Computer Vision Project

Christian Altrichter, Jury Andrea D'Onofrio, Francesco Huber

June 13, 2023

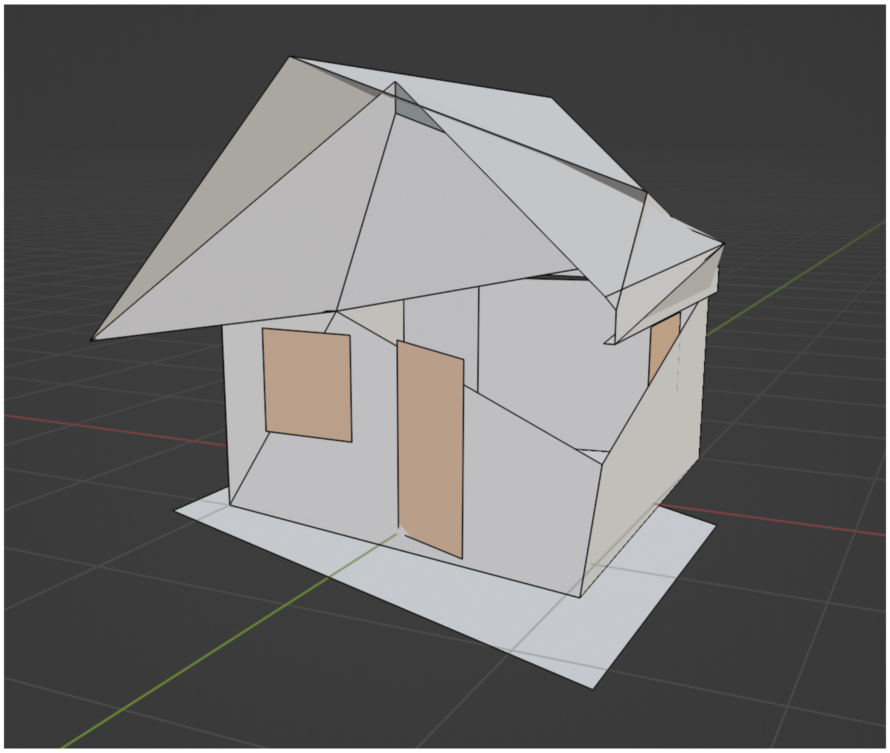


Figure 1: "Failure is another stepping stone to greatness." — Oprah Winfrey

1 Part 1

1.1 Task 1.1

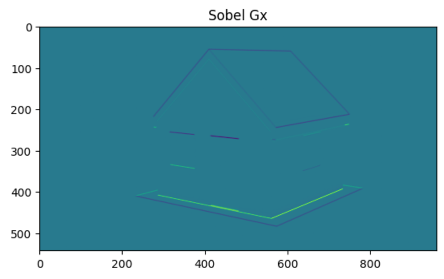
The first task was to identify two sets of parallel lines. To do this, we have taken two fundamental steps to identify and extract the lines:

1. Canny Edge detection.

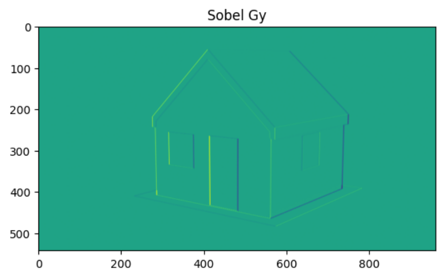
2. Hough transform.

As outlined in the course, Canny Edge detection consists of various subparts. As pre-processing, we initially considered using a Gaussian Filter. However, as the image is already computer generated, thus, the edges clear and sharp, any filter application would depreciate the quality of our edge detection. Thus, we opted against a pre-processing of the image.

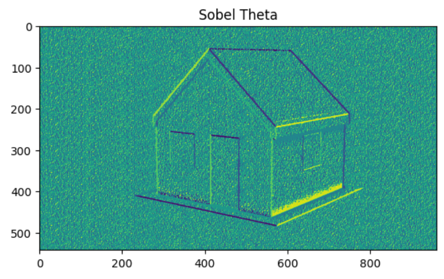
The first step of the Canny Edge Detection is to apply the Sobel Operator, which applies by convolution the derivative in x and y direction. Thus, we get the rate of change in the respective pixels. Per pixel, we then compute the gradient magnitude as well as the theta, which represents the direction of the gradient.



