Karthik.Uppuluri@rice.edu Karthik Uppuluri

Mid-Term Report

ELEC 547 Computer Vision

1. Describe the various steps in setting up stereo based depth estimation in detail.

Step1: Images

We need 2 or more images (2D) of the same 3D scene (which is static) taken from different viewpoints.

Step2: Rectification of stereo pair

Epipolar geometry: As shown in the figure below potential matches for point x_L on the left image will lie on the epipolar line x_R-e_R. But this search problem is in 2D (as the epipolar lines are skewed figure.2) and it will be computationally very expensive. To avoid such a computational mess, the images can be rectified so that epipolar lines are aligned with axis and parallel (figure.2). This can be achieved by computing homographies which can re-project the images such that the image planes are parallel to each other and to the baseline.

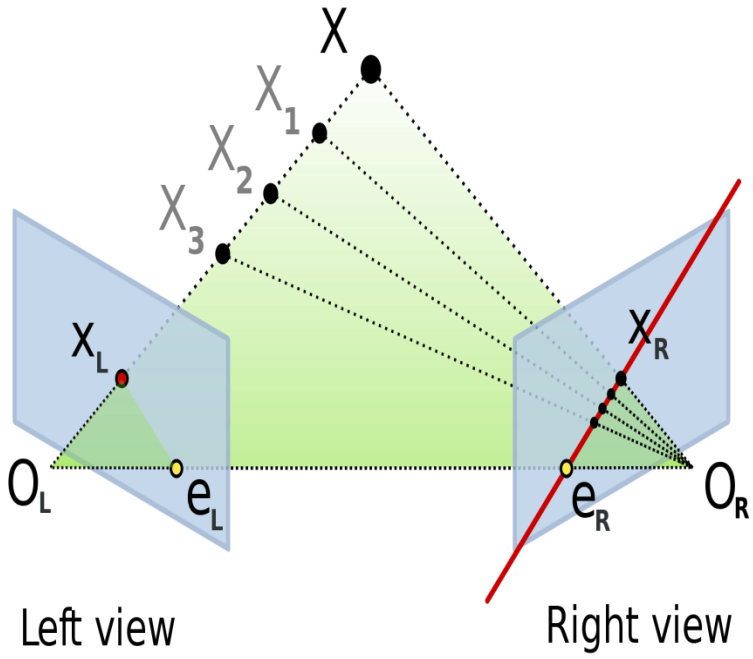


Figure.1 Epipolar Geometry [1]

Step3: Finding point correspondences

Once the stereo pair is rectified, the correspondences or the potential matches for each pixel in the left image to that of its right can be obtained by using any one of the similarity metrics defined below,

a) SSD (sum of the squared differences in the intensities of the pixels or a patch of pixels)
b) SAD (Sum of the absolute differences in the intensities of the pixels or a patch of pixels)

c) Normalized correlation.

For each pixel in the left image, can be matched with corresponding pixels on the epipolar line or a small patch around each pixel in the left image can be matched to all the patches around the pixels on the epipolar line (right image). Once the similarity metric is computed the potential match will be a global minimum in the case of SSD and SAD, and a global maximum in the case of normalized correlation.

