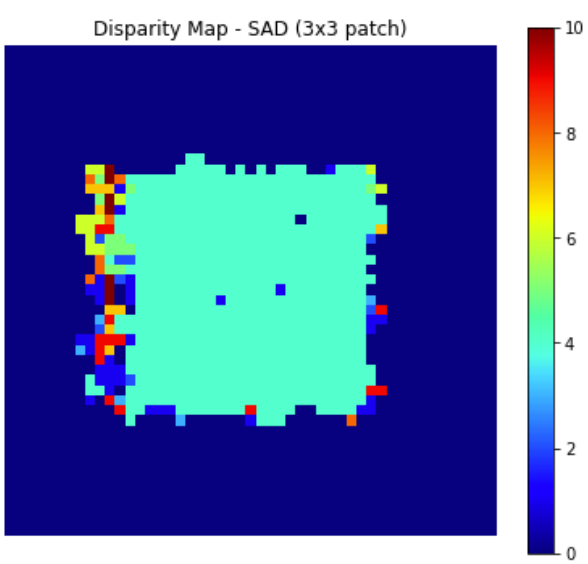
CS 6476 Project 4

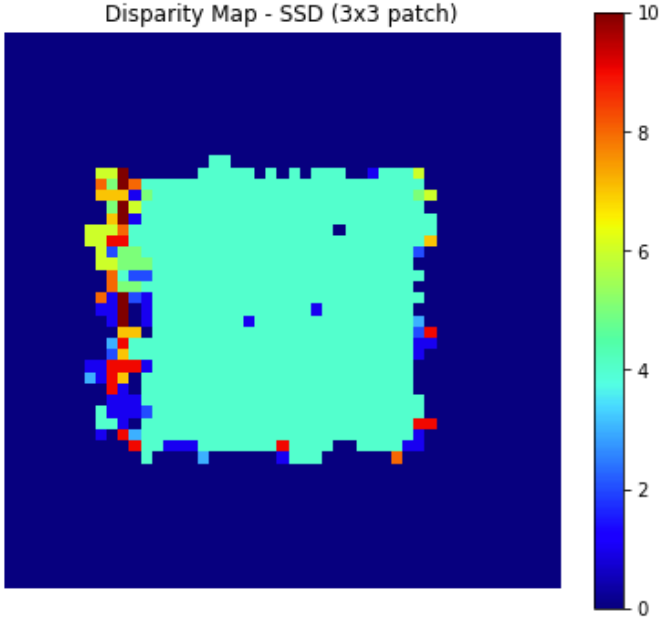
Bojun Yang
byang301@gatech.edu
byang301
903254309

Part 1: Simple stereo by matching patches

[insert visualization of random dot stereogram disparity map using 3x3 blocks with SAD]

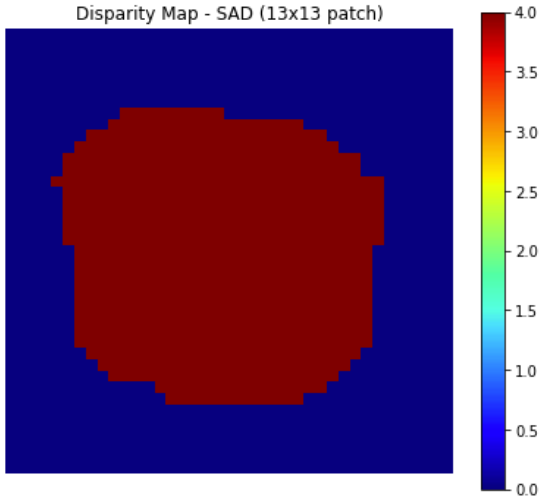


[insert visualization of random dot stereogram disparity map using 3x3 blocks with SSD]

