Project #2 – Planar Augmented Reality

Omri Gruman, 318965621 Mark Klaiman, 208734715

1 Overview

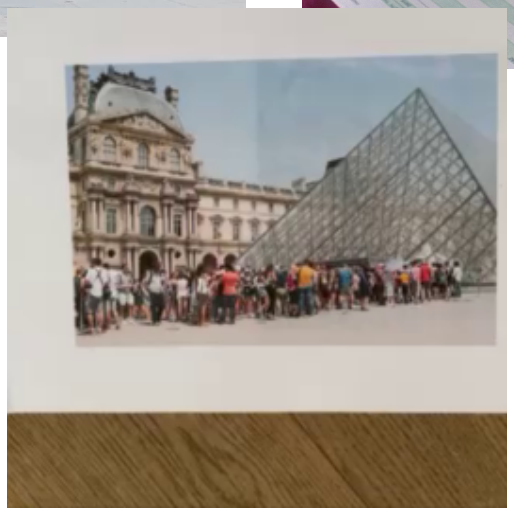
In this project we are given several tasks regarding planar augmented reality. First, we are asked to project an image (e.g., a dog) onto a video of a template image (e.g., the Louvre). Then, for the harder part of the project we need to project a given 3D model onto the video from the first task. But before we can complete the second assignment, we also need to calibrate our camera.

2 Perspective Warping

In the first part of the project, we try to match key features of both the template image as well as each frame of the video in order to calculate the geometric transformation between the two. After such transformation is found, we warp the image of the dog onto the frame to replace the template image.