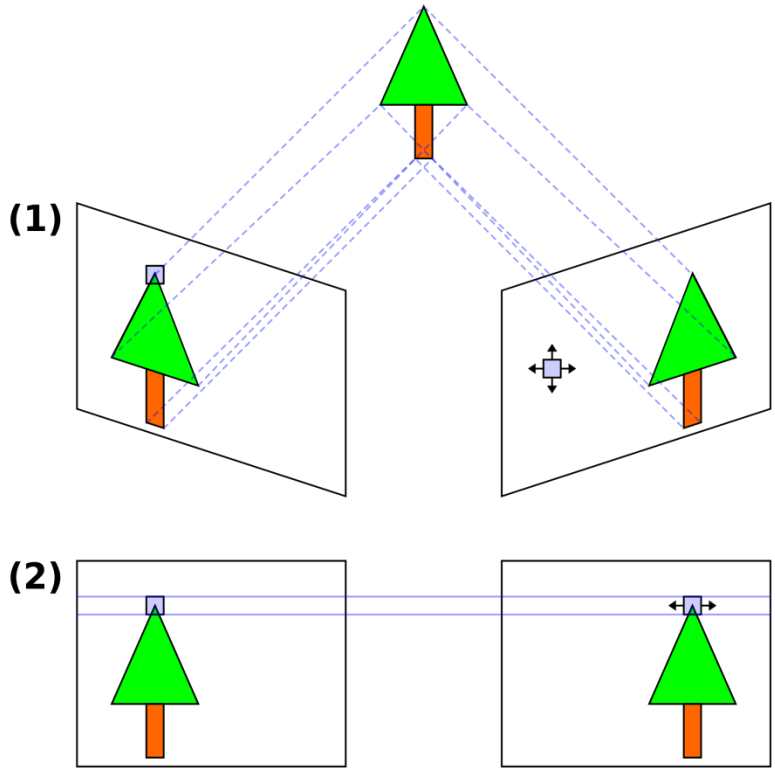


Figure.2 Correspondence search to 2D (before rectification) and 1D after rectification [2]

Step4: Find the depth at each point using triangulation method

Once correspondences (x and x') are found at each pixel in the left image, depth at that pixel can be found using triangulation as shown in figure.3.

$$\frac{B}{Z} = \frac{B-(x-x')}{Z-f}$$

$$B(Z-f) = BZ-(x-x')Z$$

$$\text{disparity } (x-x') = \frac{B.f}{Z}$$

Eventually depth Z can be recovered from the same equation.

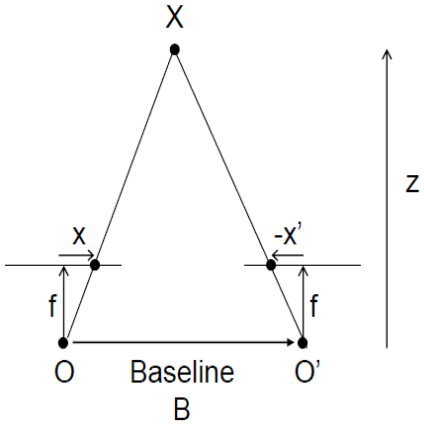


Figure.3 Triangulation method [3]

2. Describe an algorithm for rectification—why is rectification necessary? Can any two arbitrary cameras be rectified? If so how?

Rectification: Basic Algorithm [4]

Steps:

1. Left camera is rotated so that the epipoles go to infinity along the horizontal axis.
2. The same rotation is applied to the right camera.
3. Right camera is once again rotated by R.
4. Scale in both the camera frames is adjusted.

A triple of mutually **orthogonal unit vectors** is to be constructed to build rectification matrix (homography). Let the first vector e1 is given by the epipole. From the figure (1) we can see that epipole e_L coincides with the direction of translation (T).

$$e_1 = \frac{T}{||T||}$$

since e_2 is an orthogonal unit vector to e_1

$$e_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} [-T_y, T_x, 0]^T$$

Since e_3 is orthogonal to e_1 and e_2

$$e_3 = e_1 \times e_2$$

$$R_{rect} = \begin{pmatrix} e_1^T \\ e_2^T \\ e_3^T \end{pmatrix} \quad \text{equation (1)}$$

R_{rect} is the rotation matrix which rotates the left camera about the projection center in such a way that the epipolar lines become parallel to horizontal axis.

Steps for Image rectification:

1. Build R_{rect} matrix as shown in equation (1)
2. $R_l = R_{rect}$ (left Image) and $R_r = R R_{rect}$ (right Image)
3. For each pixel in the left image $p_l = [x, y, f]^T$

   compute $R_l p_l = [x', y', z']$

   rectified point is given by $p_l' = \frac{f}{z'}[x', y', z']$
4. Repeat the same processes for pixels in the right image using R_r and p_r

Image distortion is the major drawback of rectification. Zhang's algorithm describes a way of computing homographies with minimal distortion in images.

Rectification algorithm by Zhang [5]:

Goal: Compute homographies which will make the search for correspondences on skewed lines to same scan lines with minimal distortion in the rectified image.

