Aerial Robotics Kharagpur, Task 2.2

Venkatesh Naresh

***Abstract*— Task 2.2 deals with an important aspect for autonomous drones: depth perception. The first part of the task involves generating a plot showing variation of depth, called a depth map, from two given images of the same scene. Also given the camera information, actual depth information is also to be obtained. The task involves concepts of ray optics, linear algebra, coordinate systems and the understanding of image formation. Python libraries are used to complete the task. This task has wide scope of applications. While this task deals with finding depth from images, it can also be used for real-time depth perception by finding depth continuously from video frames. Real-time depth perception is a major part of today's autonomous robotics industry.**

I. INTRODUCTION

The objective is to create a depth map using two given stereo images, each shot by a left camera and a right camera respectively. Then, using the camera projection matrices for both left and right cameras, distance of an object, present in the image (a bike) from the cameras is to be evaluated. The first task has been achieved by evaluating disparity of a point i.e., the difference in coordinates in one direction, in both images, and then finding a relationship between disparity of a point and distance of the corresponding point from the camera. Later, this information is used to plot a depth map for all points, by associating distance of each point with a colour. The second task was relatively difficult, as it required the theoretical understanding of a camera projection matrix and existence of 3 coordinates systems when mapping a point in world frame to its corresponding image in image frame. A camera projection matrix is a product of intrinsic matrix (Also called the camera calibration matrix) and an extrinsic matrix. While the extrinsic matrix is useful to map points from coordinates relative to world frame to coordinates relative to camera frame, the intrinsic matrix is used to map points from the camera frame, to the image frame. Given reference image of the bike, we search for the bike in both the images using template matching, and obtain the image coordinates of the bike. Having found both intrinsic and extrinsic matrices, in left and right cameras, we use linear algebra to relate disparities in image frame with disparities in camera frame, which ultimately yields the value of distance of the bike from the camera.

II. PROBLEM STATEMENT

The first part of the task is to create a **depth map** given two stereo images. A depth map is an image which represents distance of each point on the image with respect to the camera. **Stereo imaging** (also called as stereo scoping) is a method to create an effect of a 3D environment from at least two 2D images. Stereoscopy is implemented by the human brain; the eyes capture two images of the same scene, which are processed by the brain to give a perception of 3D depth. Stereoscopy works on the principle of Ray Optics involving lenses.Consider two convex lenses,having same focal length **f**,separated by a baseline distance **B**,creating an image of the same point **X**, which is away from the lens plane by a distance **Z**. If the respective separations of the image from the principal axis is **x** and **x'**,then we have

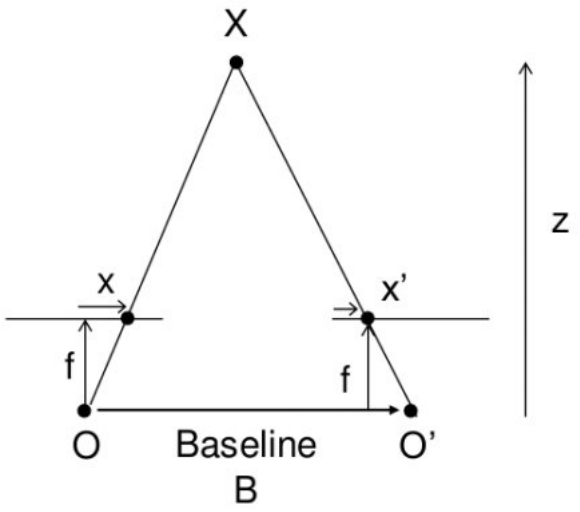


Fig. 1. Source [1]

$$disparity = x - x' = Bf/Z \tag{1}$$

The above formula states that the disparity of a point in the two images is **inversely** proportional to the distance of the point from the lens. Hence, disparity serves as an indicator of the distance of the point. Smaller the disparity between two images, larger is the distance of the point from the camera. Using disparity information of each point, the depth map of the two stereo images is to be generated. The distance is represented by brightness of particular colours, brighter the colour, closer is the point.