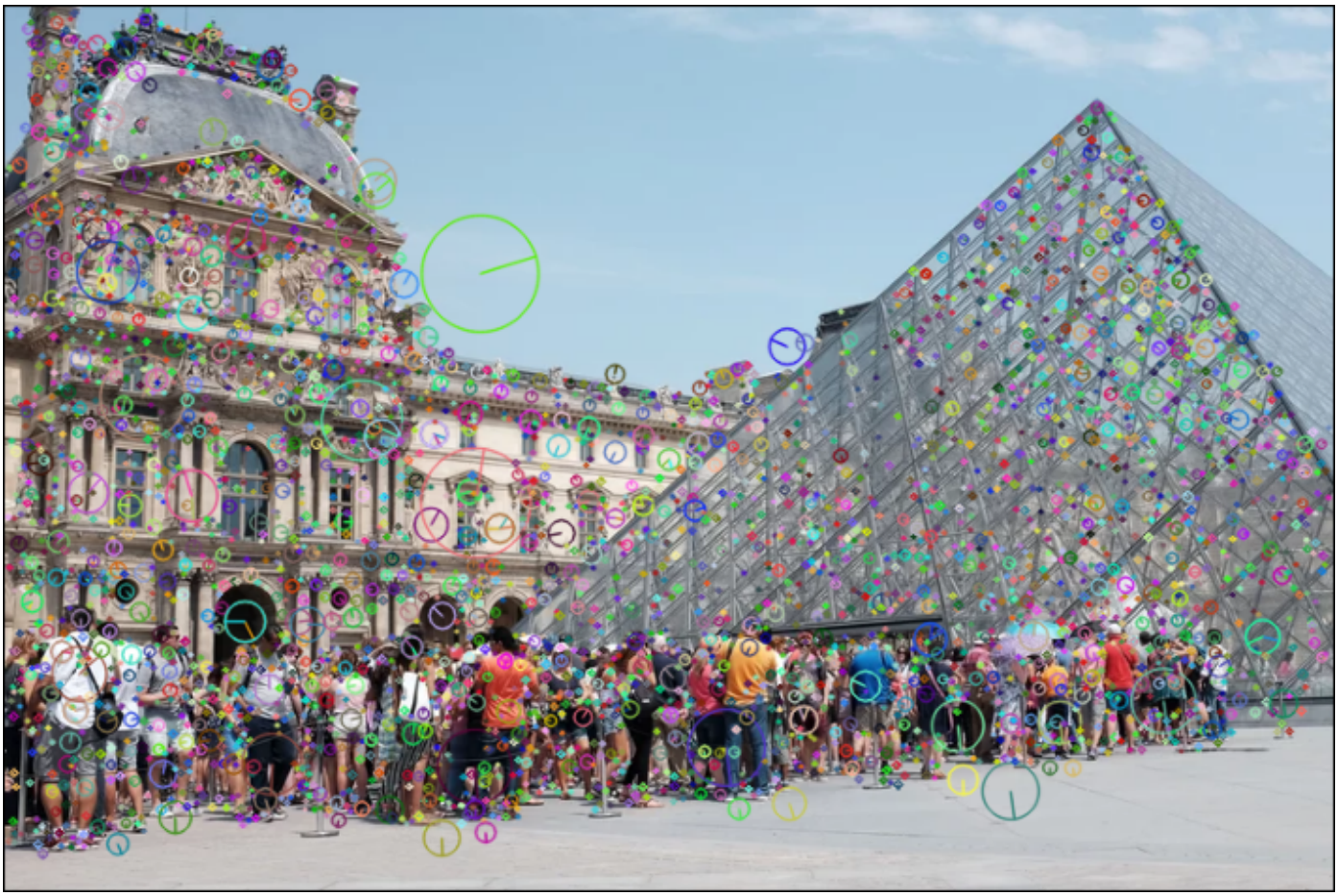
2.1 Video preparation

Before getting to code, we had to choose a template image, in this case the Louvre, and an image to warp, we chose an image of a dog. In addition, we took a video displaying the template image.



2.2 Key points extraction

After producing the images and the video, we use the SIFT algorithm to detect key features in both our template image (as shown below) and each video frame. SIFT detects features in a way that helps us locate similar features even though there are many differences between our template image and the video frame that affect different attributes of those features.



2.3 Key points matchings

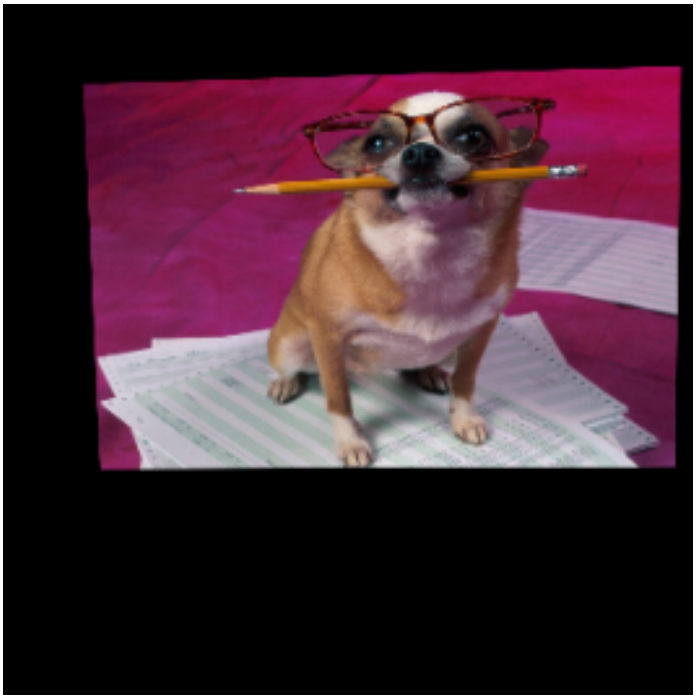
Then, we find a good match between the different key points in the template and the video frame.

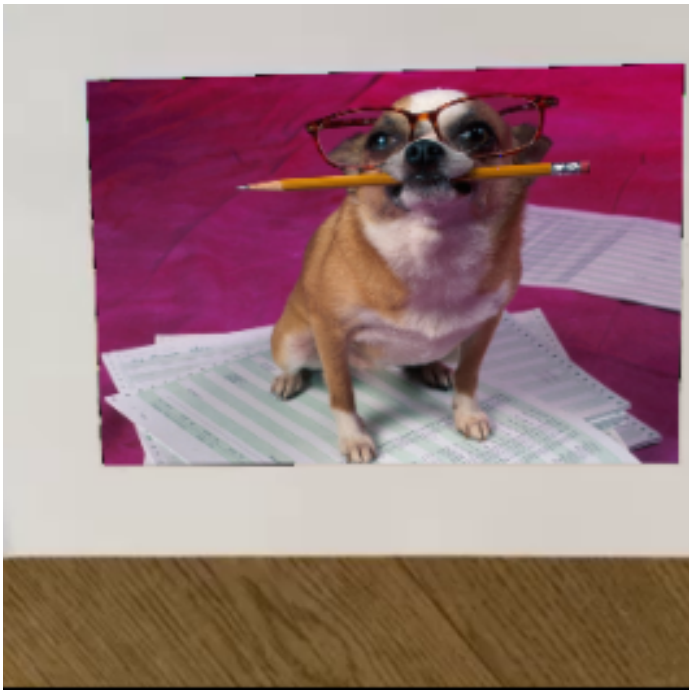
2.4 Homography finding

Using the matched key points we found, we compute a homography matrix, which represents the transformation from the template image to the video frame.

