CS 4476 Project 4

Jun Chen
jchen706@gatech.edu
903281776

Part 1

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

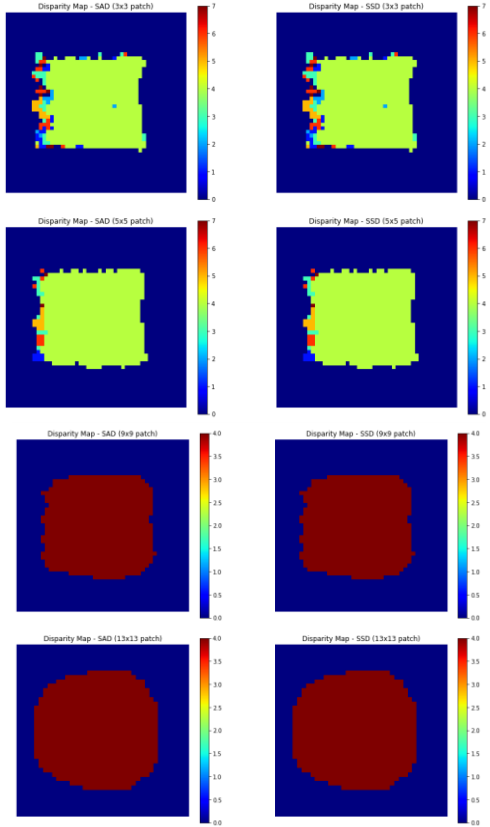
If you know the disparity of an object, you can compute the depth (distance from object to the camera by (focal length * distance between the camera center) / (disparity).

So depth = ((focal length * distance between the camera center) / (disparity).

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

Possible to see the depth as if these stereogram are sort of like a labyrinth matrix because of the thickness of the black and white dots cause eye deception of depthness in these images.

Random dot stereogram disparity maps



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

Increaseing the block size cause a large computed disparity value which causing the disparity maps to have all red. The reason is that larger block size deals with more features and values and larger horizontal, so the disparity will increase as we are summing all the disparities in the block making the object closer to camera.

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

Left edge result is poor because as the pixel on one image is shifted thus not part of the other image, so there is no possible disparity value, so the result is really poor. Another reason is that we are finding the first match in the lowest minimum error which can be incorrect because of similar patches as we are shifting left in the right image.

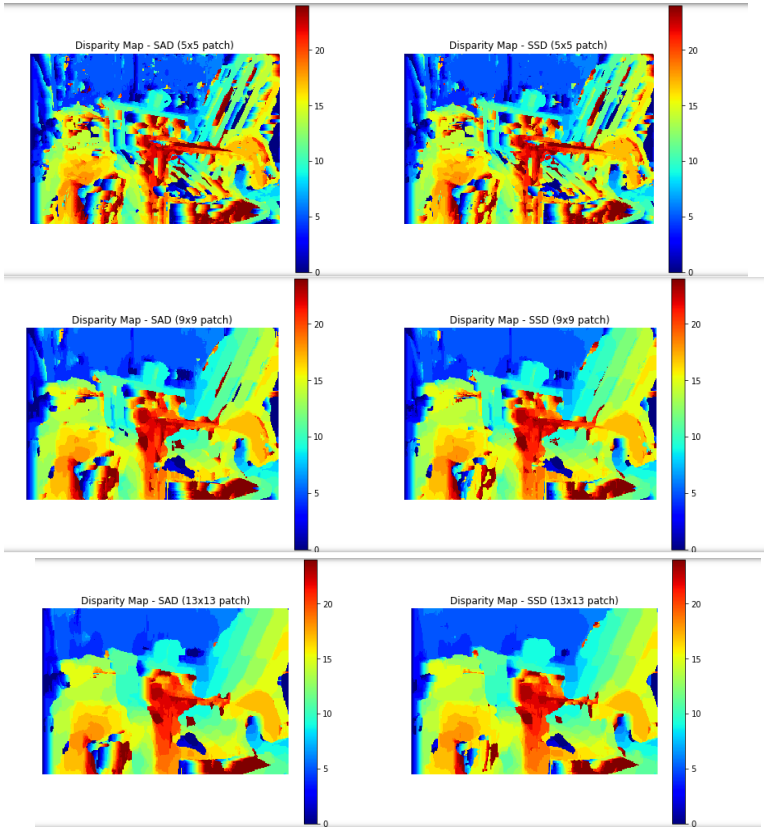
Convex error profile: Can you generalize the type of regions which will generate convex profiles?

Regions that are unique will generate a convex error profile because only itself would match not the neighboring images.

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

Regions surround a patch will have similar matches which will cause nonconvex error.