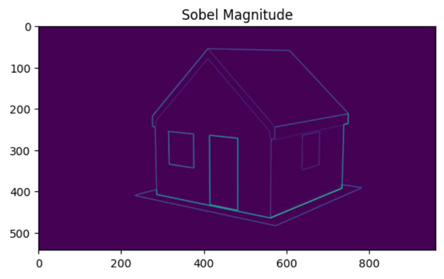
(a) Sobel - Gx



(b) Sobel - Gy



(c) Sobel - Theta



(d) Sobel - Magnitude

Figure 2: Sobel Operator and its individual components

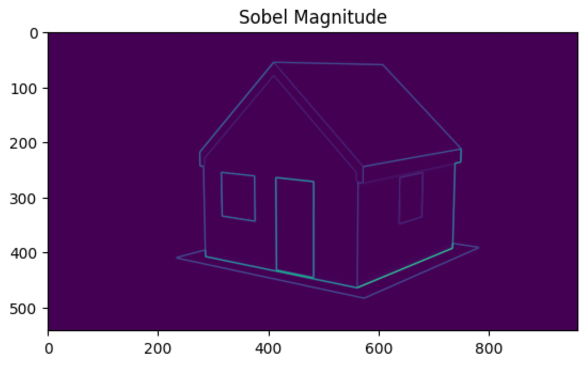
Based on the gradient magnitude and its direction, we inspected the surrounding pixels in the positive and negative direction of the gradient. For every pixel, if it was not the maximum value in its surrounding, we suppressed it by setting the pixel value to 0. Thus, ultimately thinning the edges only to its strongest values.
Subsequently, we applied the double thresholding where we differentiate and classify pixels / the detected edges into three classes based on two set thresholds:

1. Strong edge

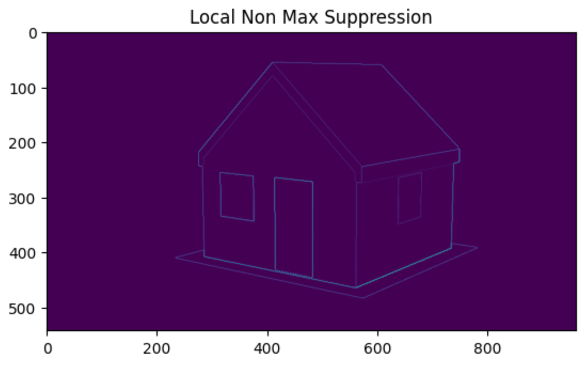
2. Weak edge

3. Non-edge

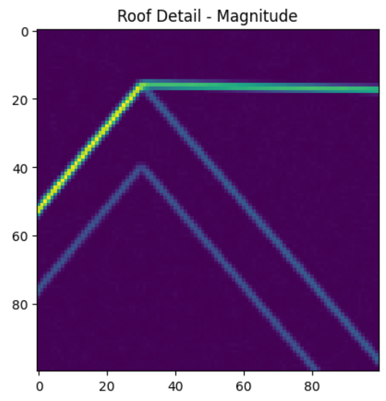
Strong and non-edges are not to be revisited after. With respect to weak edges, we have to inspect them further and detect whether they have a connection with a strong edge.



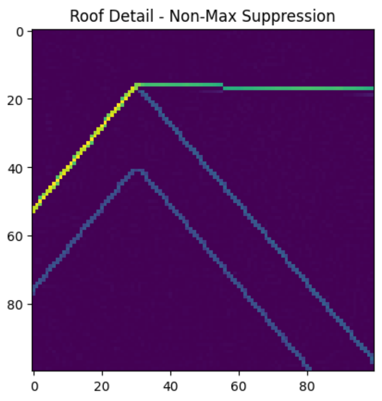
(a) Before non-maxima suppression



(b) After non-maxima suppression



(c) Detail of before non-maxima suppression



(d) Detail of after non-maxima suppression

Figure 3: Non-maxima suppression and its effects on our image

To do this, we used the connectivity analysis to connect weak edges to strong edges if there is such a connection, else to classify the weak edges as well as non-edges.

