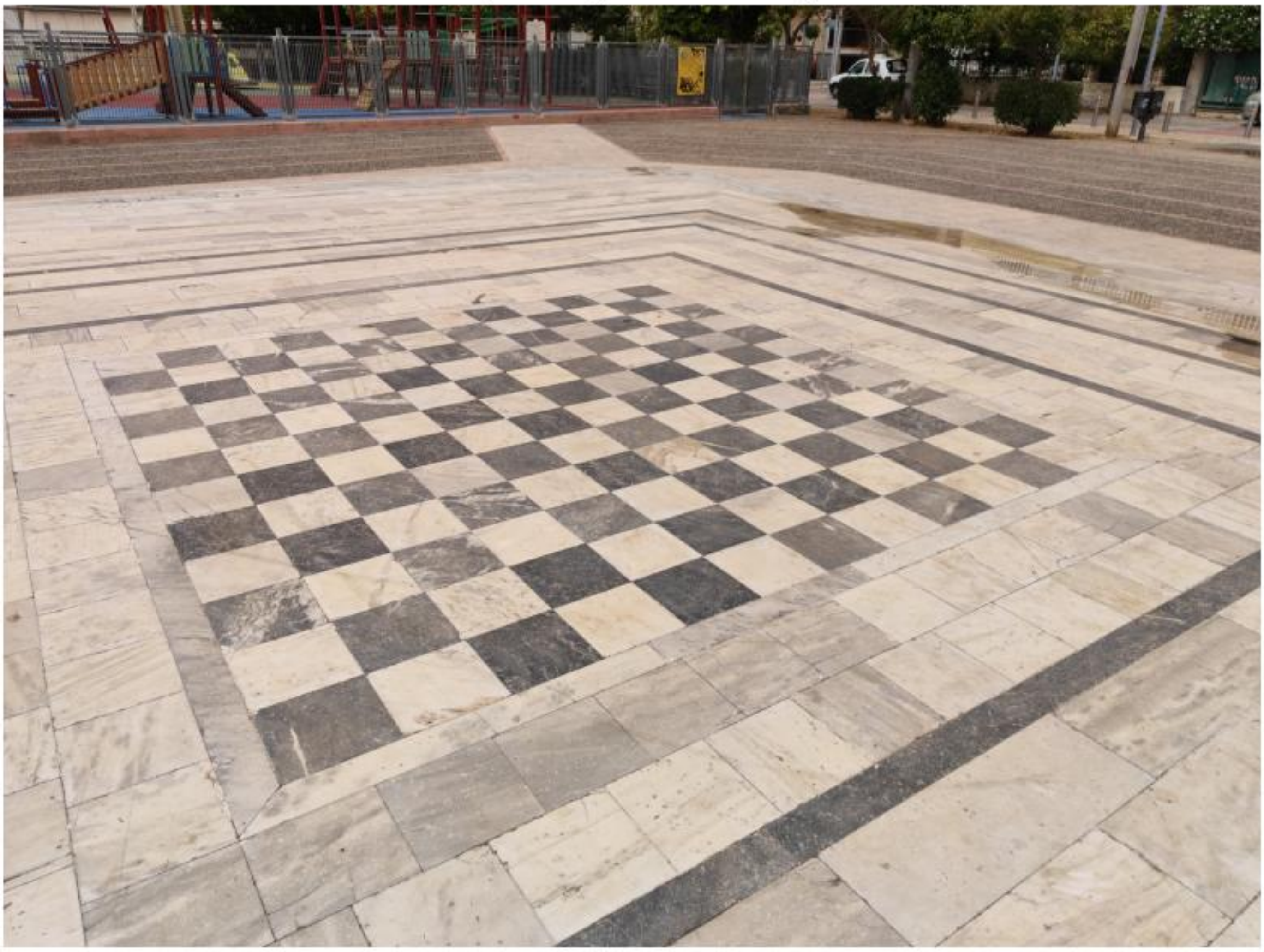
Computer Vision and Photogrammetry
3rd Task

Nikolaos Theokritos Tsopanidis aivc24022 24/06/2025

Contents

Perspectively Distorted Image



Introduction

The purpose of this paper was to calculate the projective transformation table (homography) **H** for the correction of the perspective distortion of the object projected in the photograph (chessboard). The calculation was based on a set of 11 known control points and their respective coordinates in the image.

Implementation

11 characteristic points on the chessboard were selected, where for each point, two pairs of coordinates were recorded.

