Computer Vision Assignment 6

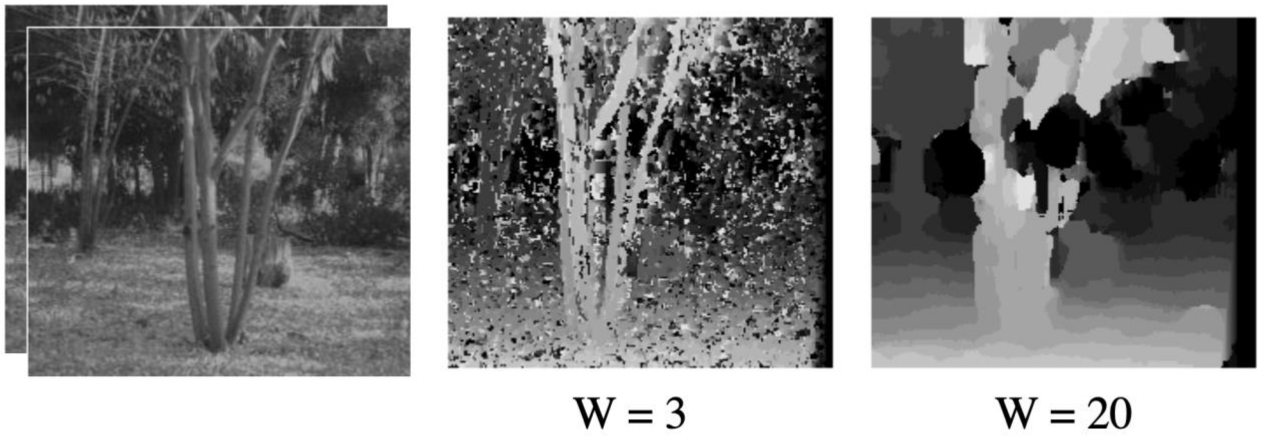
Kai-Po Lin (kl72); Department of Computer Science

Implementation

1.1 Solving Correspondence

(1) Choose a patch size and a similarity metric and provide a rationale for why you chose what you did.

⇨ I prefer to choose a patch size of 12 as my initial guess. The reason why is that on page 21 of the slide TwoViewGeometry-Stereo.pdf, the professor states that patch size with 3 is too small to choose and 20 is too big (refer to the following figure). **That is, I chose an average value of 12 as my patch size**. Moreover, I picked Sum of Absolute Difference (SAD) instead of Sum of Squared Difference (SSD) to match the two images since **I considered that SSD will enlarge the noise values to make the image messier**.



- Smaller window
  - + More detail
  - – More noise
- Larger window
  - + Smoother disparity maps
  - – Less detail

(2) Consider each of the following pixels pi=1,2,3,4,5,6 (k, m) at (row = k, col = m) of the left image and their associated patches:

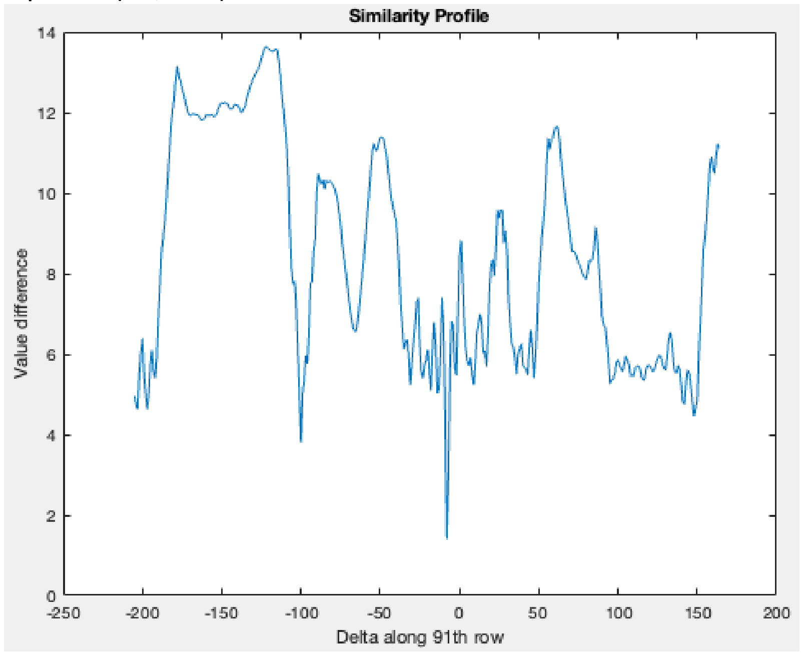
(a) p1 (136, 83)
(b) p2 (203, 304)

(c) p3 (182, 119)
(d) p4 (186, 160)
(e) p5 (123, 224)
(f) p6 (153, 338)

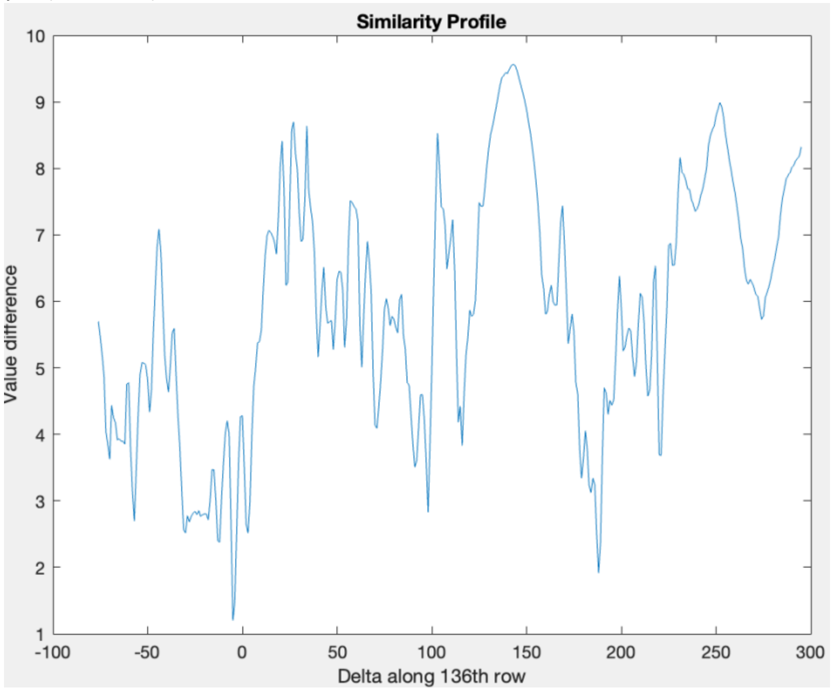
For each pixel pi (k, m) of the left image, calculate and plot the output (profile) of your similarity metric concerning disparity Δ to all the pixels along the kth row of the right image.

Using Patch Size = 12, SAD

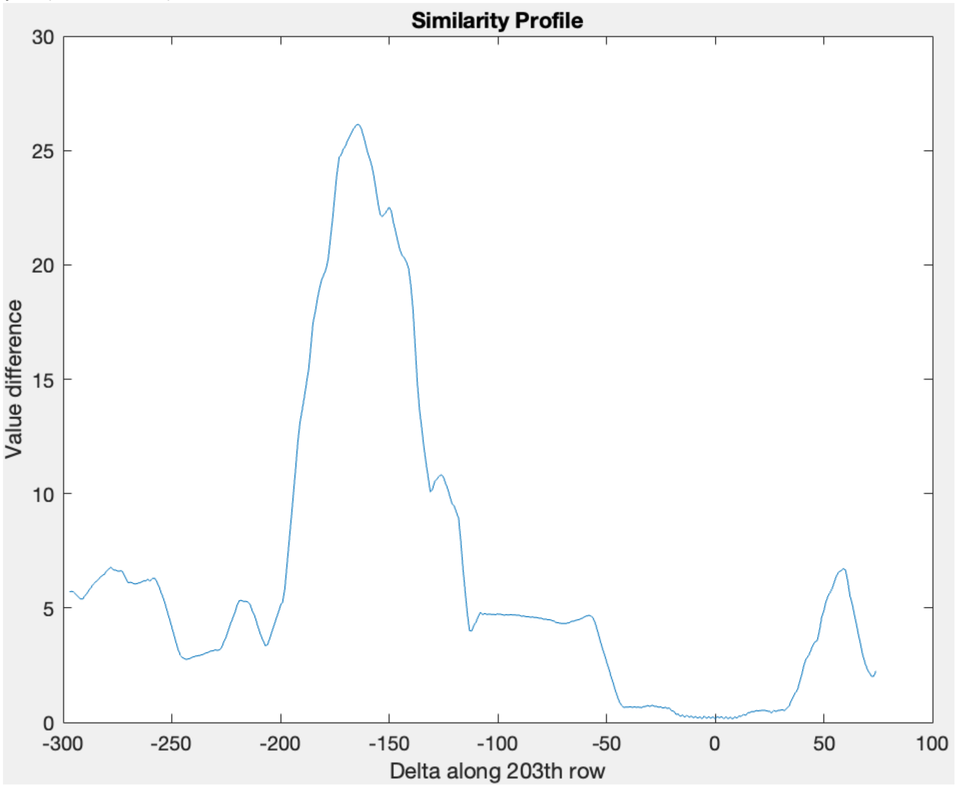
Example => (91, 213)



