Report

Sensing, perception and actuation

Homework assignment №2

Student: Oleg Rodionov

Case 04. Variant 18

Innopolis. 2019

GitHub Link

You can download the all stuff from this stuff from this link:

https://github.com/rodosha98/SPA-Homework2

ATTENTION: to launch code in Tasks 02 and 03 correctly in Matlab you should change the path to the images then you read it.

Task 01. Kalman Filter

In this task the given dataset contains measurements of centre of mass during the walking. We need to estimate proper trajectory for CoN movement using Kalman Filter.

First of all, we need to pre-process the data to prepare it for using and plot it to see a change in position from time to time. After this, we're ready to build a model.

So, the equations have the following form:

$$z_k = x_k + \eta_k$$
$$x_{k+1} = x_k + \xi_k$$

In this case we don't know information how out object has been moved(initial velocity and acceleration and their functions of time). In this difficult case we might put all unknown information from the physical model to the random variable ξk.

The array z_k represents our measurements from Kinect sensors. We can estimate the mean and variance of this distribution. The value of ξk is unknown but can be assigned to get better results.(We can assume that the mean is equal to 0 and assign σ_ξ. The second equation represents the model.

The goal is to find an optimal estimation x opt of true coordinate x_k, knowing a non precise sensor's data z_k. The following equation:

$$x_{k+1}^{opt} = K * z_{k+1} + (1 - K) * x_k^{opt}$$

K - is the the Kalman gain, which represents the weights of prediction part and measured part

We're trying to minimize the error on each step, which has this formula:

$$e_{k+1} = x_{k+1} - x_{k+1}^{opt}$$

It can be rewritten in this form:

$$e_{k+1} = (1 - K) * (e_k + \xi_k) - K * \eta_{k+1}$$

And the expectation of squared error have this form:

$$E(e_{k+1}^2) = (1 - K)^2 * (E_k^2 + \sigma_\xi^2) + K^2 * \sigma_\eta^2$$

$\xi \; and \; \eta$ are independent random variables. So, the expectation of their product is equal to 0.

Now we are ready to apply Kalman algorithm. The idea of Kalman is to get the best estimation of the real coordinate x_{k+1} between the reading of non precise sensor z_{k+1} and model prediction x^{opt}. It's iterative method. The number of iterations corresponds number of points in dataset (n = 6500). On each step we will update optimal x, optimal error and Kalman gain K. Initial values x^{opt} is equal to the first measurement, e^{opt} is equal to σ_η^2.

The expressions for updating:

$$e_{k+1}^{opt} = \sqrt{\frac{\sigma_\eta^2 * (e_{optk}^2 + \sigma_\xi^2)}{\sigma_\eta^2 + e_{optk}^2 + \sigma_\xi^2}}$$