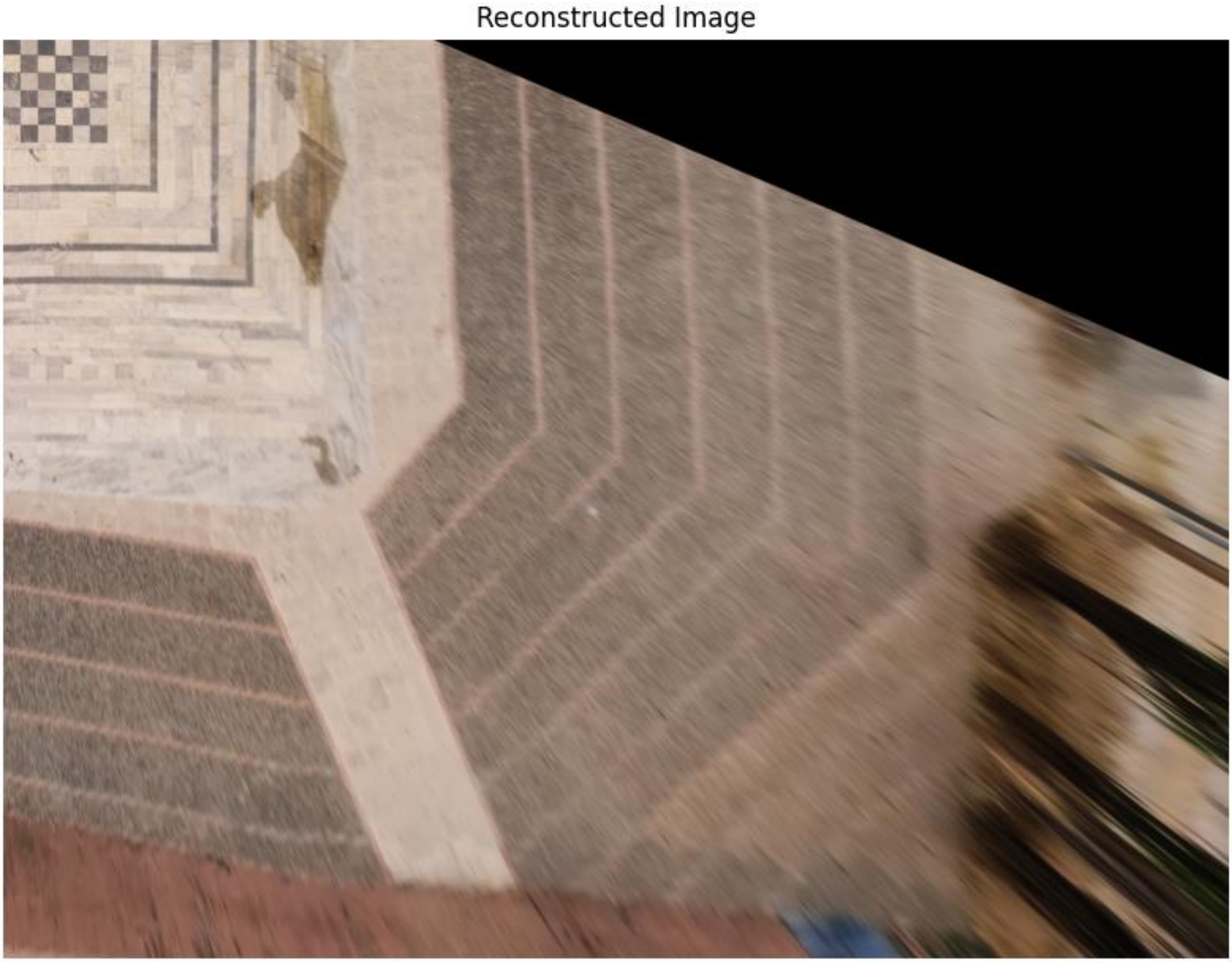
We have the actual **'obj_coords'** coordinates on the plane with an arbitrary beginning in centimeters (cm), and the corresponding **'pix_coords'** coordinates on the perspective distorted image.