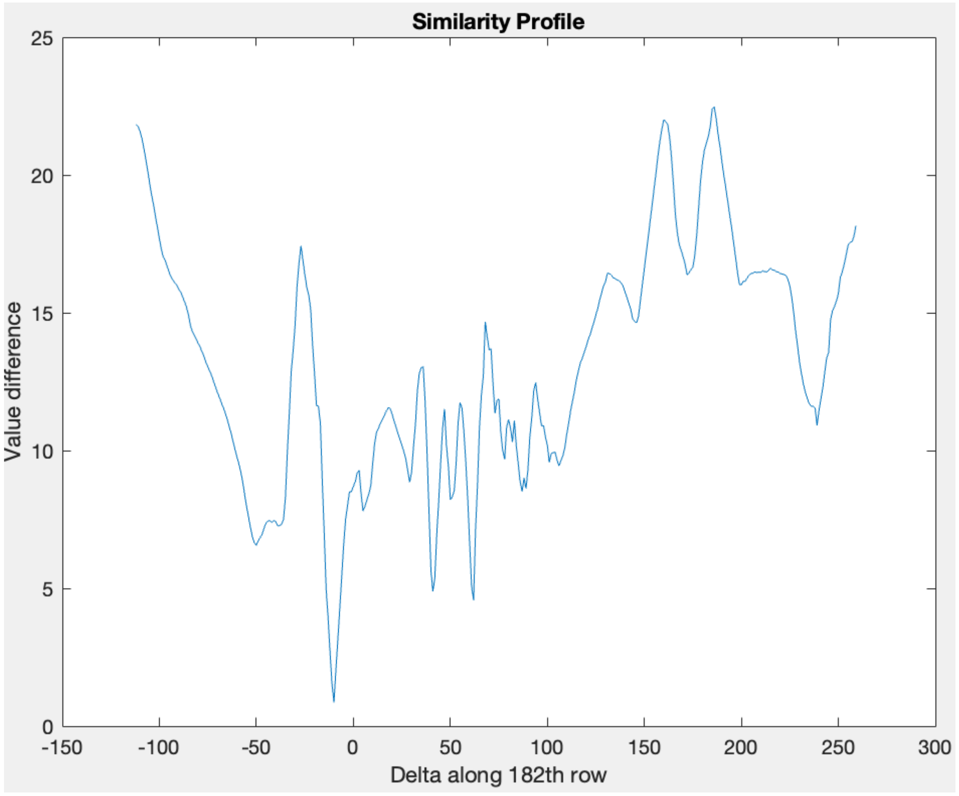
(a) p1 (136, 83)



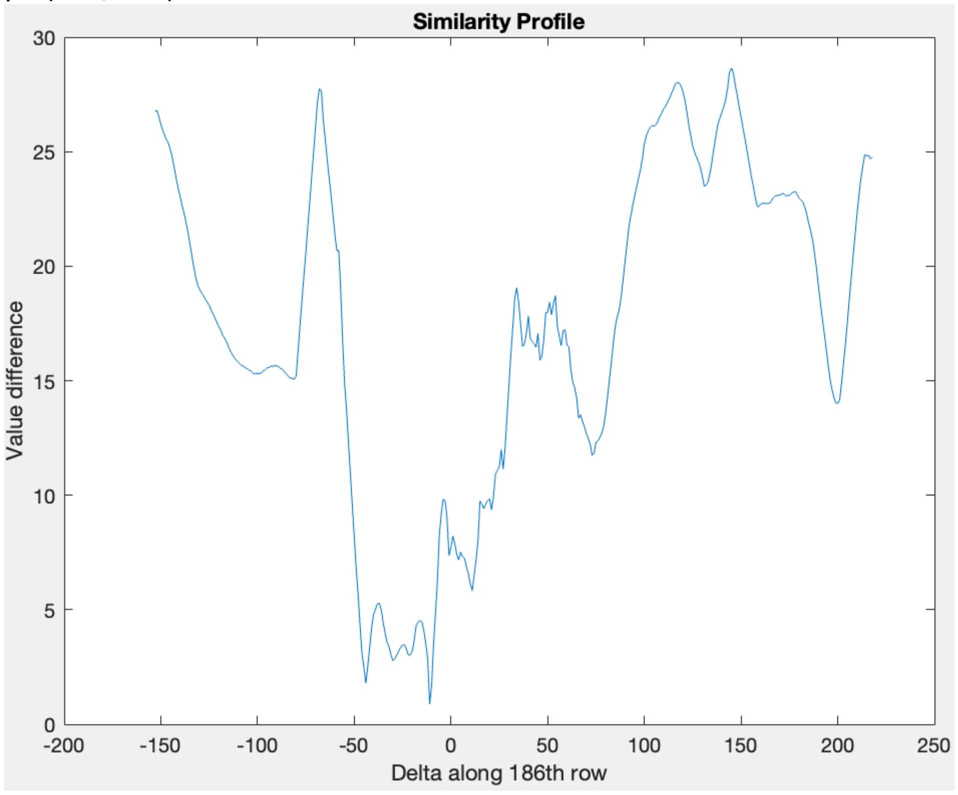
(b) p2 (203, 304)



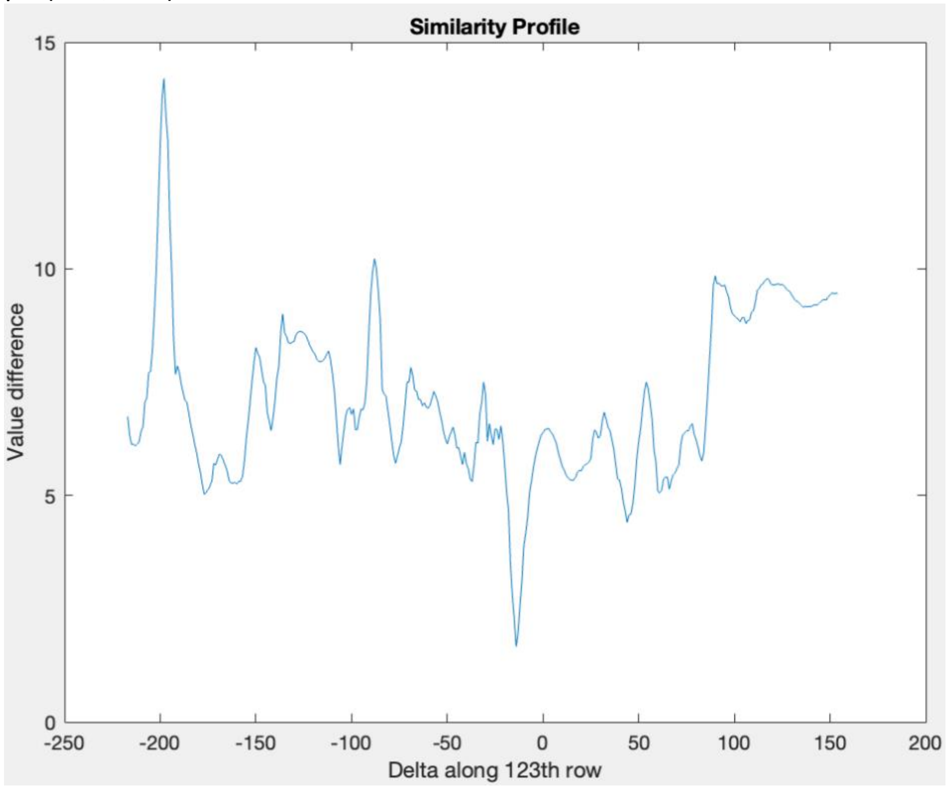
(c) p3 (182, 119)



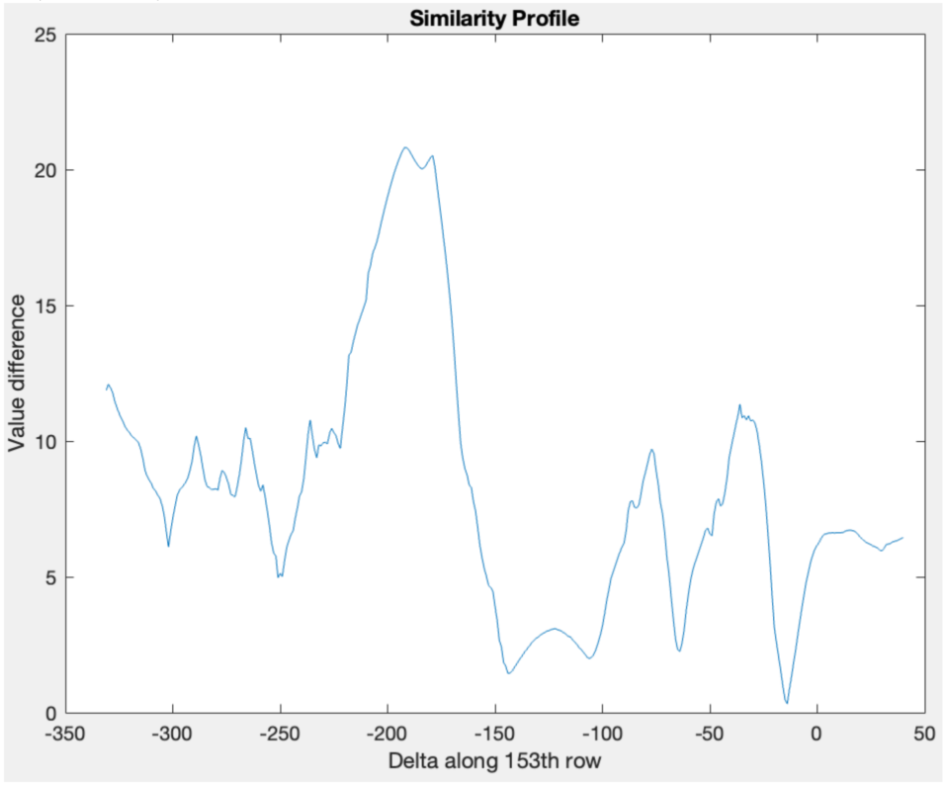
(d) p4 (186, 160)



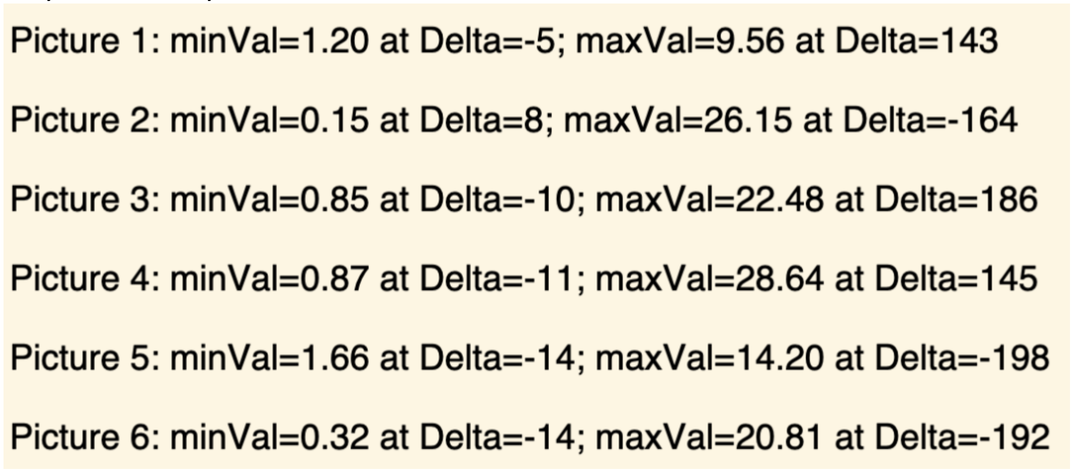
(e) p5 (123, 224)



(f) p6 (153, 338)



Interpret and give reasons for the profiles observed. Does your plot have a reliable extremum? If so, why? If not, why not?

