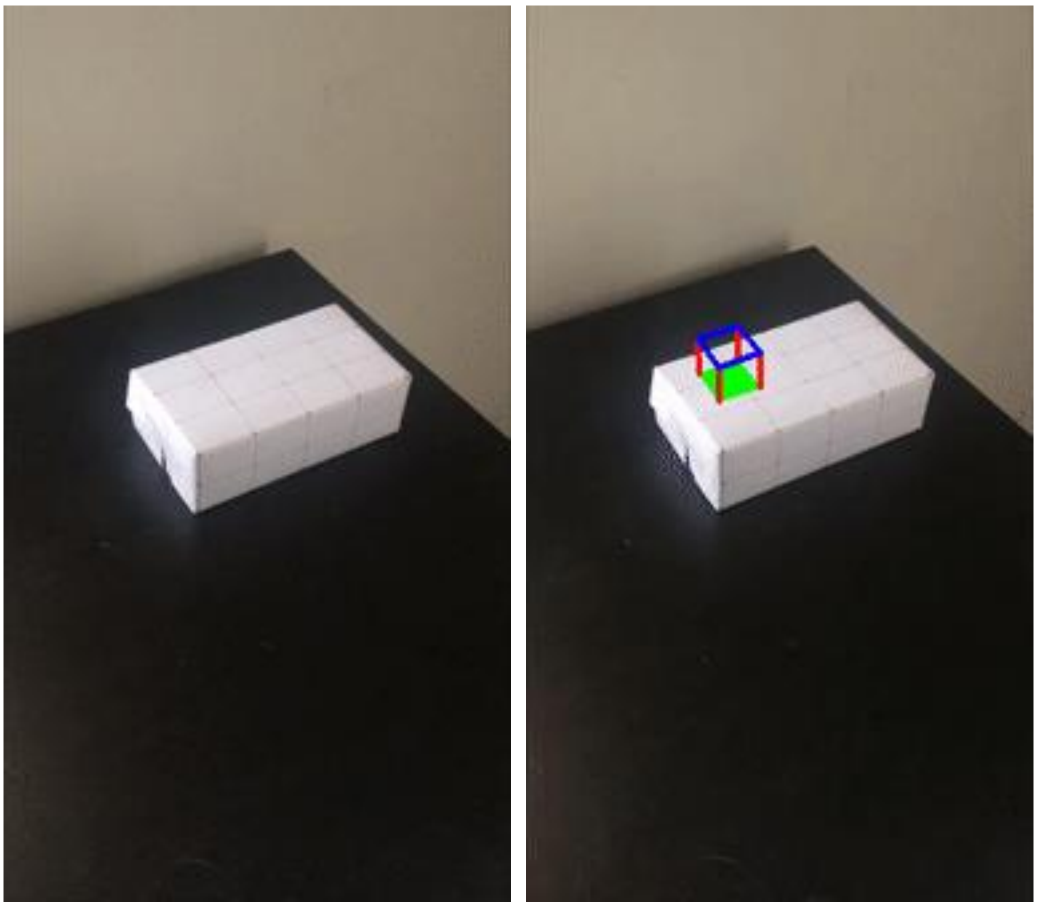
Easy Augmented Reality



(a) Input Video (b) Result

Figure 1: Example of Augmented Reality Project: Capture a video (a), insert a synthetic object into the scene and get the result (b).

In this augmented reality project, you will capture a video and insert a synthetic object into the scene, as shown in Figure 1. The basic idea is to use 2D points in the image whose 3D coordinates are known to calibrate the camera for every video frame and then use the camera projection matrix to project the 3D coordinates of a cube onto the image.

1. **Set up:** Find a flat surface and place a box on the surface. Draw a regular pattern on the box. Decide on at least 20 points from the pattern which you will mark in the image, and label their corresponding 3D points. Make sure that they are not all planar. Capture a video with the box at the center as *Figure 1 (a).*

2. **Mark and Propagate Keypoints:** Marking the points in the first image of the video and getting their 3D world points. You need to measure the length of the sides of the box and the distance between the consecutive points in the pattern. Then propagating the points from the first image to the subsequent images. The result of this procedure should be a

paired set of 2D and 3D points for every frame in the video.

3. **Calibrate the Camera：** Once you have the 2D image coordinates of marked points and their corresponding 3D coordinates, use least squares to fit the camera projection matrix to project the 4 dimensional real world coordinates (homogenous coordinates) to 3 dimensional image coordinates (again, homogenous coordinates). Perform this step separately for each frame in the video.

4. **Project a cube in the Scene：** Once you have the camera projection matrix, Project the axis points defined at the end of Camera Calibration into Resources and use the *draw* function to draw the cube on the image (defined just above the axis points in Camera Calibration). Note that the *draw* function takes an unnecessary parameter *corners* which it doesn't use. Just inputting the projected points for *imgpts* should suffice. This will place the cube of size 1 unit at (0,0,0). You should translate and scale the coordinates in the axes points to place the cube at a suitable location.

5. **Show Result:** Once you render the cube independently for each image, you can combine the images into a video and view the output result.

Several ways of propagating the keypoints:

There are several ways of propogating the points from the first image to the subsequent images. The end result of this procedure should be a paired set of 2D and 3D points for every frame in the video:

1. **Corner Detector:** One way to propagate the points from the one image to the next image is by exploiting the temporal signal in the video.
   First, we will detect the corners in img[i] and img[i+1] using a harris corner detector. Let us call them *ci* and *cnext* respectively. Since the points will not move by a large amount in consecutive frames, we can find the closest point (in pixel space) from the set *cnext* for every point in *ci*. We will further accept only those points for which the pixel space distance is below some threshold.

2. **Off the Shelf Tracker:** You can also use an off-the-shelf tracker. This tutorial of object tracking explains the usage of various trackers available in cv2. You can use the MedianFlow tracker: *cv2.TrackerMedianFlow_create()*. You need to initialize a separate tracker for each point. I used an 8x8 patch centered around the marked point to initialize the tracker. Note that the bbox describes the bounding box using 4 values, where the first two coordinates are the start (top left) coordinate of the box, followed by the width and height of the bounding box. Update the trackers for each new frame to get the points on the next frame. Keep a track of the points and their corresponding 3D points.

   The result of the tracked points should look something like this: