CS 4476 Project 4

Kok Jian Yu
jkok7@gatech.edu
903550380

Part 1

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

Assuming we know the baseline of the camera taking the 2 image, we can calculate depth(z) of a point in the image by using the formula

$$depth(x, y) = \frac{focal * baseline}{disparity\_map(x, y)}$$

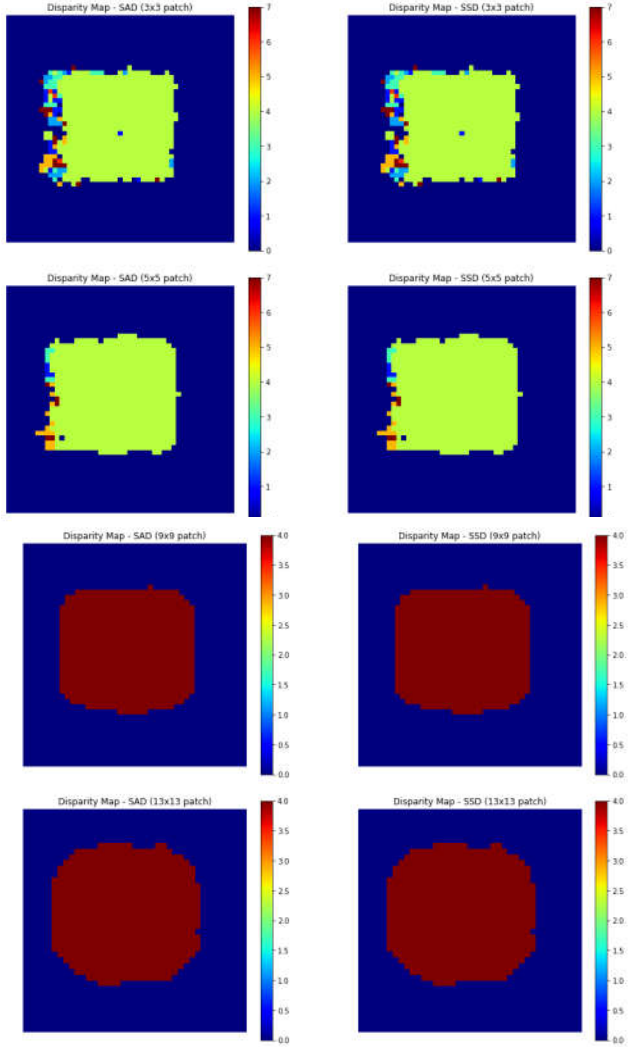
Random dot stereogram image [51x51x3] + Can you judge depth by looking them?

<copy plot here>

No. I cannot judge depth by looking.

Random dot stereogram disparity maps

<Insert all imag



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

The more you increase the block size, the more the disparity in an area becomes the same value. In a sense, the disparity maps become smoother.

This is because we are using a bigger block to calculate the disparity now. As such, the noise of the error due to coincidental matches is now reduced, and the disparity is now closer to the truth value.

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

This is because the shift of the image is to the left. When near the left edge, the error can vary greatly as the shift might make it such that some pixel is within the shift, and some pixel is outside the shift. As such, when the block_size is small, the coincidental match affects the result a lot and thus result is poor on the left edge.

Slightly outside the right edge, it will not have any matches(except coincidental) to any value on its left as the shift only starts from the start of the right edge. And when inside the right edge, there will be no noise as they will be able to find a disparity with a perfect match. As such, result is good.

