École Centrale de Nantes

CORO - Advanced Robotics

AVG Lab Report

Submitted by
Sinagaram R
Gowri UMESH

January 4, 2021

Contents

List of Figures

1. Projective Geometry

Projective transformations is any perspective projection of a plane. In the process of imaging unlike Euclidean geometry not all measures are preserved and imaging process of a camera will fall under the projective geometry.In fact Euclidean geometry is a subset of the projective geometry in which distance , orthogonality and parallelism is invariant. It is important to reconstruct the image which has invariant measures for various computer vision application such as localisation and mapping.The process is called as stratified reconstruction.
Equation 1.2 shows the mapping of 3D world co-ordinates X into the 2D image co-ordinates x via projective matrix P.

$$x = PX \tag{1.1}$$

Introducing an intermediate transform H in the above equation which maps the same World points into image points. The goal is to find matrix H that preserves measures such as parallelism, orthogonality.

$$x = PH^{-1}HX \tag{1.2}$$

Affine Rectification: Affine rectification refers to preserving parallelism in the image.Line at infinity stays invariant in both affine group and PL(3) group and hence can be used to affine recovery of the images.
Metric Rectification Metric rectification refers to preserving orthogonality in the image. Similar to affine transform an invariant has to be chosen for the rectification and geometric quantity that remains invariant in the metric rectification group are circular points I and J .

$$I = \begin{bmatrix}1 & i & 0\end{bmatrix}^T, J = \begin{bmatrix}1 & -i & 0\end{bmatrix}^T \tag{1.3}$$

These circular points are complex and reality there we cannot use a complex point from an image and hence dual conics are used which are built by a pair of orthogonal lines.

1.1 Lab Work and Results

Both Affine and Metric rectification is implemented in a single python script *Lab1_ Transformation.py*. The Affine rectification involves the following steps:

1. Choose 4 points on image which are parallel in the world frame. This is ideally selected as the corners of the square tile in Figure 1.1.

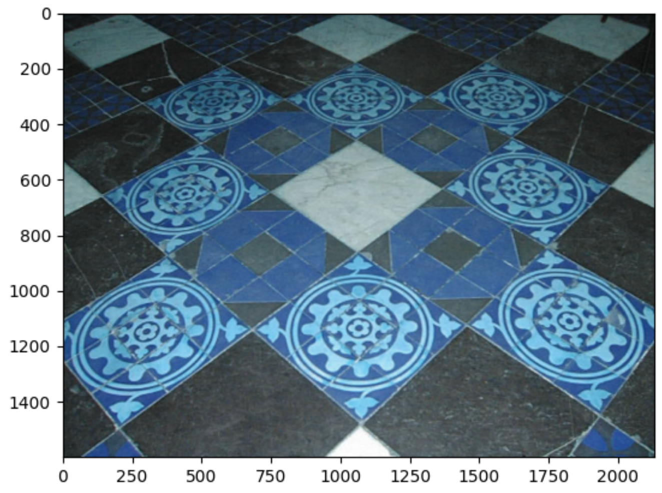


Figure 1.1: Input Perspective Image

2. Obtain 4 parallel lines from the above points (2 horizontal and 2 vertical) and there by obtaining a vanishing points.
3. Identify image of the line at infinity on projective plane (line at infinity $l_\infty = \begin{bmatrix} l_1 & l_2 & l_3 \end{bmatrix}^T$)
4. Construct the projectivity that affinely rectify image.

$$H_A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix} \tag{1.4}$$

5. Once H_A is computed it is used to projectively wrap the image.

The above results in an image that preserves parallelism and is shown in the Figure 1.2. The image of line at infinity is obtained by a pair of vanishing points.
Note:A line joining 2 points is simply the cross product between them and the point of intersection of 2 lines is also the cross product between them. The Metric rectification is implemented in following steps:

1. Transform 4 chosen points from projective image to affine image.
2. Construct constraint matrix C.constraint matrix is built using a pair of orthogonal lines l' and m' obtained from step 1. (In our case we chose one diagonal line and a parallel line from the affinely rectified image).
   C = $(l'_1 m'_1, l'_1 m'_2 + l'_2 m'_1, l'_2 m'_2)$
3. Find s by looking for the kernel of C and thereby building S, $S = \begin{bmatrix} s_1 & s_2 \\ s_2 & s_3 \end{bmatrix}$
4. Build the matrix of the imaged Dual Conic:

$${C'_\infty}^* = \begin{bmatrix} S & 0 \\ 0^T & 0 \end{bmatrix} \tag{1.5}$$

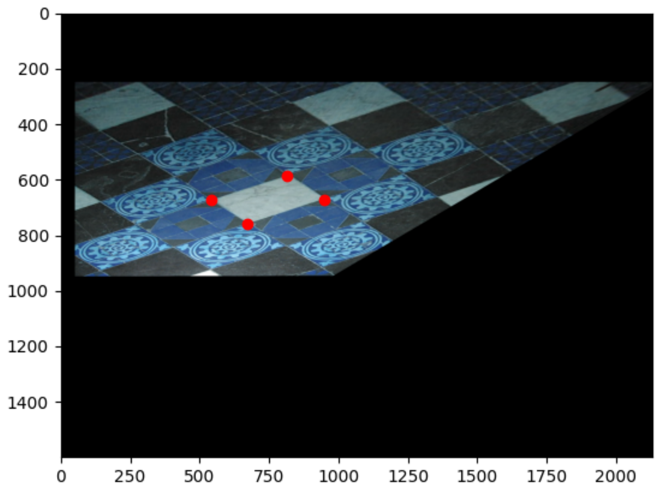


Figure 1.2: Image obtained after Affine Rectification

5. Find the projectivity H_E that do metric rectification

$$H_E \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} H_E^T \tag{1.6}$$

6. Once H_E is computed it is used to projectively wrap the image.

Figure 1.3 shows the result of metrics reconstruction

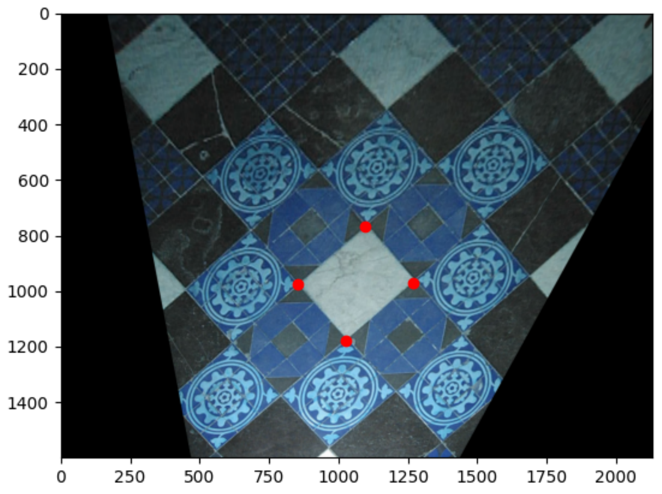
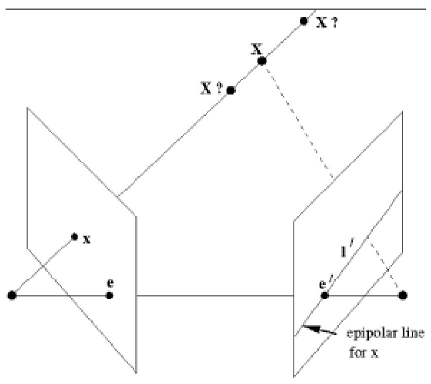


Figure 1.3: Image obtained after Metric Rectification

2. Two-View Geometry:Stereo vision

Fundamental matrix (F) and Epipolar geometry
The fundamental matrix is a 3x3 matrix that links corresponding points in stereo images. In epipolar geometry, the epipolar line of a point x on one image, lies on the other image, in which the corresponding point x' must lie. The epipolar line is the projection of all the points having the same image pale coordinates but varying depths, in the 2nd image plane.
The relation is described by line l' = Fx. Since x' lies on l', x'l' = 0 i.e. x'Fx = 0



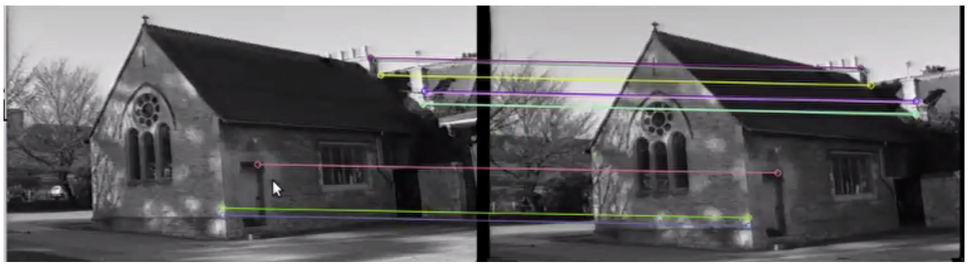
2.1 Lab Work and Results

Task 1: Visualizing epipolar geometry given F

In this lab, we have 2 images of a house with a roof. On the left image, we choose a point marked bv the red cross, using the ginput function: This point forms an epipolar line on the image on the right. The goal is to find this epipolar line using l' = Fx, where he matrix F is already provided in this file from chapel001.01.F. . This epipolar line l' for x is then drawn using plot() from pyplot. The epipoles e and e' represent the intersection of the lines linking the optical centers with the 2 image planes. They can be calculated using SVD from Fe = 0 FTe' = 0

Where the last column of the matrix V can be taken to find e or e' respectively. The epipole too lies on the epipolar line and lies at infinity if the two image planes are parallel. Suppose we have a vector l' we can draw a line by finding intersection between epipolar line and the two vertical edges to get two points and connecting them.