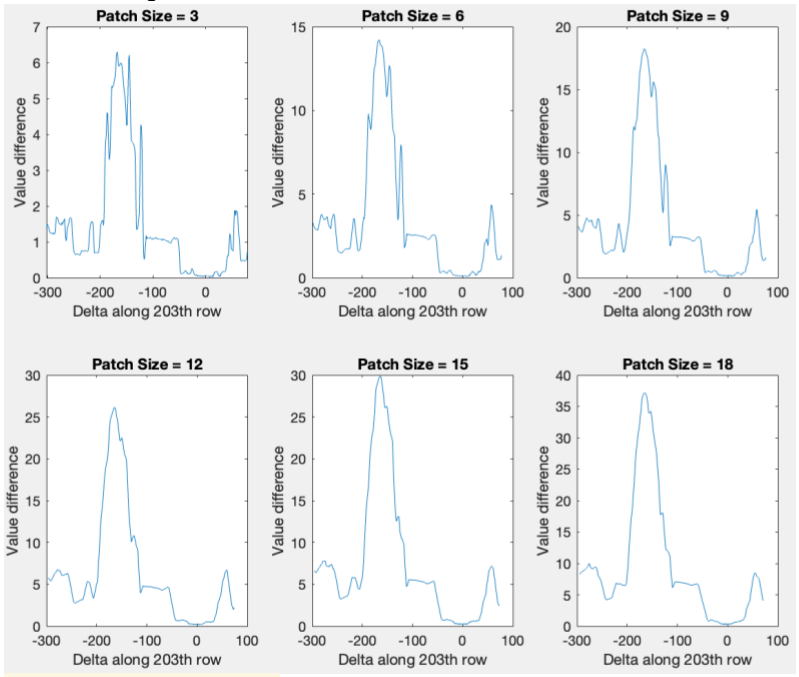


```
Picture 1: minVal=1.20 at Delta=-5; maxVal=9.56 at Delta=143

Picture 2: minVal=0.15 at Delta=8; maxVal=26.15 at Delta=-164

Picture 3: minVal=0.85 at Delta=-10; maxVal=22.48 at Delta=186

Picture 4: minVal=0.87 at Delta=-11; maxVal=28.64 at Delta=145

Picture 5: minVal=1.66 at Delta=-14; maxVal=14.20 at Delta=-198

Picture 6: minVal=0.32 at Delta=-14; maxVal=20.81 at Delta=-192
```

⇨ **I considered that the similarity profiles above have a clearly reliable extremum in most of the figures**. The only two unclear extremum values are the minimum value in p2 (row 203) and the maximum value in p4 (row 186). The neighbors in the minimum value of p2 (row 203) are like a basin, and there are lots of similar maximum values in p4 (row 186). **I think the patch size is reliable since there are 5/6 (83.3%) of the minimum delta is reasonable.** The standard I justify reasonable minimum deltas is that: **1. The minimum value of delta must <= 0 since the offset of the right image – the left image should always be negative or equal to zero. 2. The absolute delta value should be smaller than 24 since pixels with offset > 24 are likely to be matching on the wrong pixel (the offset is too large)**.

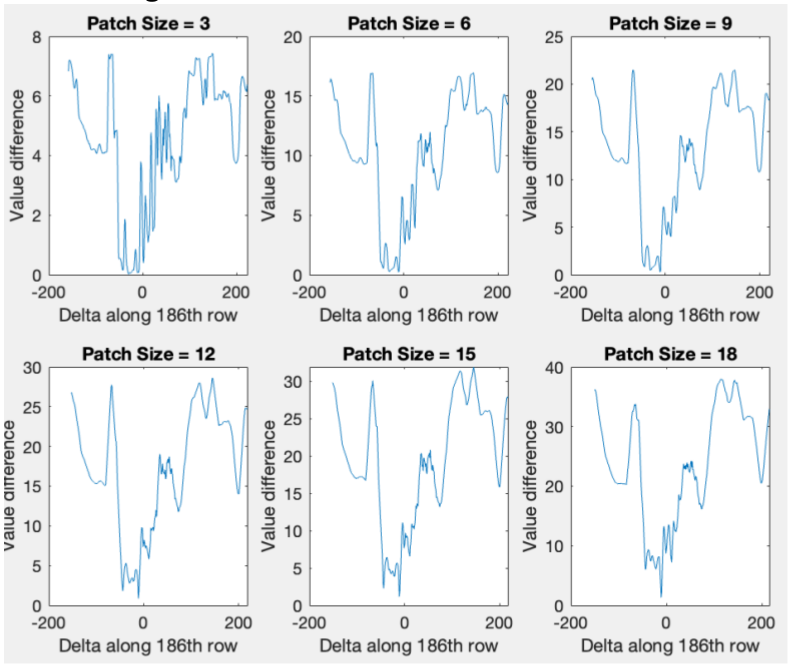
(3) From your observations of the similarity profiles, you just calculated, devise a strategy to estimate the most similar patch. Justify your choices. Include a test to reject patches that have no good matches and denote such problem pixels as undefined.

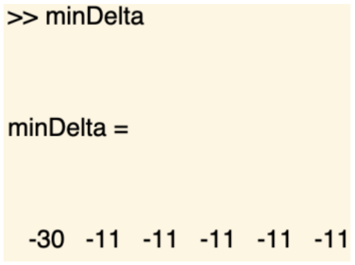
Delta Along 203rd Row



```
>> minDelta

minDelta =

     8     8     8     8    -6    -6
```

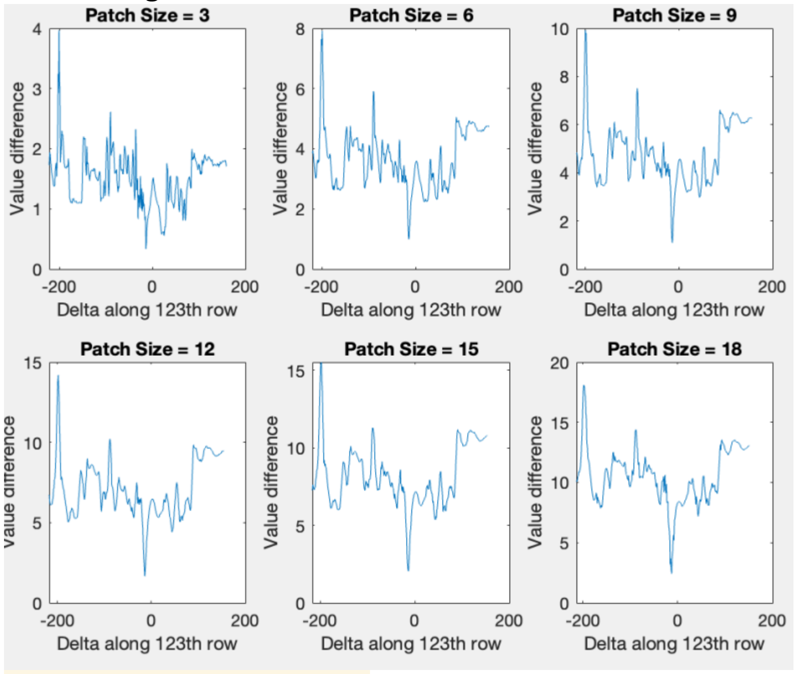
Delta Along 186th Row

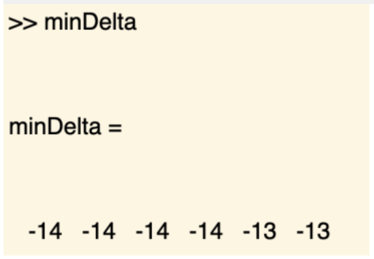




```
>> minDelta

minDelta =

  -30  -11  -11  -11  -11  -11
```

Delta Along 123rd Row





```
>> minDelta

minDelta =

  -14  -14  -14  -14  -13  -13
```

Strategy => I considered that patch size near 3~15 will be the best patch size according to the similarity profile above. The above figures are plotting the value difference of the same row (136th, 186th, 123rd row as examples) with different patch sizes. As we can see, **small patch sizes like 3, 6 will result in more complicated figure lines, and big patch sizes like 15, 18 will result in simple figure lines**.

Justify => According to the result of the min delta value, we could see that **the optimal patch size may be bigger than 3 and smaller than 15**. The reason why is that **in the result of rows 203 and 123, the min delta value starts to differ at patch size 3~15, and in the result of row 186, the min delta value starts to differ from patch size 3~6.**