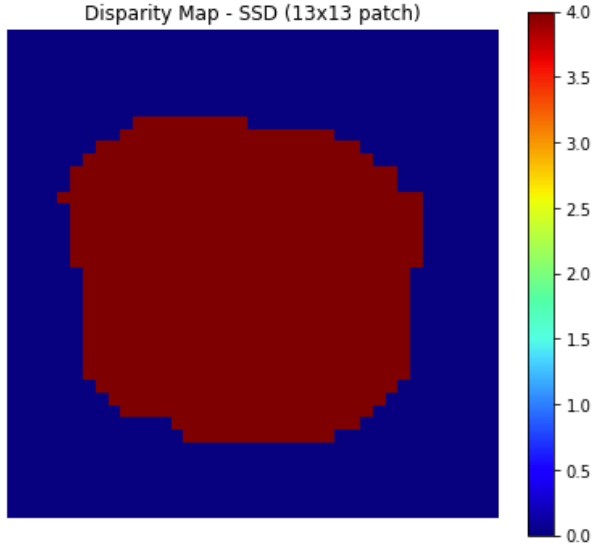


Part 1: Simple stereo by matching patches

[insert visualization of random dot stereogram disparity map using 13x13 blocks with SAD]



[insert visualization of random dot stereogram disparity map using 13x13 blocks with SSD]



Part 1: Simple stereo by matching patches

Increasing the blocksize gives a smoother disparity map around the edges of the shift. This is because a larger blocksize have a larger ratio of matches vs non-matches up to a certain extent.

It is poor around the left edge because a patch in the left image on the edge will see the same edge at a certain disparity in the right image. This is because we shifted left. When we shifted left, we filled the empty space with random bits which is very different from the right edge of the center square, thus giving us clear disparity.

Part 1: Simple stereo by matching patches

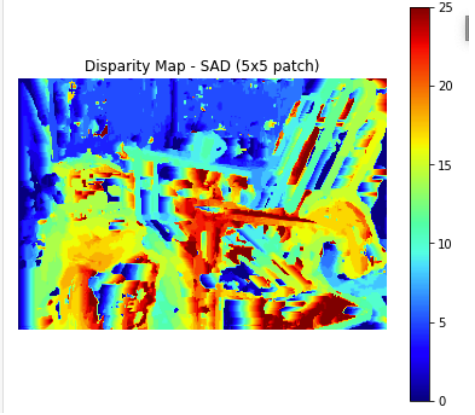
[What kind of regions will produce convex error profiles?]

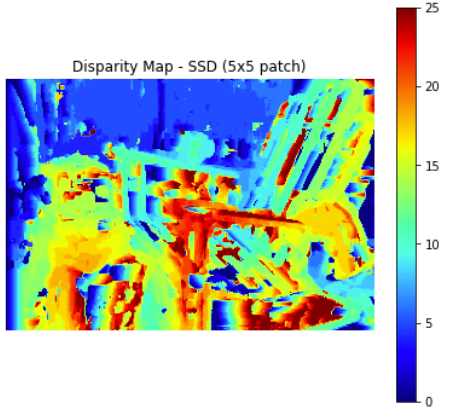
Non repetitive regions (along the horizontal) that have unique characteristics or have different pixel values.

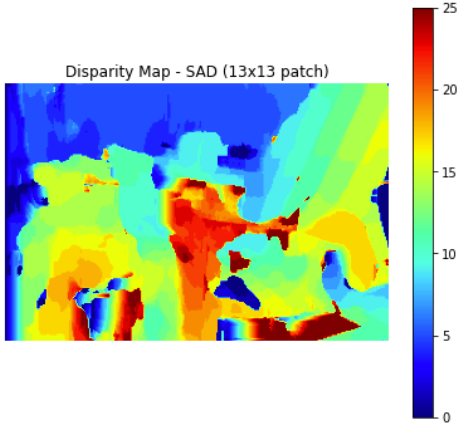
[What kind of regions will produce non-convex error profiles?]

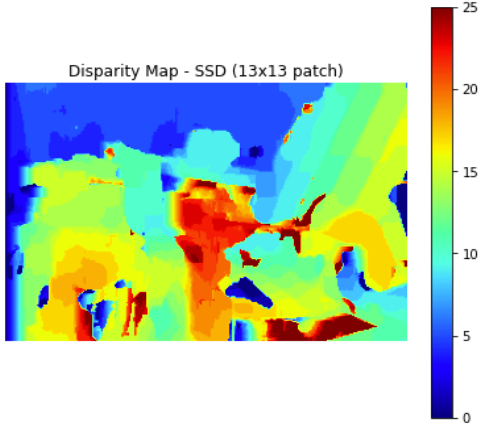
Repetitive regions that when looked at through a small window, looks similar to other regions. An example would be uniform stripes. In chair, it was the gaps between the wood planks.