For the second part of the task, simply information of disparity is not enough. As we can see from equation (1), parameters like baseline and focal length are also required. It is also possible that the lens may not be in the same plane, or they may be oriented differently in space. To consider all these factors, we define the **camera projection matrix**. It is a 3 x 4 matrix which contains information of all **intrinsic**

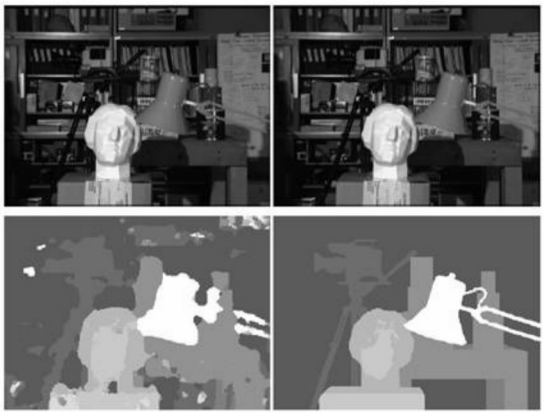


Fig. 2. Figure 2: Example of stereo images along with depth map.Source [2]

parameters(focal lengths, offset parameters and skew between axes) as well as the **extrinsic parameters**(orientation and position of the camera with respect to the world). The projection matrix **P** can be further decomposed into two matrices:

$$P = K_{3x3}[R|t]_{3x4} \tag{2}$$

Where K is the intrinsic matrix and [R—t] is the extrinsic matrix. Both K and [R—t] satisfy the following equations:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \\ 1 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [R|t] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{4}$$

where u,v represent image coordinates of a point.X_c, Y_c, Z_c represent coordinates of the actual point, in reference frame centred at the optical centre of the camera lens.X_w, Y_w, Z_w represent the coordinates of the same point relative to the world Hence, for the second part of the task, to find distance of the bike from the camera, we need to calculate Z_c.

III. Related Work

IV. INITIAL ATTEMPTS

For generating depth map, python script was written, having imported OpenCV-contrib-python library. The images were kept in the same directory as the script, and both images were read into the script using imread() function. Then, StereoBM.create() function was used to set parameters for the depth map. Finally, stereo.compute() function was invoked to generate depth map for the two stereo images.

For the second part, firstly template matching process was used.Template matching uses a reference image of the bike, called template, and searches the region in the larger image for the template. Once a match is found, a rectangle was drawn around the match, to identify it. The coordinates of midpoint of the rectangle are recorded which is later used to calculate distance of the corresponding point from camera. For this, the projection matrix was directly used to convert pixel coordinates to world coordinates(which was actually not the required quantity)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

V. Final Approach

The stereo parameters were changed to get more accurate depth map. The minimum number of disparities given by $NumDisparities$ was reduced from 16 to 0. Matplotlib library was used to generate the plot. Generating the depth map was relatively easier than finding the actual distance of the bike.

In template matching, the rectangle around the image was removed, recording only the midpoints. This was done for both left and right images.

Having understood that we require camera coordinates instead of world coordinates(as initially suggested above),a system of equations were developed to solve for distance of bike. Let u_L, v_L be the coordinates of the midpoint of the bike in the left image(henceforth referred to as coordinate of the bike) and u_R, v_R be the coordinates of the midpoint of the bike in the right image. Let X_L, Y_L, Z_L be the spatial coordinates of the bike in the left camera coordinate system andX_R, Y_R, Z_R be the spatial coordinates of the bike in the right camera coordinate system. If the world coordinates of the bike are X_W, Y_W, Z_W , we have

$$\begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} = K_L \begin{bmatrix} X_L/Z_L \\ Y_L/Z_L \\ 1 \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = K_R \begin{bmatrix} X_R/Z_R \\ Y_R/Z_R \\ 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} = [R_L|t_L] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{7}$$

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = [R_R|t_R] \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} \tag{8}$$

Here K represents the intrinsic matrix(appropriate subscript for left and right cameras). A property of K is that it is an upper triangular matrix. [R—t] is the extrinsic matrix(appropriate subscript for left and right cameras) where R is the 3x3 rotation matrix, and t is the origin of the world coordinate system, in camera coordinates,**after** rotation