OpenCV

Initially, well-known algorithms of the OpenCV library were used to calculate the **H** table and reconstruct the image. The first algorithm is **'findHomography'** where it gives the actual coordinates and the pixel coordinates, calculates the H table:

```
[[-4.01150798e+00  6.46857508e+00 -5.20893589e+02]
 [ 2.37038890e+00  1.11348604e+01 -8.29493726e+03]
 [ 9.23819681e-04 -3.11271151e-02  1.00000000e+00]]
```

Therefore, the **H** table was used for the perspective conversion of the original image with the '**warpPerspective**' algorithm, and we were able to find the corrected image visually confirming the success of the process.



Calculation with SVD

For the manual calculation of the 9 coefficients of the **H** table, the method of **Singular Value Decomposition** (**SVD**) was followed.

Initially, the table A (dimensions 22x9) was constructed, which results from the equations of collinearilty that connect the pairs of points.

$$x_2'(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{11}x_1 + H_{12}y_1 + H_{13} \quad (6)$$
$$y_2'(H_{31}x_1 + H_{32}y_1 + H_{33}) = H_{21}x_1 + H_{22}y_1 + H_{23} \quad (7)$$

We want to solve for H. Even though these inhomogeneous equations involve the coordinates nonlinearly, the coefficients of H appear linearly. Rearranging equations 6 and 7 we get,

$$\mathbf{a}_x^T\mathbf{h} = \mathbf{0} \quad (8)$$
$$\mathbf{a}_y^T\mathbf{h} = \mathbf{0} \quad (9)$$

where

$$\mathbf{h} = (H_{11}, H_{12}, H_{13}, H_{21}, H_{22}, H_{23}, H_{31}, H_{32}, H_{33})^T \quad (10)$$
$$\mathbf{a}_x = (-x_1, -y_1, -1, 0, 0, 0, x_2'x_1, x_2'y_1, x_2')^T \quad (11)$$
$$\mathbf{a}_y = (0, 0, 0, -x_1, -y_1, -1, y_2'x_1, y_2'y_1, y_2')^T. \quad (12)$$

Given a set of corresponding points, we can form the following linear system of equations,

$$A\mathbf{h} = \mathbf{0} \quad (13)$$

where

$$A = \begin{pmatrix} \mathbf{a}_{x1}^T \\ \mathbf{a}_{y1}^T \\ \vdots \\ \mathbf{a}_{xN}^T \\ \mathbf{a}_{yN}^T \end{pmatrix}. \quad (14)$$

Equation 13 can be solved using homogeneous linear least squares, described in the next section.

SVD was then applied to table A to solve the homogeneous system **Ah = 0**. **'H_appr'** (a vector of 9 elements) was found as the eigenvector corresponding to the smallest eigenvalue. The vector was reformatted to a 3x3 table and normalized, giving the final homography table.

```
[[-4.23580803e+00  6.81958908e+00 -5.45651869e+02]
 [ 2.50414811e+00  1.17769821e+01 -8.77321477e+03]
 [ 9.51551643e-04 -3.27648741e-02  1.00000000e+00]]
```

The results were almost identical to those of the OpenCV method, confirming the correctness of the manual implementation.

In addition, the coefficients of the reverse transformation were also calculated $\mathbf{H}^{-1}$ (H_inv):

```
[[ 4.11688831e+00 -1.65147050e-01  7.97517261e+02]
 [ 1.62066052e-01  5.55027574e-02  5.75369255e+02]
 [ 1.39264197e-03  1.97568681e-03  1.00000000e+00]]
```

The accuracy of the model was evaluated through the re-projection error vX, vY (v = X – X′) as well as the mean square error $\boldsymbol{\sigma_0}$. The **homogeneous** points of the image (**pix_coords**) were converted through the **H_appr** table into homogeneous coordinates of the real world and compared with the original, known coordinates (**obj_coords**).

For the calculation of the total mean square error, the formula `**sigma_0 = np.sqrt(np.sum(Errors[:, 0]**2 + Errors[:, 1]**2) / 13)**` with a denominator of 13 as it represents the degrees of freedom (2*11 points - 9 unknown parameters).