Part 1: Simple stereo by matching patches

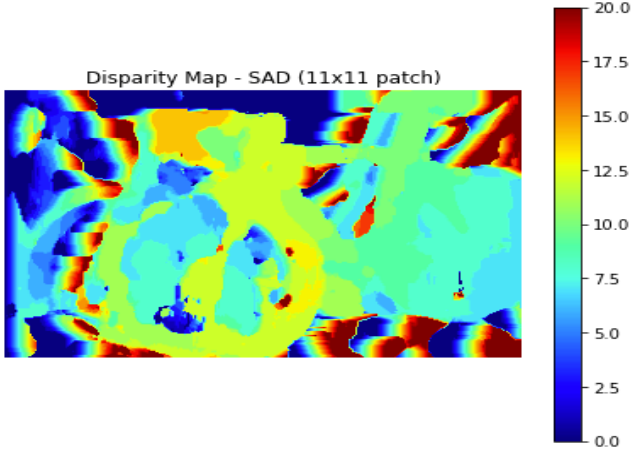


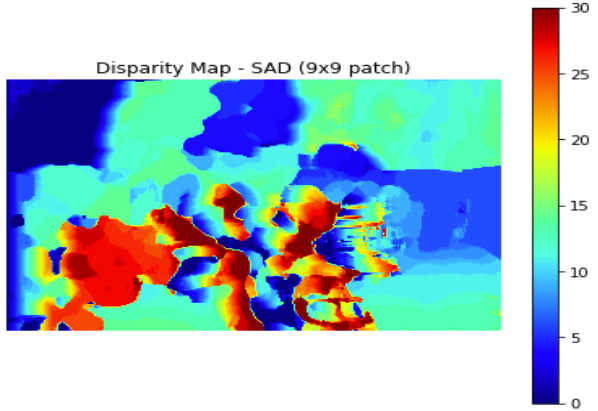


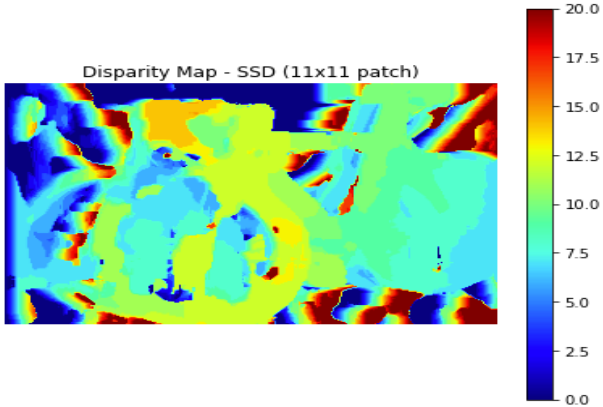


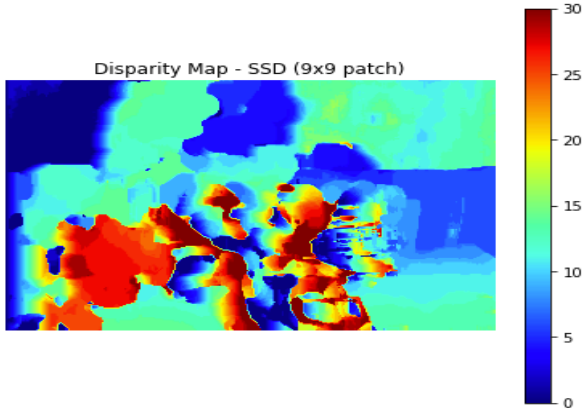


Part 1: Simple stereo by matching patches









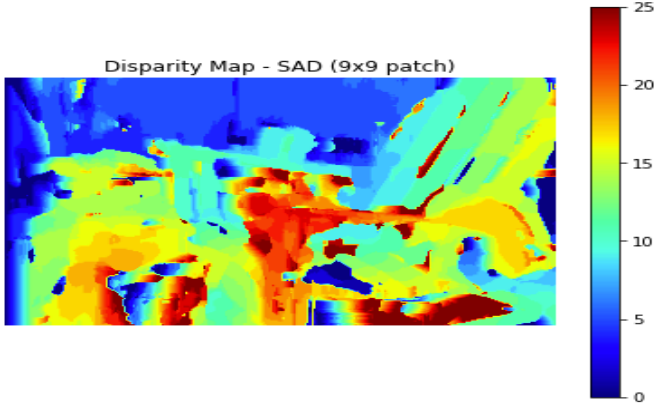
Part 1: Simple stereo by matching patches