Then, we used the Hough-Transform to extract the actual lines in Rho and Theta form. For any given pixel with the value of a white pixel (thus, an existing edge), we looped through and applied the Hough transformation. The accumulator was then built based on the rhos and thetas. We chose to populate the array by votes rather than sum of magnitudes.

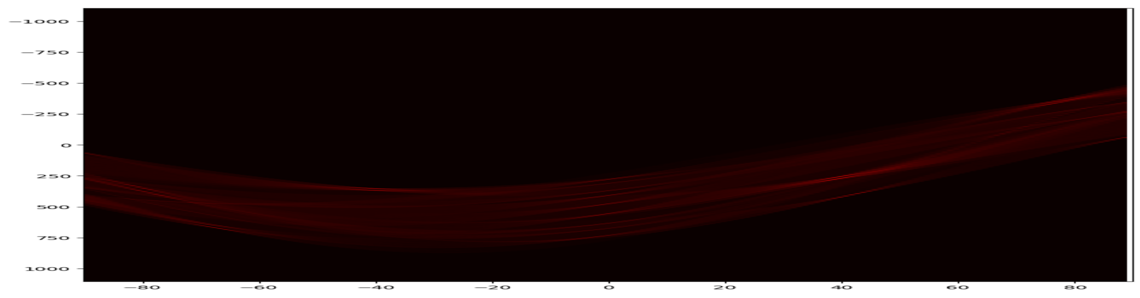


Figure 4: Results of the accumulator, Rho on the Y axis, Theta on the X axis

To then extract effectively the lines, we took the maximum value in the accumulator and extracted all rhos with their corresponding thetas for all elements under a certain percentage of said maximum value. Furthermore we also selected which lines to be shown based on a valid range of theta, easing therefore our process of discarding lines we did not want to select. After getting the intended subset of lines, we transformed them from a rho and theta representation back into a $y = mx + b$ formulation such that we could extract the Cartesian coordinates.

With a threshold of 0.4 of the maximum value in the accumulator, we retrieved the lines plotted in Figure 5.

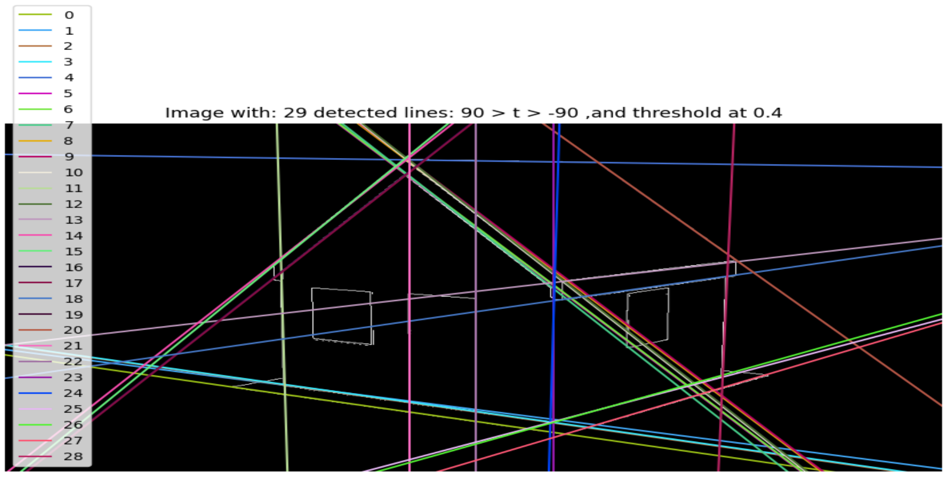


Figure 5: Detected lines based on threshold of 0.4

1.2 Task 1.2

We proceeded to manually pick a subset of lines that we knew were supposed to be parallel, but due to the distortion were not. When selecting the lines, we needed to ensure that we were taking lines from the same plane of the house. Furthermore, we paid attention such that the points at infinity are not the same for the two sets of lines.