2.5 Image perspective warping

At last, we use the homography matrix we found to warp the image of the dog to match the orientation of the template image in the video frame, and mask it onto the frame as shown below.



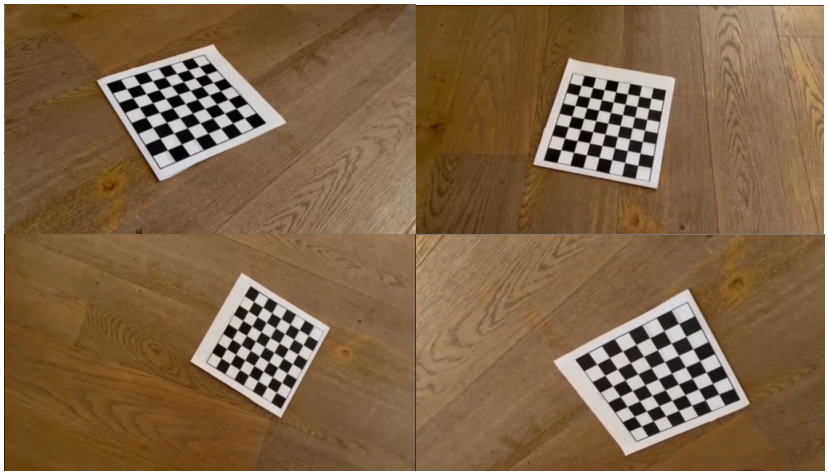


3 Calibration

As said above, before we can take our 3D model and project it onto our video frames, we need to calibrate our camera and find its camera matrix and distortion coefficients using Zhang's algorithm.

3.1 Chessboard frames preparation

First, we had to produce several chessboard frames in different orientations (as shown below). Rather than taking several photos of the board, we preferred to take a short video of it and extract different frames to make sure that our camera mode is the same for both the chessboard frames and the Louvre video. That is done so the calibration will be effective when we move to the next stage of projecting the model onto the video.



3.2 Corners finding

Then, for each chessboard image we find its internal corners as part of the algorithm's process.