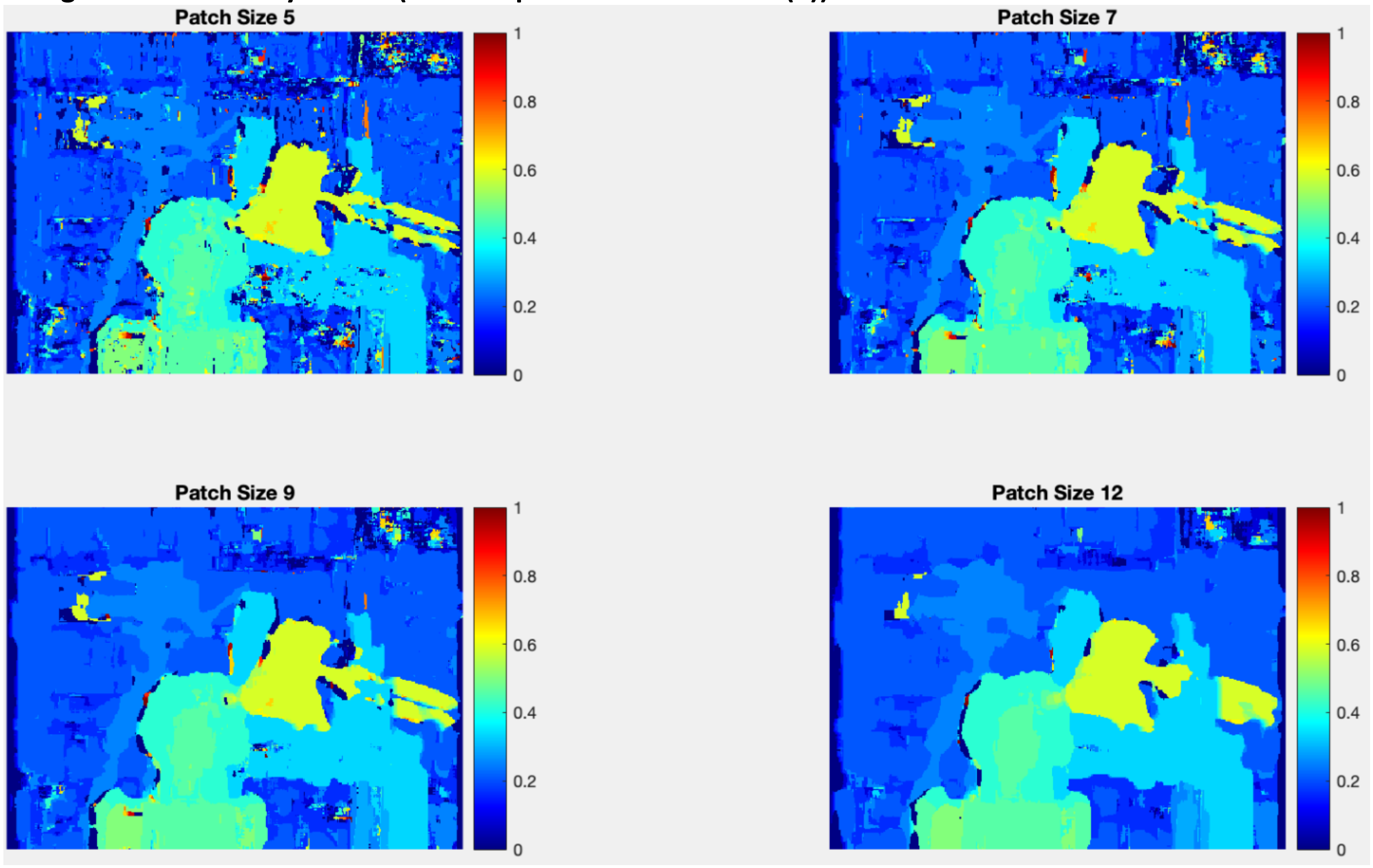
Reject patches that have no good matches, and denote such problem-pixels as undefined => My method of marking unmatched pixels is based on two strategies I've written in question (2):

1. The minimum value of delta must <= 0 since the offset of the right image – left image should always be negative or equal to zero.
2. The absolute delta value should be smaller than 24 since pixels with offset > 24 are likely to be matching on the wrong pixel (the offset is too large).

⇨ Then I will pick a patch size that has the smallest patch size but a similar number of qualified pixels.

(4) Repeat this calculation for all left image pixels. Find the point of maximal similarity in each case using your estimation strategy. The position of this estimate relative to the position of the left image pixel under consideration is the estimated disparity value, $\text{delta}_{\text{est}}$. Accumulate all these disparities (one for each left image pixel) into a matrix and display the matrix as a heatmap. This is the disparity map. Be sure to mark the undefined locations clearly in some way.

Using SAD as similarity metric (will compare SAD and SSD in (5))



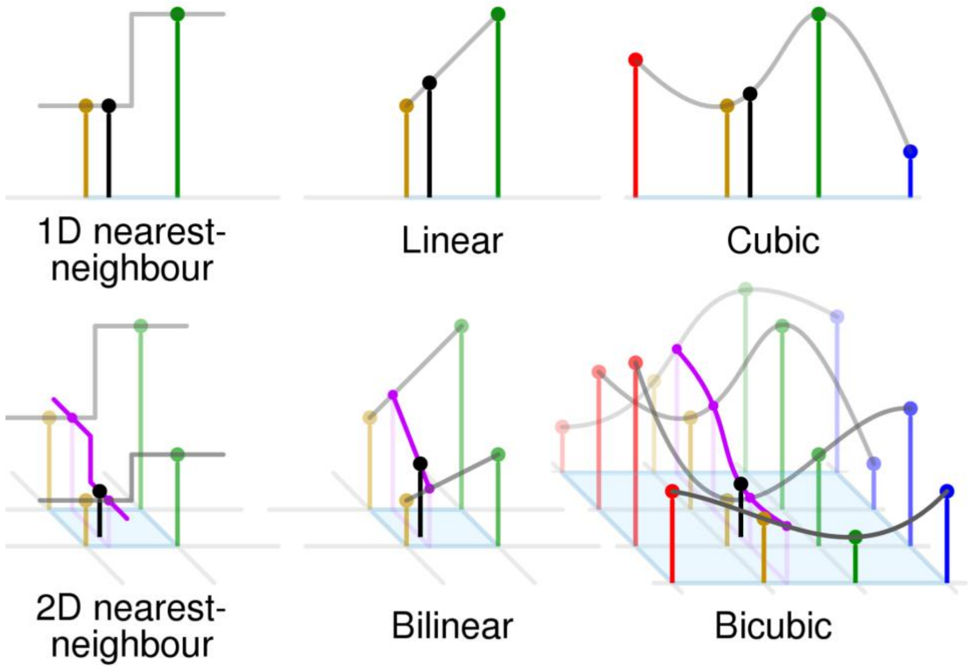
⇨ According to the figure above, we can see that if we **increase the size of the patch**, we will **obtain a smoother but less accurate and detailed result**. Conversely, if we **decrease the size of the patch**, we will have an **accurate and detailed result but with more noise in it**.

⇨ Considering the figure above, we can see that **the best setting for patch size is between 6~9** since a larger patch size (ex: 12) will distort the objects and the lower patch size (ex: 5) contain more noise in it.

⇨ Note: **The dark blue areas within the figure are the pixels I defined as problem-pixels**. I defined them by setting a threshold with a value of 24, and the delta value should be <= than 0 since (the position appears in the left image pixel) – (the position appears in the right image pixel) should always be negative.

(5) To fill in the undefined values, use an interpolation technique. There are a variety of possible options, including nearest-neighbor, linear, and cubic3. Choose one, describe how it works in your own words, and use it to estimate the unknown disparities. Provide a rationale for why you chose what you did. Display the resulting disparity map.

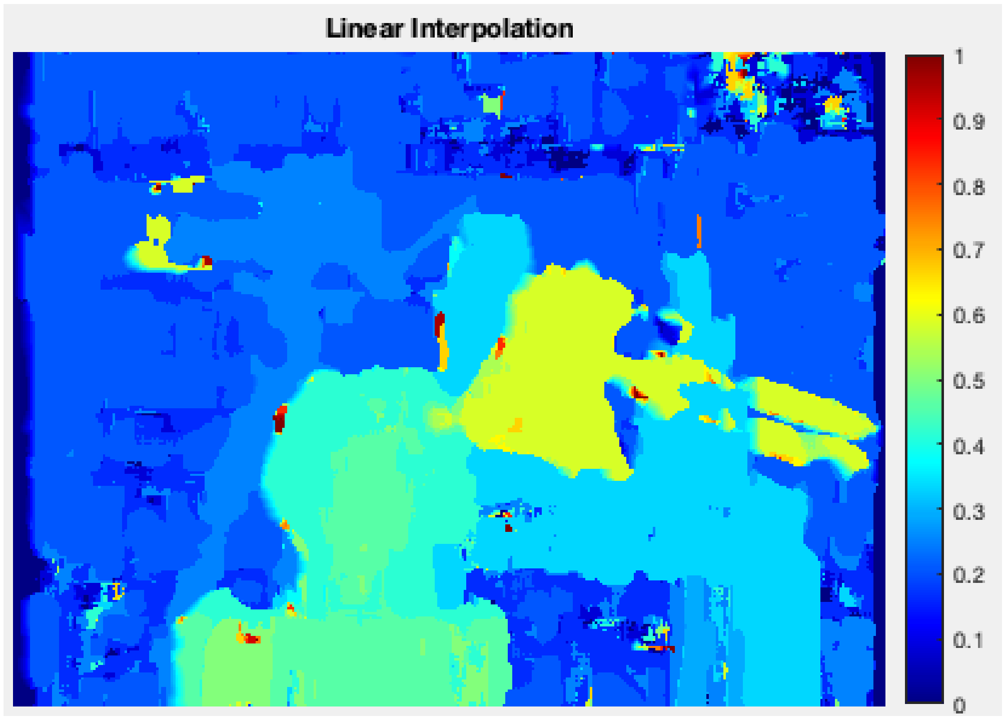
(a) There are a variety of possible options, including nearest-neighbor, linear, and cubic3. Choose one, describe how it works in your own words, and use it to estimate the unknown disparities. Provide a rationale for why you chose what you did. Display the resulting disparity map. (Cited from: http://paulbourke.net/miscellaneous/interpolation/, https://en.wikipedia.org/wiki/Nearest-neighbor_interpolation)



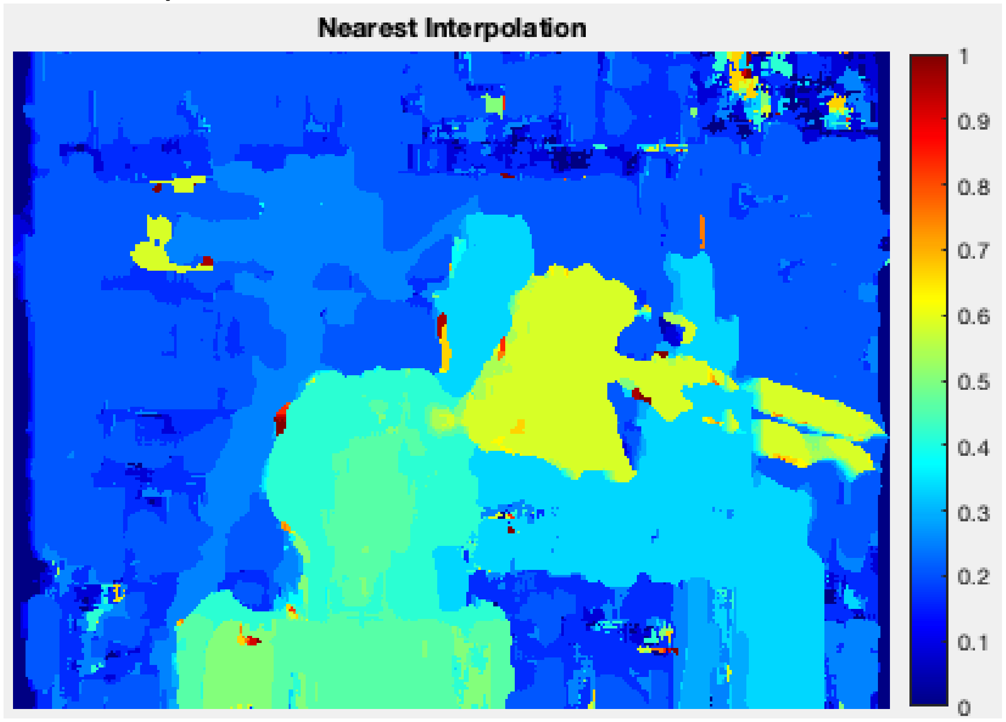
⇨ Overall, I considered **Cubic >= Linear > Nearest Neighbor** since **the result of cubic and linear interpolation performs better than the nearest-neighbor interpolation**. Considering that **Cubic interpolation is more flexible than Linear interpolation** because it contains more dimensions, **I will choose cubic interpolation to display the stereo matching**.

