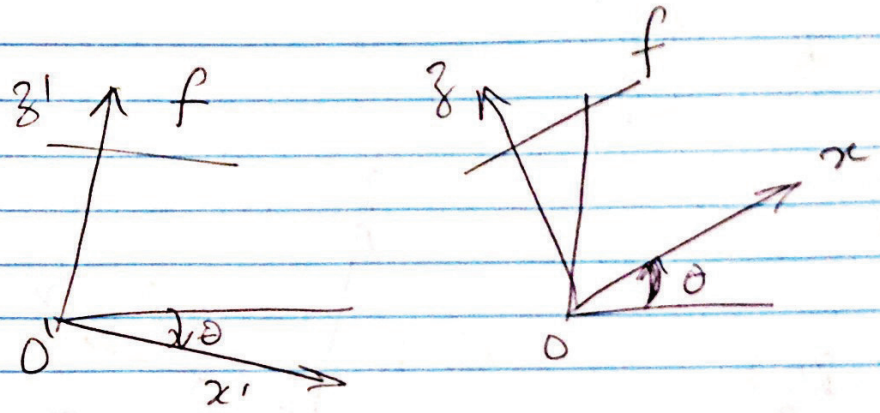
Problem 1 :



(a) $E = f(\theta) \cdot = RS$

For rotation from left camera axis to right camera axis, we have $\phi = 2\theta$ (angle of rotation).

Rotation matrix for rotation along y-axis is :

$$R = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

In our case, $\phi = 2\theta$,

$$\therefore R = \begin{bmatrix} \cos 2\theta & 0 & \sin 2\theta \\ 0 & 1 & 0 \\ -\sin 2\theta & 0 & \cos 2\theta \end{bmatrix}$$

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ 0 & 0 & -T_x \\ -T_y & +T_x & 0 \end{bmatrix}$$

For translation from O' to O, we have

$T_z = B\sin\theta$, $T_x = B\cos\theta$, $T_y = 0$.

$$\therefore \quad S = \begin{bmatrix} 0 & -B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix}$$

$$E = R \cdot S = \begin{bmatrix} \cos 2\theta & 0 & \sin 2\theta \\ 0 & 1 & 0 \\ -\sin 2\theta & 0 & \cos 2\theta \end{bmatrix} \begin{bmatrix} 0 & -B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 0 & -B\sin\theta \cdot \cos 2\theta + B\cos\theta \cdot \sin 2\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ +\cancel{B\sin\theta \sin 2\theta} & B\sin 2\theta \sin\theta + & 0 \\ 0 & B\cos 2\theta \cdot \cos\theta & \end{bmatrix}$$

$$E = \begin{bmatrix} 0 & B\sin(2\theta - \theta) & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos(2\theta - \theta) & 0 \end{bmatrix}$$

$$E = \begin{bmatrix} 0 & B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix}$$

(b) To compute the right epipole on the left image plane,

$e_R^T E = 0 \Rightarrow$ Assuming $e_R^T = [u, v, f]$;

We have

$$\begin{bmatrix} u & v & f \end{bmatrix} \begin{bmatrix} 0 & B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix} = 0$$

$$v \cdot B\sin\theta = 0 \quad \text{— (1)}$$
$$u B\sin\theta + f \cdot B\cos\theta = 0 \quad \text{— (2)}$$
$$-vB\cos\theta = 0 \quad \text{— (3)}$$

From equation (2), we have :

$$u = -f \cdot \cot\theta .$$

From (1) & (3) we have $v = 0$.

Also, since we know that the planes $x-z$ and $x'z'$ are co-planar, the line joining $O-O'$ must also lie on this plane.

Thus, both epipoles lie on $x-z$ & $x'-z'$ plane

$\therefore v = 0$ for both epipoles.

$$e_R = \begin{bmatrix} -f \cdot \cot\theta \\ 0 \\ f \end{bmatrix} \text{ wrt left camera axes.}$$

To compute left epipole on the right image plane,

$E e_L = 0$. We assume $e_L = \begin{bmatrix} u \\ v \\ f \end{bmatrix}$.

$$\begin{bmatrix} 0 & B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ f \end{bmatrix} = \begin{bmatrix} v \cdot B\sin\theta \\ uB\sin\theta - fB\cos\theta \\ vB\cos\theta \end{bmatrix} = 0$$

$\therefore e_L = [f\cot\theta, 0, f]$ wrt right camera axes.

(c) (u', v') = point on left image.

Corresponding epipolar line in right image = ?

$$\tilde{l}_R = E \cdot p_L = \begin{bmatrix} 0 & B\sin\theta & 0 \\ B\sin\theta & 0 & -B\cos\theta \\ 0 & B\cos\theta & 0 \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

$$\tilde{l}_R = \begin{bmatrix} v'B\sin\theta \\ u'B\sin\theta - B\cos\theta \\ v'B\cos\theta \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Assuming a point on right image (u, v) through which epipolar line $\tilde{l}_R$ passes, we have equation of line:

$$uv'\sin\theta + (u'\sin\theta - B\cos\theta) \cdot v + v'\cos\theta = 0$$

Problem 2: Window based stereo matching algorithm for a rectified pair of images.

In my code for template matching, I have chosen the left image as the reference image. An image region in the left image is assigned to the template and the program tries to find a match for the template in the right image. The template eventually traverses through and finds corresponding matches for all points in the entire left image.

Window size: The parameter template_size determines the size of the window that looks for matches. I experimented with different template sizes and disparity window sizes. The template sizes had a direct impact on the quality of output of the disparity map. Shown below in Figure 1 are a few experiments with different window sizes. For the given image pair, template_size = 20 was found to give the best disparity map.