Task 2: Estimating F using matched Keypoints
In this exercise there is no longer access to Fundamental matrix but many matched points between the two images from which we should estimate F.



Using the ORB keypoints and descriptors we can find a number of matches between the two images and from matched points we can create a linear homogeneous equation of the elements of fundamental matrix F. F is 3x3 and in this linear equation, it flattens into a 9x1 vector.

Each row of A is calculated using the pixel coordinates of corresponding points, so a correspondent will help – having atleast 8 pairs of points is necessary to find the fundamental matrix using SVD with a scale factor as F has 9 unknowns. It is enough to solve with 8 DOF with a scale factor as F satisfies a homogeneous linear equation x'Fx = 0. A set of linear equations is stacked up, one equation for each correspondence.

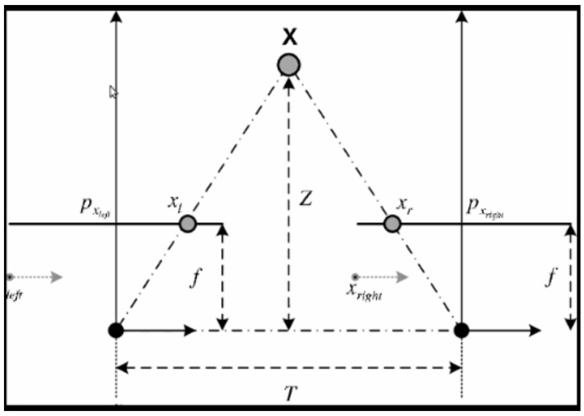
Keypoint normalization needs to be performed before estimating F. Normalization is a similarity transformation of keypoints so that keypoints have zero mean and an RMS from origin of sqrt 2 . If the center of mass coincides with origin, we find a scale transform and the RMS of keupoint to origin is sqrt2.

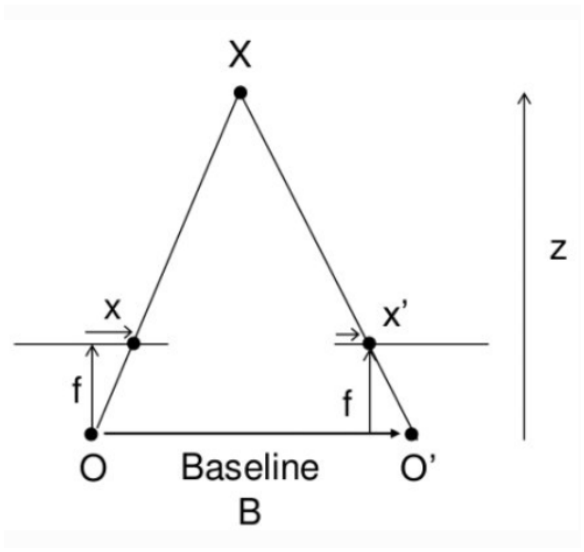
Disparity map and 3D reconstruction

This exercise aims to reconstruct a point cloud from 2 images captured by a stereo rig. The rig is shown below.

A stereo rig is a pair of cameras placed such that they have the same orientation and their x and optical axes are parallel to each other. The right camera is distance T translated from left camera usually on the baseline of stereo rig.

Due to this spatial configuration of 2 cameras we don't need the fundamental matrix to calculate the link between two corresponding xL and xR. The relation between xL and xR is

