Final Project: Stereo Matching

林义涵 (学号：13331158 计应1班)

Language

The main language I select is **C++**.
The main tool is **opencv**.
And the platform is **visual studio 2013** on **Windows 10, 64-bit**.

Basic Task

Evaluate the Quality of My Disparity Maps

I need one "ground truth" disparity map and my disparity map. Since I use **opencv**, the main data structure is **Mat**.

I need to traverse every pixel in the standart map and my map, and calculate the **absolute value** between them. If the absolute value is larger than **3**, increase the counter. And divide the counter by the total pixel number, the result is the **percentage of bad pixels**.

Sum of Squared Difference (SSD)

I need to traverse the images in the sub-directories. So I use some **directories operation API** on windows.

The max offset is 79, and the window size is 7*7. First, read two input images and convert them to grey images. Build a vector called **min_ssd** and make some initialization. Build a new Mat called **disp** to load my disparity map.

Second, according to every offset, build a Mat called **tmp** to load the right-eyed image or the left-eyed image after shifting. Build a Mat called **sd** to make some preprocess, which means to calculate the **absolute value** on the pixels between the Mat **tmp** and the other-eyed image and square it.

Third, for every window to every pixel on the Mat **sd**, calculate the sum called **ssd**. If the sum is less than the according value on the **min_ssd**, replace it and change the according value on **disp** to the **current offset * 3**.

The **matching cost function** is:

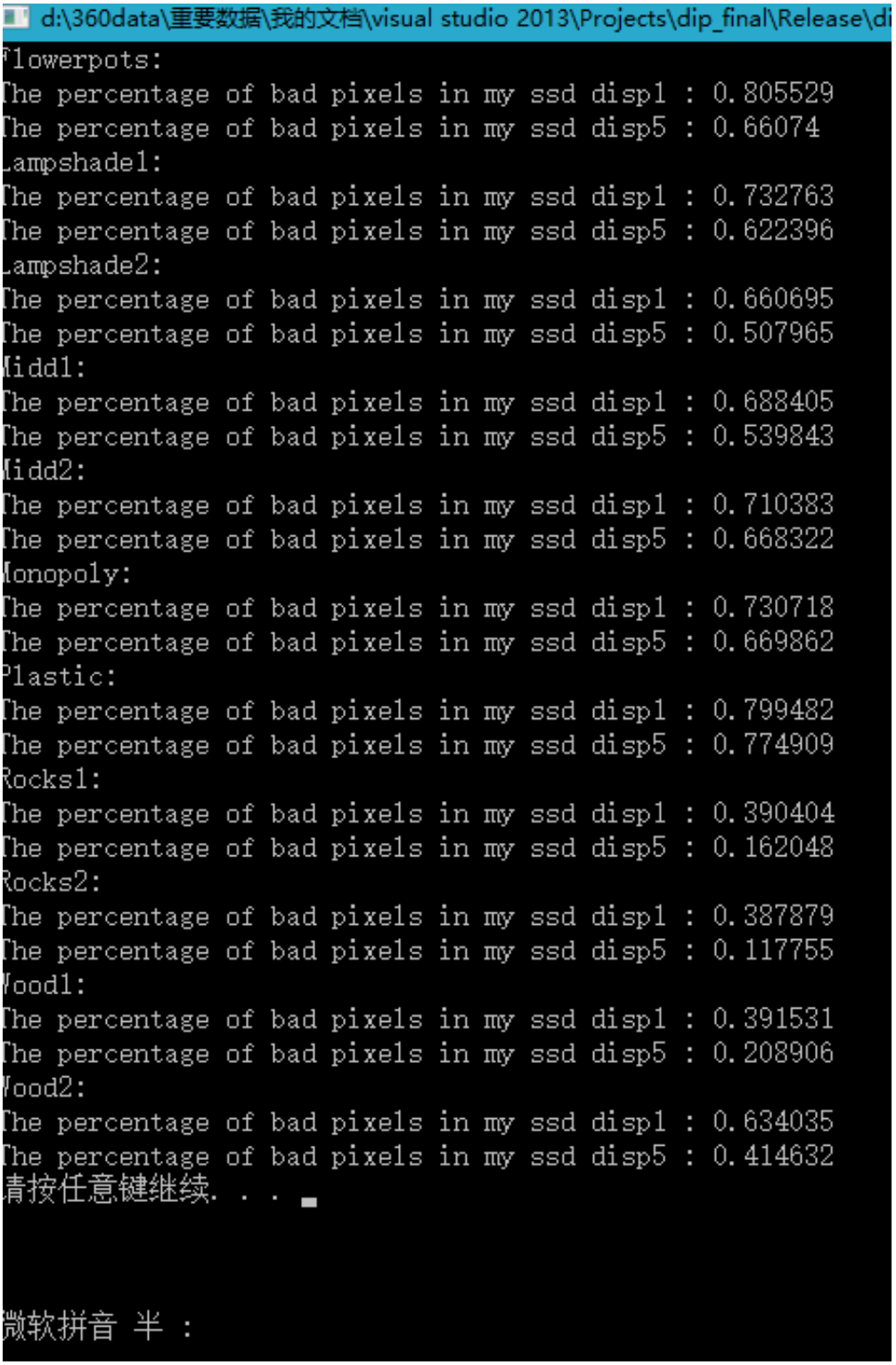
$$dist(F_L(x,y), F_R(x-d,y)) = sum((F_L(x,y) - F_R(x-d,y))^2)$$