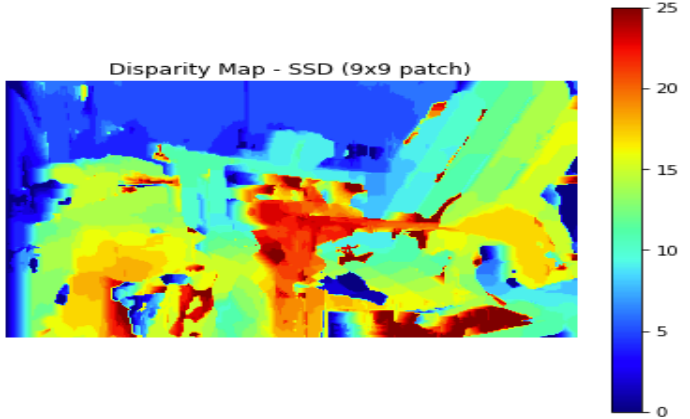
For each pixel, I calculated the disparity by looping through all with-in-bounds patches and calculating the similarity error using the given function. I simply put these in an array where the index of the similarity error was the disparity (since we had disparity from 0 to max_search_bound). The argmin of this array is the disparity with the least similarity error.

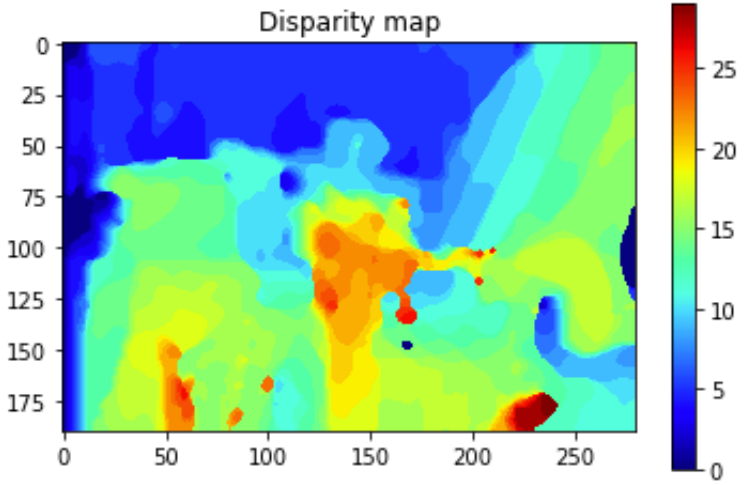
Since our disparity map uses a similarity function that is imperfect, we will get results poorer than the ground truth.

Because we are limited to a square window shape, irregular shapes that shift will always have some values that give high similarity error. Our similarity function is also imperfect.

Part 1: Simple stereo by matching patches







Part 1: Simple stereo by matching patches

Smoothing performs better on the chair, which is a large static structure. I think it does better because it is isolated from other objects and has a clear distinction between the chair and the background.

It performs worse on areas that have many multiple objects that shift, like the small table with books on top. The smoothing merged the objects into one and we can only see a vague outline.

Part 1: Simple stereo by matching patches

[Describe how semi-global matching works.]

Semi global matching performs line optimization using multiple directions and to calculate an aggregated cost using the disparities from each direction.

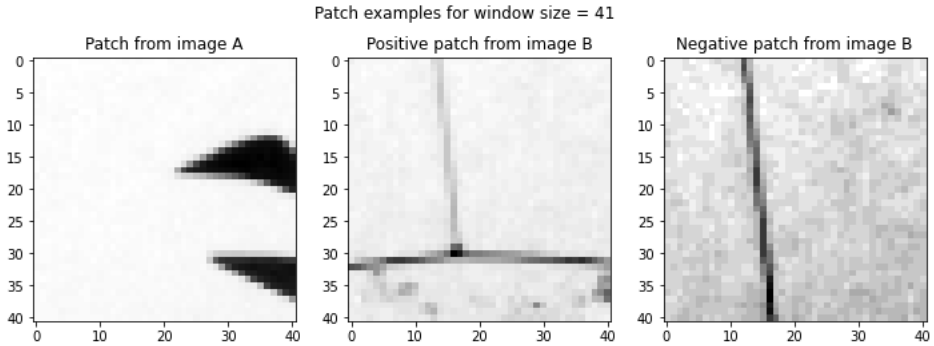
Yes, because SGM calculates the disparities in multiple different directions.