Algorithm:

Let epipole at i be $= [1\ 0\ 0]^T$

Fundamental matrix is given by

$$F = [i]_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Compute homographies H and H' such that fundamental matrix $F = H'^T [i]_x H$

Rectified image points for each image can be computed using equations $\bar{m} = Hm$ & $\bar{m}' = H'm'$ (m- original point and m' projected point). H and H' can be decomposed into 3 transforms a special projective transform which captures all the distortion due to projective transform and the distortion is minimized(minimizing weight over all the pixels), a similarity transform after which the images are rectified and a shearing transform to preserve the resolution.

Necessity of Rectification: Searching for correspondences on skewed lines is a 2D search problem which is computationally very expensive. Rectification of the images is necessary because it reduces the 2D search problem to a 1D search problem (i.e. epipolar lines are just horizontal scanlines).

Arbitrary cameras: Yes any two arbitrary cameras can be rectified using any one of the above algorithms mentioned.

3. Describe an algorithm for correspondence search using stereo.

There are many ways in which we can look for correspondences using stereo

Approaches to Find Correspondences [6]

1. Intensity Correlation-based approaches
2. Edge / feature matching approaches
3. Dynamic programming
4. Energy minimization / Graph cuts
5. Probabilistic approaches

Algorithm:

Step1: Choose a pixel/patch (window of pixels) on the left image.

Step2: For the pixel/patch chosen compare it with all the pixels on the corresponding epipolar line which is on the right image. The similarity between the pixels/patches can be obtained using squared differences/sum of squared differences (in case of patch)/ normalized correlation between the intensities of the pixels. The best corresponding match will occur at global minimum in the case of SSD or SAD and global maximum in the case of normalized correlation.

Step3: The above steps are to be repeated for each pixel in the left image.

4. Describe a dynamic programming based method for finding correspondences in stereo --- Comment on the three variants you read about (a.ka. the method described in (b) and the two methods described in (c))

Dynamic Programming based method for finding correspondences for stereo [6]:

Choose a row in the left image. And for each pixel in the row

Step1: Chose a similarity metric to construct the cost matrix.

Any similarity metric like SSD, SAD or normalized correlation can be used to construct a cost matrix. Each row of the cost matrix corresponds to difference in pixel intensities between the pixel chosen in the left image and all the pixels in the epipolar line. So if there are N pixels in the row chosen and N pixels in the corresponding epipolar line. The cost matrix will be an NXN matrix.

Step2: Build the cumulative cost matrix (D) and a matching matrix (M)

Cumulative cost matrix and matching matrix are built using the pseudo code mentioned below. The occlusion penalty considered for experimenting in the programming part is 1.

Step3: Back trace over the matching matrix to find the optimal path which eventually gives disparity at each pixel.

Once the matching matrix is created back trace over the matrix from last pixel to first and find the optimal path using the pseudo code below.

```
occulusion_Penalty = const;
for i = 1:N
  for j = 1:N
    min_1 = D(i-1, j-1) + cost_mat(i, j);
    min_2 = D(i-1, j) + occulusion_Penalty;
    min_3 = D(i, j-1) + occulusion_Penalty;
    cmin = min(min_1,min_2,min_3);
    D(i, j) = cost_mat(i, j) + cmin;
     if (cmin == min_1)
      M(i, j) = 1;
       elseif (cmin == min_2)
        M(i, j) = 2;
         else
          M(i, j) = 3;
       end
  end
end
```

1. Pseudo code for cost matrix[6]

Back tracing for optimal path [6]

```
[m n] = size(match_matrix);
disparity = 0;
while (m ~= 0 && n ~= 0)
   val = match_matrix(m,n);
    if (val == 1)
     m = m-1;
     n = n-1;
    list_y(1,n) = disparity;
     elseif (val == 2)
       disparity=disparity-1;
       m=m-1;
       list_y(1,n)=disparity;
        else
          disparity = disparity+1;
         n = n-1;
         list_y(1,n)=disparity;
    end
end
```

List_y gives the disparity of the each row in the left image and the same steps are repeated for remaining rows.

Three variants of dynamic programming discussed are

a. Intra-scanline search (correspondences within same scanlines)
b. Inter-scanline search (correspondences along connected edges)
c. Maximum-likelihood

Intra scanline search based dynamic programming searches for correspondences (optimal path) within the same scan-lines which is a 2D search problem. Whereas the inter-scanline uses one more constraint that, two points which are on a vertically connected edge, should most likely lie on a vertically connected edge in the right image. This addition of the constraint makes the search for optimal path a 3D search (over a stack of 2D image planes/correspondences over vertically connected edges).