Given that,

$$P_L = K_L[R_L|t_L] \tag{9}$$

and

$$P_R = K_R[R_R|t_R] \tag{10}$$

we define P_s as the 3x3 leftmost sub-matrix of P. Then, we have

$$P_{s,L} = K_L R_L \tag{11}$$

and

$$P_{s,R} = K_R R_R \tag{12}$$

We define, C_L and C_R as column matrices denoting the 4^{th} columns of P_L and P_R respectively, then we have

$$t_L = K_L^{-1} C_L \tag{13}$$

and

$$t_R = K_R^{-1} C_R \tag{14}$$

From equations (7) and (8), we can write

$$\begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} = R_L \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + t_L \tag{15}$$

$$\begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = R_R \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + t_R \tag{16}$$

Multiplying equations (15) and (16) with R_L^{-1} and R_R^{-1} respectively and subtracting, we get

$$R_L^{-1} \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} - R_R^{-1} \begin{bmatrix} X_R \\ Y_R \\ Z_R \end{bmatrix} = R_L^{-1} t_L - R_R^{-1} t_R$$

Which can be re-written as

$$Z_L R_L^{-1} \begin{bmatrix} X_L/Z_L \\ Y_L/Z_L \\ 1 \end{bmatrix} - Z_R R_R^{-1} \begin{bmatrix} X_R/Z_R \\ Y_R/Z_R \\ 1 \end{bmatrix} = R_L^{-1} t_L - R_R^{-1} t_R \tag{17}$$

Substituting equations (5) and (6),we get

$$Z_L R_L^{-1} K_L^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} - Z_R R_R^{-1} K_R^{-1} \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = R_L^{-1} t_L - R_R^{-1} t_R \tag{18}$$

using (11) and (12), we have

$$P_{s,L}^{-1} = R_L^{-1} K_L^{-1}$$

$$P_{s,R}^{-1} = R_R^{-1} K_R^{-1}$$

And, along with substituting (13) and (14), equation (18) becomes

$$Z_L P_{s,L}^{-1} \begin{bmatrix} u_L \\ v_L \\ 1 \end{bmatrix} - Z_R P_{s,R}^{-1} \begin{bmatrix} u_R \\ v_R \\ 1 \end{bmatrix} = P_{s,L}^{-1} C_L - P_{s,R}^{-1} C_R \tag{19}$$

This implies, that K and R need not be evaluated to find Z_L and Z_R. The image coordinates of the bike and the projection matrices themselves are sufficient to find the distance of the bike.

On simplification, if the equation looks like

$$Z_L \begin{bmatrix} m_L \\ n_L \\ o_L \end{bmatrix} - Z_R \begin{bmatrix} m_R \\ n_R \\ o_R \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Then Z_L and Z_R are solutions of the system of linear equations

$$\begin{bmatrix} m_L & -m_R & 0 \\ n_L & -n_R & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Z_L \\ Z_R \\ b_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

In this way, distances from the left camera as well as the right camera can be calculated.

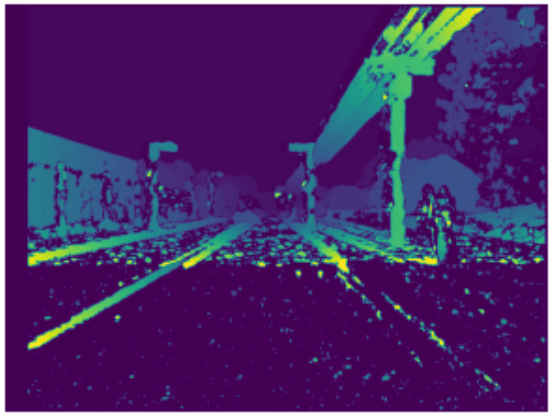
VI. Results and Observation

For the given stereo images

Fig. 3. Image from Left camera

Fig. 4. Image from Right camera

Depth map generated is



A brighter green colour indicates the closer points, while darker blue colour indicates the farther points. According to

provided projection matrix data, we have,

$$P_{s,L} = P_{s,R} = P_s = \begin{bmatrix} 640 & 0 & 640 \\ 0 & 480 & 480 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_L = \begin{bmatrix} 2176 \\ 552 \\ 1.4 \end{bmatrix} C_R = \begin{bmatrix} 2176 \\ 792 \\ 1.4 \end{bmatrix}$$

