Assignment 4

CSCI B657 – Computer Vision

Pravilla-murugem-sriksrin

Pradeep Ravilla
Manikandan Murugesan
Srikanth Srinivas Holavanahalli

1. Files Submitted

Sr. No.	File Name	Included/Modified
1	Render.cpp	Modified
2	Segment.cpp	Modified
3	Stereo.cpp	Included
4	Right_stereo.png	Included
5	Result.png(stereo image)	Included

- Render.cpp - contains code for core logic for code for creating stereograms in part1
- Segment.cpp – contains the code for core logic for Naïve and mrf implementation in part 2
- Stereo.cpp – contains the code for core logic for stereo implementation of part 3

2. How to run the code?

- Part 1 – Creating Stereograms
  1) ./render image.png disp.png
- Part 2 – Naïve and mrf implementation
  1) ./segment input.png seeds.png
- Part 3 – Stereo
  1) ./stereo left_image.png right_image.png gt.png

Note: To run disparity image obtained from part 2, 3, we need to run each disparity image sequentially and see result.png to view the result.

3. Brief Outline of the algorithm , results and analysis

Part 1

- Read the input image and disparity image
- For each pixel in the image Repeat,
  - Multiply each pixel in disparity image with a constant factor
  - Add this intermediate result to row value of corresponding pixel in input image.
  - We will get shifted pixel values, these pixel values are considered as pixels in right image

- Now, we access the red channel of input image, green and red pixels of the right image to form the stereogram image.

Part 1 Results

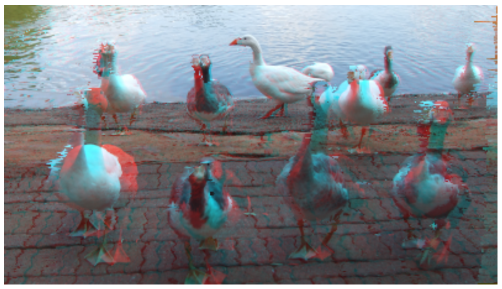

Fig 1. Stereogram obtained

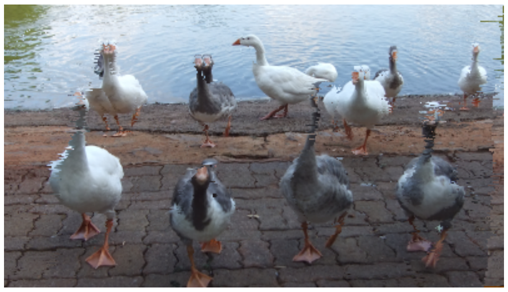

Fig 2. Right Stereo Image

Part 2

Naïve segment

- Read the input image, fg pixels and bg pixels
- Calculate the mean and variance of fg pixels using the input image
- For each of the pixel points in fg and bg , assign 1 and 0 respectively in disparity image
- For each of other pixels, repeat
  - Compute the Gaussian probability function for each pixel using mean and variance
  - We are assuming beta value as 22
  - Whichever is the lesser of Gaussian pdf and beta, we assign 0 or 1 correspondingly to that pixel in disparity image
- We return the result to the output segmentation function, which segments the image as foreground and background.

MRF segment

- Read the input image, fg pixels and bg pixels
- Utilize the mean and variance from naïve code
- Initialize all messages as zeroes for each label (0, 1) for each pixel in the image for each of the neighbor in 4 directions
- Repeat the following steps 100 times
  - For each pixel in the input image, for each label a pixel can take(0,1), for each of its neighbors, calculate the energy function that a pixel will send to its neighbor in consideration utilizing the previous messages sent from each of its neighbors
  - The pixel which receives messages, will get 2 values , each for its label
  - These 2 messages are saved for next iteration, noting from which neighbor it received message form.

- At the end of 100 iterations, the energy values a pixel receives from each of its neighbors will be stabilized and its own energy function value will be stabilized.
- Now, we calculate the label for each pixel. We compare the energy values of 0 and 1 and then assign the label for each pixel.

  Results

  Our MRF segment works well for all the images apart from woodpecker. Woodpecker image even though works fine, doesn't produce good results as compared to the other sets.