Part 2: Learning-based stereo matching

```
MCNET(
  (conv): 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()
  )
  (classifier): Sequential(
    (0): Linear(in_features=27104, out_features=384, bias=True)
    (1): ReLU()
    (2): Linear(in_features=384, out_features=384, bias=True)
    (3): ReLU()
    (4): Linear(in_features=384, out_features=1, bias=True)
    (5): Sigmoid()
  )
  (criterion): BCELoss()
)
Testing for MCNET: "Correct"
```

ReLU activation function is 0 for $x < 0$ and x for $x > 0$. ReLU outputs the input directly if it is positive, otherwise it will be zero. We use it because we negative feature values have no meaning to us, and disparity values should be positive.

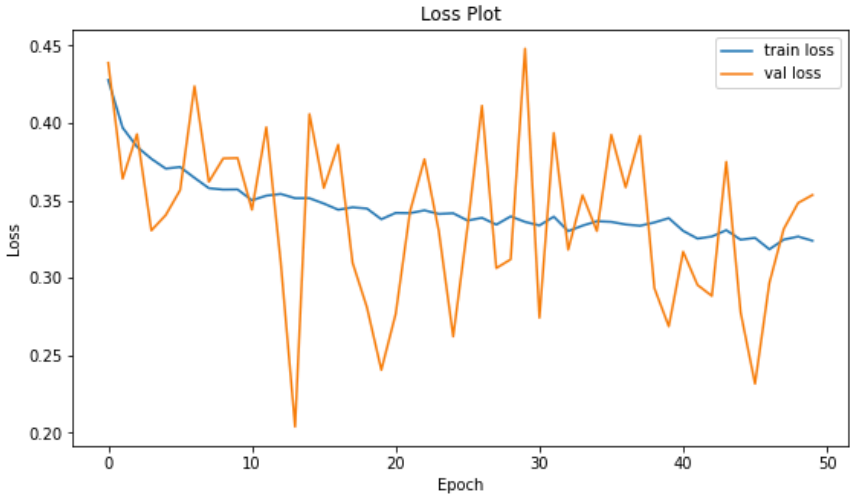
Part 2: Learning-based stereo matching

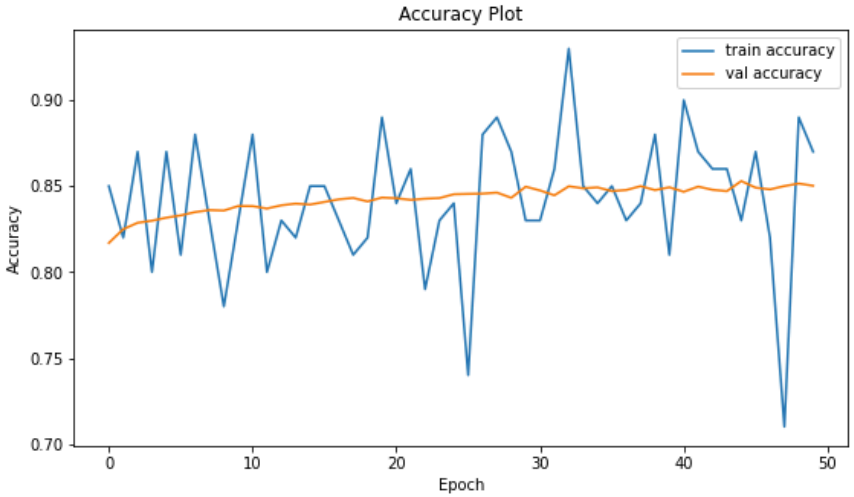


[Given a true disparity map for each stereo pair, how do we extract positive and negative patches for training?]

We extract positive and negative patches by using the true disparity for positive patch sand changing the true disparity by a random value for negative patches.

