Assignment 4- Calibration and Augmented Reality

The assignment aims to find the distortion coefficients which occur because of imperfections in the lens or wide-eye feature, intrinsic camera matrix, and then using these values to transform our 3-D visualized/assumed objects back to 2-D image plane. To accomplish these tasks the project is divided into 7 tasks. Each task we will discuss further –

TASK 1 –

To find the inner corners of the checkerboard. To find these inner corners which are 9 in the columns and 6 in the rows we use OpenCV standard functions –

1. **findChessboardCorners –**

   It takes an image as input and uses an algorithm to locate the internal corners of the chessboard. The function returns a Boolean value indicating whether it was able to find the corners, as well as the locations of the corners if they were found. This function is commonly used in applications like camera calibration, where precise knowledge of the location of the chessboard corners is important for accurate measurement and analysis.
   Used these flags –

- *CALIB_CB_ADAPTIVE_THRESH:* This flag enables adaptive thresholding, which adjusts the threshold value based on the local pixel intensity of the image. This can help improve the detection of chessboard corners in images with varying lighting conditions.
- *CALIB_CB_NORMALIZE_IMAGE:* This flag normalizes the image to improve the accuracy of the corner detection algorithm. Normalization rescales the pixel intensities of the image so that they fall within a certain range, which can help to reduce the effects of noise and other image artifacts.
- *CALIB_CB_FAST_CHECK:* This flag enables a fast check method for detecting corners, which can significantly speed up the corner detection process. This method uses a simple algorithm to quickly check whether the input image contains a chessboard pattern before performing more computationally expensive operations.

2. **cornerSubPix –**

   It takes an image, the original corner locations, and some other optional parameters as input and uses an iterative algorithm to refine the corner positions to sub-pixel accuracy. This function is useful in applications where precise knowledge of the corner locations is important, such as camera calibration or object tracking. By refining the corner positions to sub-pixel accuracy, it can improve the accuracy of subsequent image processing operations that rely on those corner locations.

   Termination criteria used is - TermCriteria(CV_TERMCRIT_EPS + CV_TERMCRIT_ITER, 30, 0.1)) where epsilon value is 0.1 and max iterations are 30.

TASK 2 –

After finding the inner corners using the method described in task1 we draw a circle on each point to highlight it. Here are some calibrated images screenshots –