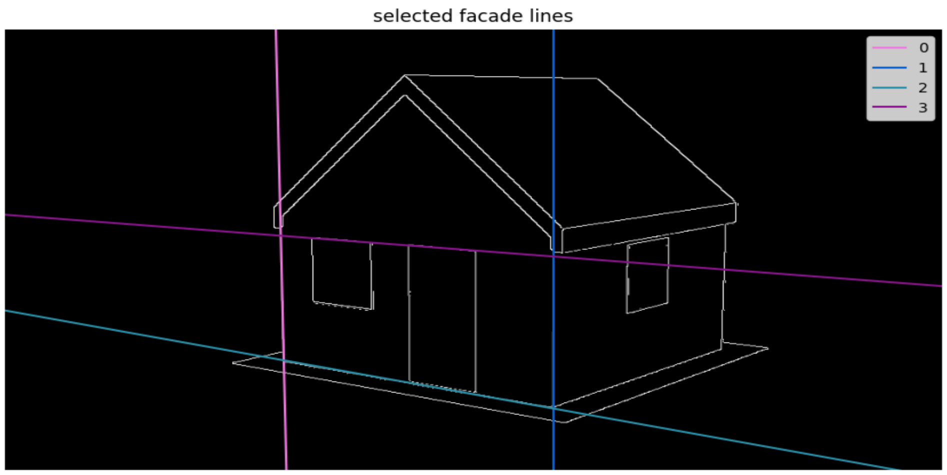


Figure 6: Selected lines to determine the vanishing line

We then intersected the lines by applying the cross product to the two horizontal and two vertical lines.

	X	Y	W
Horizontal line intersection	14238.0227	563	1
Vertical line intersection	140.5172	-957.0371	1

Table 1: Points at infinity

1.3 Task 1.3

Lastly, we have computed the infinity line by applying the cross product between the two determined points at infinity in task 1.2.

X	Y	W
-0.0001	0.001	1

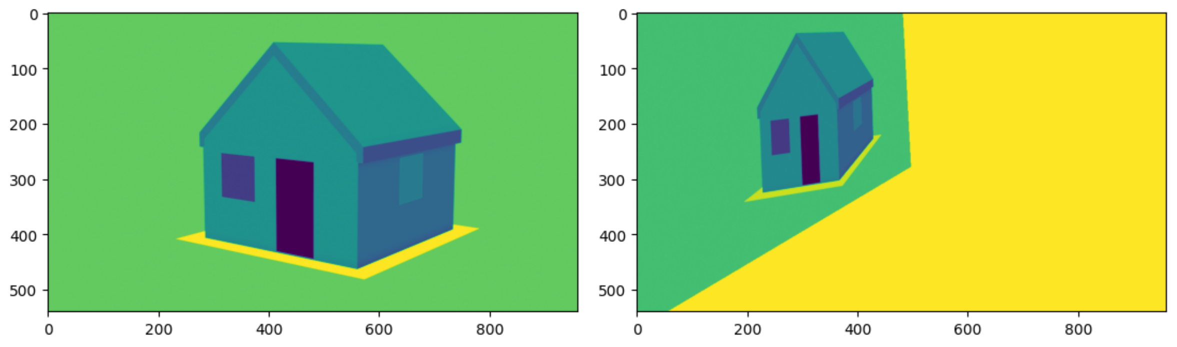
Table 2: Homogeneous coordinates of the vanishing line

Lastly, we have created the matrix H, where $l = (l_1, l_2, l_3)$ is the vanishing line. The last row of matrix H is then filled with homogeneous coordinates of the vanishing lines as follows:

$$\begin{bmatrix} 1. & 0. & 0. \\ 0. & 1. & 0. \\ -0.0001 & 0.001 & 1. \end{bmatrix}$$

1.4 Part 1. Results

Given the matrix H, we have then applied a bi-linear inverse mapping to perform the affine rectification. The results of the rectification can be seen in Figure 7.



(a) Before affine rectification

(b) After affine rectification

Figure 7: Results of the affine rectification

Since an affine rectification preserves the area ratios, we could then determine the size of the door.
Ultimately, by counting the pixels of the affine rectified image, we have found that the size of the door is approximately:

$$1.95 sqm$$

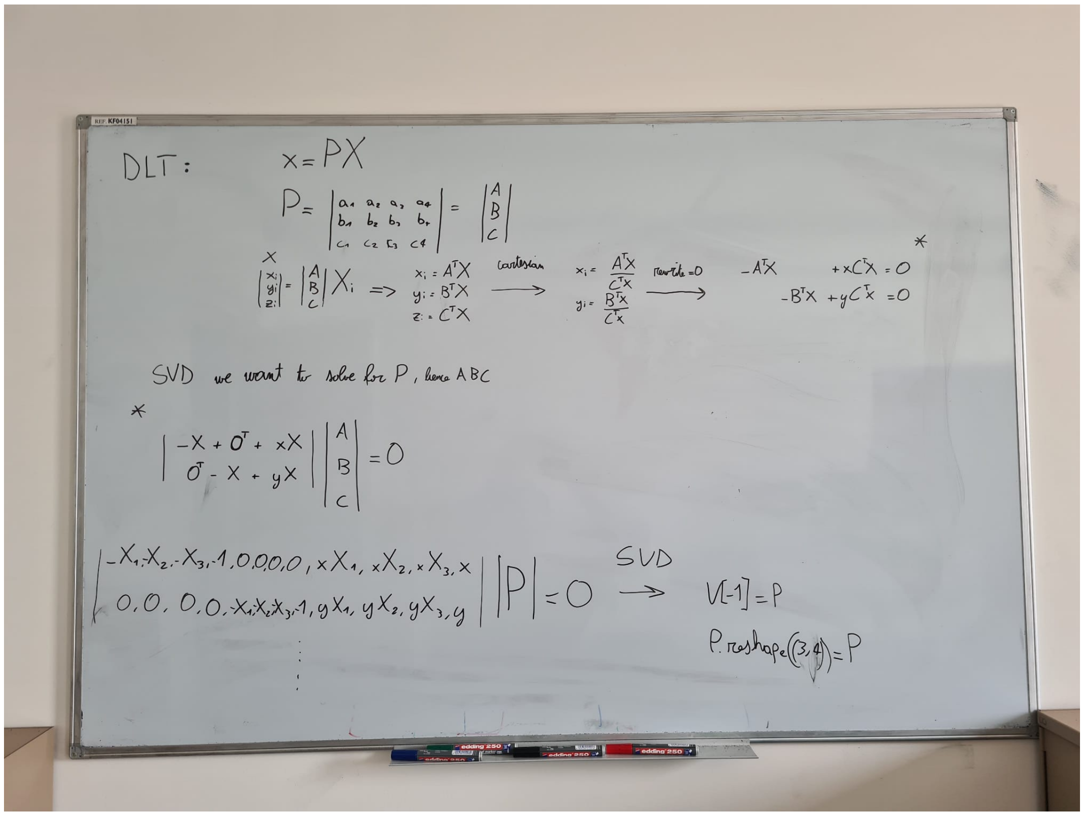
2 Part 2.

2.1 Task 2.1

We decided on manually reading the coordinates of the indicated points via Matplotlib as we previously demonstrated in task 1.2 that we are capable of getting any point on the image as a result of selecting the correct lines from the Hough-Transform and applying the cross product to obtain their intersection.

2.2 Task 2.2

We started the process of obtaining the projection matrix by deconstructing the equation x = PX in the manner indicated by Figure 8. This allowed us to create a series of linear equations corresponding to the 2D points for a total of 20 rows and 12 entries, 2 rows per point representing x and y. After applying a SVD decomposition on said matrix we have extracted the last column of V in order to obtain the solution to the linear problem and have reshaped it into a 3 by 4 matrix. The last passage gave us the projection matrix P.


REF. KF04151

DLT:

$$x = PX$$

$$P = \begin{vmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{vmatrix} = \begin{vmatrix} A \\ B \\ C \end{vmatrix}$$

$$\overset{x}{\begin{vmatrix} x_i \\ y_i \\ z_i \end{vmatrix}} = \begin{vmatrix} A \\ B \\ C \end{vmatrix} X_i \Rightarrow \begin{matrix} x_i = A^TX \\ y_i = B^TX \\ z_i = C^TX \end{matrix} \xrightarrow{\text{cartesian}} \begin{matrix} x_i = \frac{A^TX}{C^TX} \\ y_i = \frac{B^TX}{C^TX} \end{matrix} \xrightarrow{\text{rewrite} = 0} \begin{matrix} -A^TX \quad + xC^TX = 0 \\ -B^TX + yC^TX = 0 \end{matrix} \quad *$$

