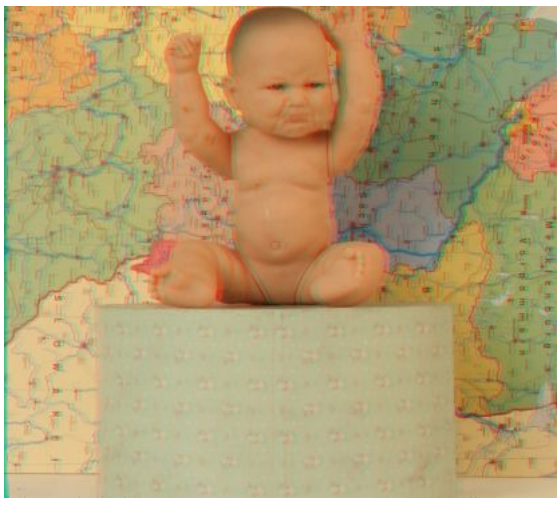
CSCI-B657: Computer Vision
HW4: Stereo

Ishtiak Zaman (izaman@indiana.edu)


Introduction

In this assignment, we have worked with stereo from two different perspectives: rendering stereo from depth maps, and creating depth maps both from single images and from stereo pairs. The major parts are:

- Creating stereograms (Part 1)
- Background-foreground segmentation (Part 2)
- Inferring depth from stereo (Part 3)

Creating Stereograms (Part1)

In this part, given an image and corresponding disparity map, we assume that the given image belongs to the right eye. To get the image for left eye, we need to shift the pixels according to the disparity map to the right.

We scaled down the disparity values multiplying by a constant disparity factor of 0.05 before transformation so that the left eye image does not shift too much compared to the right eye image. Two or more pixels can end up on the same position. To overcome this, we sorted the pixels from lowest disparity to highest disparity, and shifted the pixels in that order, so that lowest disparity (deepest depth) background pixels would be rendered first and may be overwritten by highest disparity (lowest depth) foreground pixels.

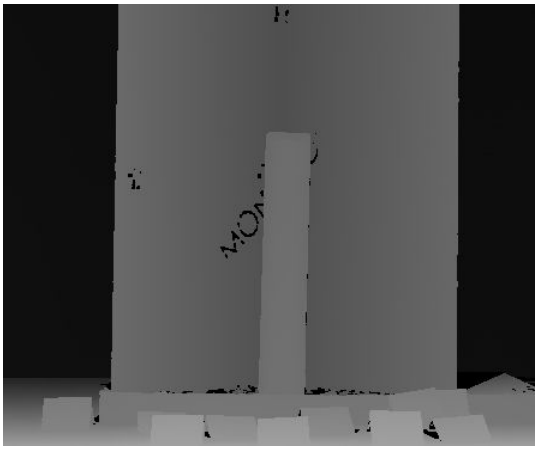


Fig: The original image and the disparity

Fig 1: After the transformation

Fig 2: After blending the empty regions

After the transformation, some of the regions never filled up and stays black (Fig 1). We kept track of those pixels and blend them up using a gaussian filter (Fig 2).

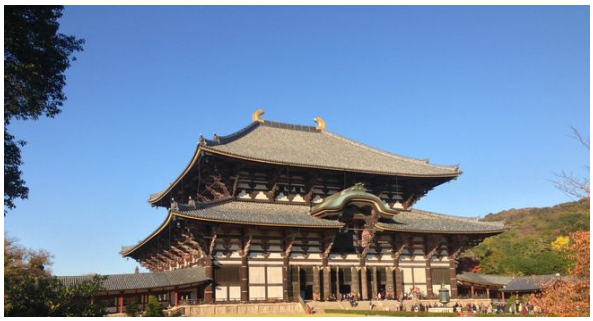
Finally to generate the stereogram, we took the red channel from the left eye image, and green and blue channels from the right eye image to the stereogram as the following:

Fig: The final stereograms

Compile: make render
Run: ./render <main_image> <disparity_image> [optional: disparity factor]
The disparity factor is 0.05 by default which can be changed on runtime.

