CS 4476 Project 4

Ashwin Rathie
ashwin.rathie@gatech.edu
903281887

Part 1

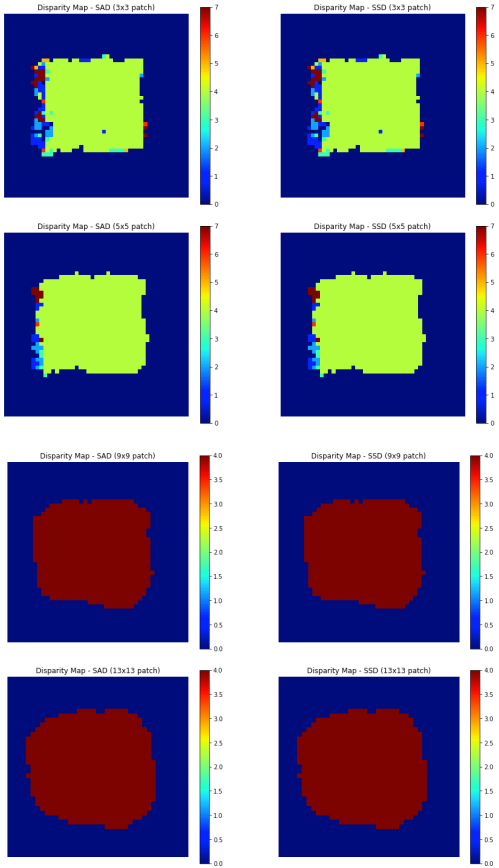
What is the relation between disparity map and depth? [Text/Equations/Drawing whatever you feel is relevant]

Generally, the higher a value on a disparity map is, the closer the corresponding point in the image is to the camera. On our disparity maps, this means the warmer the color, the closer the point, and the cooler areas are farther.

Random dot stereogram image [51x51x3] + Can you judge depth by looking them?

You cannot really judge depth just be looking at them.

Random dot stereogram disparity maps



What is the effect of increasing the block size? Explain the reasoning behind it?

In the case of the random dot stereogram, increasing the block size generally increases the SSD/SAD. This is because the calculate SSD and SAD methods calculate the sums of their respective disparities (absolute or squared) over the entire patch. So, if the patch increases, the overall disparity is going to increase, especially in the case of the random dot stereogram as there are a high number of differences in the center square.

Random dot stereogram: Why is the result poor on the left edge and not on the other edges?

In creating the random dot stereogram, we we instructed to move
“a block around the center block in the right image by 'disparity val' to the left”. The right edge was filled in with random values and the top and bottom edges are irrelevant in this implementation because we are only dealing with a horizontal shift. This mean that on the left edge, there were values that were essentially deleted completely, whereas that is not the case anywhere else.

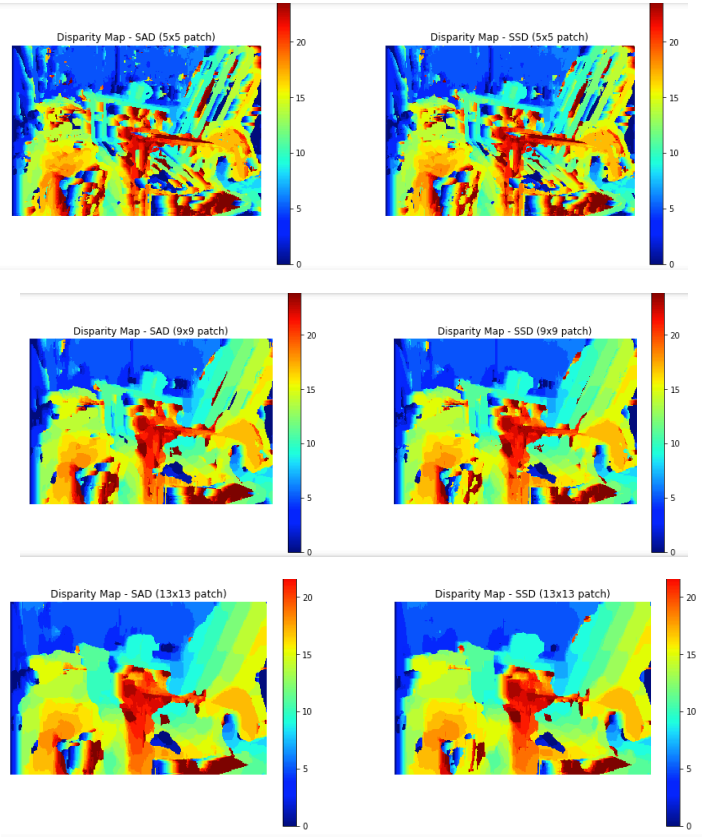
Convex error profile: Can you generalize the type of regions which will generate convex profiles?