SVD we want to solve for P, hence A B C

$*$

$$\begin{vmatrix} -X + 0^T + xX \\ 0^T - X + yX \end{vmatrix} \begin{vmatrix} A \\ B \\ C \end{vmatrix} = 0$$

$$\begin{vmatrix} -X_1, -X_2, -X_3, -1, 0, 0, 0, 0, xX_1, xX_2, xX_3, x \\ 0, 0, 0, 0, -X_1, -X_2, -X_3, -1, yX_1, yX_2, yX_3, y \\ \vdots \end{vmatrix} \begin{vmatrix} P \end{vmatrix} = 0 \xrightarrow{\text{SVD}} V[-1] = P$$

P.reshape((3,4)) = P

Figure 8: Composition of the DLT procedure for the Projection Matrix

The results of $x = PX$ were then verified in a function called `compare_points`.

2.3 Task 2.3

This task was obtained by applying the RQ decomposition as shown by the class handout 17 at slide 60. We have corrected for any negative diagonal values in K by negating the respective column in K and row in R. C was extracted following the formula: $C = -M^{-1}@p4$ which was obtained by rearranging an existing equation also shown in the handout 17 at slide 60.

2.4 Part 2. Results

As requested please find below our results.

IMAGE 1

K1: $\begin{bmatrix} 1354.4565 & 6.1987 & 528.5405 \\ 0.0 & 1345.1017 & 272.8836 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$

R1: $\begin{bmatrix} 0.8196 & 0.573 & 0.0067 \\ 0.095 & -0.1243 & -0.9877 \\ -0.5651 & 0.8101 & -0.1563 \end{bmatrix}$

C1: $\begin{bmatrix} 4.7427 & -6.6825 & 2.1104 \end{bmatrix}$

IMAGE 2

K2: $\begin{bmatrix} 1381.8476 & 0.8099 & 497.1722 \\ 0.0 & 1376.6241 & 254.218 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$

R2: $\begin{bmatrix} 0.4229 & 0.9062 & 0.0005 \\ 0.0398 & -0.0181 & -0.999 \\ -0.9053 & 0.4225 & -0.0437 \end{bmatrix}$

C2: $\begin{bmatrix} 6.5535 & -3.2084 & 1.4354 \end{bmatrix}$

3 Part 3

3.1 Task 3.1

After getting the images coordinates x_i and x'_i, $i = 1,...,10$ from the second part of project in `house1.png` and `house2.png`, we used the 8-Point-Algorithm to reconstruct the fundamental matrix F relating to the two views.

Once again similarly to how we did for the projection matrix, we obtained the fundamental matrix F by deconstructing the equation $x'Fx = 0$ for every point x and x' like it was shown in the handout 21, slide 20. This allowed us to create a series of linear equations corresponding to the 2D points for a total of 10 rows and 9 entries.
After applying a SVD decomposition on said matrix we have extracted the last column of V in order to obtain the solution to the linear problem and have reshaped it into a 3 by 3 matrix.
Since we obtained a matrix with rank(3), we enforced rank(2) on the fundamental matrix.

Utilizing the matrix V and U from the SVD decomposition, we also computed the two epipolar points (e and e') by knowing that e is the right null space of F and e' is the left null space. Then, e and e' allowed us to compute the canonical pairs $\hat{P}$ and $\hat{P}'$.

We also conducted a verification of our results in a function called `estimate_error` which applied the formula $x'Fx = 0$, the results were not always close to zero but it is to be expected due to the imprecision of our manual selection of the points in the 2 images.

3.2 Task 3.2

In this section we were required to compose the canonical camera pair $\hat{P}$ and $\hat{P}'$, the latter was composed with the previously mentioned e'. Subsequently, we estimated the 3D coordinates by applying a linear triangulation through an adjusted DLT on a 2D point correspondence and the two camera pairs. In accordance to the equation shown in section 12.2 of the Book we applied the SVD on the constructed matrix from the linear triangulation. The last column of V represented the new estimated point that had to be normalized by its fourth component.

We collected new estimates for every point correspondence and stored them for later use.

3.3 Task 3.3

In order to obtain a 3D homography, we have followed our previously shown process of obtaining the equation for DLT by deconstructing the formula $x' = Hx$, as can be see in Figure 9.