  Naïve segment

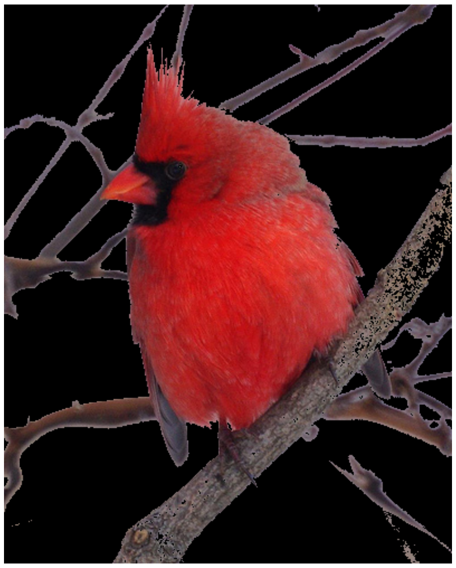
Fig3. Foreground segment image

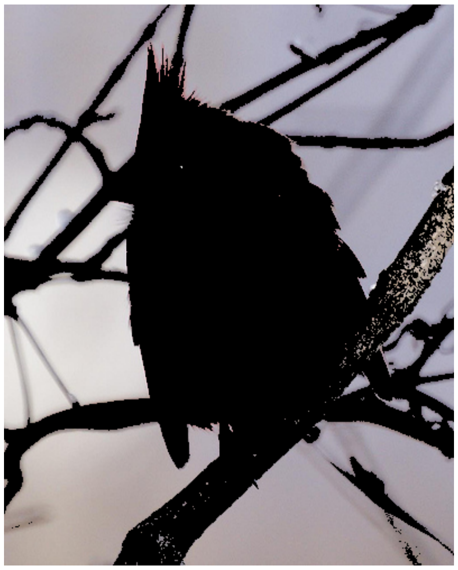
Fig 4. Background segment image

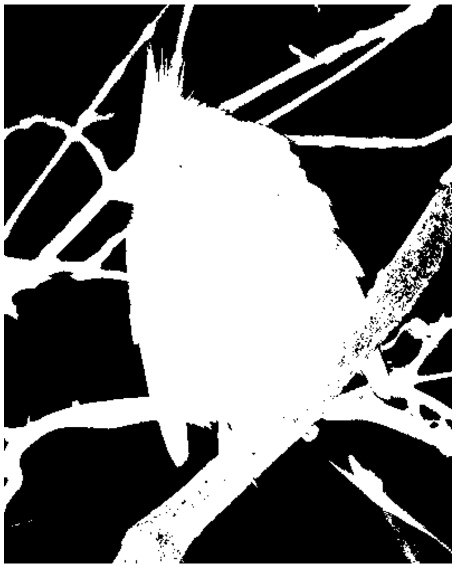
Fig 5. Disparity Image

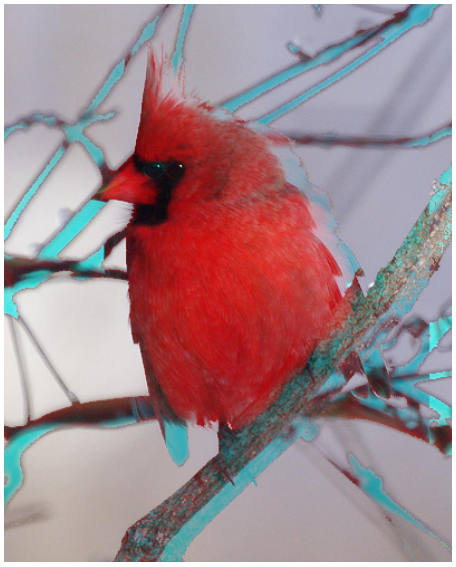
Fig 6. Stereogram using disparity image

Mrf Segment

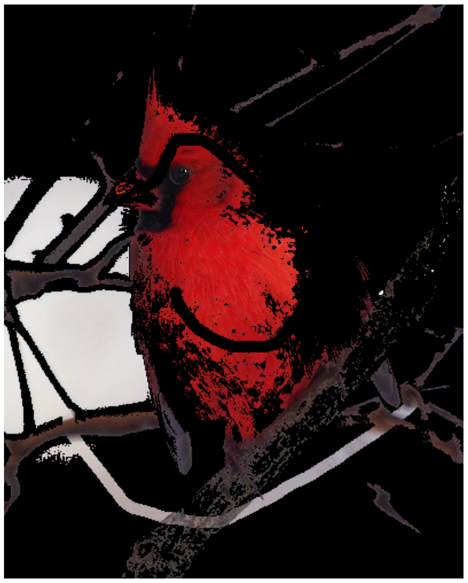

Fig 7. Foreground segment image

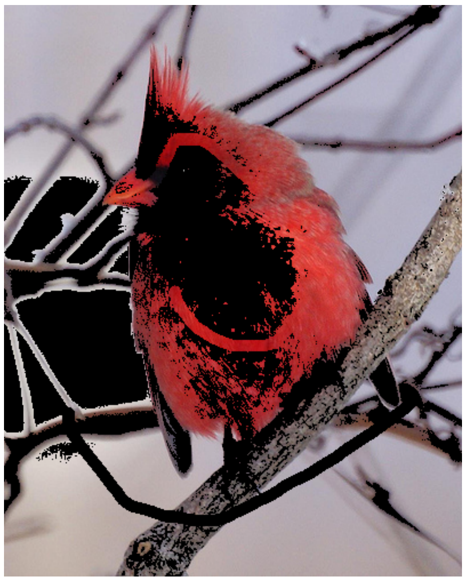

Fig 8. Background segment image

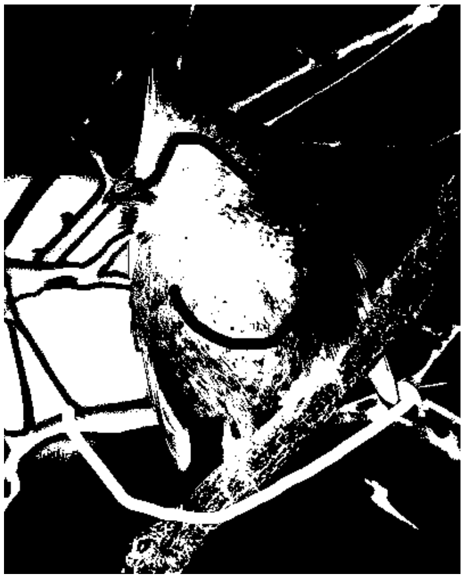

Fig 9. Disparity Image

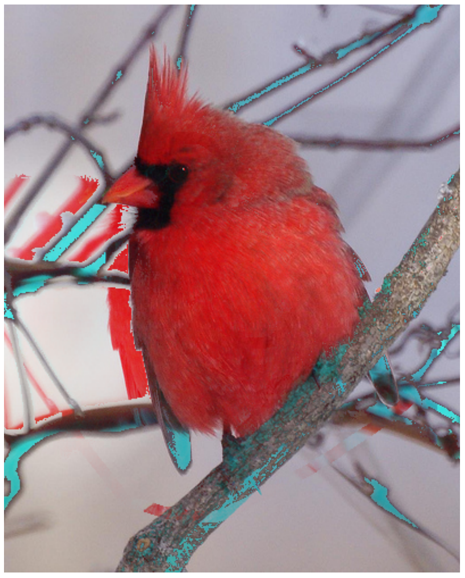

Fig 10. Stereogram using disparity image

2.3) Run your code from Part 1 on a disparity map generated by this part for a few of the sample images. How well does it work?

Answer: Fig 6, 10 above show how stereogram works using disparity image from results of Naïve and mrf respectively.

The results are good , if MRF is run for more iterations, we might get better disparity images and better results.

<u>Part 3</u>

- Read the input images image1 and image2
- Initialize all messages as zeroes for each label (0, 1) for each pixel in the image for each of the neighbor in 4 directions
- Here distance function is calculated differently, we take a **window** of 5x5 across each pixel in the scanline from left image and right image, compute the sum of all pixels in each window in left and right images respectively, take sum squared error of sums obtained from 2 windows