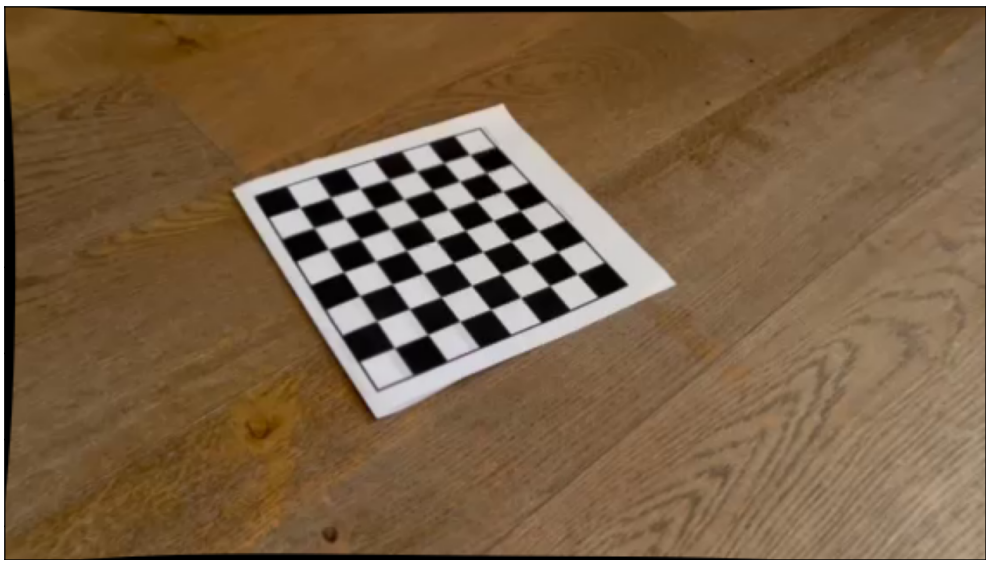
3.3

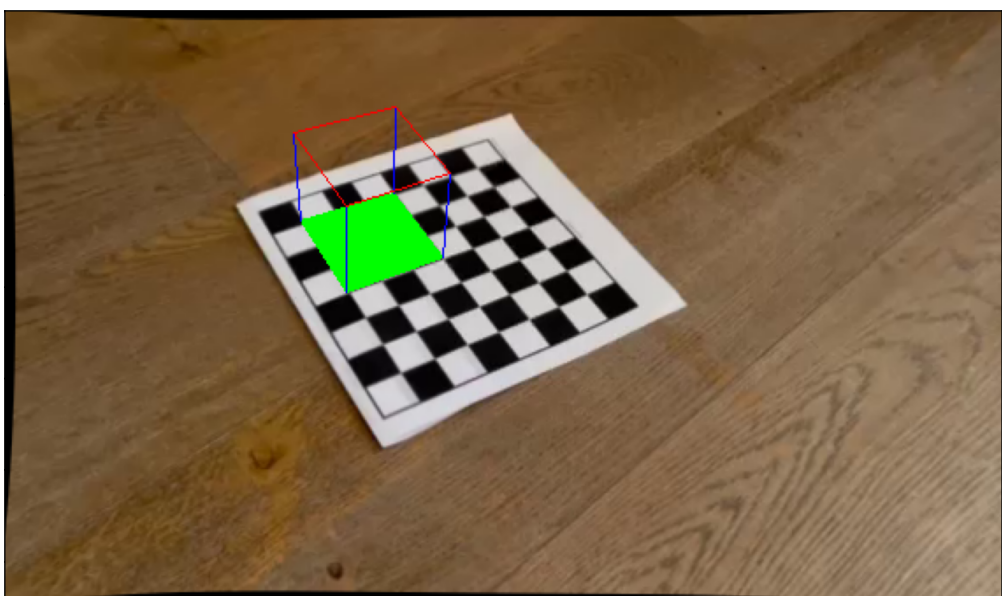
Camera calibration

After we locate our chessboard corners, we calculate our camera matrix and our distortion coefficients. Now we can draw different 3D models onto our video frame!

3.4 Cube projection

To verify our calibration process, we undistort our image and draw a simple cube onto our chessboard image.





4 3D model projection

Finally, our final part of the project requires us to use the same homography calculation from before in order to select the optimal pairs of key points between the template image and the video frame. This is done to solve the Perceptive-n-Point problem to estimate the camera position in space relative to origin, in this case the top left corner of the template image in the video.

4.1 Key points selection

First, we use the homography found in the first section to select the best pairs of key points that obey the homography to remain with the optimal matches between the template image and the video frame.

4.2 Perceptive-N-point solving

Next, we use the *solvePnP* method from opencv library to calculate the rotation and translation of our camera relative to the real world coordinates.

Perspective-*n*-Point is the problem of estimating the pose of a calibrated camera given a set of *n* 3D points in the world and their corresponding 2D projections in the image. The camera pose consists of 6 degrees-of-freedom (DOF) which are made up of the and 3D translation of the camera with respect to the world. This problem originates from camera calibration and has many applications in computer vision and other areas, including 3D pose estimation, robotics, and augmented reality.

Given a set of *n* 3D points in a world reference frame and their corresponding 2D image projections as well as the calibrated intrinsic camera parameters, determine the 6 DOF pose of the camera in the form of its rotation and translation with respect to the world. This follows the perspective projection model for cameras:

$$s\, p_c = K\,[\,R\,|\,T\,]\, p_w\,.$$

where:

$p_w = [\,x \quad y \quad z \quad 1\,]^T$ is the homogeneous world point, $p_c = [\,u \quad v \quad 1\,]^T$ is the corresponding homogeneous image point, ***s*** is a scale factor for the image point, ***K*** is the matrix of intrinsic camera parameters (where ***fx*** and ***fy*** are the scaled focal lengths, ***lambda*** is the skew parameter which is sometimes assumed to be 0, and ***(u0, v0)*** is the principal point), and ***R*** and ***T*** are the desired 3D rotation and 3D translation of the camera (extrinsic parameters) that are being calculated.

This results in the following equation:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

The PnP problem has different variants, including P3P which uses only 3 points, EPnP which stands for Efficient PnP, SQPnP (i.e., Sequential Quadratic), and an additional method that uses RANSAC.

4.2.1 P3P

When *n* = 3, the P*n*P problem is in its minimal form of P3P and can be solved with three-point correspondences. However, with just three-point correspondences, P3P yields up to four real, geometrically feasible solutions. For low noise levels a fourth correspondence can be used to remove ambiguity. The setup for the problem is as follows.

Let *P* be the center of projection for the camera, *A*, *B*, and *C* be 3D world points with corresponding images points *u*, *v*, and *w* $\alpha = \angle BPC, \beta = \angle APC, \gamma = \angle APB$ $p = 2\cos\alpha, q = 2\cos\beta, r = 2\cos\gamma$ $a' = |AB|, b' = |BC|, c' = |AC|$. Let *X* = |*PA*|, *Y* = |*PB*|, *Z* = |*PC*|, .