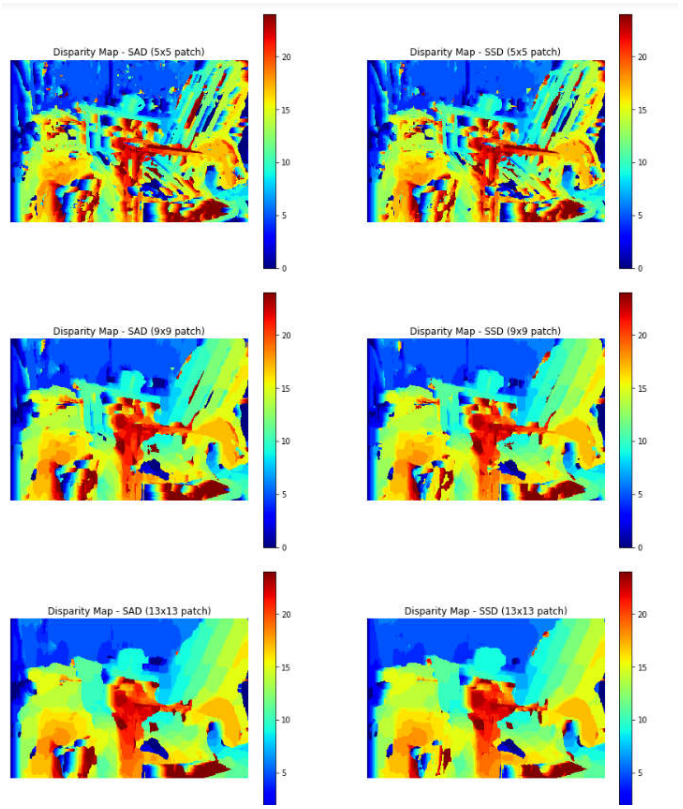
Convex error profile: Can you generalize the type of regions which will generate convex profiles?

Regions with extremely distinct features that only occur once in the image. For example a single red dot in a white image.

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

Regions that have a pattern that reoccur in the image multiple times.

Disparity map for Set 1 (for 3 patch sizes)

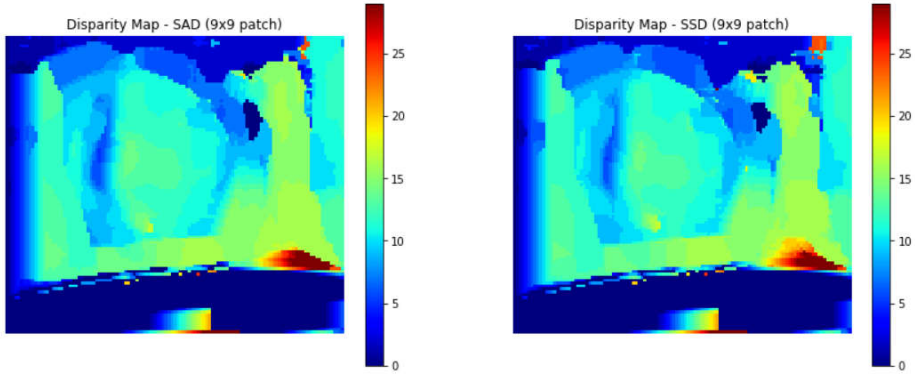


Set 1: Can you think of an explanation as to why the backrest of the chair appears blocky?

We are calculating the disparity of windows that is a square. As a result, the output will be appear blocky.

Set 4 disparity maps (only one patch size)

<Copy images here>



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?

There is an area near the right bowling pin head where the disparity map is red in color (high disparity).
This is to due to the coloured dots of the wall in the background behind occluded in the second image. This resulted in the disparity value in that area behind a lot higher than it actually is.

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 be related to the depth?

The bottom part of the image is whiter in set 4 compared to set 3.
This resulted in the disparity in the bottom of the image being close to 0.
As such, disparity calculation won't be related to depth in situations where the region looks exactly the same to its surrounds. For example, a white wall.

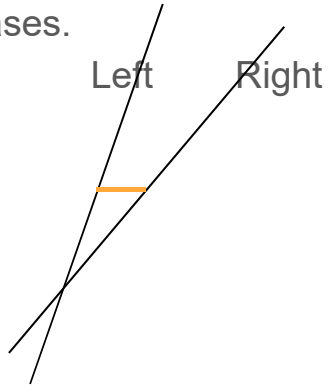
Set 6: Effect on block size on the stairs in disparity maps

The higher the block size, the smoother the disparity map appears to be at the edges of the objects.

Set 6: Gradual shift in disparity values on the wall

This is because the wall is taken at an angle. The wall that is further away will appear to have a “higher” shift due to the angles of the 2 image being longer.