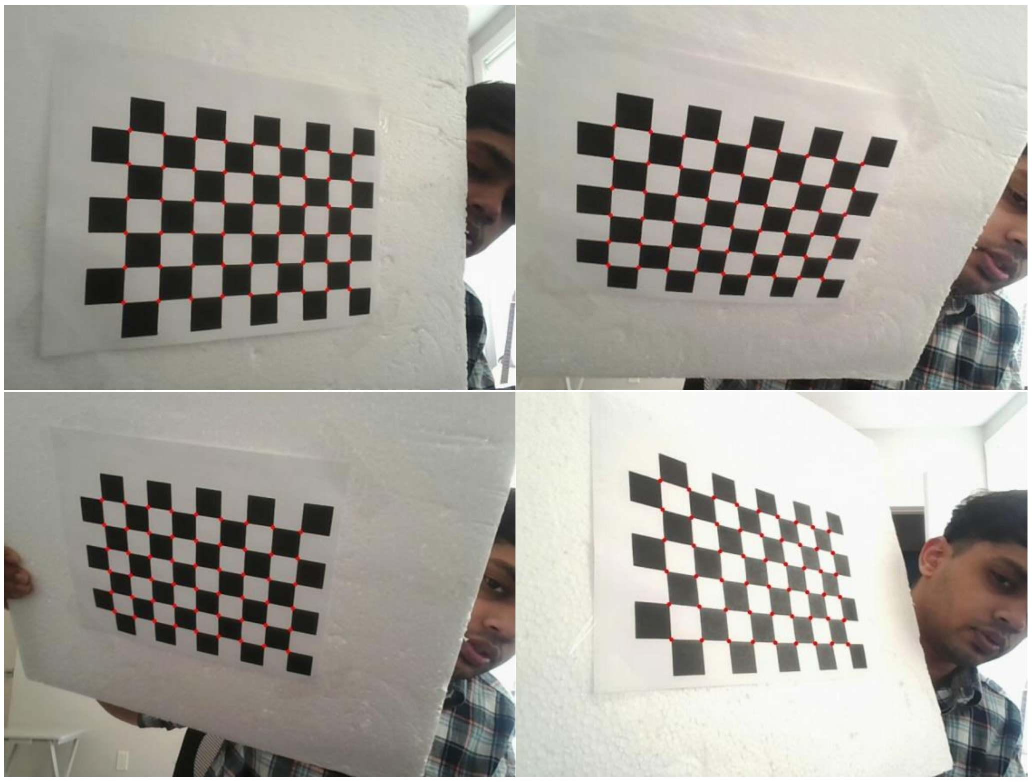
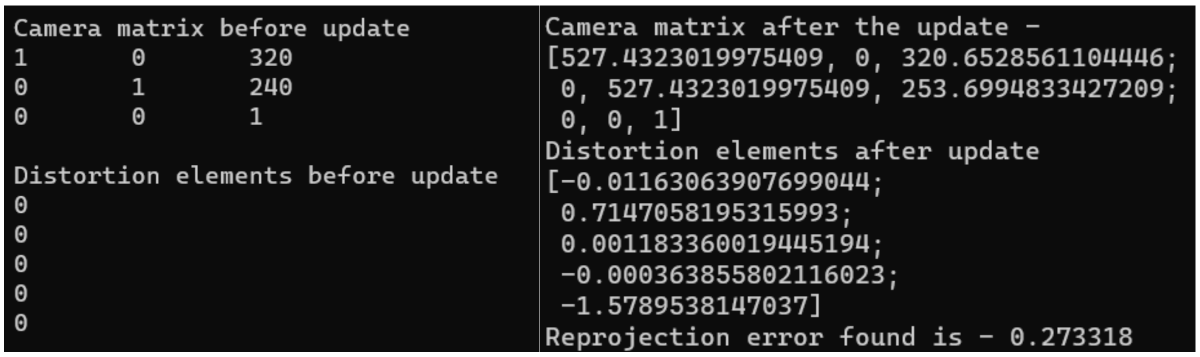


Fig 1. – Inner corners found screenshots.

TASK 3 –

calibrateCamera is a function in OpenCV that takes a set of calibration images with a known pattern and estimates the intrinsic and extrinsic parameters of a camera. The function uses the correspondences between the pattern points and their image coordinates to calibrate the camera, which involves estimating the focal length, principal point, lens distortion coefficients, and the position and orientation of the camera in 3D space. The output of the function is the calibrated camera matrix, distortion coefficients, and other parameters that can be used for image rectification, 3D reconstruction, and other computer vision tasks.



```
Camera matrix before update
1       0       320
0       1       240
0       0       1

Distortion elements before update
0
0
0
0
0
```

```
Camera matrix after the update -
[527.4323019975409, 0, 320.6528561104446;
 0, 527.4323019975409, 253.6994833427209;
 0, 0, 1]
Distortion elements after update
[-0.01163063907699044;
 0.7147058195315993;
 0.001183360019445194;
 -0.000363855802116023;
 -1.5789538147037]
Reprojection error found is - 0.273318
```

Fig 2. – Camera intrinsic matrix and distortion matrix before and after update.