Background-foreground segmentation (Part 2)

In this semi supervised part, we are given an image and a seed map with blue and red pixels that belongs to the foreground and background respectively. We have to make a decision for every pixels whether they belong to the foreground or background. The task is done in two different ways:



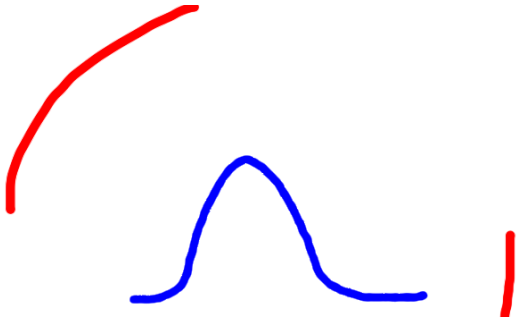


Fig: The original image and the seed map (red as background, blue as foreground)

Naive Approach: From the seedmap, we first labeled all the pixels with blue color as foreground and red color as background. Then we computed the mean and variance of the known foreground pixels to get a Gaussian probability density function. Then for all unlabeled pixels, we computed the cost for labeling foreground as $-\log N(I(x,y);\mu;\sigma)$ and background as β, and chose the label with the minimum cost. By default, β is set as 25 which can be changed on runtime.

MRF Approach: In this approach, we pre-computed the distance function as the naive function. And to guarantee smoothness between neighbor pixels we computed another pairwise cost function with Potts model, that will be α if two neighbor pixels are with different label and 0 otherwise. α is 10 by default which can be changed during runtime. We solved the problem using loopy belief propagation, which might not give an optimal solution but works fast and pretty good for a real case. We iterated through the top left of the image to bottom right and passed message to all four directions to figure out which label gives the minimum energy between label 0 and 1. We did this iteration 5 times and converged to a nice result, which is a good tradeoff between naive approach and smoothness. The following figures show the result for both naive and MRF approaches.