The description and the result of the three interpolation methods respectively =>

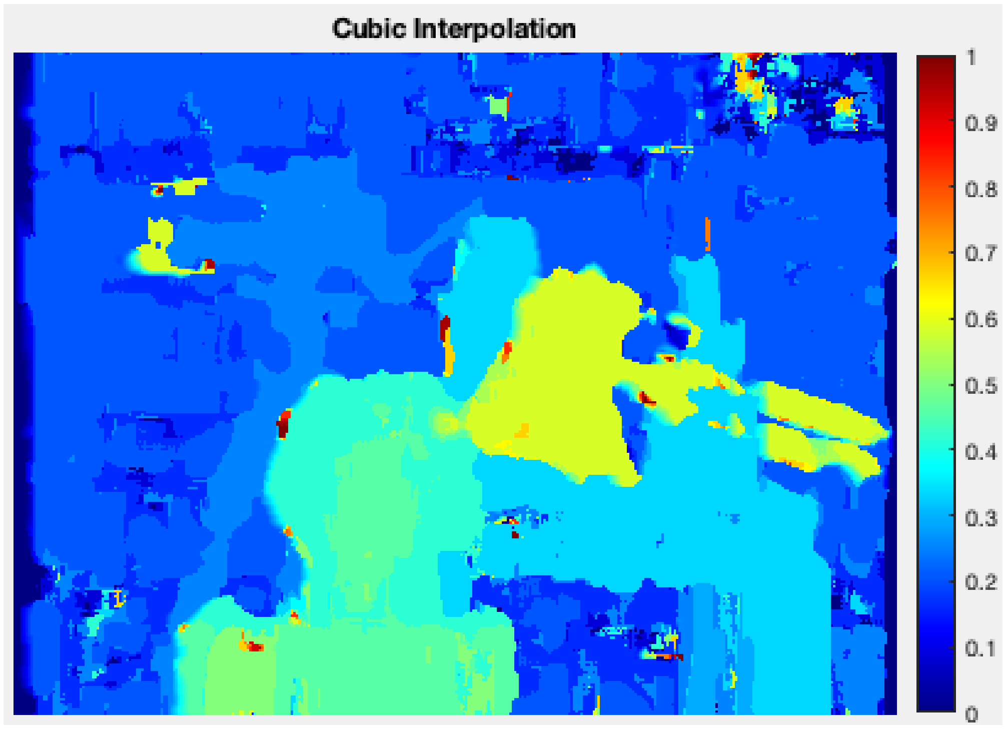
(i) **Linear Interpolation:** Linear interpolation is the simplest method of getting values at positions in between the data points. **The points are simply joined by straight line segments. Each segment (bounded by two data points) can be interpolated independently.**



(ii) **Nearest Neighbor:** The nearest neighbor algorithm **selects the value of the nearest point** and does not consider the values of neighboring points at all, **yielding a piecewise-constant interpolant.**

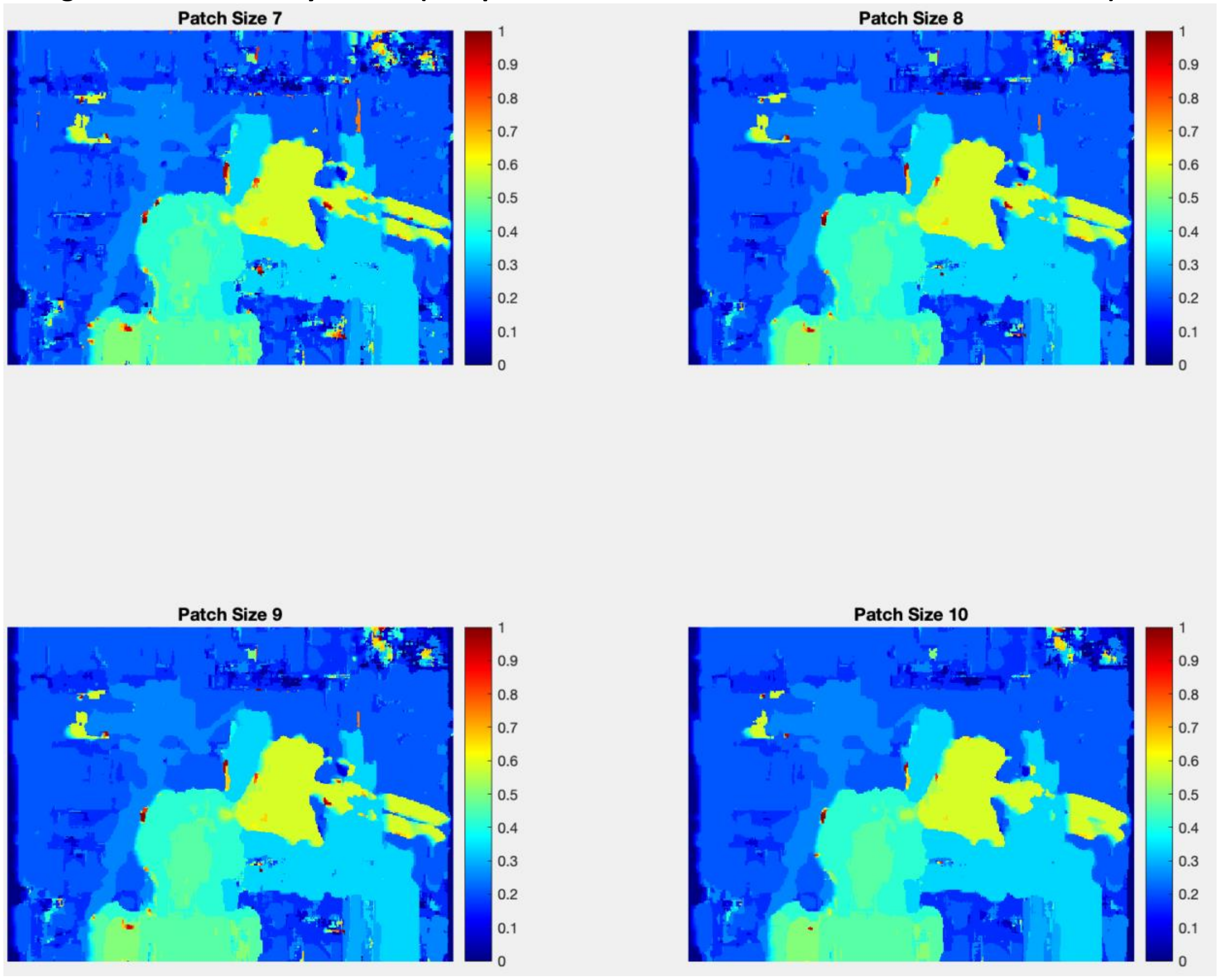


(iii) **Cubic Interpolation:** Cubic interpolation is the simplest method that **offers true continuity between the segments**. As such it requires more than just the two endpoints of the segment but also the two points on either side of them.



(b) To fill in the undefined values, use an interpolation technique.

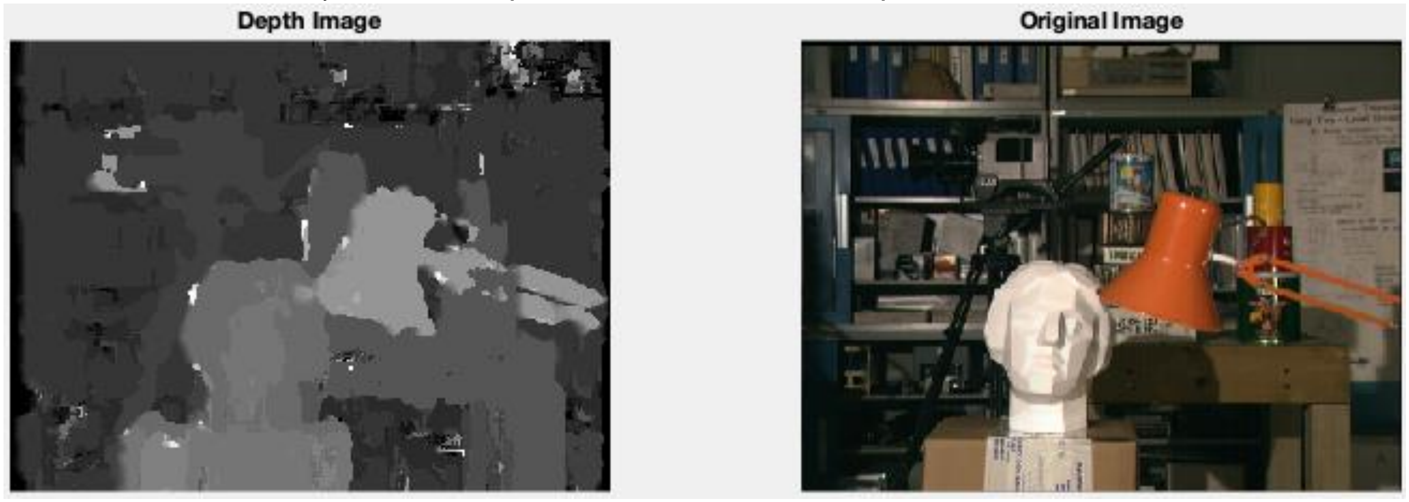
Using SAD as a similarity metric (compare the difference between SAD and SSD below)