However it should be noted that the ultimate formula in the image is not the correct one as we have made a mistake with the order of the row arguments, we have fixed it by moving the first 4 elements for all three rows at the end. This ensured that the row multiplication happened in a correct order such that the variables were computed accurately as well.

This process produced a matrix A of dimensions 30 by 16, with 3 rows per point as per x,y,z. Our matrix A was of rank 9, so in accordance to the handout 21 slide 10, we continued by applying SVD and taking the last column of V, we then reshaped it into a 4x4 matrix, obtaining the 3D homography matrix H.

To verify our results we then applied the formula $x' = Hx$ to ensure that any estimated 3D points would then be converted to its correspondence. This produced positive results with minimal variation from the actual given coordinates.

3.4 Task 3.4

We have applied the linear triangulation on the new given points that were once again manually picked using the correct canonical camera pair P and P' as follow: $\tilde{P} = PH^{-1}$. Our Y point estimates can be found in Figure 11.

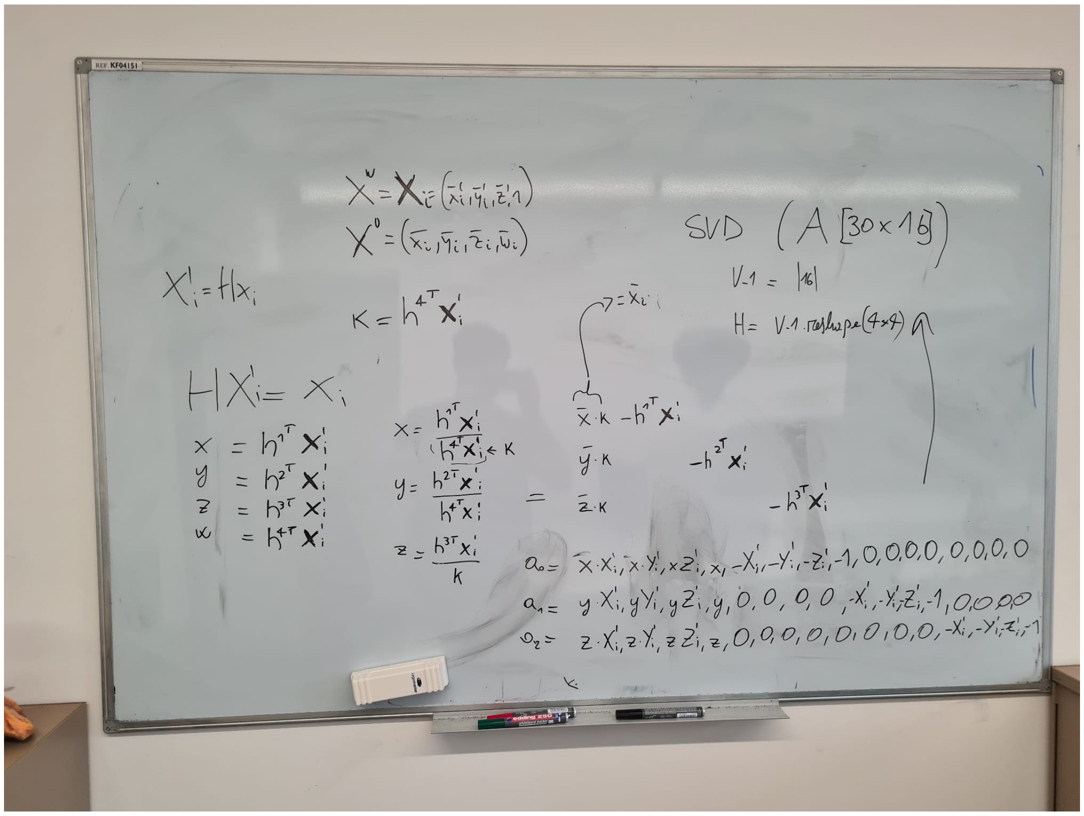


Figure 9: Composition of the DLT procedure for 3D Homography matrix

3.5 Part 3. Results

As discussed in section 3.1

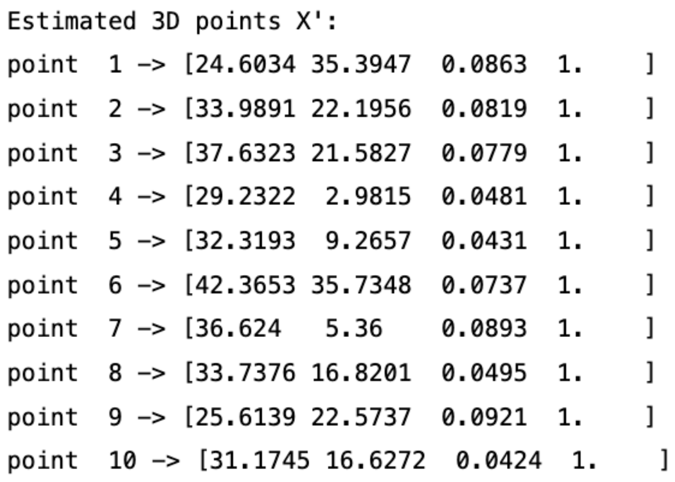
$$F: \begin{bmatrix} 0 & 0 & -0.0057 \\ 0 & 0 & 0.0372 \\ 0.0072 & -0.0344 & -0.9987 \end{bmatrix}$$

3.5.1 Canonical Results