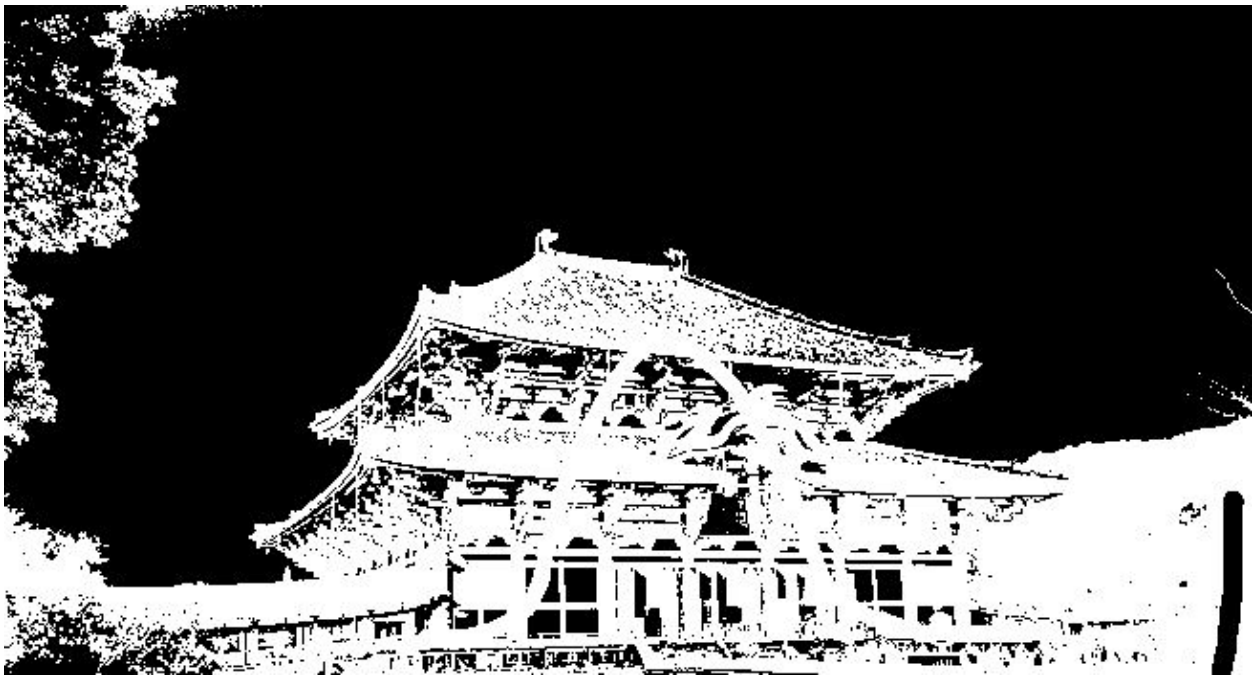


Fig: Disparity by naive approach

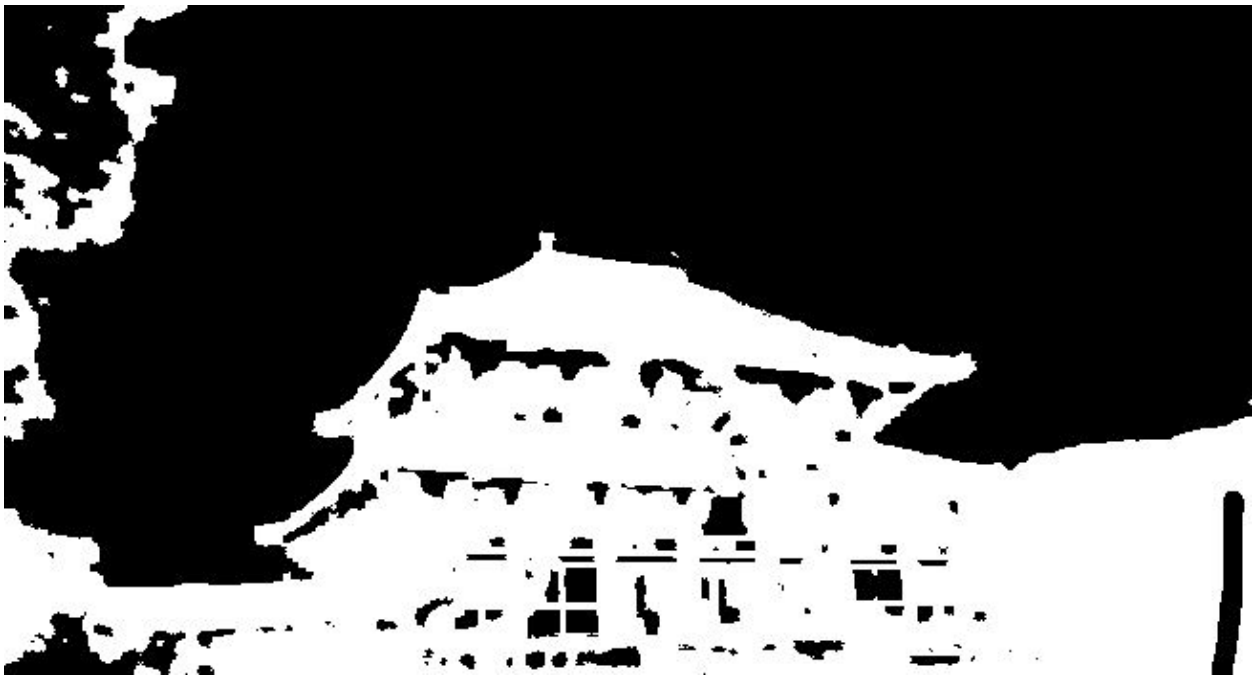


Fig: Disparity by MRF approach

From the two images, we can see that MRF results in a smoother disparity than the naive approach. We can change the smoothness by tweaking the α value during runtime.

Compile: make segment

Run: ./segment <main_image> <seed_image> [optional: β] [optional: α]
By default, β is 25, α is 10.

Stereogram from the Segmentation: We ran part1 to get stereograms from the disparity generated from part 2, here are a few sample images.

Fig: Stereogram for Parakeet MRF segmentation

Fig: Stereogram for nara MRF segmentation

Inferring Depth from Stereo (Part 3)

In this part, we are given two stereo images for left eye and right eye. We generated the disparity map between the two images using naive approach and MRF approach, and compared to the ground truth result give.