Let the orange line represent the disparity value between left and right wall. As right wall goes further, disparity increases.



Smoothing

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

You can clearly see the chair outline is the output without smoothing, but you can't see it clearly in the output with smoothing.
This means that the disparity of the chair, and the objects around the chair are all relatively similar. Therefore, it looks to me that the output of the chair image after smoothing is better as it better represent the disparity of objects that do not have distinct outlines in the image better.

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

It performs better on regions of image where the object is larger. For example the box in the chair image. This is probably because it can smooth out the results of the entire region containing the box as most of it probably have similar disparity, instead of looking at every blocks individually. Which will probably make the result slightly more accurate.

Smoothing(contd.)

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

Regions where the objects are extremely small.

The values in these area will probably not appear much, as such the values might get smoothen out by its surroundings if the block size for the smoothing filter Is too large.

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

Assuming we know the direction of both shift, I believe that it will still work as we just have to calculate and smooth out both shift individually.

Part 2

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

<Text here>

```
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()
)
```

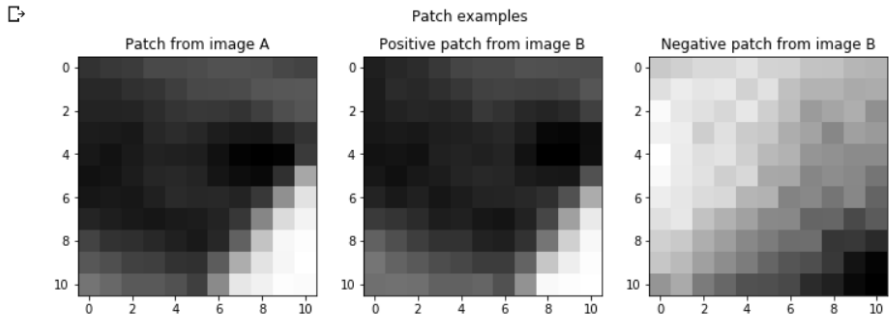
What's the purpose of ReLU and why do we add it after every conv and fc layers?

NN by itself without any activation function (ReLU in this case) will simply be a huge linear mapping from input to output.

ReLU is a non-linear activation function that will allow the NN to be a nonlinear mapping. Which will allow it to learn and solve more complex task.

Show the patch example from your gen_patch function.

<Copy figure image here>



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

Retrieve the actual disparity.

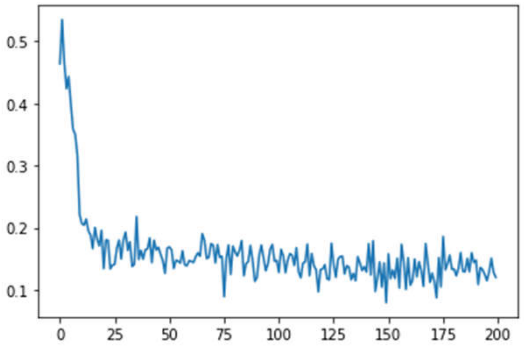
Create a disparity offset to be used for the negative patch that is a random value from 5 to 25.

To get positive patch, simply get the patch from image b from axis, with horizontal dimension minus by actual disparity.

To get negative patch, simply get the patch from image b from axis, with horizontal dimension minus by actual disparity and adding the disparity offset.

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

<Copy figure image here>




```
Final average validation loss: 0.22560543
```

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

Higher learning rate made the training loss plot have more variance (plot looks more unstable), but loss is reduced quickly over the epochs.

However, learning rate that is too high resulted in a situation where the training loss is constantly high, and never drops.

Lower learning rate made the plot have lower variance, but loss is reduced slower over the epochs.

Higher learning rate → Hard to hit minimum loss, values might change too much, resulting in a higher loss than before as it moves past the minimum point.

Lower learning rate → Easier to get to minimum loss, but trains slower as moves slowly.

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

<Text here>

SGD only have a single fix learning rate during training, while Adam uses 2 learning rates based on the average first moment and the average second moment of the gradients.

Training loss in SGD seems to drop at a slower rate compared to Adam when given the same learning rate

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?