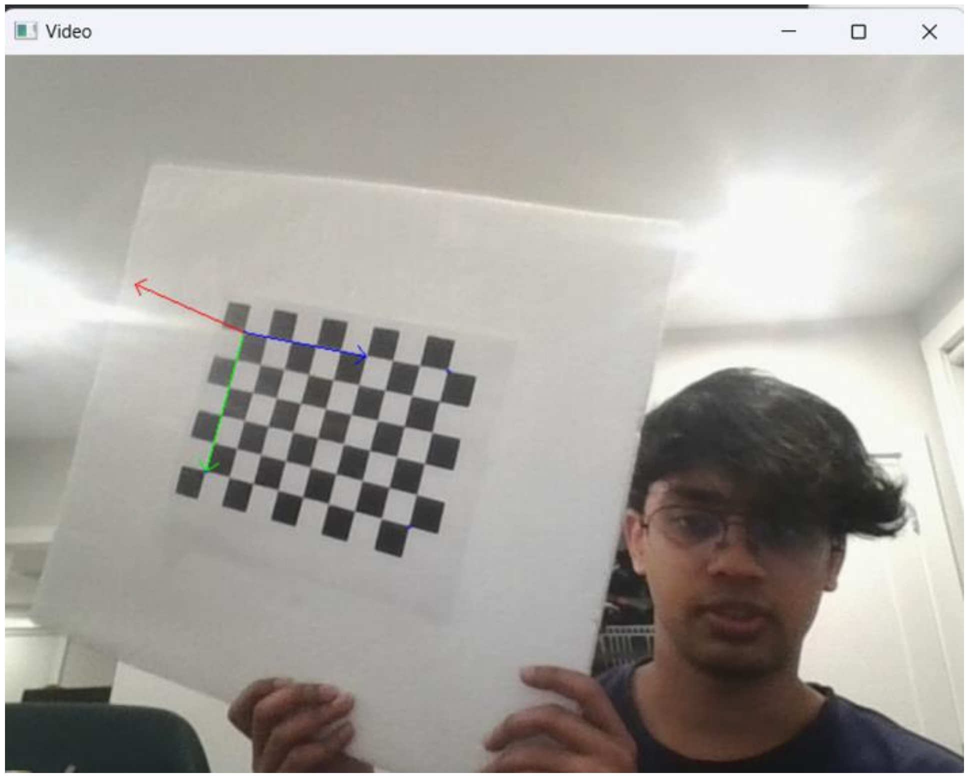
TASK 4 –

Now for each frame whenever our standard checkerboard is shown we can estimate the camera pose i.e. rotation matrix and position from the checkerboard. This is calculated using solvePNP OpenCV function. The solvePNP function returns a boolean value that indicates whether the solution was successful or not. If successful, the function sets the rvec and tvec output parameters to the estimated rotation and translation vectors, respectively. CALIB_FIX_ASPECT_RATIO is used as a flag to fix the aspect ratio of the camera's intrinsic matrix during the calibration process.

TASK 5 –

Now, as we have information about our intrinsic and extrinsic transformation matrix. We can project any point in our 3D world to a 2D image plane of the frame. To do this we use projectPoints function of OpenCV which takes in a set of 3D points, a camera matrix (which represents the intrinsic parameters of a camera), a rotation vector, a translation vector, and an optional distortion vector, and returns the corresponding 2D image coordinates of the projected points. We have projected both the corner points (in blue maybe difficult to observe) and our 3D axis on the checkerboard.