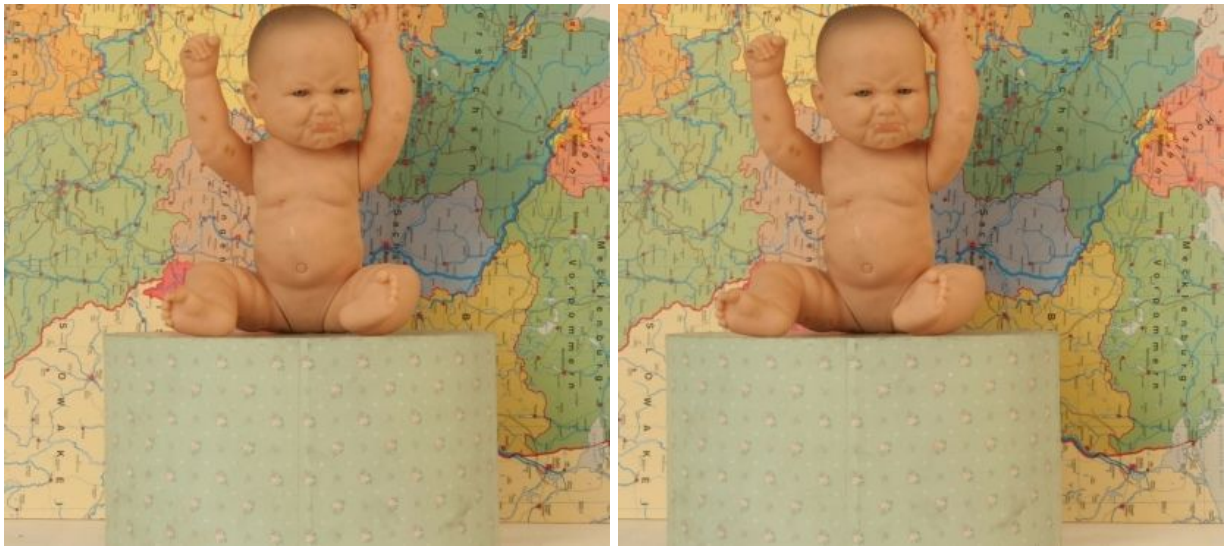


Fig: Given left eye image and right eye image of the stereo

Naive Approach: In this method, we every pixel in the left image, we searched up to 50 pixels for a pixel similar in the scanline through left direction. We defined similarity by the difference between all three color value difference sum squared within a short window around the pixel. We label the disparity of each pixel with the value within 0 to 50 that is the least costly when we computed the cost.

MRF Approach: In this approach, we pre-computed the distance function as the naive function. And to guarantee smoothness between neighbor pixels we computed another pairwise cost function, that will be α if two neighbor pixels are with different label and 0 otherwise. α is 1000 by default which can be changed during runtime. We solved the problem using loopy belief propagation, which might not give an optimal solution but works fast and pretty good for a real case. We iterated through the top left of the image to bottom right and passed message to all four directions to figure out which label gives the minimum energy between label 0 upto 50, which is our *max_disparity*. We did this iteration 5 times and converged to a nice result, which is a good tradeoff between naive approach and smoothness. The following figures show the result for both naive and MRF approaches.