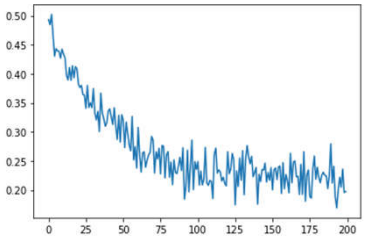
<Text here>

Higher. We trained on the training data, which means that it can probably predict the training data better compared to the test data.

To get a better validation loss, we need to allow the model to generalise better to the test data, and not overfit to the training data. One way to do this can be to reduce the learning rate (Not too much) to prevent overfitting.

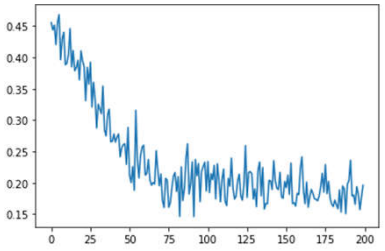
Copy the training loss plot and the final average validation loss of the networks with different window sizes

WS=5



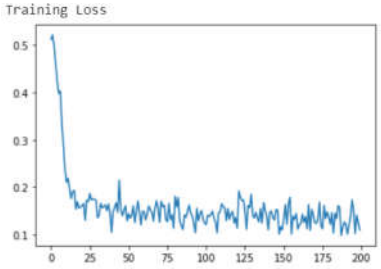
Final average validation loss: 0.30490804

WS=9



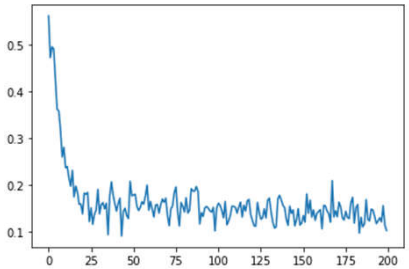
Final average validation loss: 0.25129825

WS=11



Final average validation loss: 0.23003954

WS=15



Final average validation loss: 0.22348274

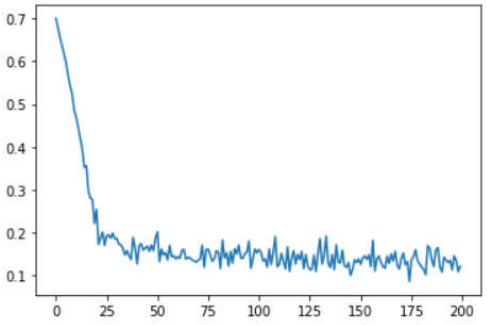
What's the effect of varying the window size on performance? Do you suppose there is an optimal window size for all images? Explain why or why not.

Increasing the window size seems to reduce the validation loss.

I think that there will be an optimal window size for all images. Looking at the learning rate for the 4 different sizes, it can be observed that the training loss decreases at different rate and plateau at different points. However, the training loss for 11 and 15 is relatively similar. As such, I believe that the optimal window size is around these 2 values as increasing window size doesn't improve the loss anymore.