This forms triangles *PBC*, *PAC*, and *PAB* from which we obtain a sufficient equation system for P3P:

$$\begin{cases} Y^2 + Z^2 - YZp - b'^2 & = 0 \\ Z^2 + X^2 - XZq - c'^2 & = 0 \\ X^2 + Y^2 - XYr - a'^2 & = 0 \end{cases}.$$

4.2.2 EPnP

Efficient P*n*P (EP*n*P) is a method that solves the general problem of P*n*P for *n* ≥ 4. This method is based on the notion that each of the *n* points (which are called reference points) can be expressed as a weighted sum of four virtual control points. Thus, the coordinates of these control points become the unknowns of the problem. It is from these control points that the final pose of the camera is solved for.

4.2.3 SQPnP

SQPnP is a non-minimal, non-polynomial solver which casts P*n*P as a non-linear quadratic program. SQPnP identifies regions in the parameter space of 3D rotations that contain unique minima with guarantees that at least one of them is the global one. Each regional minimum is computed with sequential quadratic programming that is initiated at nearest orthogonal approximation matrices.

SQPnP has similar or even higher accuracy compared to state-of-the-art polynomial solvers, is globally optimal and computationally very efficient, being practically linear in the number of

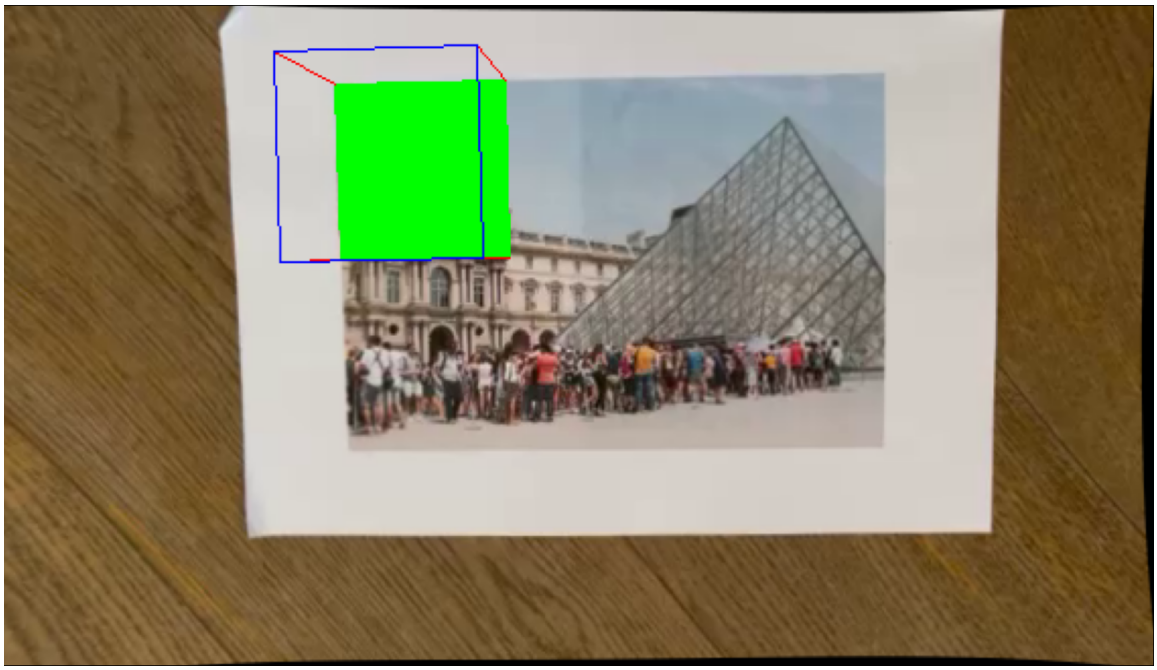
supplied points *n*.

4.2.4 RANSAC

P*n*P is prone to errors if there are outliers in the set of point correspondences. Thus, RANSAC can be used in conjunction with existing solutions to make the final solution for the camera pose more robust to outliers.

4.3 3D projection

At last, after acquiring the rotation and translation of the camera for in the video frame, we are free to draw 3D models onto our video. First, as always, we start with something simple like a



cube, and continue with a slightly more complicated model like a drill.

5 Assumptions

During the project we made several assumptions while calculating the homography and projecting 3D models onto the video.

- Our projection surface is planar (i.e., the printed image of the Louvre).
- The chessboard images and the video were taken by a camera with same parameters (camera matrix values & distortion coefficients).

6 Conclusion

In conclusion, we tried to solve several problems regarding planar augmented reality. First, we projected a dog image onto a video that displayed an image of the Louvre. Then, we calibrated our camera using the camera matrix and the distortion coefficients we found. At last, we were able to draw a cube as well as a 3D model of a drill onto the video frames.