Here are the screenshots for the same –



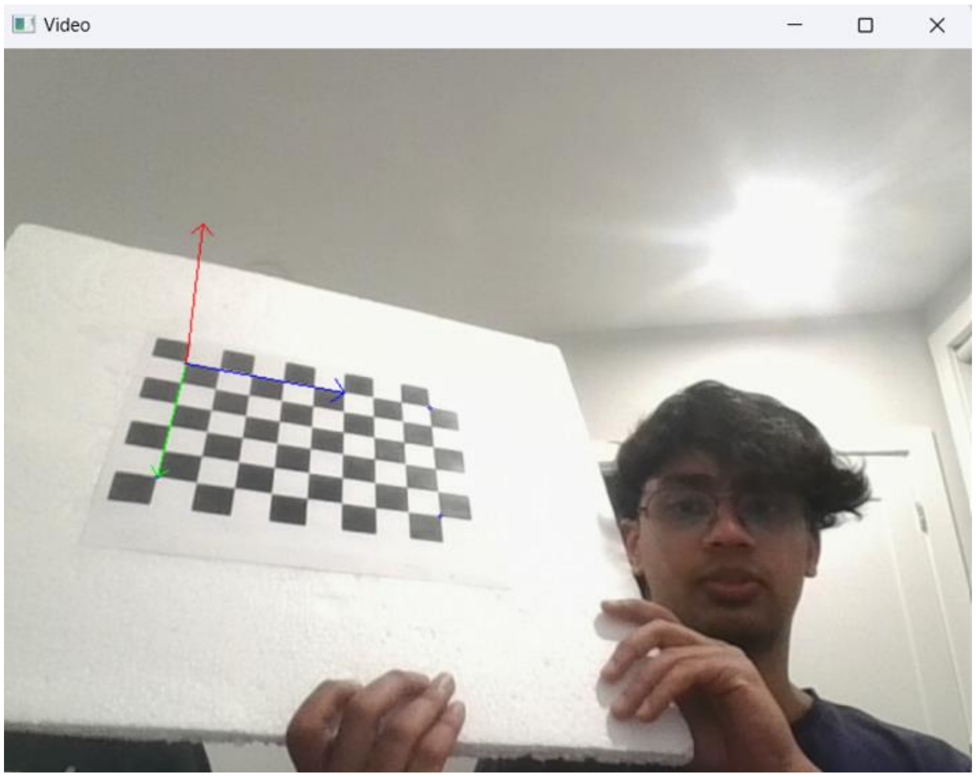
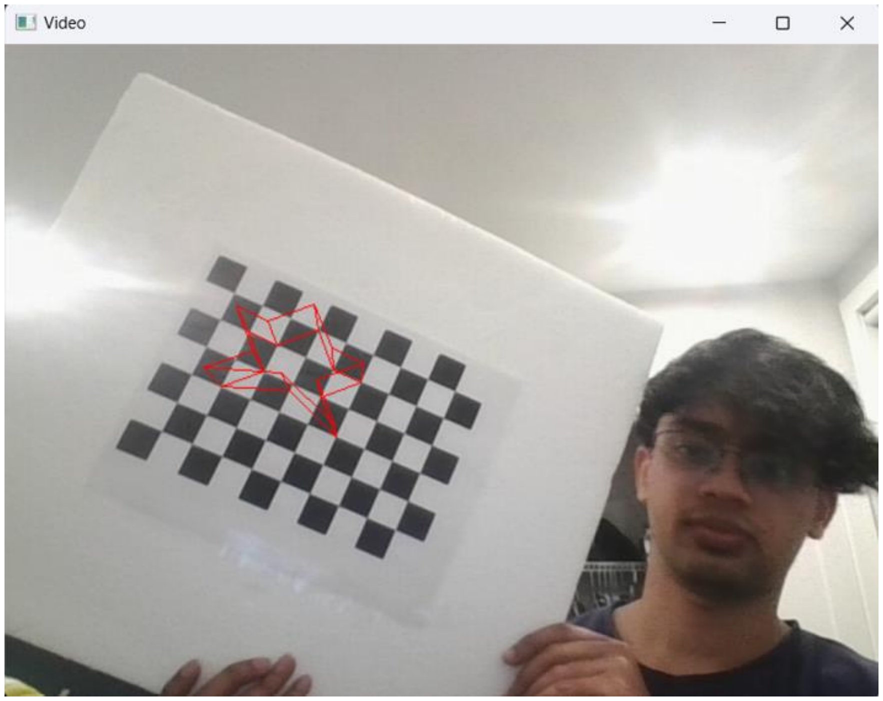


Fig 3. – 3D axes and co-ordinates of our inner 'cornerest' corners

TASK 6 –

Projecting a 3D object to our checkerboard. Like the corner points and 3D axis we imagine our 3D object and then project it to the image plane. We have made a ***star*** as our 3D object.