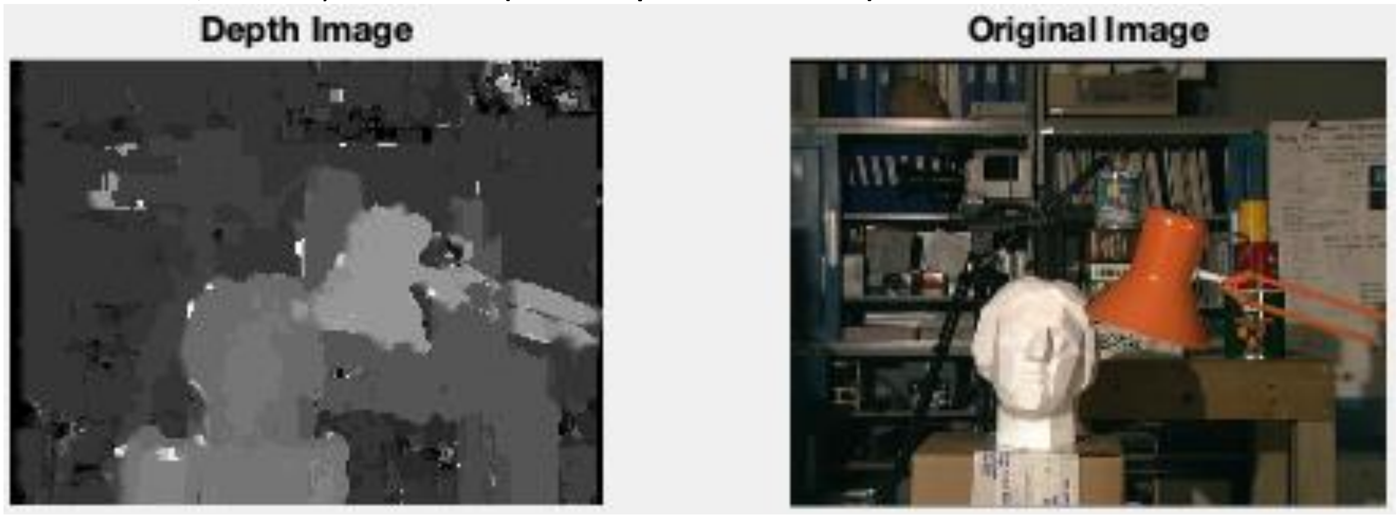
⇨ Referring to the figure above, **I think that Patch Size 8 has the best result since it remains the detail of the lamp bracket and reduces some noise pixels in patch size 7**.

Best Result for SAD (Sum of Absolute Difference) and SSD (Sum of Squared Difference)

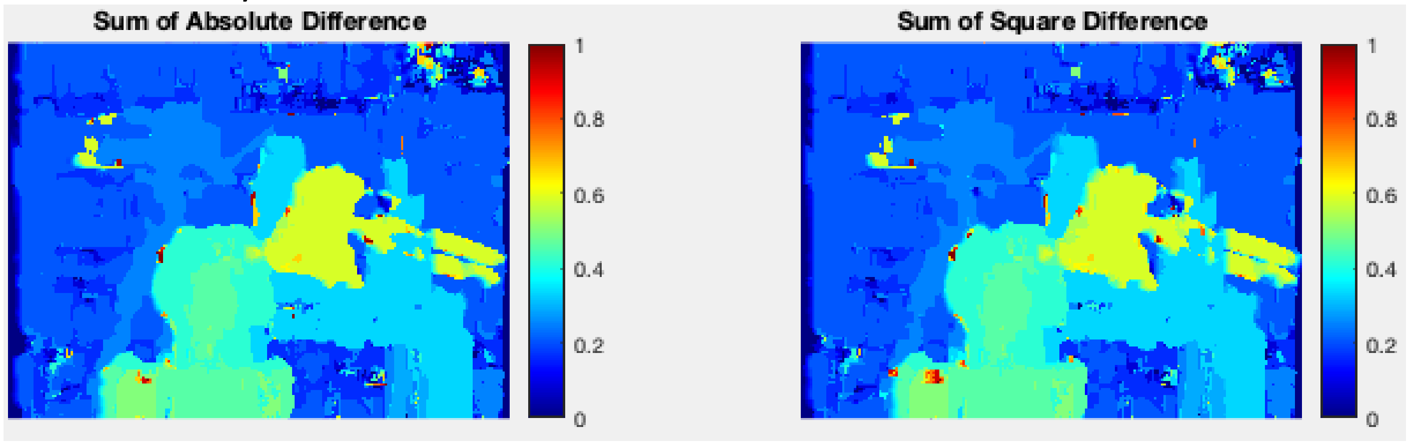
Patch Size: **8x8**, Similarity Metric: **SAD (Sum of Absolute Difference)**



Patch Size: **8x8**, Similarity Metric: **SSD (Sum of Squared Difference)**



Compare Sum of Absolute Difference v.s Sum of Square Difference using colormap (display image with scaled colors)



⇨ According to the figure above, we can see that both SAD and SSD perform well on display the depth, **the difference between the two are minor**. However, overall, **SSD is more sensitive to noise than SAD**. The reason why is that **SSD will enlarge the noise points to make those points more obvious**. Moreover, **both SAD and SSD are influenced by noise points, especially near the objects' boundaries.**

To sum up, I think that **I will choose patch size 8x8 with Sum of Absolute Difference to find out the maximal similarity between the two images** (Initial guess: 12x12 with SAD).

(6) The technique described above attempts to find each pixel correspondence independently. It uses a hard constraint of epipolar geometry and a soft constraint of patch similarity to find the correspondence. But, in addition, we can also have other soft constraints like smoothness, uniqueness, and ordering of correspondences. These constraints help to relate between pixels on the same row (epipolar line). Briefly ex- plain the three constraints (smoothness, uniqueness, and ordering) and mention the cases when they would fail.
(Cited from: https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT11/node5.html#:~:text=Uniqueness%20Marr%20and,zone%20%28Fig.%C2%A07%29)

(a) **Smoothness**

⇨ For the intensity-based approach, which is our implementation, the matching pixels must have a standard to ensure all the pixels are qualified. For example, I use 24 as the threshold value to ensure that **any pixel that exceeds an offset of 24 will be recognized as the problem pixel**.

(b) **Uniqueness**

⇨ **A given pixel or feature from one image can match no more than one pixel** or feature from the other image. That is, in our implementation, **we only consider the pixel with minimum disparity difference as our only matching pixel**.

(c) **Ordering**

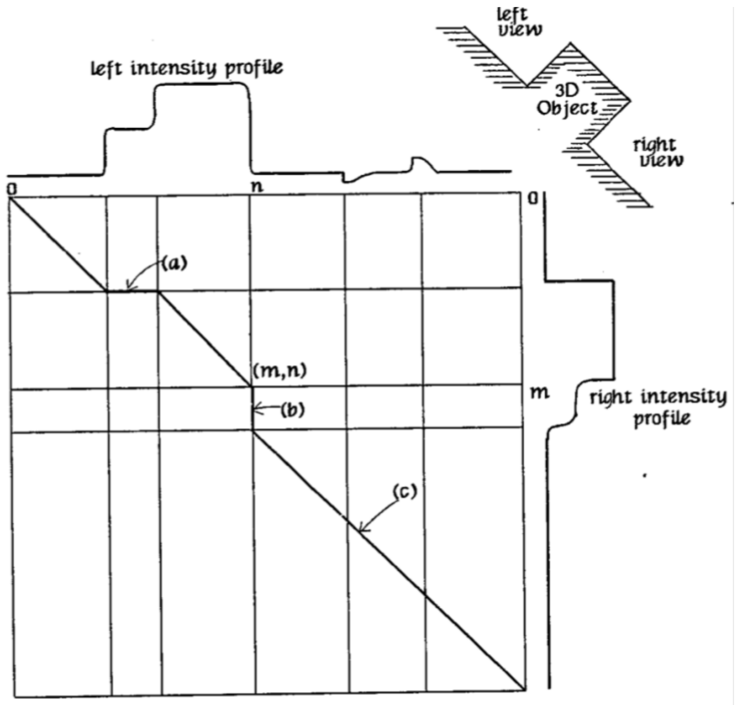
⇨ **This means that if the given pixel is from the left image, then the corresponding pixel in the right image can only appear on the right side of this given pixel's index**. This is the like the **minimum value of delta must <= 0 constrain** I set in my implementation, which marks the pixels as problem pixels if the index of the pixel we found at the right image appears in the left side of the given pixel.

1.2 Scanline Stereo

(1) Read the included paper4, "Stereo by intra- and inter-scanline search using dynamic programming," until section 3.2. What are the key points of this portion of the paper? With their approach in mind, how could smoothness, uniqueness, and ordering constraints be incorporated to improve the quality of correspondences? Note: although this paper discusses edge correspondences, many of the concepts can be applied to patch correspondences also.
(Cited from: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.4704&rep=rep1&type=pdf)

Introduction: When we know the camera model, we can simplify the image-to-image correspondence problem into a set of **scanline-to-scanline problems**. We call this search **intra-scanline search**. Moreover, the problem of finding a correspondence between pair of stereo images can be cast as that of finding a matching surface (Le., a set of matching paths), in which the cost of the matching surface is defined as **the sum of the costs of the intra-scanline matches on the 2D search planes**.

(a) **Key Points:**
1. A **dynamic programming** technique can handle the intra-scanline search efficiently.
2. Our stereo algorithm tries to find an **optimal set of paths that minimizes the matching cost under the inter-scanline constraint**.
3. It **involves two searches**: one is inter-scanline search for possible correspondence of connected edges and the other is intra-scanline search for correspondence of edges under the constraint given by connected edges.



⇨ In the figure above, (a) corresponds to the case in which a visible part in the left image is occluded in the right image. That is, it means that we cannot find a

(b) **Smoothness**
⇨ The dynamic programming approach will not affect the smoothness since we can still set up the threshold of the matching pixels to ensure all the pixels are qualified.

(c) **Uniqueness**
⇨ The dynamic programming approach will not affect the uniqueness since **only the pixel in the right image with minimum disparity difference will be matched with the pixel in the left**.

(d) **Ordering**
⇨ The dynamic programming approach ensures that **the pixels of the right image that appears in the right will not match to a pixel in the left image that is on the left-hand side of their**

precede pixels correspondences. The path will go from the top left to the lower right and never turn back to the opposite direction.

(2) Implement dynamic programming to solve the patch correspondence problem. They refer to this as the “intra-scanline search” in the paper.

⇨ The dynamic programming method I implemented is the Geiger et al method (refer to the figure of part c below), which further improved the smoothness at the corresponding pixels.