Maximum likelihood based algorithm[7]

Goal: Find correspondences between the stereo pair

Cost:

The cost is divided into two terms one which represents the cost of matching the feature (feature can be anything, for example it can be intensity of the pixel or some high level feature) and a fixed occlusion

penalty for an unmatched measurement (function of probability of occlusion). Dynamic programming along with two constraints (uniqueness and ordering) is used to obtain the optimal solution (optimal locations/ maximum likelihood of correspondences).

For images which have more vertically connected structures like building etc., inter-scanline strategy works very well due to the vertical connectedness constraint. Images which require this constraint should use the inter-scanline strategy, else intra-scanline works reasonably well. ML stereo algorithm is computationally more efficient as it can be implemented with just pixel intensity differences as features, without any need for high level features or windows.

5. How do modern stereo algorithms such as those in (d)(e)(f) and (g) differ from dynamic programming based methods? What are the salient differentiating factors that result in improved performance?

Answer: Any stereo correspondence or matching algorithm does four basic steps[8]

a. Constructing cost matrix
b. Cost aggregation
c. Disparity optimization.
d. Disparity refinement.

All modern stereo algorithms use optimization algorithms like belief propagation, graph cuts and iterative methods like iterative-semi global matching. The question actually boils down to which optimization algorithm would give better results is it dynamic programming or graph cuts or belief propagation.

Even though dynamic programming is faster it has many constraints like the problem should have overlapping sub-problems and optimal sub-structure to obtain an optimal solution. Whereas graph-cuts and belief propagation give an optimal solution even though such constraints mentioned above are not met.

Salient points which made difference in performance:

Adding more constraints to the cost function:

a. How compatible the disparity value is with respect to differences in intensities at respective pixels
b. How compatible the disparity values with respect to the disparity values of the neighbors.

Better optimization techniques like graph-cuts, belief propagation and iterative semi-global matching which have lesser constraints to obtain an optimal solution when compared to dynamic programming. Result validation checks using uniqueness constraints

Reduction of search area, by constraining the maximum and minimum disparity values lead to improved speeds. And few algorithms clustered the depth maps to obtain smoother results.

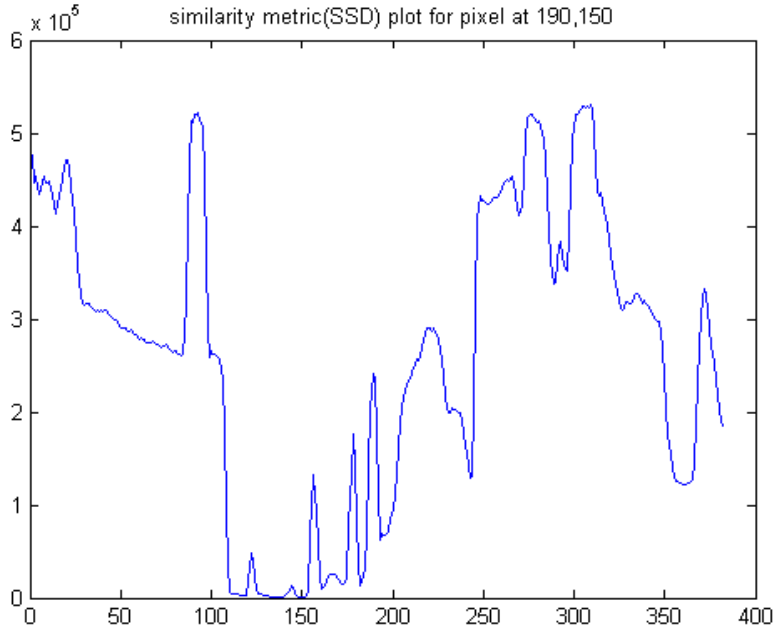
These are some of the salient points which made difference in performance.

Karthik.Uppuluri@rice.edu Karthik Uppuluri

Programming Assignment

1. **Patch size chosen is 3x3**
   Reason: (Empirical) direct pixel based similarity measurement is very sensitive to noise, even though windows of size more than 3 gave a smoother disparity map for tsukuba image 3X3 gave an optimal result.
   Similarity metric: SSD (sum of squared differences)
   Reason: Computationally more efficient when compared to correlation. And it is not much different from Sum of Absolute differences.