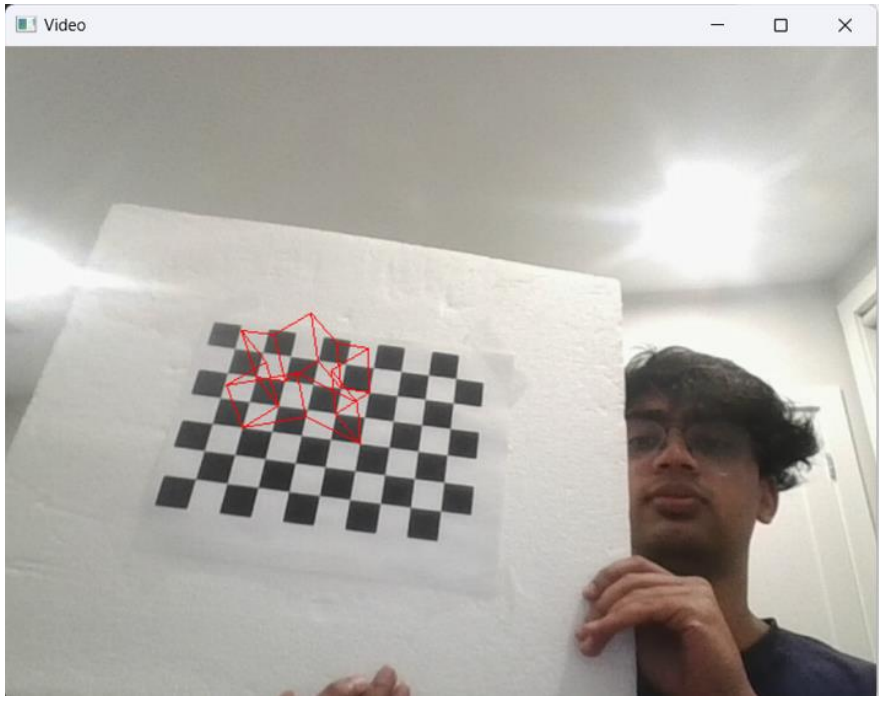


Fig 4. – Our 3D object star projected to the checkerboard plane

TASK 7 –

We have used Harris corner detector as the feature finder. The Harris corner detector is a popular method used in computer vision to detect corner features in an image. It works by computing the local variations in intensity in different directions at each pixel, and then looking for regions where these variations are high in all directions. This is done by computing a corner response function using the gradients of the image and a window function, and then comparing this response function to a threshold to determine whether a pixel is a corner or not. The Harris corner detector is widely used in applications such as feature extraction, object recognition, and image alignment.