$$\hat{P}: \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\hat{P}': \begin{bmatrix} -0.0011 & 0.0053 & 0.1525 & -0.9883 \\ 0.0072 & -0.034 & -0.987 & -0.1527 \\ 0 & 0 & -0.0376 & 0 \end{bmatrix}$$

Estimated 3D points X': Figure 10



```
Estimated 3D points X':
point  1 -> [24.6034 35.3947  0.0863  1.     ]
point  2 -> [33.9891 22.1956  0.0819  1.     ]
point  3 -> [37.6323 21.5827  0.0779  1.     ]
point  4 -> [29.2322  2.9815  0.0481  1.     ]
point  5 -> [32.3193  9.2657  0.0431  1.     ]
point  6 -> [42.3653 35.7348  0.0737  1.     ]
point  7 -> [36.624   5.36    0.0893  1.     ]
point  8 -> [33.7376 16.8201  0.0495  1.     ]
point  9 -> [25.6139 22.5737  0.0921  1.     ]
point  10 -> [31.1745 16.6272  0.0424  1.     ]
```

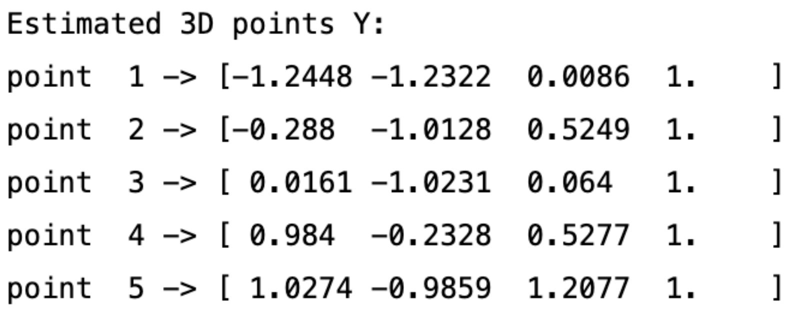
Figure 10: X estimation via Canonical Camera pair P_hat, P_hat'

3.5.2 Real Results

P: $\begin{bmatrix} -459.0328 & -665.4225 & 46.5488 & -2443.7306 \\ 11.9372 & -25.9871 & 773.0405 & -1899.1124 \\ 0.3101 & -0.4363 & 0.0931 & -4.7263 \end{bmatrix}$

P': $\begin{bmatrix} 2.6012 & 28.5849 & -0.5011 & 74.9312 \\ -3.6931 & 0.8858 & -26.7752 & 64.6347 \\ -0.0176 & 0.0083 & -0.001 & 0.143 \end{bmatrix}$

Estimated 3D points Y: Figure 11



```
Estimated 3D points Y:
point  1 -> [-1.2448 -1.2322  0.0086  1.     ]
point  2 -> [-0.288  -1.0128  0.5249  1.     ]
point  3 -> [ 0.0161 -1.0231  0.064   1.     ]
point  4 -> [ 0.984  -0.2328  0.5277  1.     ]
point  5 -> [ 1.0274 -0.9859  1.2077  1.     ]
```

Figure 11: Y points estimation via Camera pair P,P'