2. The plot clearly doesn't have an extremum which is way different from other extremums. This is because the patch considered over left image is not unique (there are many patches which look similar to this patch in the right image (over the face which doesn't have any texture)). Disparity value is 11.



3. Disparity and depth map figures for tsukuba image

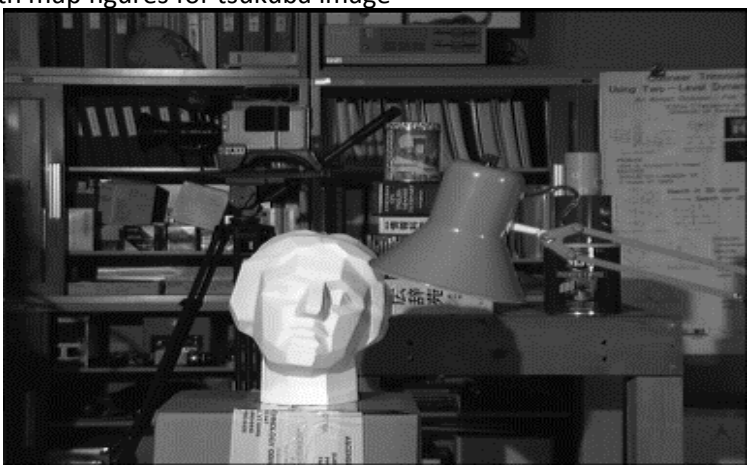


Figure1. Input Image

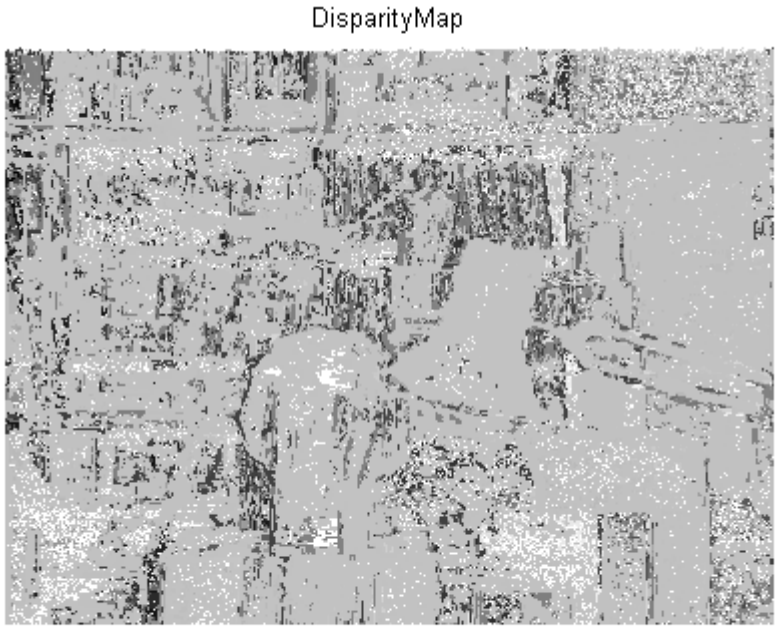


Figure2. Disparity map using windowing (3x3) and SSD (**white pixels don't have reliable extremum**)