where F∗(x,y) is a feature vector extracted from I∗ at location (x,y) (e.g. a 5×5 patch centered at

(x,y)).

The quality of my disparity maps is:

```
Aloe:
The percentage of bad pixels in my ssd disp1 : 0.217735
The percentage of bad pixels in my ssd disp5 : 0.105747
Baby1:
The percentage of bad pixels in my ssd disp1 : 0.434559
The percentage of bad pixels in my ssd disp5 : 0.277227
Baby2:
The percentage of bad pixels in my ssd disp1 : 0.422944
The percentage of bad pixels in my ssd disp5 : 0.304044
Baby3:
The percentage of bad pixels in my ssd disp1 : 0.634789
The percentage of bad pixels in my ssd disp5 : 0.417156
Bowling1:
The percentage of bad pixels in my ssd disp1 : 0.72973
The percentage of bad pixels in my ssd disp5 : 0.696986
Bowling2:
The percentage of bad pixels in my ssd disp1 : 0.599091
The percentage of bad pixels in my ssd disp5 : 0.435739
Cloth1:
The percentage of bad pixels in my ssd disp1 : 0.300039
The percentage of bad pixels in my ssd disp5 : 0.0089442
Cloth2:
The percentage of bad pixels in my ssd disp1 : 0.614718
The percentage of bad pixels in my ssd disp5 : 0.286742
Cloth3:
The percentage of bad pixels in my ssd disp1 : 0.365604
The percentage of bad pixels in my ssd disp5 : 0.066887
Cloth4:
The percentage of bad pixels in my ssd disp1 : 0.460514
The percentage of bad pixels in my ssd disp5 : 0.153811
Flowerpots:
The percentage of bad pixels in my ssd disp1 : 0.805529
The percentage of bad pixels in my ssd disp5 : 0.66074
Lampshade1:
The percentage of bad pixels in my ssd disp1 : 0.732763
The percentage of bad pixels in my ssd disp5 : 0.622396
Lampshade2:
微软拼音 半 :
```



Normalized Cross Correlation (NCC)

The **formula of the NCC matching cost** is:

$$N_{cc}(d) = \frac{S_{RL} - S_R S_L / m^2}{\sqrt{(S_{RR} - S_R S_R / m^2)(S_{LL} - S_L S_L / m^2)}}$$