$$x_{k+1}^{opt} = K_{k+1} * z_{k+1} + (1 - K_{k+1} + * x_k^{opt}$$

$$K_{k+1} = \frac{e_{optk+1}^2}{\sigma_\eta^2}$$

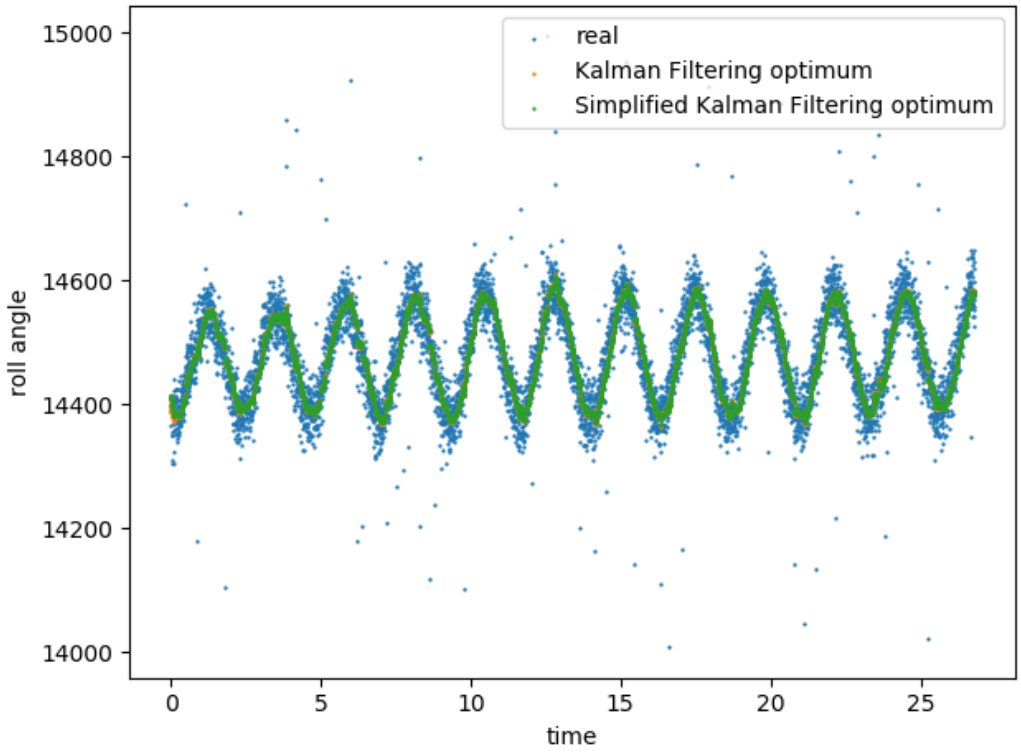
The results of this algorithm is represented by the plot on the figure 1.

We see that with increasing of the step k the Kalman coefficient always stabilizes to the certain value Kstab.

Figure 1 - Kalman filtering results

Because of it, we can assume that this coefficient is constant and implement simplified Kalman filter algorithm. The result is also on the figure 1. The trajectories, obtained with this 2 methods are similar. It's happens, because K stabilise very fast.

In repository on GitHub you can find the implementation of this code in Python.



Task 02. Camera Calibration

Calibration gives confidence in the measurement results you're getting. For camera it's also allows to remove distortions and find the object's dimensions and the distance to it. To solve this problem I used Computer vision and Image processing toolbox in Matlab. The goal is to find the inartistic and extrinsic parameters on camera on my MacBook laptop. I used it because it hasn't autofocus, but the quality of photos is not so good.

I made 40 photos of chessboard, because during the calibration will be needed to drop some photos.

The input information to Matlab is photos and the size of square on the chessboard, in my case it's 33 mm. For each image toolbox defines the chessboard and add there the image coordinate frame on the top left corner. It means, that now we know the coordinates of each point on the chess plane. After that we can use the formulas to define intrinsic and extrinsic parameters.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \left[R | T \right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

Because of there too many photos, computer can define the extrinsic parameters for each case and intrinsic in general.

Intrinsic matrix K contains information about Focal length, optical center and the skew coefficient.

$$K = \begin{pmatrix} f_x & 0 & 0 \\ s & f_y & 0 \\ c_x & c_y & 1 \end{pmatrix}$$

Because of fact, that our camera isn' t the ideal pinhole, we have to take into account tangential and radial lens distortion.

Main steps of the algorithm:

1. Images reading

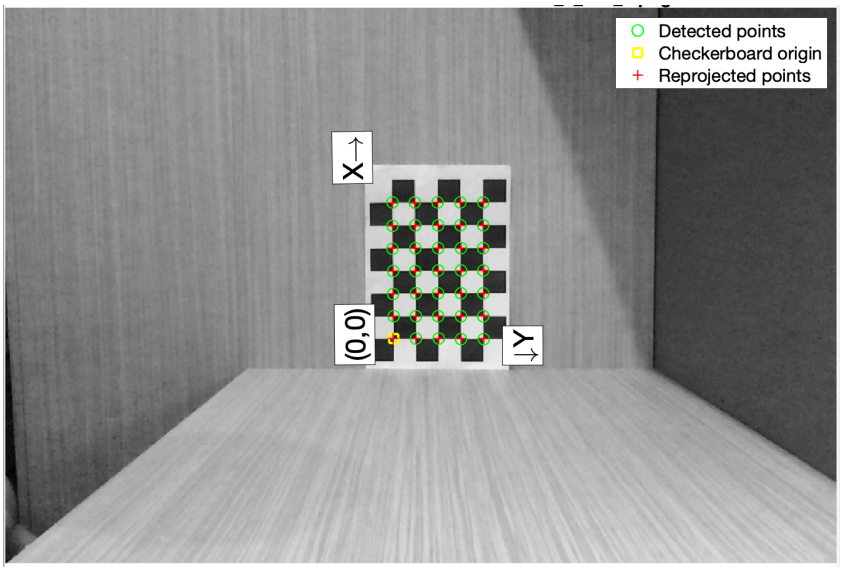


Figure 2 - Camera Calibration example

2. Chessboard detecting on each image

3. Computer obtains image's size and generate world frame, knowing the size of square
4. Calibrate the camera and show the results.

Results of calibration:

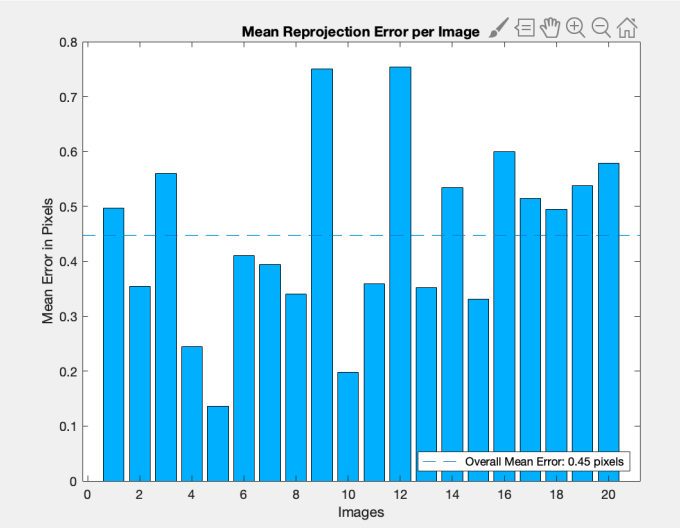


Figure 3 - Mean error in pixel after recalibration

As I mentioned before, I had to drop some photos to reduce the error from 3 pixels to one.

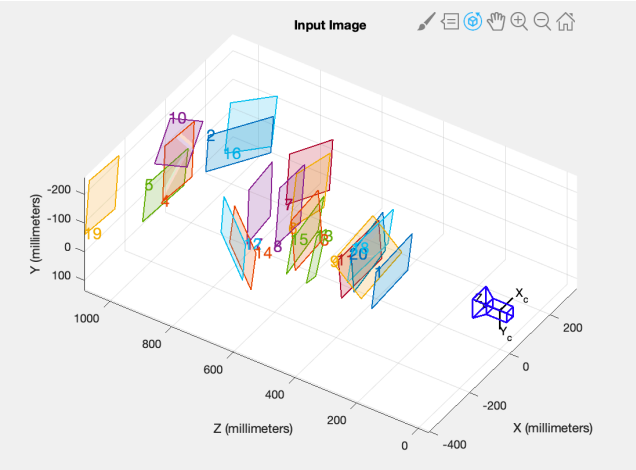


Figure 4 - Extrinsic parameters

Object definition

As an object, I used a perfume box, which has a height of 130 mm and a width of 95 mm. After the calibration, we can define the size of object and the distance to it from the camera.

The main steps:

1. Take a photo of the object with chessboard, so that the plane of the board coincides with the front side of the object. It's needed to compute the extrinsic parameters to go over from image frame to camera frame. The image plane will be located in the top left corner of the board.

Figure 5 - Box with the chessboard

2. Read the image and remove distortion(Because we've already known the intrinsic parameters of camera.

3. Object detecting. You can apply many different methods to detect the image, I've chosen the HSV conversion and rectangle detecting. For this, I've surrounded the object with black sheets, which will be transformed in black, and my object, which has the dark blue color, after converting will be white.