A convex error profile will generally be generated by regions that are non-repeating or non-patterned. For example, a region with the tip of a pencil is not particularly patterned or repeated, for there is only one pencil tip and no real pattern. This is the case because a convex error profile has one “minima”, so to speak, in the disparity measurements because there is only one window in the search space that matches the patch well.

Nonconvex error profile: Can you generalize the type of regions which will generate non-convex profiles?

A non-convex error profile will generally be generated by regions that are repeating or patterned. For example, a region of a tiger's body is particularly patterned or repeated, for there is many similar looking stripes that form a pattern. This is the case because a non-convex error profile has multiple similar "minima", so to speak, in the disparity measurements because there are multiple windows in the search space that matches the patch well.

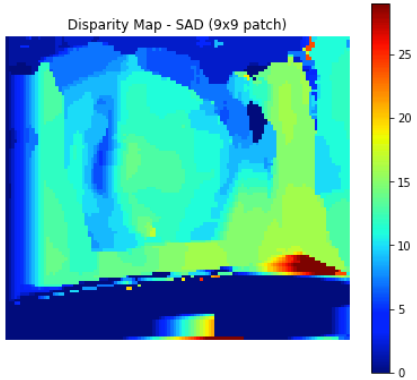
Disparity map for Set 1 (for 3 patch sizes)

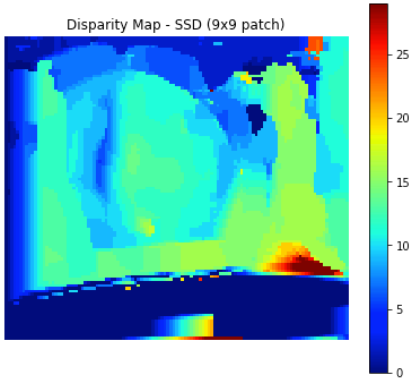


Set 1: Can you think of an explanation as to why the backrest of the chair appears blocky?

By blocky, we mean that each of the 4 panels on the backrest have distinctively different disparity values from each other but have a more or less homogenous disparity value among the areas of an individual backrest, particularly when the patch size is 9x9 and 13x13. This may be because if the patch size is just wider than one panel of the backrest such that a patch centered on the right of a panel includes the same amount of dark gap as a patch centered on the left of the panel, then the disparity values will be roughly the same, creating homogeneity over a panel.

Set 4 disparity maps (only one patch size)





Set 3, 4: peculiar behaviour of the disparity maps near right bowling pin: what do you see in input there and can you explain disparity map there?

The disparity maps seems to not be able to outline the head of the right bowling pin, showing a vague, deformed green blob rather than a clear head of a bowling pin. This is likely the input image shows that the bowling head pin has no distinct structures or patterns compared the the background and the bowling pin base to the right. This means that the shifted position on the bowling pin head from one image to another doesn't really show and clear disparity differences because the background is so similar to the head itself.

Set 3, 4: What was the change between set 3 and set 4? What effect did it have on the disparity? Can you generalize the reasons where disparity calculations wont

It looks like set 4 is a brightened version of set 3. This means that if something is whitish in set 3, it will appear as a much purer white in set 4. This resulted in the disparity map dropping to zero or near-zero on the bottom of the image and part of the bowling ball, because now the intensity in those regions is more homogeneous, as there is an intensity limit (i.e. 230 + 25 = 255 and 245 + 25 = 255).