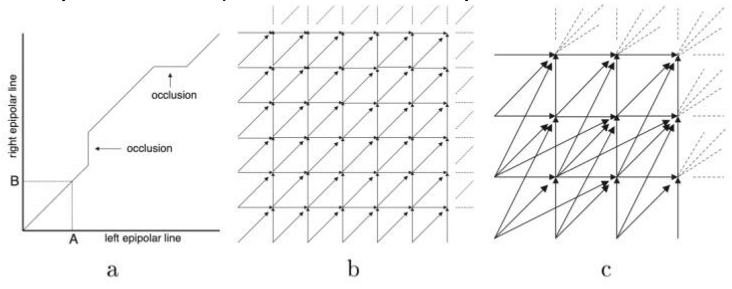
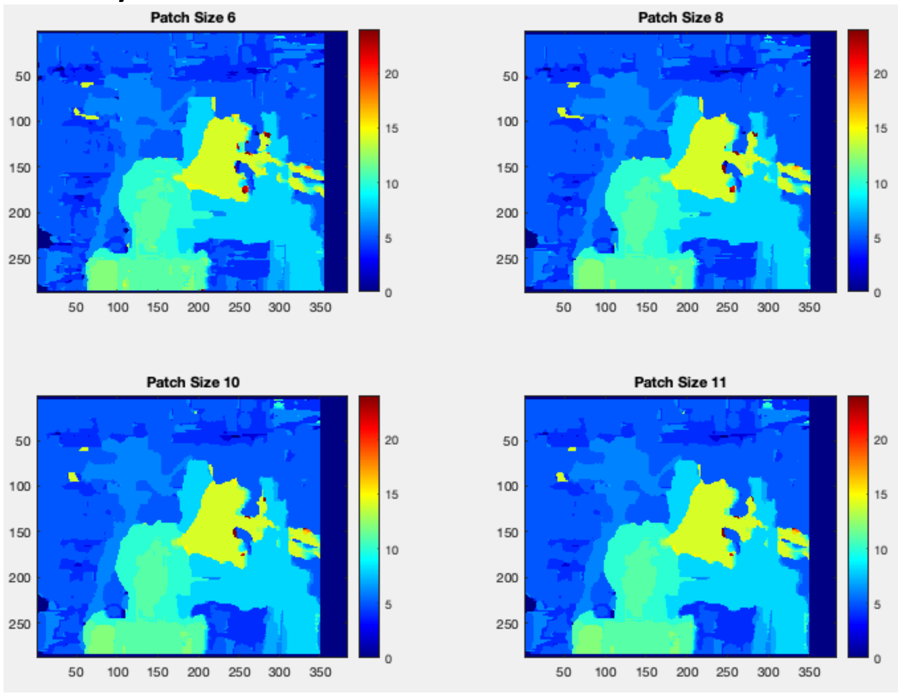


Figure 2: *Dense epipolar matching.* **(a)** *the path defines a matching between the two epipolar lines, e.g. point A is matched to point B. Horizontal and vertical arcs represent occlusions.* **(b)** *the connectivity of the graph of the Cox et al method, each vertex represents a putative matching of a left and right pixel and has three input arcs and three output allowing horizontal, vertical and diagonal moves.* **(c)** *the connectivity of the Geiger et al method, here more complex movements can be made.*

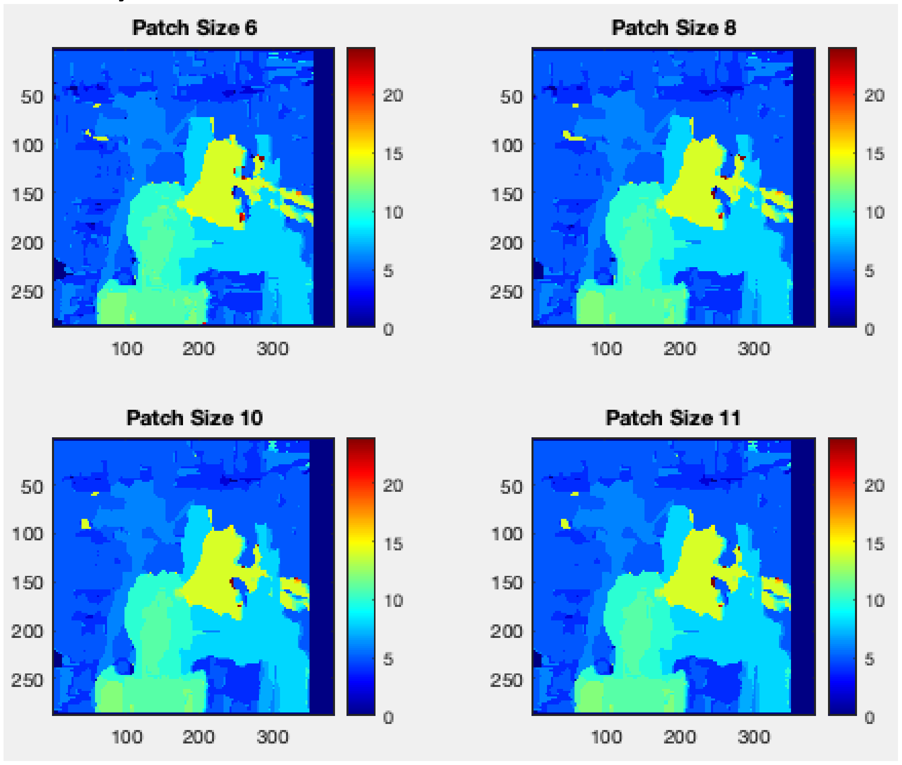
(Cited from: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Criminisi_techrep2002a.pdf)

Similarity Metric: SAD



⇨ I considered that **patch size 8~10 is the best** since a larger patch size (ex: 10) will distort the objects and the lower patch size (ex: 6) contain more noise in it.

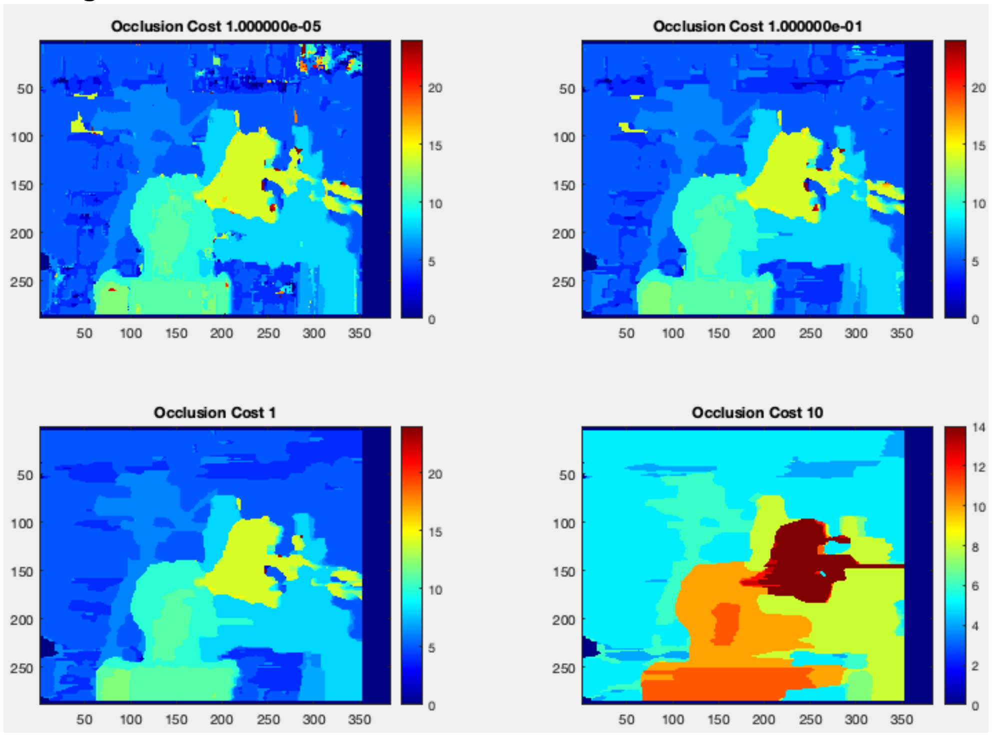
Similarity Metric: SSD



⇨ I think SAD works better than SSD since **SSD contains more noises like the figure shown**.

⇨ Note that since dynamic programming is to scan diagonally on the two images, there will be some occlusions at the right side of the output images.

Setting Different Occlusion Costs:



⇨ I think occlusion cost = 0.1 gives the best result.

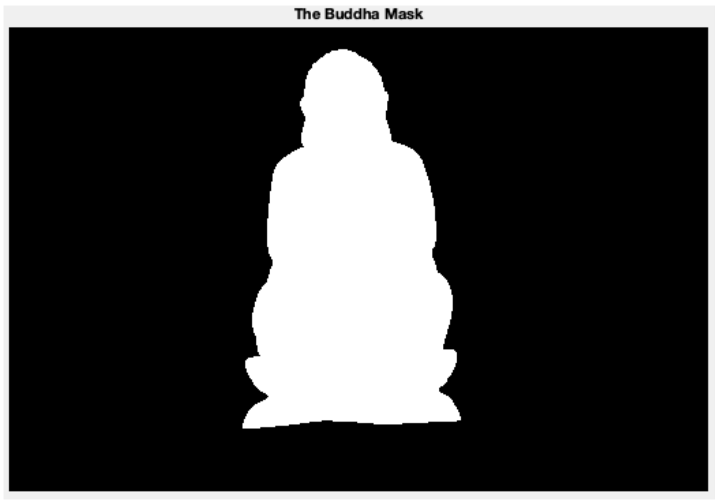
Grad Credits: Photometric Stereo for Lambertian object

In this section, you will implement a simple version of photometric stereo for Lambertian objects in MATLAB. To aid you, radiometric calibration and discovering lighting angles have been performed for you. That is, image intensities have been made linear and normalized to account for differences in distance between the light source and object and you have been provided with the lighting angle for each view (lighting.mat).