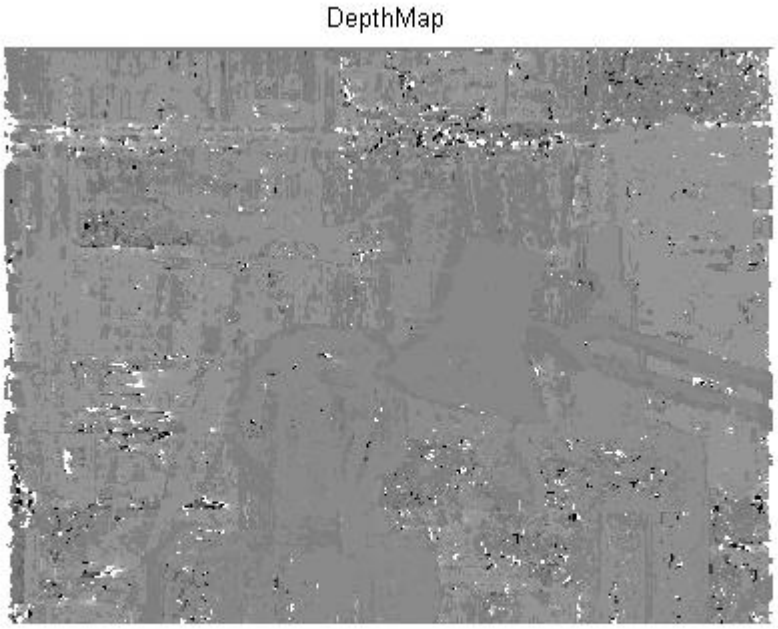


Figure3. Depth Map

4. Nearest neighbor interpolation to fill unknown values

Figure.4 Disparity map after interpolation (replacing with any one of the four neighbors)

5. Dynamic programming result

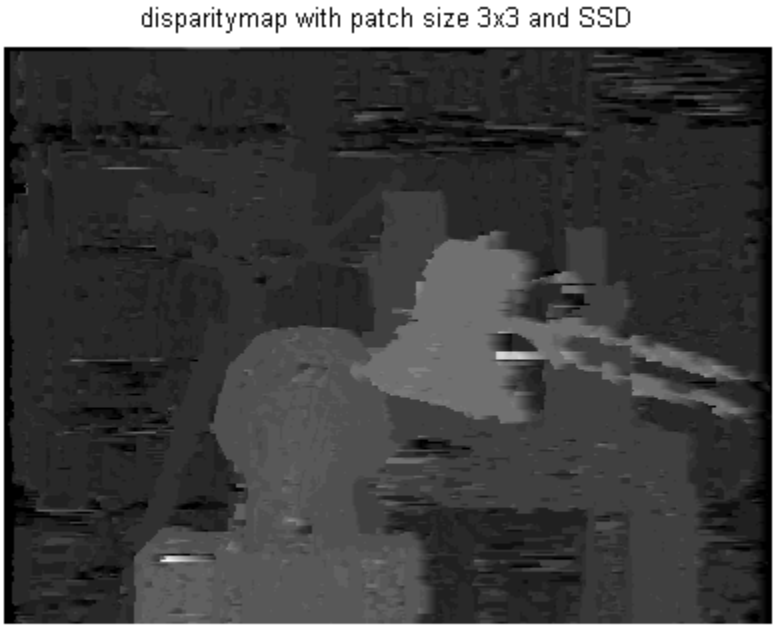


Figure.5 Disparity Map after dynamic programming

Patch size chosen: 3x3, feature: Pixel intensities and implementation algorithm is same as mentioned in question 4 theory assignment.

6. Results on some other images from other data sets

Rectified right and left Images are obtained from [9]