Set 6: Effect on block size on the stairs in disparity maps

It seems that a lower block size produces more clear stair-like structures in the disparity maps. This is because a smaller window allows the algorithm to identify more detailed differences among the parts of an individual stair rather than try to look at a overly-general area.

Set 6: Gradual shift in disparity values on the wall

We are able to see a gradual shift in disparity values on the wall because the lighting along the wall gradually shift from light to dark (left to right). This allows the algorithm to effectively localize where in a search window we can match our patch, as the intensities of any region on the wall is not going to be the same as another region to its right or left due to the change in lighting.

Smoothing

1. Compare these results on the chair image qualitatively to the output of the chair image without smoothing.

The smoothed results look much more blobby and less structured/defined than the non-smoothed result. However, this also means that its looks far less patchy and noisy.

2. What regions of the image does smoothing seem to perform better on and why do you think that is?

The backrest of the chair is better with smoothing than no smoothing because it now shows a more gradual, continuous transition from close to far rather then the blocky, panel by panel transition. This is because with smoothing, the border along the panels are now penalized for having such different disparity values on one panel compares to the adjacent panel. The base of the chair also shows a significant improvement. It is a generally homogeneous region but without smoothing, it was very noisy. Smoothing enforces that nearby pixels should have closer disparity values.

Smoothing(contd.)

3. What regions of the image does smoothing seem to perform worse on and why do you think that is?

Smoothing performs worse on the two books to the left of the chair because after smoothing, the books are now shown as generally homogeneous blobs, losing the differences in disparity that would have otherwise represented the face that some parts of the book are farther away. This is because Semi-Global Matching is now forcing nearby pixels to have more similar disparity values, so when there are small but relevant differences in the disparities on the books, they are smoothed away.

4. Would smoothing still work for images with both a horizontal and vertical shift?

In order to make SGM work, we hold the cost volume of a point as a tensor of dimensions (H, W, D), where D is the max search bound (the size of the shift that represents the whole search region). If we had vertical and horizontal shift, we should have to add another dimension to cost volume to represent the vertical shift. Furthermore, smoothing compares a pixel's disparity value to its adjacent pixels. To determine which pixels are "adjacent" in a vertically and horizontally shift, we would have to know just how the images were shifted. Overall, it would be much harder and computationally expensive but I suppose it could be possible.

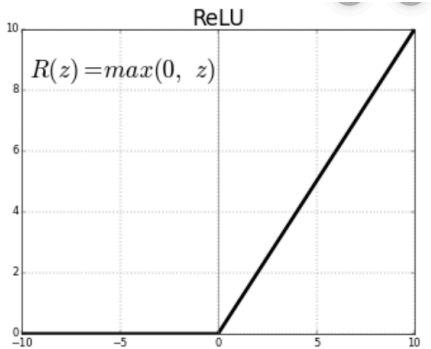
Part 2

Copy the output of print(net_tr) from the notebook here. You can use screenshot instead of text if that's easier.

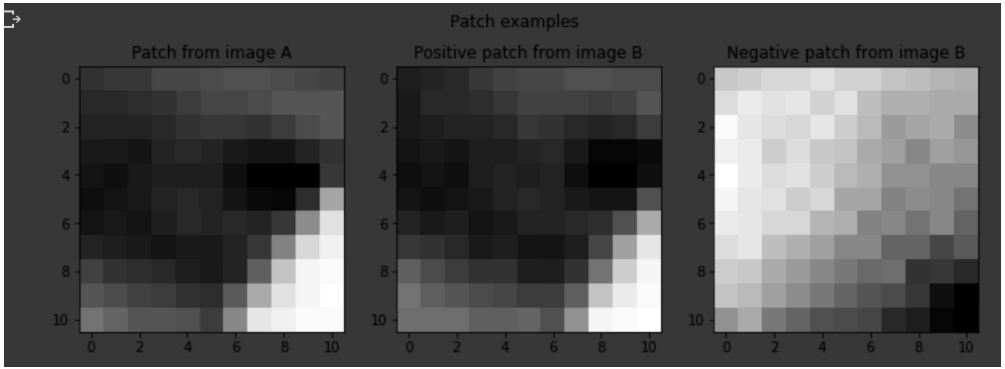
```
MCNET(
  (net): Sequential(
    (0): Conv2d(1, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU()
    (6): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU()
    (10): Reshape()
    (11): Linear(in_features=27104, out_features=384, bias=True)
    (12): ReLU()
    (13): Linear(in_features=384, out_features=384, bias=True)
    (14): ReLU()
    (15): Linear(in_features=384, out_features=1, bias=True)
    (16): Sigmoid()
  )
  (criterion): BCELoss()
)
```

What's the purpose of ReLU and why do we add it after every conv and fc layers?