Then, from equation (19), we have

$$\begin{bmatrix} Z_L u_L - Z_R u_R \\ Z_L v_L - Z_R v_R \\ Z_L - Z_R \end{bmatrix} = \begin{bmatrix} 0 \\ -240 \\ 0 \end{bmatrix}$$

So, $Z_L = Z_R = Z$

A very important observation here, is the **choice of the coordinate system**. We assume that the Z axis of the camera frame is outward from the camera.Hence,Z is positive.From the above equation we also get $u_L = u_R$ and $v_L < v_R$.

In digital images, by default, the origin is taken at the **top left corner** of the image, with **x-axis horizontally towards right**, and **y-axis vertically down**

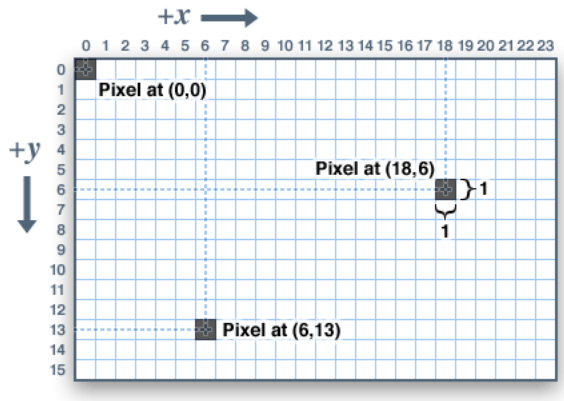


Fig. 5. source: https://towardsdatascience.com/Image Geometric Transformation In Numpy and OpenCV

Consequently, the midpoint coordinates I obtained,following the above convention were $(u_L, v_L) = (562, 502)$ and $(u_R, v_R) = (551, 502)$. But this pair of solutions doesn't satisfy the above relations, resulting in a contradiction. To ensure that the above relations were true, I considered the coordinate system at **top right corner** of the image, with **x-axis vertically down**, and **y-axis horizontally leftward**,implying that Z-axis will be outward from the camera. The new coordinates which I obtained were $(u_L, v_L) = (494, 710)$ and $(u_R, v_R) = (494, 721)$. Now the above relations get satisfied

Finally we get

$$Z = \frac{-240}{v_L - v_R} = \frac{-240}{-11} = 21.818\, metres$$

The value of Z coming positive **verifies** our assumption of the coordinate system. Hence, the bike is approximately at a distance of 21.818 metres from the camera

VII. Future Work

There are multiple parameters associated with StereoBM, which require tuning, to get better and smoother depth map, like texture threshold,speckle range and size,uniqueness ration, filtering [1]. Additionally other algorithms like StereoSGBM can also be used for developing depth maps. Template matching too offers other methods, based on how the disparities are normalised. Further, we can scan the rectangular region enclosing the bike, and find coordinates of the point with maximum disparity, this will offer a slightly lower value than the calculated distance, this will enable the bike to come out faster from the minimum safe distance threshold, and then, the receiver of the camera data can act accordingly.

Conclusion

The task was based on an important part of autonomous perception: computer vision. Generating a depth map gives a representation of how far points are from the camera. Later, camera projection matrices were used to find distance of an object from the camera. Various python libraries like OpenCV-contrib-Python, NumPy and Matplotlib were used. While Matplotlib was used to generate the depth map, NumPy was used for linear algebra calculations and OpenCV-contrib-Python gave the necessary functions regarding template matching and processing images. Main difficulties faced in this task were understanding the theory behind camera projection matrix, and the choice of the coordinate system. Overall, this task has a wide scope, and has several applications like self-driving cars, aerial survey of an agricultural plot, underwater exploration and so on.

References

[1] OpenCV Official Website. Opencv: Depth map from stereo images.
[2] Manuel Dominguez-Morales, Angel Jiménez-Fernandez, Rafael Paz-Vicente, Alejandro Linares-Barranco, and Gabriel Jimenez. *Stereo Matching: From the Basis to Neuromorphic Engineering*. 07 2012.