(a) (b) (c)

Figure 1
Variation in disparity map with change in template size. Darker regions have high disparity and lighter regions have low disparity. (a) Template size = 20, Maximum disparity= 40; (b) Template size = 9, Maximum disparity= 20; (c) Template size = 31, Maximum disparity= 50;

Disparity window: When looking for a match, the template does not look along the entire row since it is unlikely that the matching point will be very far from the template position. The parameter max_disp_colsize (maximum disparity) fixes the maximum distance from template center along which the template looks for a match. For the given stereo pair the best value was found to be 40, for a template size of 20. In general, for a given template size we observe reasonable performance when maximum disparity is set to 1.5 to 2 times the template_size (for the given image pair. This varies according to the stereo pair).

Matching criteria:

I implemented 3 different matching criteria - sum of absolute differences (SAD), sum of squared differences (SSD) and normalized cross correlation (NCC). From Figure 2, it can be qualitatively observed that the performance of SSD, SAD NCC is comparable since all three matching criteria are extremely noisy. NCC is qualitatively the best matching criteria due to slightly lesser object edge noise. However, as can be seen in the table above, the very high computational complexity of NCC offsets its advantage. Since SAD and SSD generate disparity maps of similar quality, SAD, which is the least computationally expensive matching criterion, is chosen.

	SAD	SSD	NCC
template_size	20	21	21
max disparity	40	40	40
time taken	19.79 sec	67.39 sec	3 hrs

Improvements on template matching:

To improve performance, I additionally defined two other matching criteria – gradients of image along x and y directions respectively. By matching gradients along with raw intensity values across left and right images noise is reduced to a large extent as seen in Figure 2, which is a linear combination of the intensity disparities and gradient disparities.

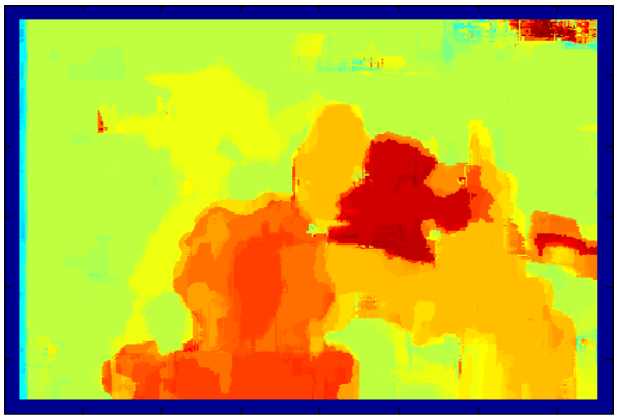


Figure 2.

Linear combination of intensity disparity map and gradient disparity map.
Dark red regions have higher disparities. Light yellow regions have lower disparities.

Use of graph cut segmentation:

Due to the large presence of noise, I tried using a graph-cut based segmented image to refine the disparity map further. For image segmentation, I used the kernel graph-cut code by Ayed[1] and MATLAB wrapper for Veksler et al.'s implementation of Graph Cut algorithm[2],[3],[4] provided by Bagon[5]. Ayed's code implements multi-region graph cut image segmentation according to the kernel-mapping formulation in Salah et al. [6]. Ayed's kernel graph-cut code segments an image into 'k' (parameter that is chosen by user) regions based on color. The left image is segmented into k segments, for each of which we find the dominant disparity in our computed disparity map. Thus, a less noisy disparity map is generated, as shown in Figure 3.

Figure 3. Disparity map produced after image segments are assigned with dominant disparities.

Advantages, Disdavantages:

The disadvantage of this method is that, by assigning an image segment with the dominant disparity value, we lose the true disparity values of every point in our image. Thus, when we compute the depth map by multiplying by the appropriate values based on camera intrinsics, we will not get a true distance map. We may have to re-calibrate the depth map intensity values with ground truth to get an accurate distance map/ depth map. Also, it can be noticed that the video camera (farthest object in scene) is lost in the segmentation. This, I believe, can be fixed by fine-tuning the segmentation and other parameters for this image.

The advantage is that we get a quantitative measure of relative distances of objects from the camera. Also, the disparity map is less noisy, and we have clear segmentation of objects present at different depths, which was not the case with simple case of template matching. We get relatively good results for a simple, computationally inexpensive algorithm. Our implementation runs in less

than a minute. The stereo matching takes 20 seconds once the segmented images are pre-computed.

With use of better energy minimization techniques and application of smoothing filters, the quality of our results can be improved greatly.

References:

[1] Ismail Ben Ayed. Kernel graph cut image segmentation. 2012. http://www.mathworks.com/matlabcentral/fileexchange/38555-kernel-graph-cut-image-segmentation.

[2] Yuri Boykov, Olga Veksler, Ramin Zabih. Efficient Approximate Energy Minimization via Graph Cuts. IEEE transactions on PAMI, vol. 20, no. 12, p. 1222-1239, November 2001.

[3] Vladimir Kolmogorov and Ramin Zabih. What Energy Functions can be Minimized via Graph Cuts? European Conference on Computer Vision (ECCV), May 2002

[4] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), September 2004.

[5] Shai Bagon. Matlab Wrapper for Graph Cut. 2006. http://www.wisdom.weizmann.ac.il/~bagon

[6] M. Ben Salah, A. Mitiche, and I. Ben Ayed. Multiregion Image Segmentation by Parametric Kernel Graph Cuts, IEEE Transactions on Image Processing, 20(2): 545-557 (2011).