Fig: Naive disparity map for Baby image

Fig: MRF disparity map for Baby image

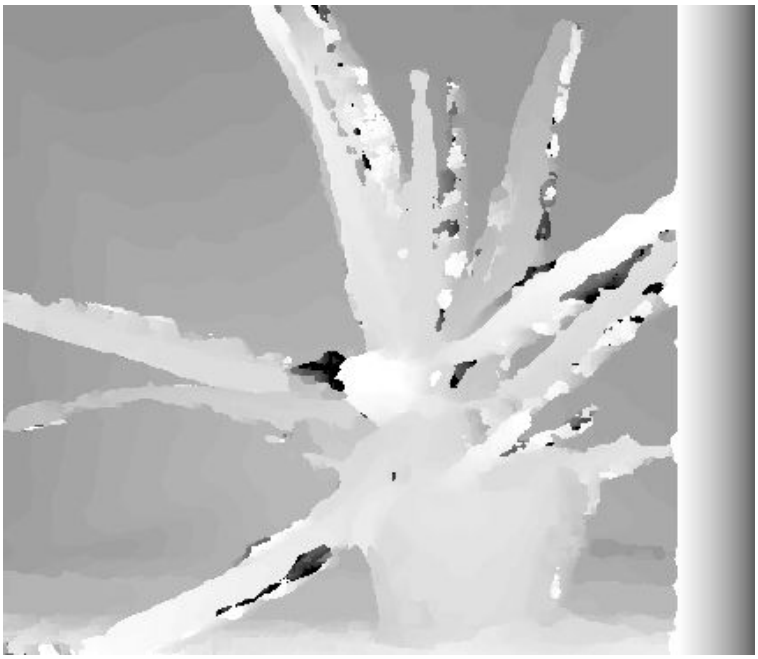


Fig: Naive disparity map for Aloe image

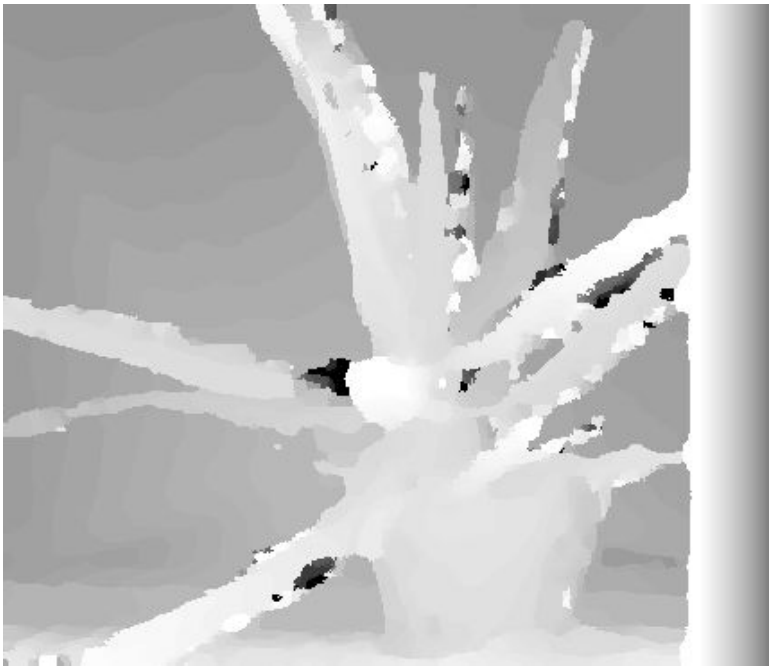


Fig: MRF disparity map for Aloe image

From the images, we can see that MRF results in a smoother disparity than the naive approach. We can change the smoothness by tweaking the α value during runtime.

Compile: make stereo

Run: ./stereo <left_eye_image> <right_eye_image> [optional: ground_truth_imge] [optional: α]
By default, α is 1000. The ordering of <left_eye_iamge> and <right_eye_image> must be preserved.

Stereogram from the Segmentation: We ran part 1 to get stereograms from the disparity generated from part 3, here are a few sample images.

Fig: Stereogram generated by one Baby image and MRF disparity map

Fig: Stereogram generated by one Aloe image and MRF disparity map

Quantitative Error: While running the program, we also generated the quantitative error compared to the ground truth value generated. We got the following error for the images:

Baby Image:
Naive stereo technique mean error = 329.648
MRF stereo technique mean error = 355.194

Aloe Image:
Naive stereo technique mean error = 258.645
MRF stereo technique mean error = 275.241

Flowerpot Image:
Naive stereo technique mean error = 512.986
MRF stereo technique mean error = 521.963

In all cases, the MRF techniques gives a slightly more error value due to the smoothness, as in the ground truth image, neighbor pixels are not same most of the time.

Conclusion

We can conclude that, the MRF approaches work better than the naive in all cases due to the smoothness. Although loopy BP is not guaranteed to converge, it worked pretty well practically in our cases.

References

1. Lecture Notes and Slides
2. Wikipedia