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): 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=384, bias=True)
    (18): ReLU()
    (19): Linear(in_features=384, out_features=1, bias=True)
    (20): Sigmoid()
  )
  (criterion): BCELoss()
```



Final average validation loss: 0.22451021

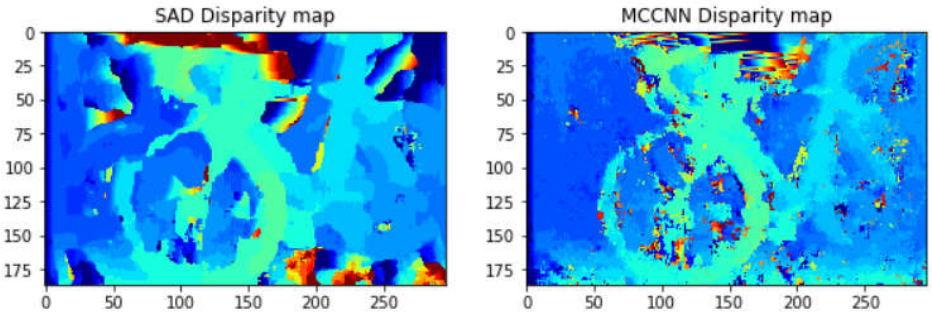
What layers did you add? What's the effect of adding more layers without adding more data? Why?

I added 2 FC layer with input and output size of 384. With a ReLU layer after each FC layer.

There seems to be not much effect due to adding more layers, just a slight drop in validation loss. This is probably because the network has already extracted most of the features that it possibly could from the image. As such, adding more layers will not be able to improve the network much further. In fact, it could possibly even lead to overfitting.

Show the disparity map comparison between SAD and MCCNN

<Copy figure image here>



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

I think SAD looks betters. While it might appear more chunky and the outline of the bicycle might not appear as distinctly as MCCNN, there is a lot less noise in SAD disparity map compared to MCCNN disparity map.

As such, I find SAD to be better looking.

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

<Text or screenshot here>

Metrics:	SAD	MCCNN
Average Error	23.70199	19.584524
%error > 1 pixel	95.35363988960991	93.93000990953553
%error > 2 pixels	90.88056345832135	88.04196102249358
%error > 4 pixels	81.29468614476446	77.7041444234888

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

MCCCN achieved better score on the metrics.

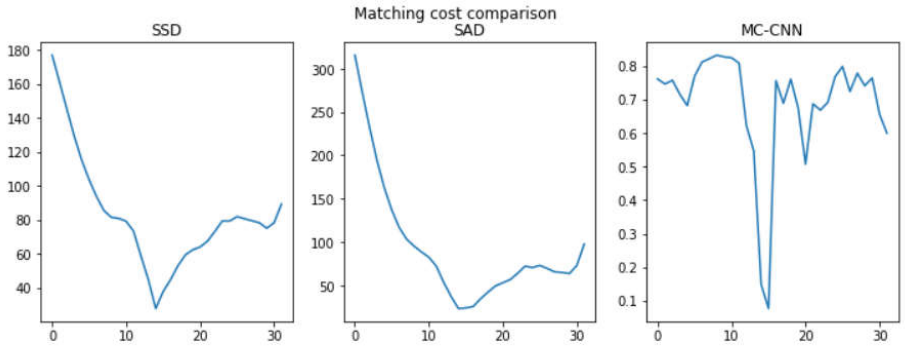
This is because of the way the matching cost is created. For SAD, it has a smoother curve towards to the lowest matching cost, while for MCCNN, it has a sharp descent to the lowest matching cost.

Therefore, for MCCNN, there is a higher chance it will get a more accurate value while for SAD, there is a chance it might get a value slightly off from the correct value.

This result in MCCNN having a better metrics.

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

<Copy figure image here>



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?

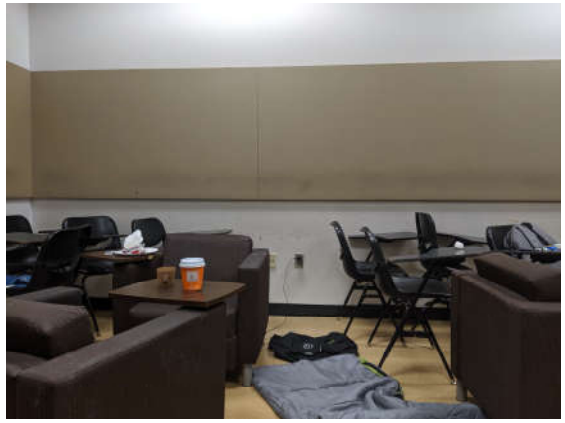
MC-CCN will be more preferred in a scenario where the objects in the image are relatively similar. This is because MC-CNN seems to be more sensitive compared to SSD and SAD. As such, it will be able to detect between the similar objects better compared to SAD and SSD.

Extra Credit

I took 2 images with a slight horizontal shift and tried the smoothing code.

I thought the image turned out quite good. The background area have a smaller disparity compared to the objects in front. This is correct as the objects in front have a higher disparity. I think this image is easy as it is a very consistent horizontal shift with minimal vertical shift.

Left



Right