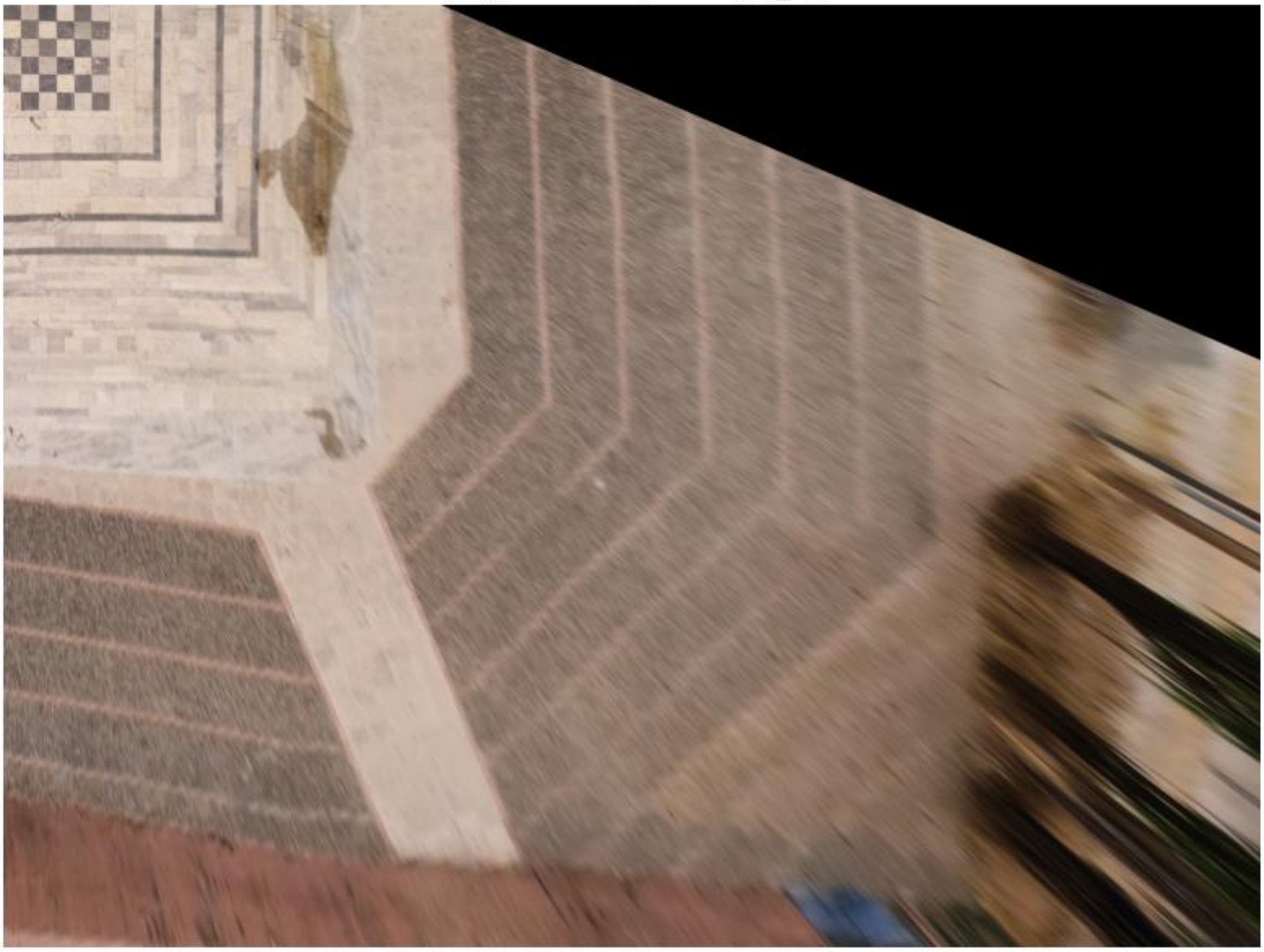
```
Errors of reprojection:
[[ 0.15763086 -0.44234962]
 [ 0.29835402 -1.62758789]
 [ 1.24245645  0.02832888]
 [-0.24181757  0.42521455]
 [-0.58948148  0.06357   ]
 [ 0.0687493  -0.34683229]
 [ 0.39347987 -0.55462641]
 [ 0.14263822 -0.71600704]
 [-1.86355336 -0.00868711]
 [ 0.50300029  2.23748441]
 [ 0.08223267  0.61571564]]

Mean Squared Error of reprojection: 1.0860117383159504
```

The final value of the error calculated was **sigma_0 = 1.0860**. This value is low and indicates that the model achieved a very accurate match between the two coordinate systems, with an average deviation of about one centimeter.

For further evaluation of the estimated **H_appr** table, the image was re-projected using the '**warpPerspective**' algorithm:

Reprojected Image using H_appr



And we notice that the visual results are remarkably similar to those we used exclusively the OpenCV algorithms to estimate the transformation table.