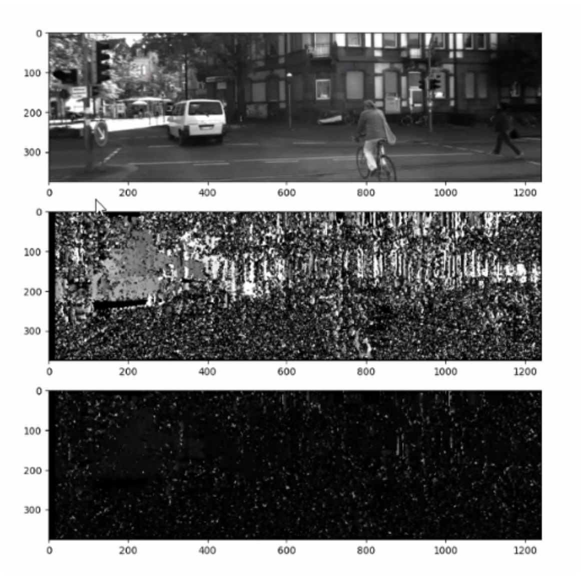


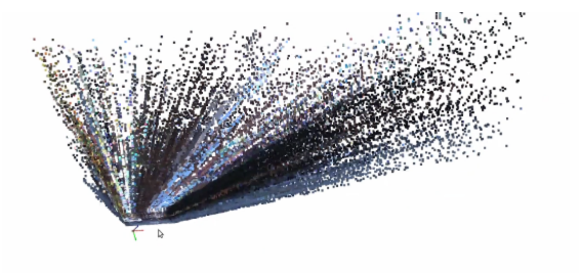


$$disparity = x - x' = \frac{Bf}{Z} \tag{2.1}$$

From this equation, we can calculate F if we know the correspondent and when you have F we can reconstruct the 3D coordinate of the point.

We find disparity for every pixel on image on left and this is the disparity map for every pixel on left image. From the disparity map, we can calculate the depth for every pixel on image on left and as the stereorig's heatmap is noisy, we cannot see clear features. From the depth of every pixel, we can reconstruct the 3D coordinates using the normalized coordinates of keypoints – we take the camera intrinsic matrix K and multiply it with the pixel coordinate in homogeneous form to obtain normalized coordinates. From this we go back to 3D coordinates by multiplying with f. After this, we obtain point cloud and inject color to see some texture on point cloud.





With this stereorig the 3D Environment can be reconstructed.

3. Homography based visual odometry

Visual Odometry is a vision based localization method that estimates ego vehicle's pose using series of images. The idea behind Homography based VO is to compute a homography H from two images as shown in Figure 3.1 that contains the camera pose.This matrix H maps a pixel in first frame to its corresponding pixel in second frame.
The process of homography based VO follows the below steps

1. Detect keypoints and compute their descriptors from the incoming frame
2. Find matches between keypoints in the incoming frame with those in the source frame (e.g. image capture when camera's pose is C0)
3. Using RANSAC, compute homography H and also get inliners (matched keypoints that agree with H)

4. Decompose H to get (R; t; n) candidates (usually 4).
5. Prune candidates that give a negative depth for at least 1 inlier.
6. There still may be two candidates, keep the one that has the normal n closer to the current estimation of the plane's normal vector.

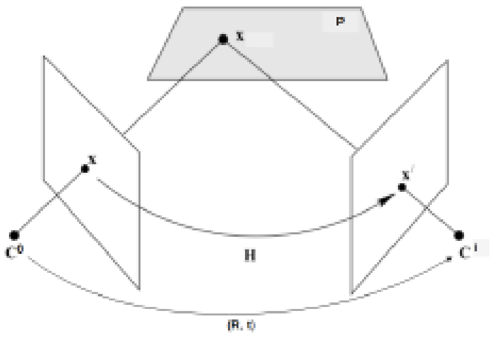


Figure 3.1: Homography induced by a pane

The detailed algorithm is explained in the Lab document.

3.1 Lab Work and Results

The implementation of the lab is done in two python scrips *main_ vo.py* and *visual_ odometry.py*.The former script invokes the latter in which *VisualOdometry* class is defined. This class implements the algorithm explained in the above section and uses openCV , orb feature detector and Flann based matcher is used in its implementation. After executing of the

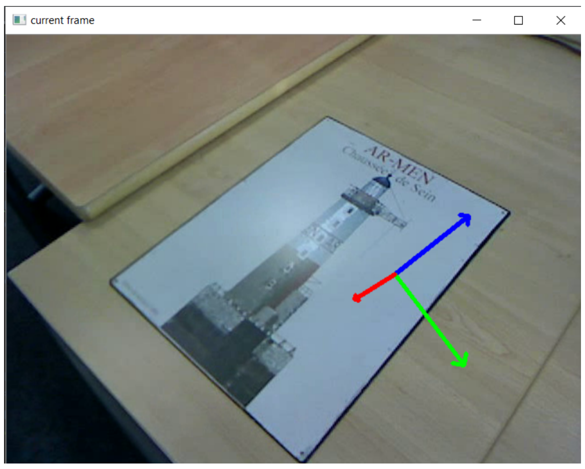


Figure 3.2: Image frame from the resultant VO video

high level steps mentioned below:

- Read the input video file
- Invoke the *run* method of class *VisualOdometry* to compute the relative transformation.

- Map the plane's local frame from C^0 to C^{inc} and project it onto the image of C^{inc}

The implementation results is analysed by drawing the plane's local frame onto every image frame. Figure 3.2 shows one such frame.

4. Conclusion

The objective of the lab was to conduct practical sessions related to Projective Geometry , Two-view geometry and Homography based Visual odometry which was completed successfully. All the results and observations were noted.This report documents the theory lab work and results of the mentioned subjects.