ReLU, or rectified linear unit, is an activation function that acts as a max(0, node output) function, eliminating negative values. This means that we can create a sparse network rather than a dense one, decreasing computational intensity. ReLU allows us to get around the vanishing gradient problem. We add it after every conv and fc layer on order to reach convergence sooner and simplify and accelerate calculation



Show the patch example from your gen_patch function.



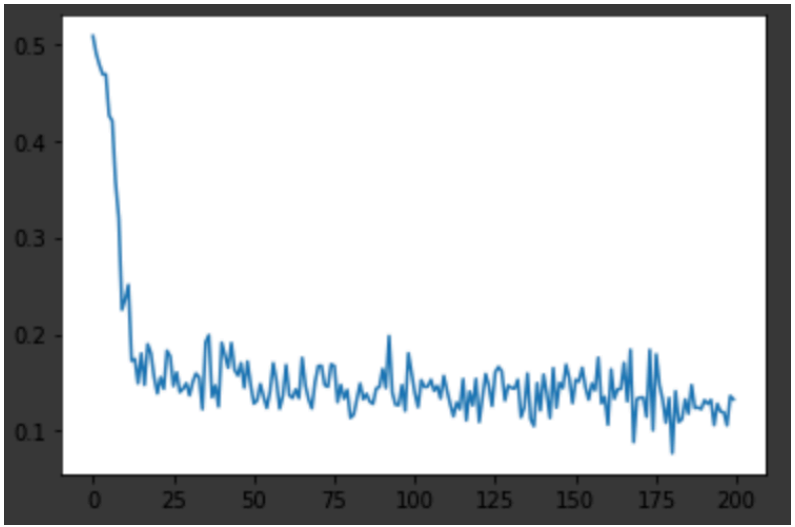
Giving a true disparity map for each stereo pair, how do we extract positive and negative patches for the training?

Positive patches, those whose center points are the same point in space, can be extracted with the equation q = (x - d + o_{pos}, y). If is P is center point of one patch with coordinate x and y, then q (where the other image's patch is to be centered), is found using the equation. D is the true disparity for the pixels. o_{pos} is a randle sample from a dataset of positive intervals.

The negative patches, patches where the center points are not same point in space, can be extracted by picking a random point o_{neg} from a dataset of negative intervals and using the following equation to create the center of the other image's patch, q = (x − d + o_{neg} , y).

Copy the training loss plot and the final average validation loss of your best network

```
Final average validation loss: 0.20953421
```



How does changing learning rate (try using large (> 1) vs small value (< 1e-5)) effect the training? Why?

A very high learning rate seems to converge to a consistent loss more quickly, but that loss is very, very high. A lower learning rate takes more epochs to converge but generally results in a much more accurate model. This is because a high learning rate is essentially telling the model to changed the learned parameters much more drastically to try to reach a solution faster, but this removes the granularity required to get precise, more correct parameter values.

What's the difference between Adam and SGD? What do you notice when switching between them?

Adam and SGD are two different optimizers. Stochastic Gradient Descent is like gradient descent but only operates on a small subset on the data, which is fine as long as we keep the learning rate low. Adam is also gradient-based but combines the advantages of RMSProp and AdaGrad.

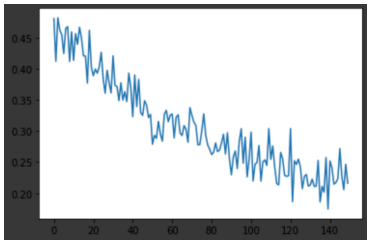
I found that Adam performed better than SGD, converging to a lower training loss faster. However, the validation loss was only .03 better for Adam.