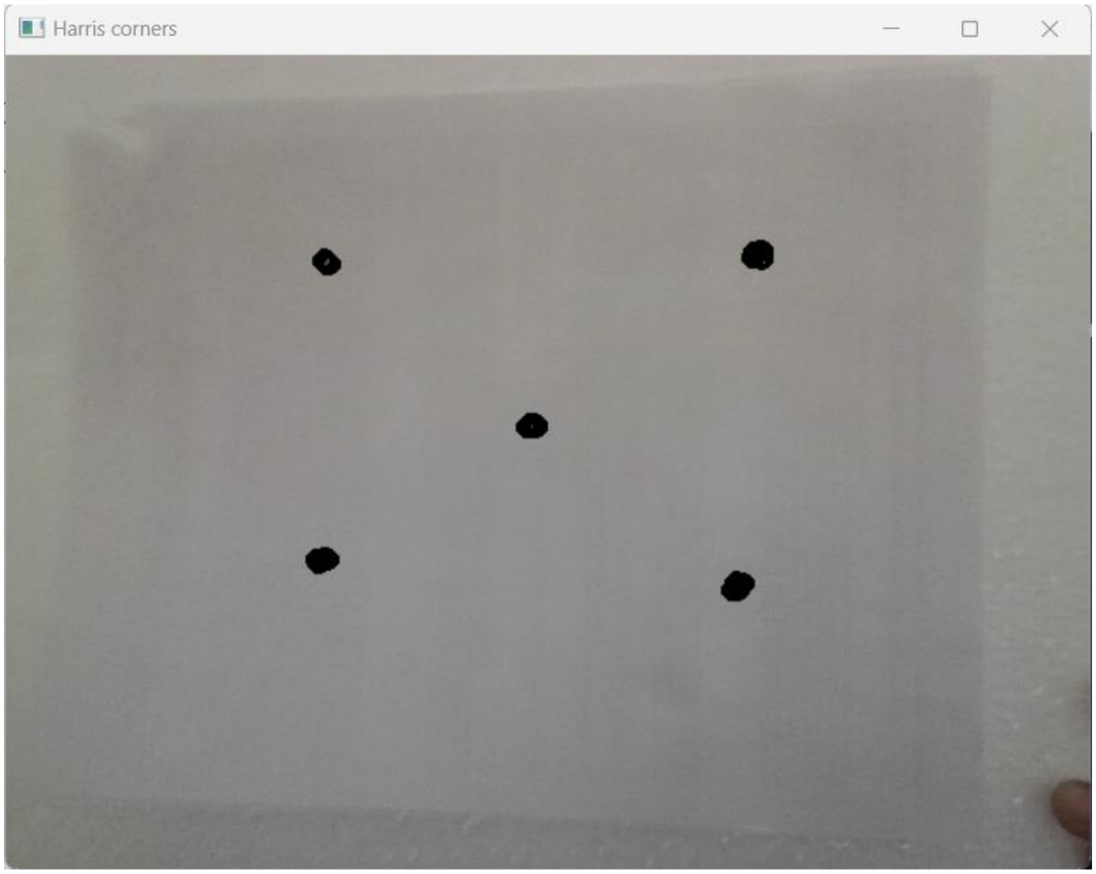


Fig 4. – Our 3D object star projected to the checkerboard plane

We tested our implementation on a custom design of dot positions. The corners detected are highlighted in the screenshots. In Augmented Reality, we can use this feature on a white background to project images/videos on the dots or any highlight which reflects in our Harris feature detector.

EXTENSION –

For extension, we have warped Pikachu on a cross 'X' drawn on white paper. This is implemented using the following steps to the frame.

1. Convert frame to grayscale for ease in computation
2. Then threshold the frame to convert to binary frame
3. Then clean the binary frame to remove noise
4. Find the cross sign, where the number of contours == '2'
5. If found, get perspective transform and warp our Pikachu to the frame
6. Mask the cross so that our Pikachu merges with the background

Additional functions used –

'getPerspectiveTransform' is a function in computer vision libraries such as OpenCV that calculates a perspective transformation matrix from four pairs of corresponding points in two images. The perspective transformation matrix can be used to transform the perspective of one image to align with the other image. In simpler terms, it helps in finding the transformation between two sets of four points in order to warp one image to match the perspective of another image.

'warpPerspective' is a function in computer vision libraries such as OpenCV that applies a perspective transformation to an image using a transformation matrix. Given the input image and the transformation matrix, warpPerspective applies the transformation to the image to produce an output image with the same size as the input image. The function is useful in tasks such as image registration, stitching, and panorama creation, where it is necessary to align and transform images taken from different viewpoints or angles.