where

$$S_{RL} = \sum_{i=1}^{m} \sum_{j=1}^{m} I_R I_L$$

$$S_R = \sum_{i=1}^{m} \sum_{j=1}^{m} I_R$$

$$S_L = \sum_{i=1}^{m} \sum_{j=1}^{m} I_L$$

$$S_{RR} = \sum_{i=1}^{m} \sum_{j=1}^{m} I_R^2$$

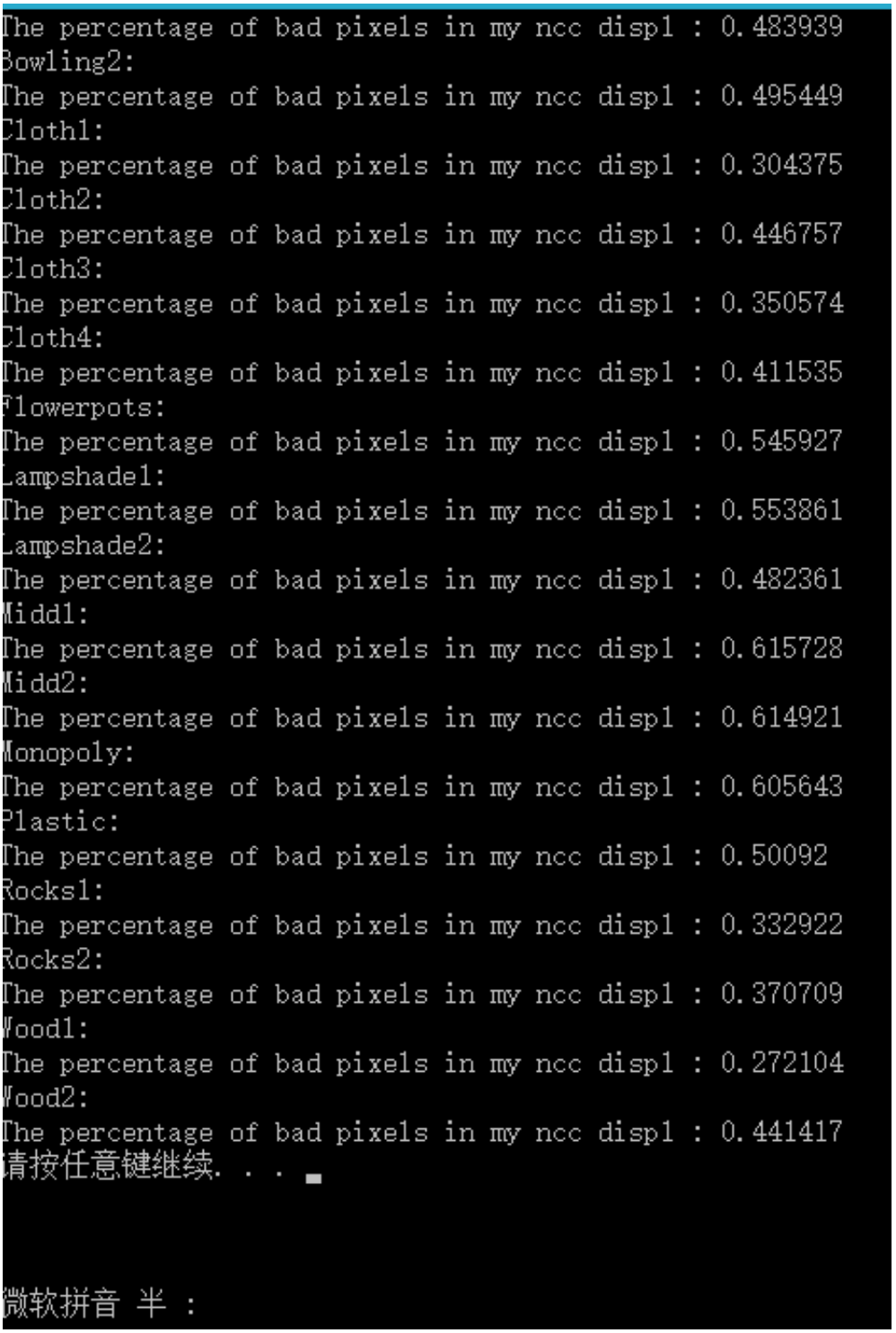
The max offset is 79, and the window size is 7*7. First, read two input images and convert them to grey images. Build a vector called **max_ncc** and make some initialization. Build a new Mat called **disp** to load my disparity map.

Second, according to every offset, build a Mat called **tmp** to load the right-eyed image or the left-eyed image after shifting.

Third, for every window to every pixel on the Mat, use the formula to calculate the parameters and comes up with a result called **sum_ncc**. If the result is larger than the according value on the **max_ncc**, replace it and change the according value on **disp** to the **current offset * 3**.

The quality of my disparity maps is:

```
Aloe:
The percentage of bad pixels in my ncc disp1 : 0.211558
Baby1:
The percentage of bad pixels in my ncc disp1 : 0.281487
Baby2:
The percentage of bad pixels in my ncc disp1 : 0.32969
Baby3:
The percentage of bad pixels in my ncc disp1 : 0.504694
Bowling1:
The percentage of bad pixels in my ncc disp1 : 0.483939
Bowling2:
The percentage of bad pixels in my ncc disp1 : 0.495449
Cloth1:
The percentage of bad pixels in my ncc disp1 : 0.304375
Cloth2:
The percentage of bad pixels in my ncc disp1 : 0.446757
Cloth3:
The percentage of bad pixels in my ncc disp1 : 0.350574
Cloth4:
The percentage of bad pixels in my ncc disp1 : 0.411535
Flowerpots:
The percentage of bad pixels in my ncc disp1 : 0.545927
Lampshade1:
The percentage of bad pixels in my ncc disp1 : 0.553861
Lampshade2:
The percentage of bad pixels in my ncc disp1 : 0.482361
Midd1:
The percentage of bad pixels in my ncc disp1 : 0.615728
Midd2:
The percentage of bad pixels in my ncc disp1 : 0.614921
Monopoly:
The percentage of bad pixels in my ncc disp1 : 0.605643
Plastic:
The percentage of bad pixels in my ncc disp1 : 0.50092
Rocks1:
The percentage of bad pixels in my ncc disp1 : 0.332922
Rocks2:
微软拼音 半 :
```