Figure6. Left Image

Figure7. Right Image

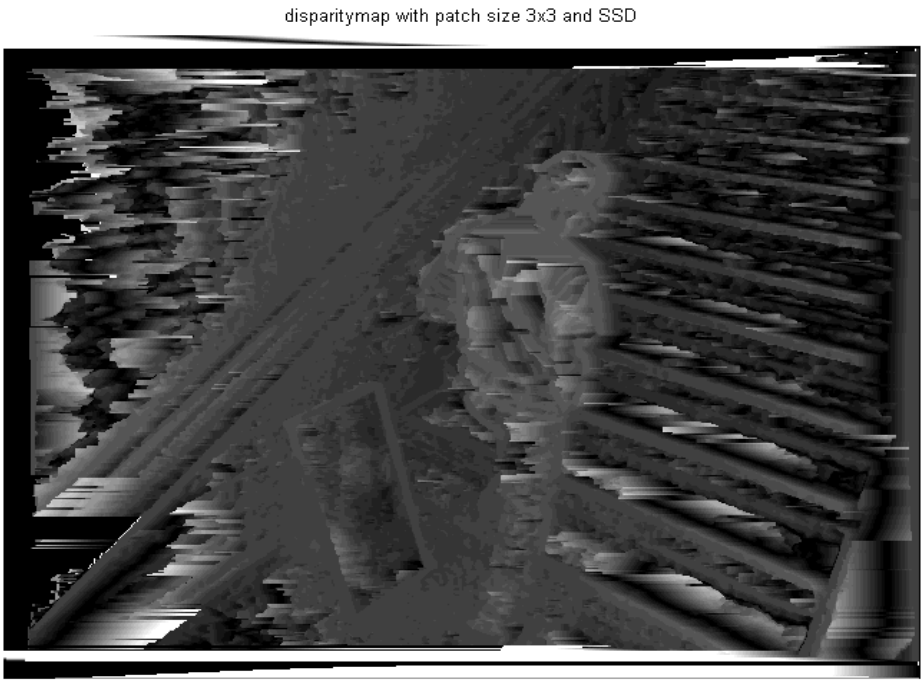


Figure8. Disparity Map

Figure9. Left Image

Figure10. Right Image

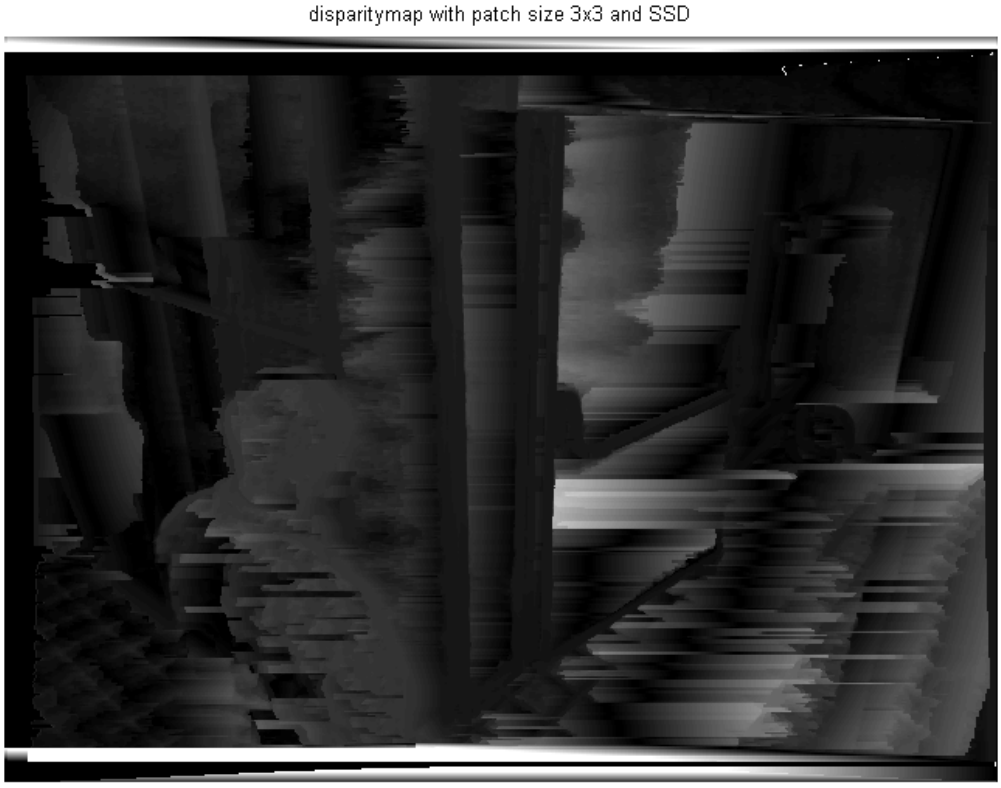


Figure11. Disparity Map

Figure12. Left Image

Figure13. Right Image

Figure14. Disparity Map

References

1. http://en.wikipedia.org/wiki/Epipolar_geometry
2. http://en.wikipedia.org/wiki/Image_rectification
3. http://en.wikipedia.org/wiki/Triangulation
4. TruccoVerri.IntroTechniquesFor3DComputerVision. Chapter 7 Stereopsis
5. C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.
6. www.cs.auckland.ac.nz/~jmor159/775/ppt/StereoCorrespB.ppt
7. Cox et. al, " A maximum likelihood stereo algorithm"
8. http://people.csail.mit.edu/billf/tappenIccv.pdf
9. http://vision.deis.unibo.it/~smatt/stereo_smp.html