Part 2: Learning-based stereo matching



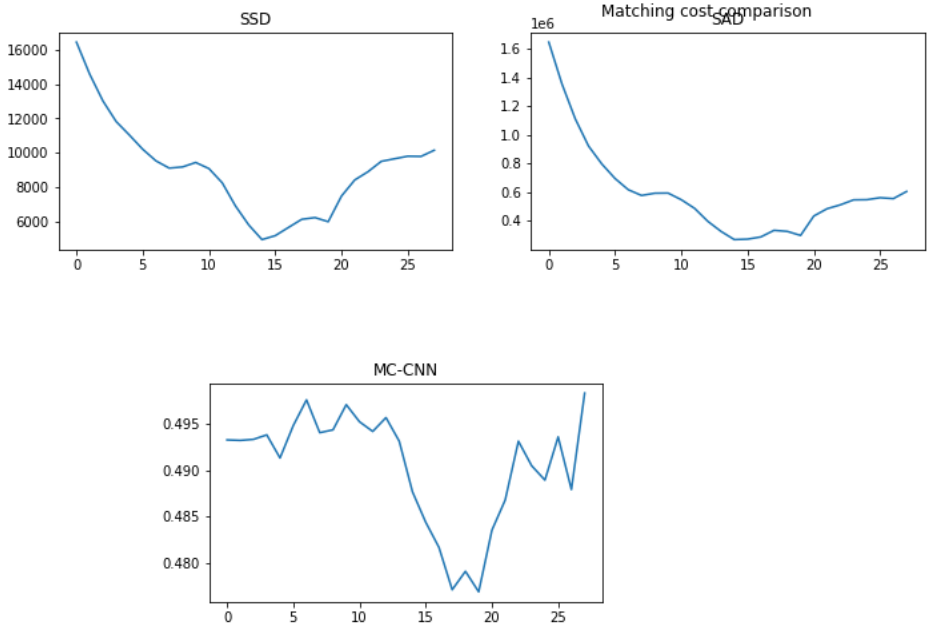


Part 2: Learning-based stereo matching

Neural nets are trained using gradient descent and the learning rate controls the size of each step of the descent. A small learning rate can cause the model to converge slowly and get stuck at local minimas. A too large learning rate can cause the model to be unstable, meaning the model overestimates and never converges to the optimal point. If a large learning is still stable, it can still cause the model to converge too quickly at a suboptimal solution.

A smaller window size takes longer to compute since there are more windows and features to consider. Window size can also affect accuracy. A small window could have false positives, thinking two patches are similar when they are not. A too large window can introduce less distinct similarity values, since a large window could always include something that did not shift. I do not think there is an optimal window size for all images because an optimal window size depends on the objects within the image. I think each image can have an optimal window size but that window size changes for different images.

Part 2: Learning-based stereo matching

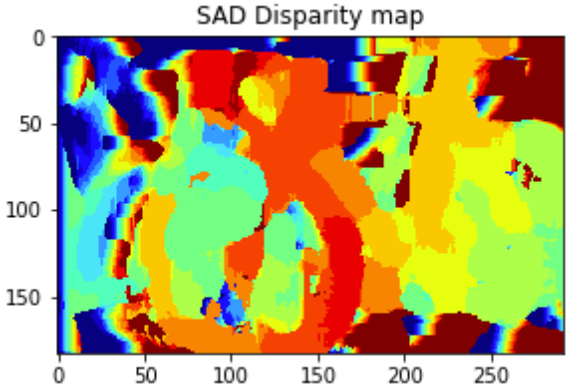


[Is MC-CNN or SAD/SSD better?]

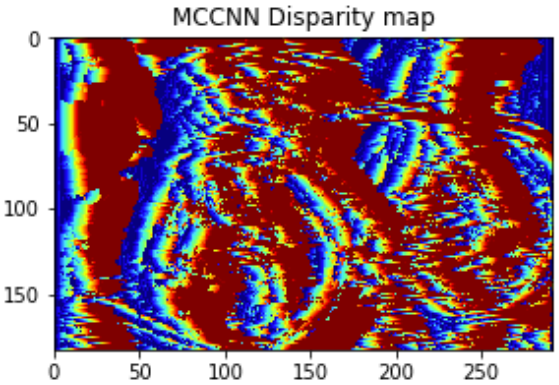
MC-CNN should be better in theory. My results were not as great but MC-CNN shows a distinct optimal solution at around 17. SSD and SAD are a bit more ambiguous.

Part 2: Learning-based stereo matching

[What techniques are discussed in the MC-CNN paper to augment the size of the training set?] The training patches are randomly rotated, scaled, and sheared. Their brightness and contrast are also changed.



sparity map



sparity map

Part 2: Learning-based stereo matching