Is your validation loss lower or higher than your training loss? What is the reason for that? How would we get a better validation loss?

My training loss was 0.131 and my validation loss was 0.2095 (my validation loss was higher). This is because the validation test is on a data point that was not trained on, that has never “been seen” by the model before. The validation loss can be improved by creating a model that is less overfitting and generalizes better.

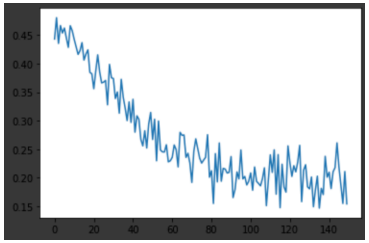
Copy the training loss plot and the final average validation loss of the networks with different window sizes

5x5



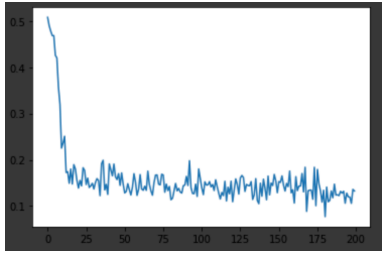
Final average validation loss: 0.6836855

9x9



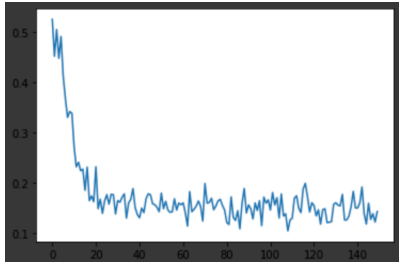
Final average validation loss: 0.6763414

11x11



Final average validation loss: 0.20953421

15x15

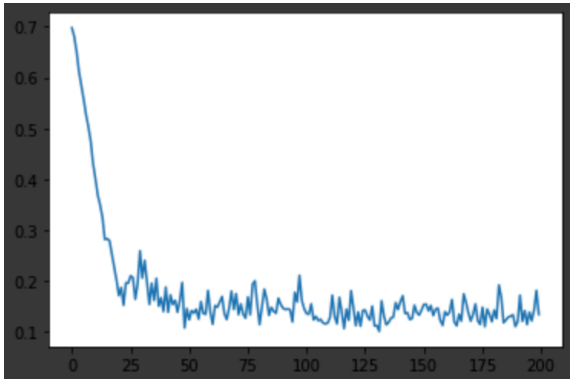


Final average validation loss: 0.6978755

What's the effect of varying the window size on performance? Do you suppose there is an optimal window size for all images? Explain why or why not.

It seems in this case that increasing window size betters the performance up to a certain point. I do not think there is an optimal window size for all images. A large window can be useful in that is gives the algorithm more information (more points) to use to match up, but in cases where the image has smaller details or repeating patterns at a larger scale, a smaller window could have better results.

Copy the training loss plot and the final average validation loss of your extended network, as well as the network layer list.



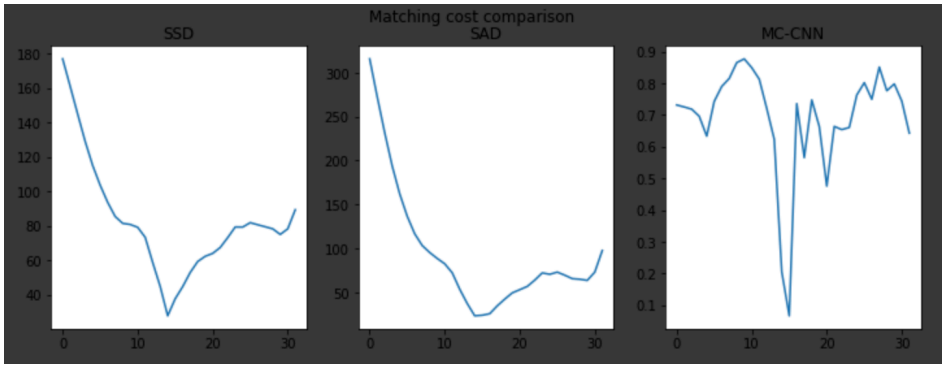

```
Final average validation loss: 0.22066398
```

```
ExtendedNet(
  (net): Sequential(
    (0): Conv2d(1, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU()
    (6): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU()
    (8): Conv2d(112, 112, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU()
    (10): Reshape()
    (11): Linear(in_features=27104, out_features=384, bias=True)
    (12): ReLU()
    (13): Linear(in_features=384, out_features=384, bias=True)
    (14): ReLU()
    (15): Linear(in_features=384, out_features=384, bias=True)
    (16): ReLU()
    (17): Linear(in_features=384, out_features=1, bias=True)
    (18): Sigmoid()
  )
  (criterion): BCELoss()
)
Testing for MCNET: "Correct"
```