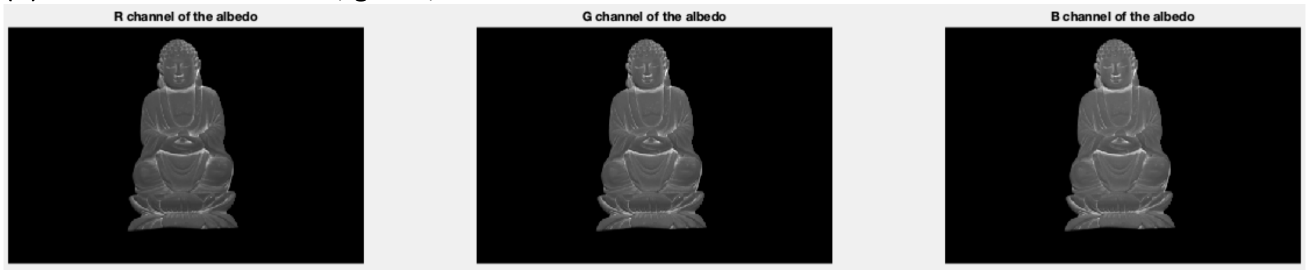
1. Read in 12 images for a given scene

2. Compute surface normal vectors for each point within a specified mask

   The Buddha Mask

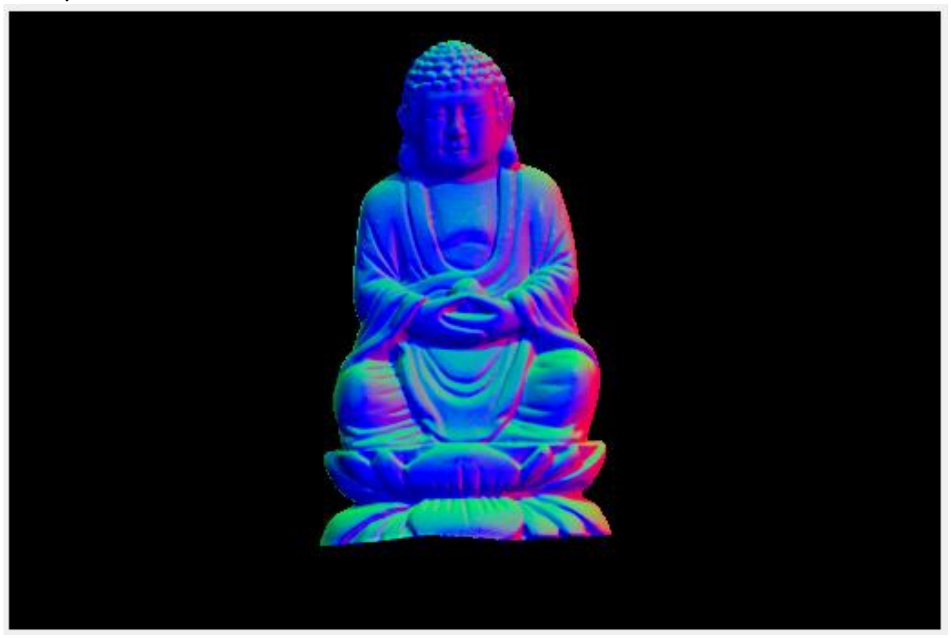


   (1) the albedo for the red, green, and blue channels

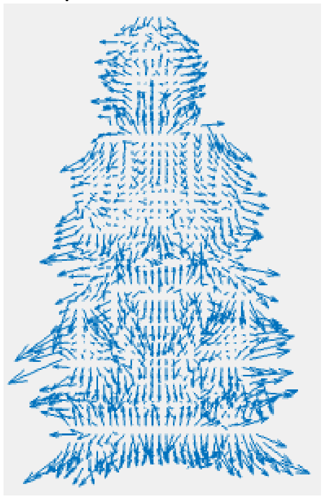


(2) surface normal vectors for the luminance channel

Compute surface normal

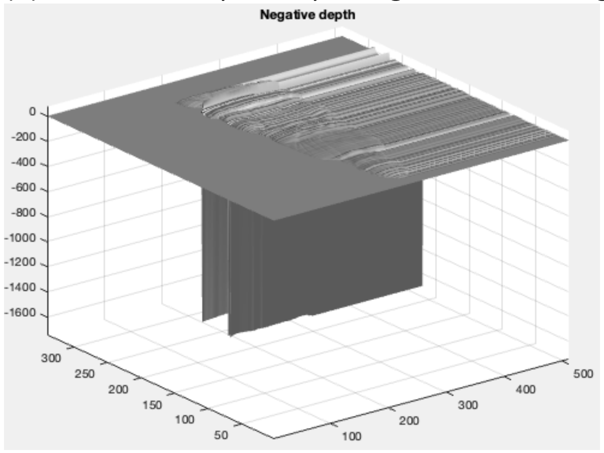


Compute surface normal vectors (setting the scale of the quiver to 6)



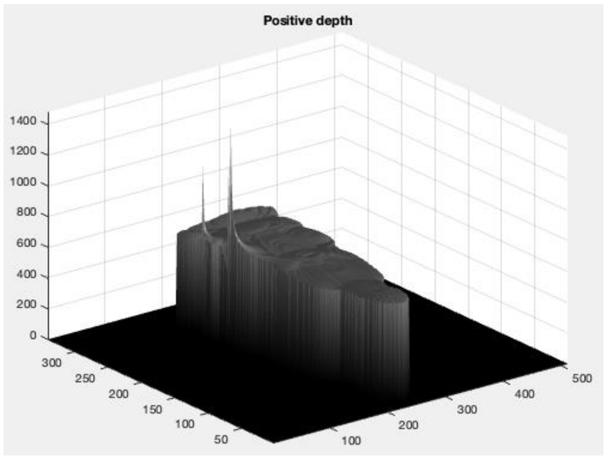
3. Recover albedo information within the region of interest

(3) recovered depth map using the direct integration of the surface normal vectors

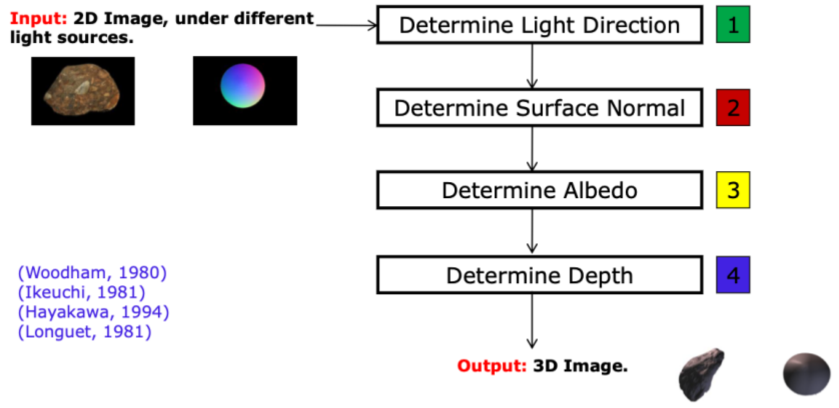


4. Find a depth map for the object.

(4) refined depth map using the provided code. Please note that the albedo is computed for each channel independently and that the surface normal vectors are computed using the grayscale representation of the image.

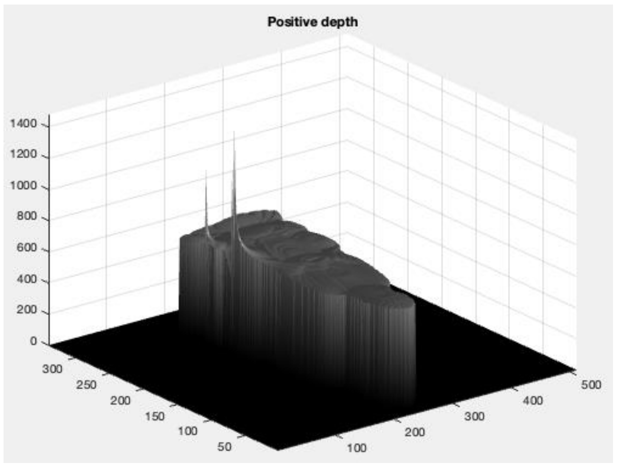


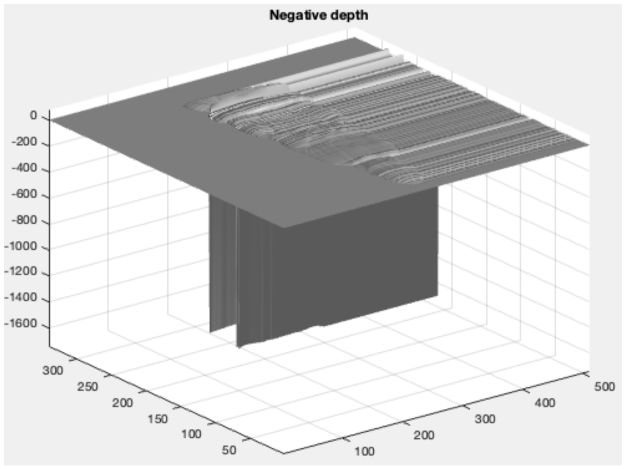
The procedure of Photometric Stereo and its Algorithm (Cited from: https://aptafti.github.io/files/3DSurfaceReconstructionUsingPhotometricStereoApproach.pdf)



5. At the end of the report, be sure to address any shortcomings of your algorithm and offer suggestions for improved performance. Do you trust the depth provided using photometric stereo? In particular, discuss how shadows can be handled within this framework though you do not have to account for shadows in your code.

⇨ I think there are still some improvements I perhaps I can do on my algorithm to achieve better results, like filtering **out the noise pixels in the depth maps** (refer to the figures below). As we can see, the positive depth map has some noises with values larger than 600 in the z-axis, and there are also some noises in the negative depth map that has value above 0. I considered that I could enhance my algorithm by **setting a threshold** to eliminate outliers or to **apply filters like the Gaussian Filter at first.**





⇨ **I trust the depth provided using photometric stereo since the outputs look reasonable**, and nothing seems weird on the results. Though there are still some improvements like to denoise the depth images, the overall results look great with merely a few noise pixels in it.

[Issue]

The biggest issue I encountered in this assignment is to **find out what constraints I should add on to ensure smoothness, uniqueness, and ordering**. The prerequisite of defining the constraints is to recognize what occlusion is. Moreover, in section 1.2 **the dynamic programming part, I could not realize what the paper is trying to tell at first**. Eventually, I learn the reason and how to use dynamic programming on stereo matching on YouTube and finally got my work done.

Another issue in this assignment is that **it takes too much time in 1.1 to scan all rows in the right image**. The algorithm took me about 20 mins to compare 9 different patch sizes. Hopefully, dynamic programming in 1.2 did not take that much time by reducing the search space from the whole row to a manually specified number of pixels (I specified 24 pixels).

Lastly, I think **the workload of this assignment is too large**. I considered that the workload is twice as the previous assignments. I spent about the whole week comprehensively accomplishing this assignment. I anticipate professor could adjust the workload in the following assignments.

[Surprise]

The assignment demonstrates the techniques of reconstructing the 3D world based on a 2D perspective. I think this is super useful since plenty of applications like automatic cars rely greatly on depth image construction from the 3D scene to perform its object detection on it.