- This distance function is used in the next few steps of the algorithm
- Repeat the following steps 100 times
    - For each pixel in the input image, for each label a pixel can take[0,1,2…50], for each of its neighbors, calculate the energy function that a pixel will send to its neighbor in consideration utilizing the previous messages sent from each of its neighbors
    - The pixel which receives messages, will get 50 values , each for its label
    - These 50 messages are saved for next iteration, noting from which neighbor it received message form.
    - We had reused code written for part 2.2
- At the end of 100 iterations, the energy values a pixel receives from each of its neighbors will be stabilized and its own energy function value will be stabilized.
- Now, we calculate the label for each pixel. We compare the energy values of 0 and 1 and then assign the label for each pixel.

Results

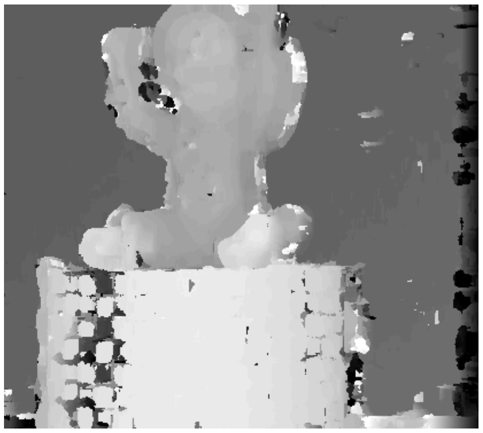
Fig 11. Naïve disparity image

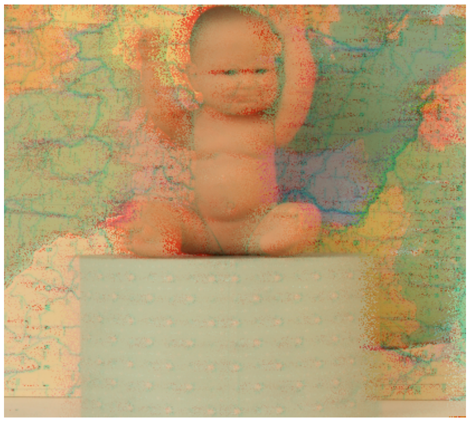
Fig 12. Stereogram using disparity image

Quantitative results

	Quantitative Error
Naïve stereo technique	83.2148
MRF technique	883.853

Our algorithm for Part 3 takes approximately 15 minutes to run.