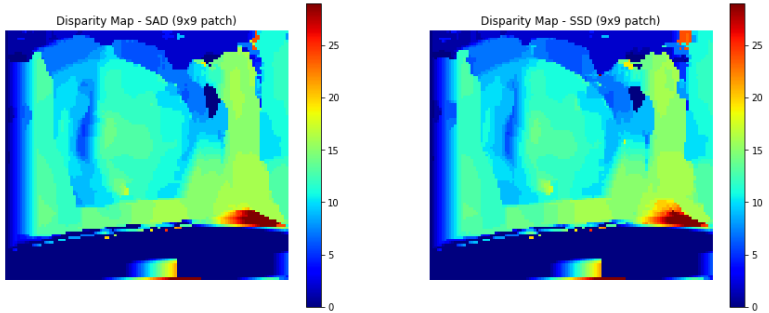
Disparity map for Set 1 (for 3 patch sizes)



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

The blocky comes from the shadows of the objects or angles of objects where distance to the camera changes which are hard to decipher the depth from different angles of the camera centers making pixels hard to differentiate for the patches.

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?

I see that the there is a large disparity at the bottom of the right bowling bin. I think this is because the right image is shifted with new features that are not in the left image causing a large disparity making that part of the image very close to camera. Also the patches for the right bowling pin are very similar to surrounding patches, but not accurately matching, thus causing a irregular bowling pin shape.

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

The front part of the image which is a table or cover is turning from color of the bowling pins to white which was photoshopped. The front part of the image had almost no disparity after the change. If all the pixels in the image where the same but lighting or contrast differ, the disparity values won't be affected by depth.

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

Increase block size allows less blocky in the disparities, the sides of the stairs become more linear in disparity values. In the 3X3, we see disparities going from red to blue in larger quantities than in the 7X7 patch.

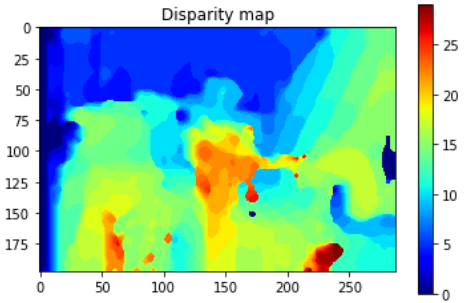
Set 6: Gradual shift in disparity values on the wall

I think the contrast of the wall or the lightning of the wall allows us to see the disparities on the wall. If the wall was just one color flat, then you cannot determine the disparity of the wall.

Smoothing

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

The back of the chair is smooth into a more linear gradient disparity. The blue cloth with books on top section is smooth out rather than having a blocky section facing the camera in the image without the smoothing. The chair arms are less blocky as the sudden blue under the chair arms have decrease in size.



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

The smoothing performs on sections with good lighting in the picture. Patches that are similar smooths out pretty well. The front of chair and back of the chair had nice smoothing because disparity were similar to their neighbors.The background smooth out well because of the similar brightness. In addition, our patches are square, objects can be rotated which will cause large error in smoothing.

Smoothing(contd.)

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

Smoothing will perform worse at section of the image where there are different brightness in the image. Patches that are shadows and rotated can cause bad smoothing effect. These areas have great contrast in similarity to surrounding patches even though they are close to the camera, the smoothing will think that these are negative patches.

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

The semi-global block matching constraints are only 1 dimension so we can only do horizontal shift not both vertical and horizontal shifts.

Part 2

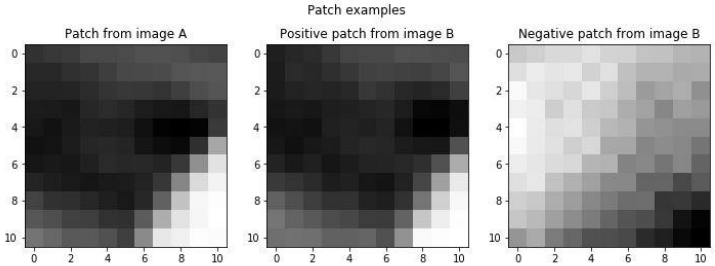
Copy the output of print(net_tr) from the notebook here. You can use screenshot instead of text if that's easier.

```
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?

It's an activation function which filters out data that is not useful. It sets where on layer ends and a new layers begin so we add it after every conv and fc layers. The Relu thresholds negative values to zeros and non-negative values to its output. The Relu is nonlinear because of its thresholding.

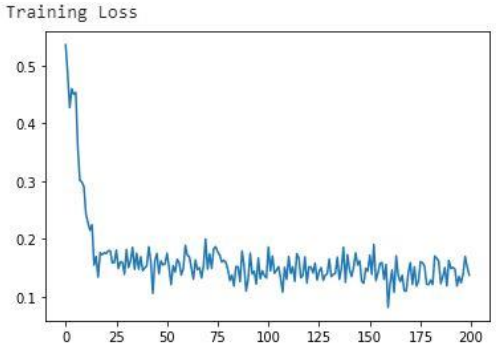
Show the patch example from your gen_patch function.



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

We can extract a pair of patches that have same center and same disparity. The negative patch can be obtain from a offset from the center of the positive patches so the negative patch is not center on the positive patch and does not match the positive patches is disparity.

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