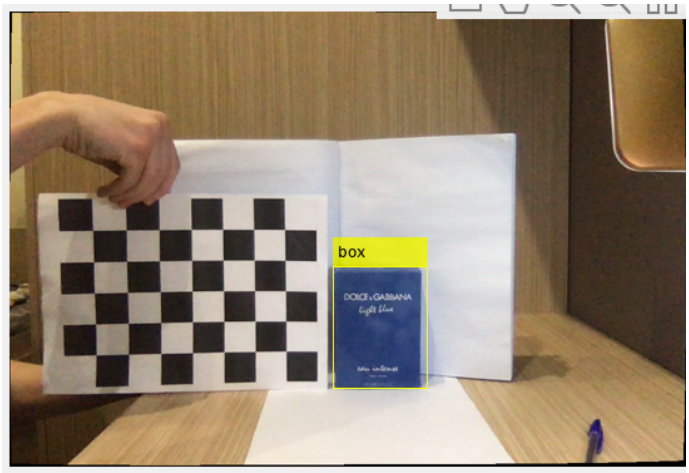


Figure 6 - HSV conversion of the image

Now it's easy to detect it.

4. Chessboard detection. Now we know rotation and translation matrices to get the image plane. After this we define the origin of undistorted image.

5. Define the rectangle's left top and right down points and convert it to the world frame. Now we can estimate the height and the width of the box.

The values of obtained width and height: w = 101 mm, h = 139 mm. The error is about 5-6 mm. I think this is connected with quality of the camera, not precise photo with chessboard and method of detection, and also calibration errors.

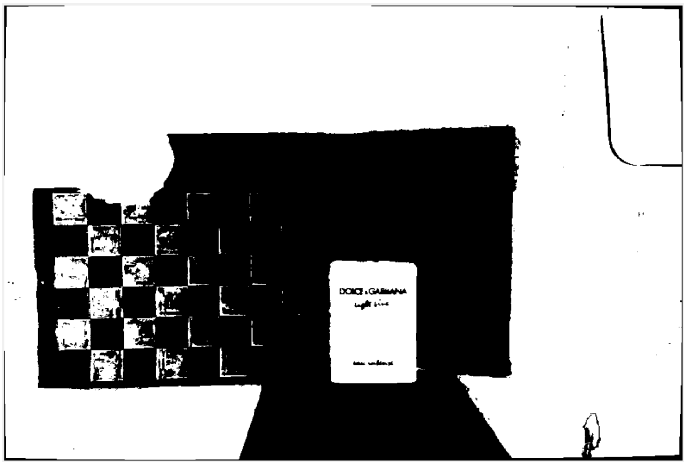


Figure 7 - Object detection

6. Define the distance to the object. It's equal to the distance to the image plane (chessboard plane). We need to convert the box centre point to the world frame and estimate the norm between that point and the camera. The value is equal to 670 mm.

Task 03. Stereo vision

Case 18.The provided dataset contains left and right photos of the street with the car from 2 cameras. Firstly, it's needed to find the fundamental matrix. Some intrinsic parameters of the camera are known (Focal length = 2.8 mm) and baseline is also known.

The idea of the stereo vision is simple - we just use 2 cameras except 1 and take a photo of some object from 2 cameras. After that we can estimate the size of this object and the distance to it (depth).

