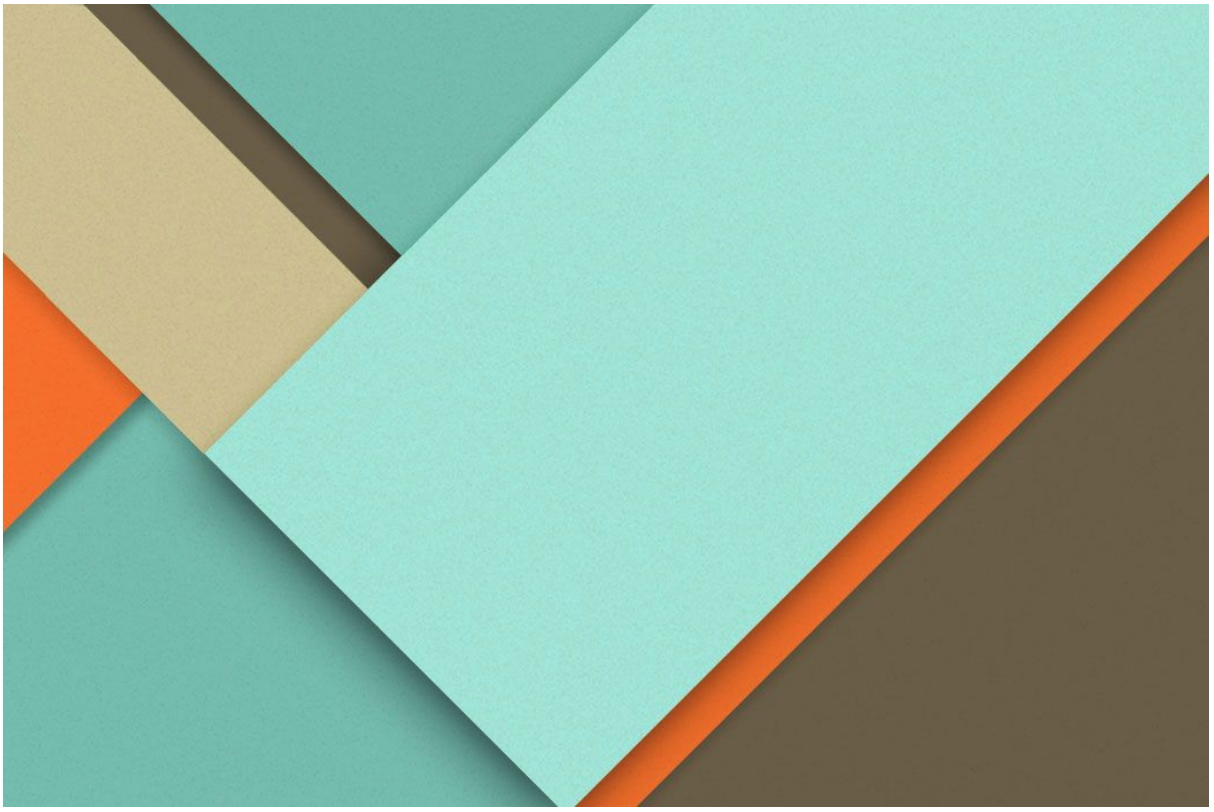

Assignment 3 - Augmented Reality

23.10.2019

Ritesh Saha
2017CS10481

Aditya Senthilnathan
2017CS50403

IIT Delhi
Dept. of Computer Science and Engineering

Augmented Reality Application

Uploaded files Description:

video3.py: Python code to run marker detection on live camera feed or to generate video for question 4.

helper.py: Consists of helper functions.

Steps:

0. Estimate Camera Calibration Matrix

We took 10 images of chessboard in various positions and estimated camera calibration matrix from these images.

1. Finding Contours in the Image

We found contours in the image by resizing the image for efficiency and then performing global thresholding on it and then detecting contours on the image and then performed polygonal approximation of contours and rejected all those with no. of points not equal to 4 and with perimeter lesser than a certain threshold.

2. Validating Candidate Contours

We performed performed perspective transform to obtain the portion of the contour inside the bounding box and compared it with our list of model markers using SIFT feature detector and we rejected the ones which didn't have matches greater than 15.

Feature matching

To establish a homography between two images, we first need to find a set of correspondences between them. One common way of doing this is to identify "interest points" or "key points" in both images, summarize their appearances using descriptors, and then establish matches between these "features" (interest points combined with their descriptors) by choosing features with similar descriptors from each image. Extracted keypoints and descriptors using the SIFT interface to compute interest points and descriptors, and using one of the descriptor matchers OpenCV provides under a common interface. We found it useful to visualize the matched features between two images to see how many of them look correct. RANSAC technique to try to reduce the number of outliers. The fewer there are, the better the fit of your homography will be, although we were never be able to eliminate all of them.

Fitting the homography

Once established correspondences between two images, used them to find a “best” homography mapping one image into the frame of another. cv2.findHomography, which can compute the least-squares-best-fit homography given a set of corresponding points. It also offers robustified computation using RANSAC or the Least-Median methods.We tweaked the robustifiers' parameters to get the best results.

3. Estimating Homography and Perspective Projection

By mapping correspondences from the corners and midpoints between corners of the model marker matched to the contour corners and midpoints we were able to estimate the homography matrix and from there we were able to extrapolate the projection matrix. From the projection matrix we were able to find the x,y,z axes of the marker

4. Projecting the Object

We read the data pertaining to the wavefront object from a file stored locally and projected it onto the estimated center of the marker.

5. Moving Object from Marker to Marker

Knowing the axis of the markers using standard 3D points we used cv2.PerspectiveTransform to do so. After this we approached in the direction of 2nd Plane Z Axis. We stopped as we approached the slope of one of the axes of the 2nd plane.