What layers did you add? What's the effect of adding more layers without adding more data? Why?

I added an additional fully connected layer and an additional ReLU layer right before the final layer. Adding more layers without more data could be beneficial, but depending on the capacity of the network, could also harm performance. If you add more layer than what is sufficient, your model could overfit to the training data and generalize poorly to the unseen testing data, resulting in worse testing performance.

Show the matching cost comparison plot between SSD/SAD/MCCNN

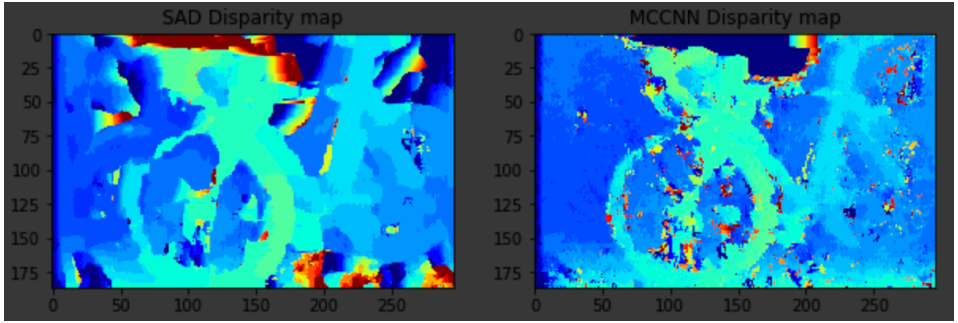


Based on the matching cost comparison plot, how are SSD, SAD, and MC-CNN different? Can you think of a scenario where matching with MC-CNN would be prefered over SSD/SAD?

SSD and SAD both have more stable-looking trends that lead the a single pointed minima and not many other local minima. MC-CNN, on the other hand, is very jagged and erratic and has many local minima, although the global minima is still very clear. MC-CNN could be preferred when you need a clear single minima point, such as when there is pattern. While there are more local minima in MC-CNN, the overall disparity is still higher than SSD and SAD with just a single point going much lower (the matching point), leading me to believe it is more robust.

Show the disparity map comparison between SAD and MCCNN

SAD Disparity MCCNN



Qualitatively, between SAD and MCCNN, which disparity map do you think looks better? Explain your reasoning.

I think that MCCNN looks better because it is less blotchy, although it is more spotty, making it look more defined like a bicycle even with the spotted noisiness. Also, from intuition of the true image itself and qualitative comparison to the ground truth disparity, it looks more accurate such that disparity values are more closely correlated with depth (higher disparity correlated with closer to camera), whereas the SAD disparity map (left) has random red blotched on the top and bottom.

Show the quantitative evaluation between SAD and MCCNN (average error, etc.)

Metrics:	SAD	MCCNN
Average Error	23.715473	19.027502
%error > 1 pixel	95.35304052254153	93.56472897953095
%error > 2 pixels	90.88016388027576	87.6816748180588
%error > 4 pixels	81.29468614476446	77.27406525375869

Quantitatively, between SAD and MCCNN, which one achieve better score on the metrics? Why do you think that's the case?

MCCNN achieved better scores across the board. I think this is because it excelled in areas of the image a disparity error profile could show multiple low local minima, because as we saw earlier, the MCCNN disparity error profile had a higher overall disparity with a singular very low point of disparity (the matched region).

Extra Credit

<Discuss What extra credit you did and analyze it. Include Images of results as well >