Input/output of our implementation –

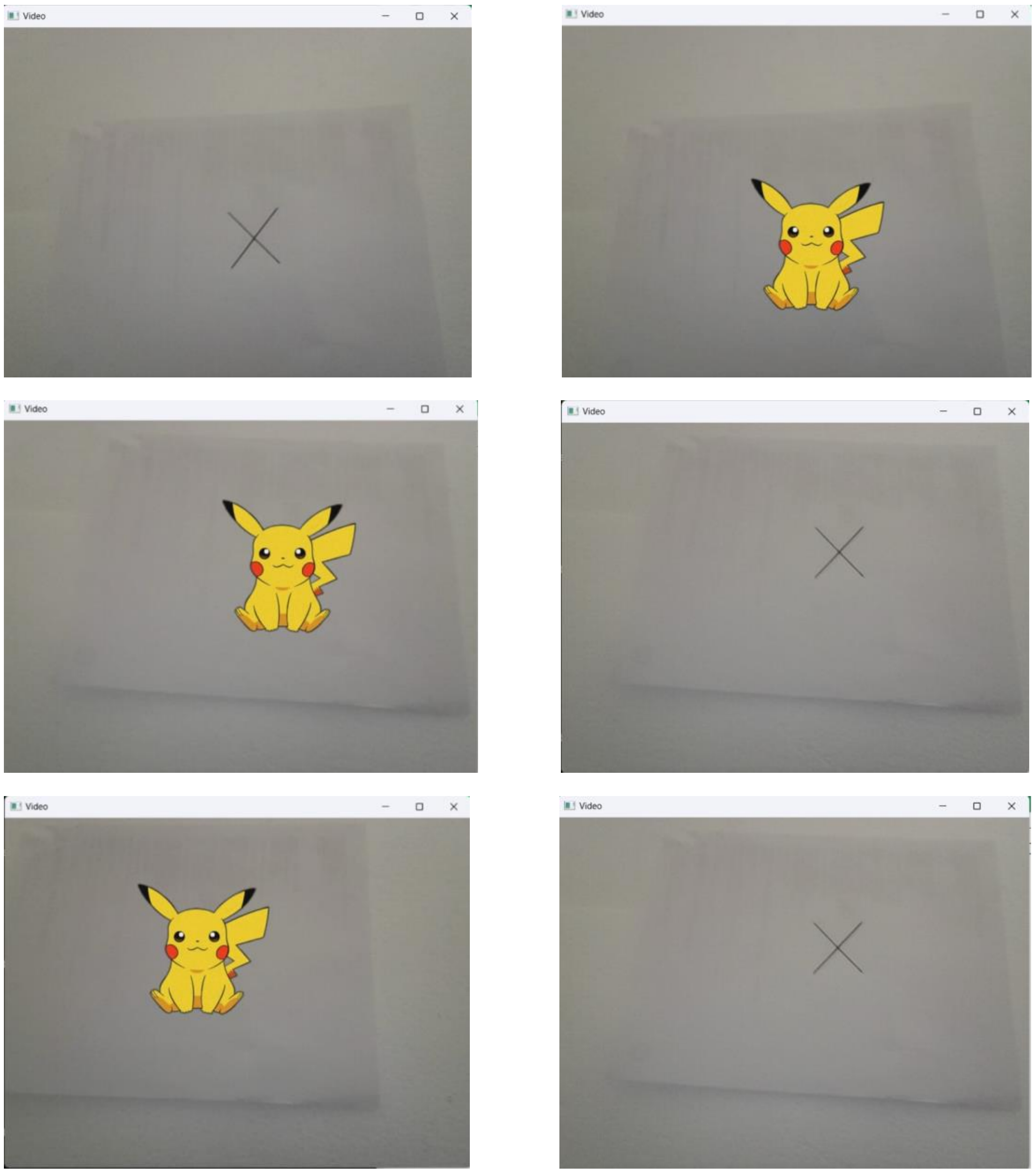


Fig 5. – Extension – Warping image on cross ' X '

Reflection –

This assignment was slightly easier than previous one but faced some difficulty in datatypes. Overall, the assignment was fun and with greater understanding development in transformations.

Acknowledgement –

Used Professor's lectures, stack-overflow and google for debugging.