CSE-483: Mobile Robotics **Monsoon 2019**

Stereoscopic Vision

Prepared by: Shovan. S, Devansh. G

In this module we discuss how a 3-D rendition of a scene can be done by using 2 steresoscopic cameras. The main idea is to have 2 cameras placed a small distance apart. We take 2 images, 1 from each camera and by looking at similiar points on both these images, we can estimate the depth of the point and thus create a approximate 3-Dimensional rendition of the scene.

1 Setup

As shown below, this is how a general stereoscopic setup is done. It includes two cameras kept a distance b apart. The focal length of these cameras is f. We see that v_L and v_R depict the point P's camera co-ordinates in the respective camera planes.

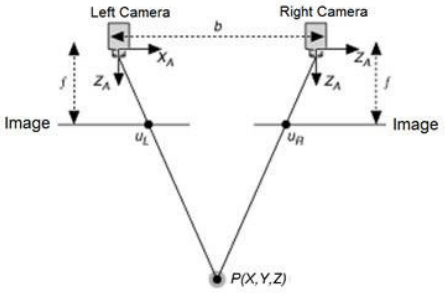

Figure 1: A typical steroscopic setup

In the next section, we will see how we can use these different camera co-ordinates of the same point to approximate its depth.

2 Disparity

Disparity measures the distance between similar points on the left and right camera images. Using this distance, we can estimate it's depth in the real world.

We take the origin to be at O_1. Using the concept of similar triangles, we see that :

$$\frac{f}{Z_P} = \frac{x_1}{X_P} = \frac{-x_2}{b - X_P} \tag{1}$$

Solving these 2 equations and eliminating X_P, we get :

$$Depth = Z_P = \frac{bf}{x_1 - x_2} \tag{2}$$

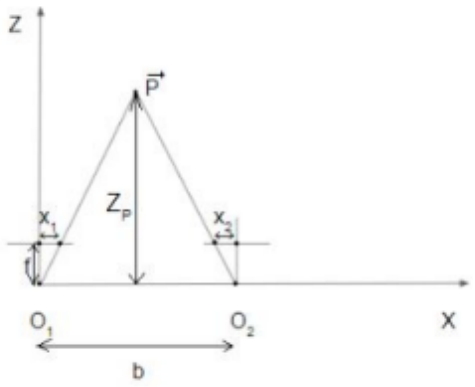

Figure 2: Disparity

So we see that, given the camera distance (b), focal length (f), and disparity ($x_1 - x_2$), we can calculate the depth of a point in 3-D space.

2.1 General Case

Referring to the earlier diagram, we take Origin at O_1. Then the Projection Matrix for Camera 1 $P_1 = K_1(I|0)$ And the Projection Matrix for Camera 2 $P_2 = K_2(R|T)$

Assuming the rays formed by from the cameras to x_1 and x_2 respectively intersect at P,

$$\lambda_1 x_{1h} = P_1 \vec{P_h} \tag{3}$$

$$\lambda_2 x_{2h} = P_2 \vec{P_h} \tag{4}$$

Since cross product of parellel lines is zero, we get:

$$[x_{1h}]_\times P_1 \vec{P_h} = [x_{2h}]_\times P_2 \vec{P_h} = 0 \tag{5}$$

where $[x_{1h}]_\times = \begin{bmatrix} 0 & -1 & y_1 \\ 1 & 0 & -x_1 \\ -y_1 & x_1 & 0 \end{bmatrix}$

In equation 1, both the equations contribute 2 equations each. So we have 4 equations and 3 unknowns for each pixel pair. Then we can solve for P_h using SVD, and taking the vector corrresponding to the smallest eigen value. Now, disparity for each point can be calculated using, $d_i = \frac{bf}{Z_P i}$

If the rays dont intersect, non-linear approach like LM algorithm can be used to minimize reprojection error.

$$loss = min((x_1 - \pi_1(\vec{P}))^2 + (x_2 - \pi_2(\vec{P}))^2) \tag{6}$$

Result of this linear approach can be used as initialization for the non-linear approach

3 Rectification

Image rectification is a transformation process used to project two-or-more images onto a common image plane. This process has several degrees of freedom and there are many strategies for transforming images to the common plane.

It is used in computer stereo vision to simplify the problem of finding matching points between images (i.e. the correspondence problem).

Even in commercial stereo cameras the left and right images are never perfectly aligned. In practice, it is convenient if image scanlines are the epipolar lines. Stereo rectification warps the left and right images into new rectified images, whose epipolar lines are aligned to the baseline.

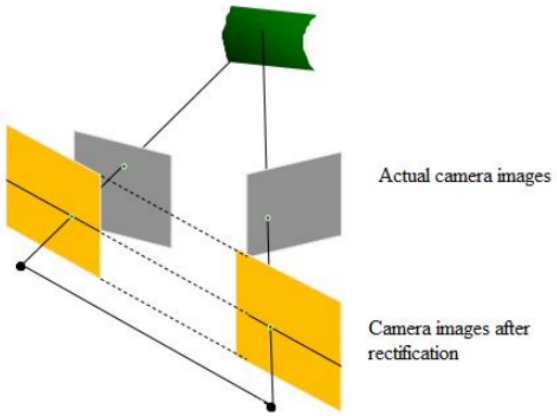

Figure 3: Image Rectification

Given a pair of stereo images, image rectification is a transformation that makes epipolar lines collinear and parallel to x-axis. Image Rectification is widely used in stereo vision because it speeds up matching because searching horizontal epipolar lines is easier than general epipolar lines.