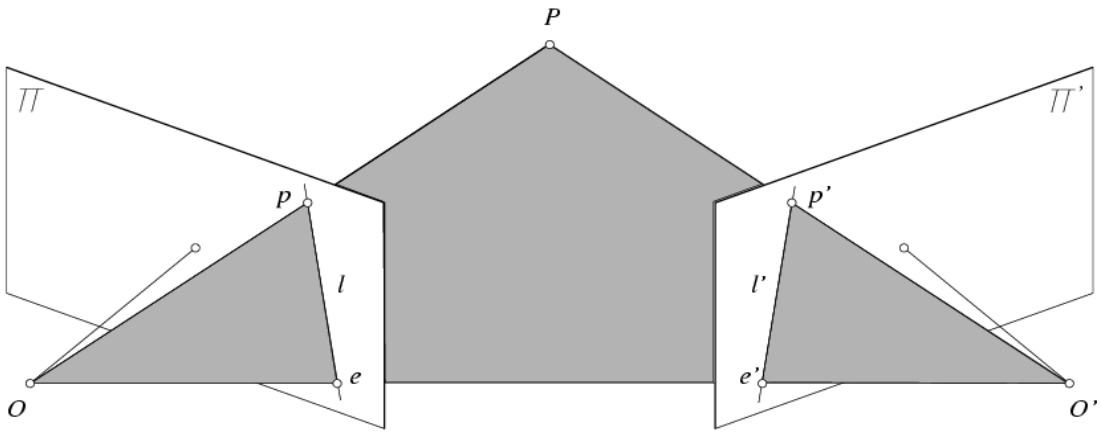


Figure 8 - Stereo vision scheme

I've implemented algorithm in Matlab. To find F we need to find corresponding points on both pictures. So the main steps are:

1. Image reading
2. I used Speeded-up robust features algorithm (SURF) to determine key points. For this it's needed to convert a picture in grey scale.
3. Surf Implementation (On the picture there are 10 points for example). After this we extract features(key points) into arrays.

Figure 9 - SURF algorithm implementation

4. Next Step is to match two arrays with points to find corresponding one.

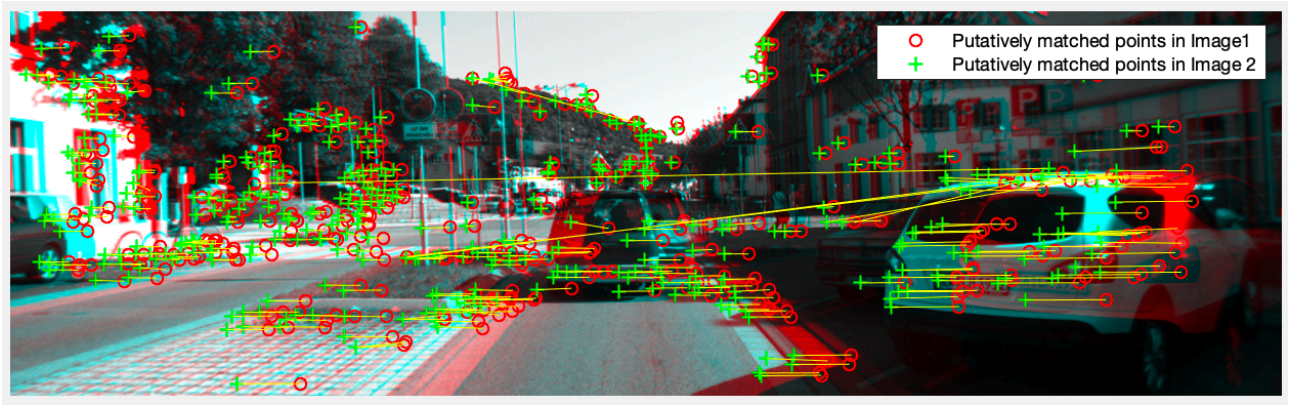


Figure 10 - matched points on overlay photos

5. And after that we create arrays of corresponding points xl, yl, xr, yr. One point has 2 different pairs of image coordinates on the left and right pictures.

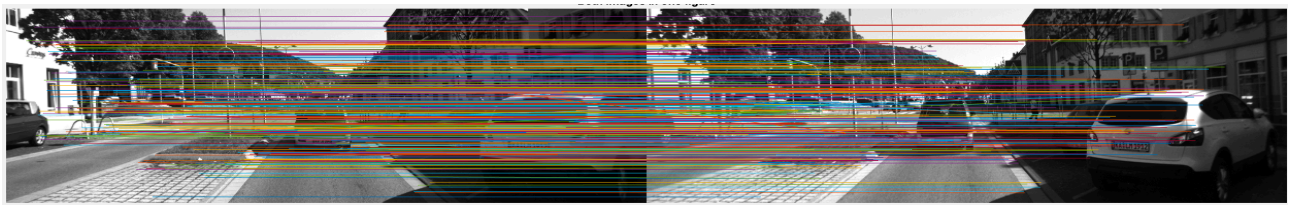


Figure 11 - matched points on both photos separately

6. Implement the 8-points algorithm to find Fundamental matrix F or Essential matrix E. The idea is to solve a system of homogeneous linear equations. Write down the system of equations and solve f from equation Af=0 using SVD. det(F) = 0.