```
Final average validation loss: 0.21055686
```

Window size 15

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

Both the very large and very small learning rate cause the training loss to fluctuation between two values. The large learning rate cause the training loss to fluctuate between 6 and 8 while the small value learning rate cause the training loss to fluctuate between 0.46 and 0.58. The large learning rate is overshooting the minimum of the data, and the small learning is moving toward the minimum too slow and its fluctuating on a plateau.

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

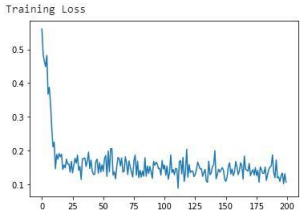
Adam is doing computation over all the weights. Each with it’s individual learning rate. SGD choosing a small subset or a random sample to perform computation and there is one learning rate for the dataset.

The Adam was faster to reach the plateau minimum and had smaller fluctuations at the plateau. The SGD had larger fluctuation when reach the plateau of minimum.

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?

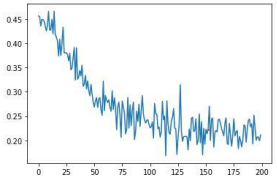
The final average validation loss is higher than the training loss because the we may be overfitting the data with not the optimal window seize or weights. We can get a better validation by increasing the window size by a little bit or train on more real world images to get a better model.

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



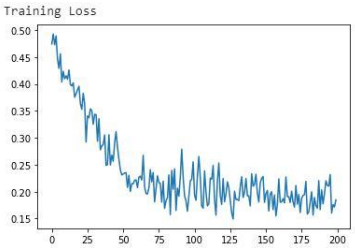
Final average validation loss: 0.24477014

Window size 11



Final average validation loss: 0.3048416

Window size 5



Final average validation loss: 0.2545141

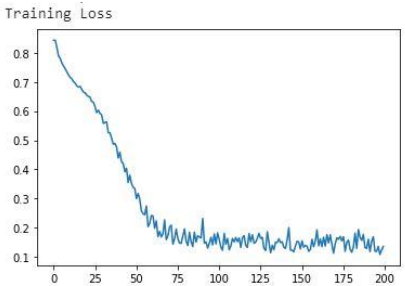
Window size 9

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.

Lowering the window size decrease the performance with increase of average validation loss. I don’t think there is a optimal window size for all image because different images have different amount of features, color gradients, brightness and depth which can affect the disparity values.

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

```
ExtendedNet(
  (net): Sequential(
    (0): 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()
    )
    (1): Sequential(
      (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()
)
Testing for MCNET: "Correct"
```

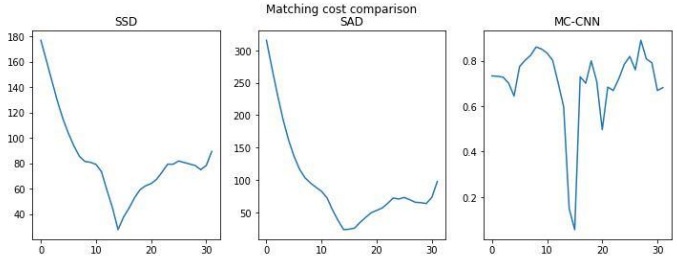



```
Final average validation loss: 0.24016154
```

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

I added a fully connected linear layer with a Relu layer. Adding this layer lower the final average validation loss from 0.24477014 to 0.24016154. Adding more layers allows for extraction of more features to a certain point. Then, without any new data, this the model will overfit leading to errors because of extra false positives.

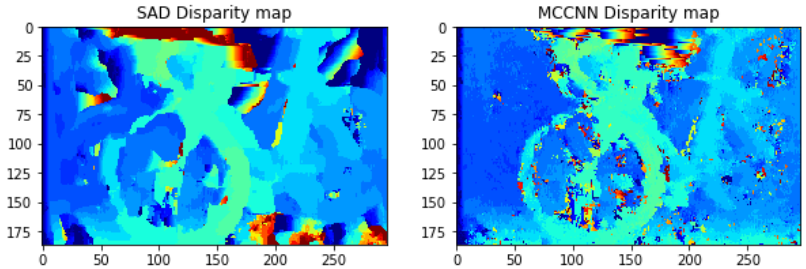
Show the matching cost comparison plot between SSD/SAD/MCCNN



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?

The SSD is the sum of square of the difference of the patches while SAD is the sum of absolute difference. The MC-CNN is a match cost convolution neural network with find the difference in image intensity of a position in the image to all the disparities under consideration in a patch. MC-CNN would work well in if the image has an incredible amount of similar patches in surrounding locations for a image.

Show the disparity map comparison between SAD and MCCNN



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

The SAD looks better qualitatively with smoother disparity gradients. The MCNN is blocky with a lot of noise in the image. The wheel of the bike on the MCCNN is very noisy due to the metal threads intersecting the middle of the wheel.

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

Metrics:	SAD	MCCNN
Average Error	23.70199	19.53587
%error > 1 pixel	95.35363988960991	93.93040948758112
%error > 2 pixels	90.88056345832135	88.02817557992094
%error > 4 pixels	81.29468614476446	77.58606911101876

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

MCCNN achieved a better score because SAD only sum of the absolute difference while MCCNN is learning from the CNN while also finding the difference in image intensity between two patches.

Extra Credit

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