In this case, we don't keep origin at one of the cameras. So,

$$\text{Projection Matrix of Camera 1 } P_1 = K_1 R_1^T [I| - T_1]$$
$$\text{Projection Matrix of Camera 2 } P_2 = K_2 R_2^T [I| - T_2]$$

We need a new $\hat{R}$ and $\hat{K}$ such that :

$$\hat{P_1} = \hat{K}\hat{R}^T[I| - T_1] \text{ and}$$
$$\hat{P_2} = \hat{K}\hat{R}^T[I| - T_2]$$

Assuming the new Rotation Matrix $\hat{R} = [\hat{r_1}^T, \hat{r_2}^T, \hat{r_3}^T]^T$.
We know that the X axis has to be parallel to the baseline. So,

$$\hat{r_1}^T = \frac{T_1 - T_2}{||T_1 - T_2||}$$

Taking new Y axis perpendicular to X axis and old Z axis of the left camera,

$$\hat{r_2}^T = r_3 \times \hat{r_1}^T$$
$$\hat{r_3}^T = \hat{r_1} \times \hat{r_2}^T$$

Finally, the new camera matrix can be taken as,

$$\hat{K} = \frac{K_1 + K_2}{2}$$

In the non rectified image, the camera pixels were calculated as,

$$\lambda_1 x_1 = K_1 R_1^T [I| - T_1] \text{ and } \lambda_2 x_2 = K_2 R_2^T [I| - T_2]$$

But now, camera pixels in the rectified plane will be

$$\hat{\lambda_1}\hat{x_1} = \lambda_1 \hat{K}\hat{R}^T R_1^{-T} K_1^{-1} x_1 \text{ and } \hat{\lambda_2}\hat{x_2} = \lambda_2 \hat{K}\hat{R}^T R_2^{-T} K_2^{-1} x_2$$

4 3-D Point Clouds

As discussed earlier, by knowing the disparity values of a particular point in the image, we can find it's depth in the world frame. This can be used to regenerate a 3-D render of the scene. A way to represent several disparity values is to color code them like in the image attached below.

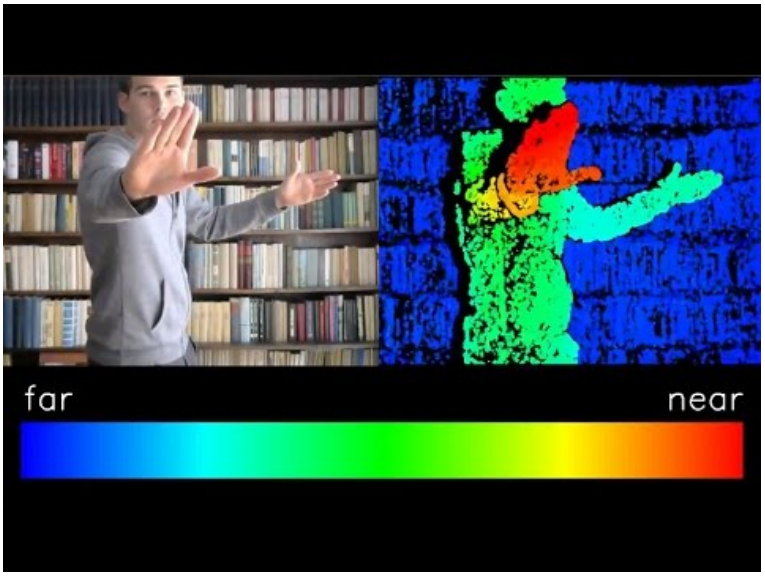

Figure 4: Color Coded Disparity Map

This concept can be extended a little further. As we saw previously, given the values of b, f and the disparity value, we can easily calculate the depth of that point.

$$Z_P = \frac{bf}{x_1 - x_2}$$

We can also calculate:

$$x = \frac{b(x_1 + x_2}{2(x_1 - x_2)}$$
$$x = \frac{b(y_1 + y_2}{2(x_1 - x_2)}$$

With these values of x and y, we can compute the point cloud.

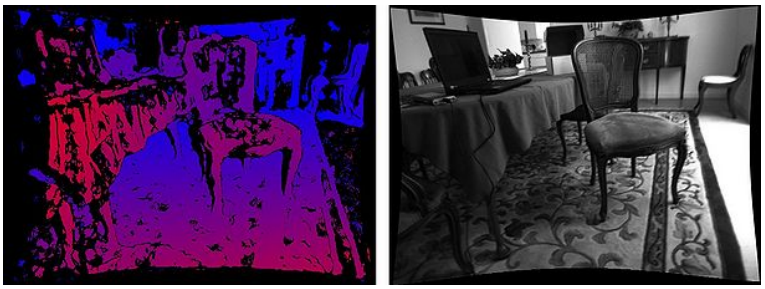

Figure 5: A sample image with it's Disparity Map

Figure 6: The corresponding point cloud representation

References

[1] Image Rectification: http://www.close-range.com/docs/Image_rectification.pdf

[2] Lecture Slides

[3] ETH Zurich Stereo : http://rpg.ifi.uzh.ch/docs/teaching/2018/07_multiple_view_geometry_1.pdf

[4] Image Rectification - Wikipedia : https://en.wikipedia.org/wiki/Image_rectification