$$F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -0.006 \\ -0.0032 & 0.0054 & 0.0754 \end{pmatrix}$$

7. Draw Epipolar lines . Knowing F, we can determine the equations of epilines and draw it on the left and right images.

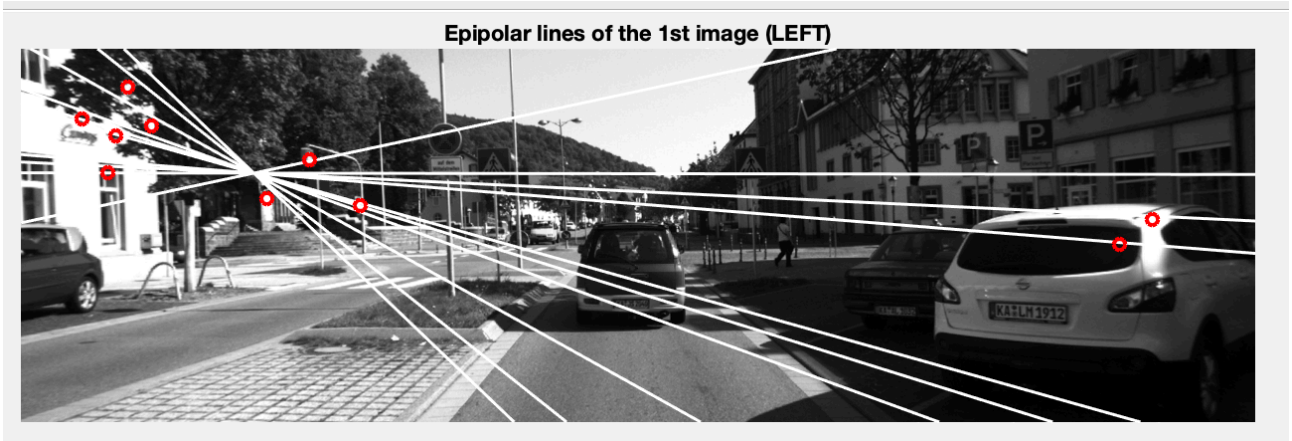


Figure 12 - Epipolar lines for the first image

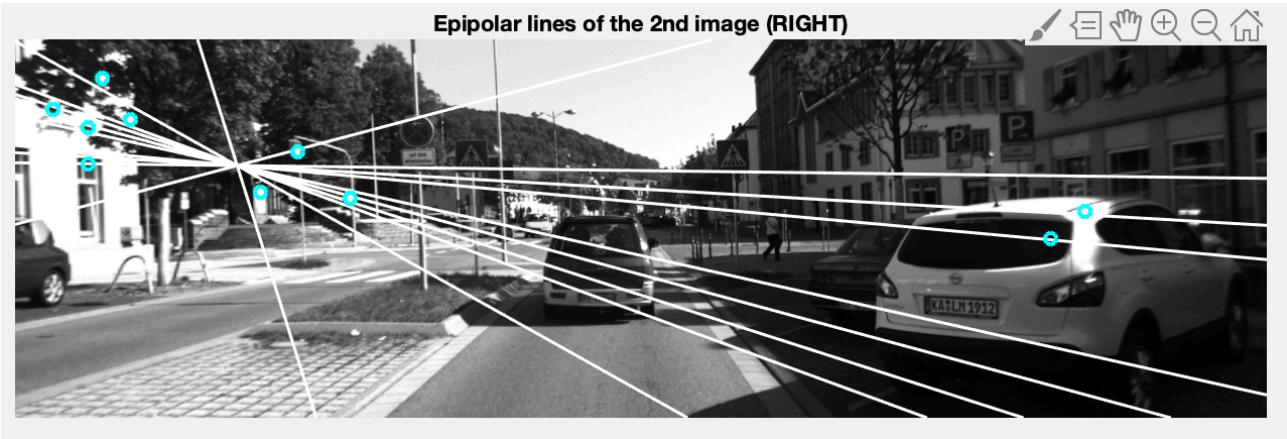


Figure 13 - Epipolar lines for the second image

8. Next step is to find The Disparity Map.

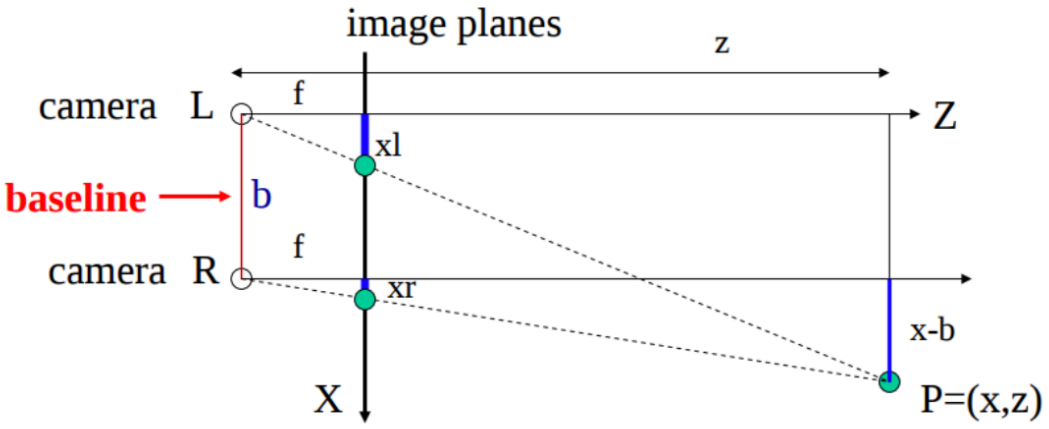


Figure 14 - Disparity and depth definitions

The disparity can be estimated as: d = xl - xr. The communication between them is governed by the fundamental matrix F. So we can use different methods to obtain rectified stereo pair image and disparity map.

Figure 15 - Rectified images

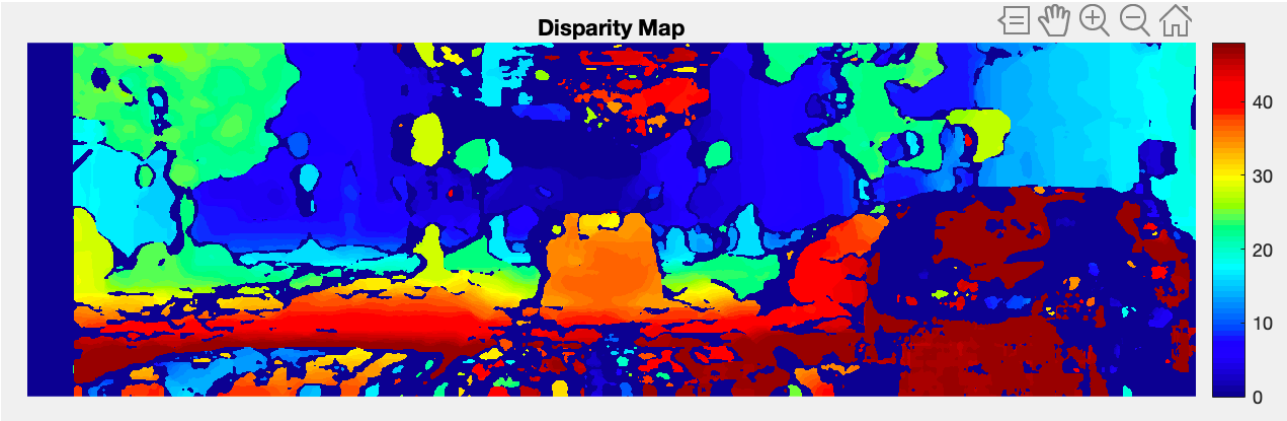


Figure 16 - Disparity map

So with disparity map we can estimate the depth (the distance) to any point.

$$disparity = x - x' = \frac{B \cdot f}{z}$$

Where B is the baseline (10cm), f is the focal length and z is a depth.