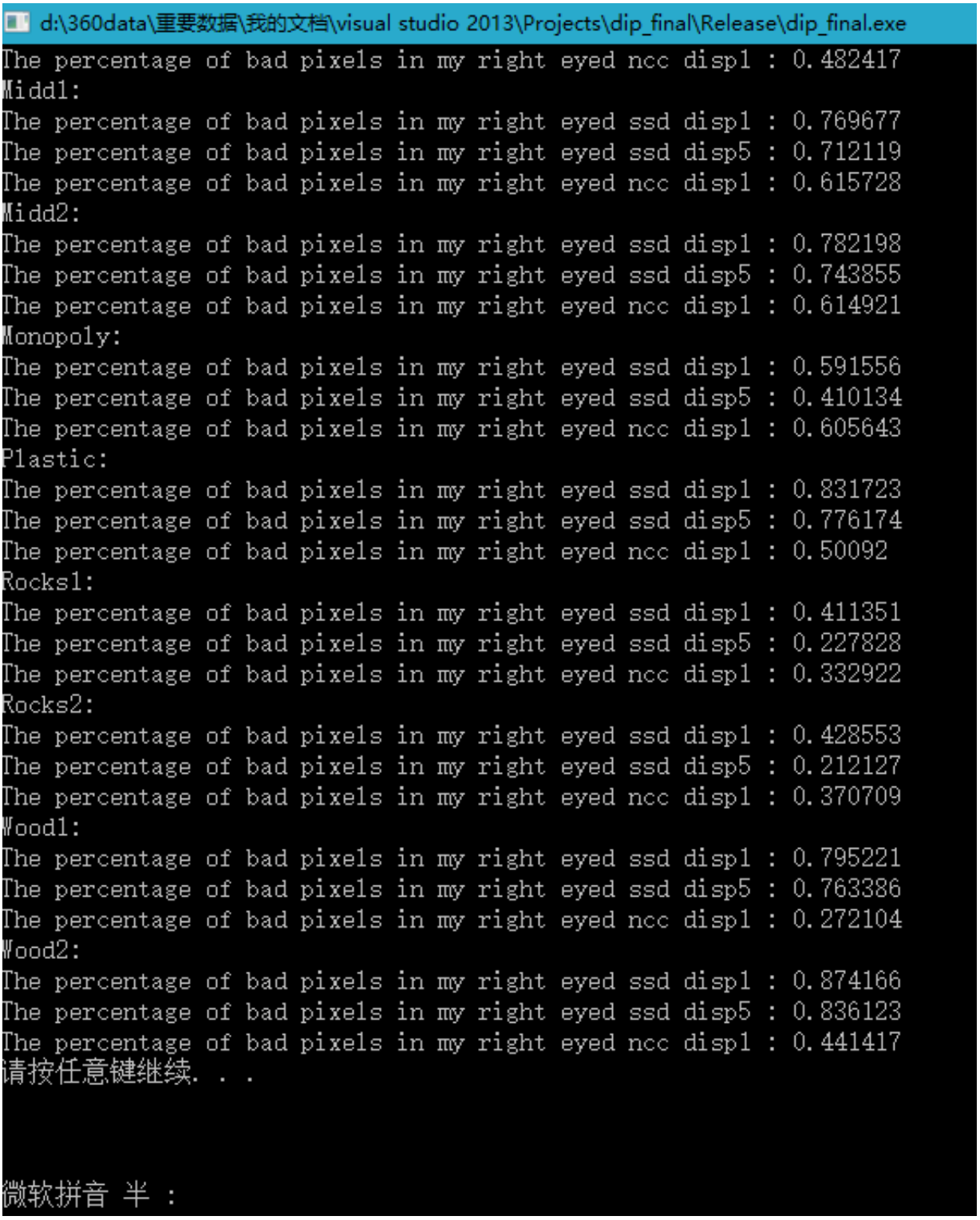


Add a Constant Amount of Intensity to Right Eye Images

The constant amount of intensity I added is 10. Here are the results:

```
Aloe:
The percentage of bad pixels in my right eyed ssd disp1 : 0.236635
The percentage of bad pixels in my right eyed ssd disp5 : 0.152231
The percentage of bad pixels in my right eyed ncc disp1 : 0.211558
Baby1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.531614
The percentage of bad pixels in my right eyed ssd disp5 : 0.613141
The percentage of bad pixels in my right eyed ncc disp1 : 0.281487
Baby2:
The percentage of bad pixels in my right eyed ssd disp1 : 0.526117
The percentage of bad pixels in my right eyed ssd disp5 : 0.513448
The percentage of bad pixels in my right eyed ncc disp1 : 0.32969
Baby3:
The percentage of bad pixels in my right eyed ssd disp1 : 0.751215
The percentage of bad pixels in my right eyed ssd disp5 : 0.648061
The percentage of bad pixels in my right eyed ncc disp1 : 0.504694
Bowling1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.847424
The percentage of bad pixels in my right eyed ssd disp5 : 0.852246
The percentage of bad pixels in my right eyed ncc disp1 : 0.511725
Bowling2:
The percentage of bad pixels in my right eyed ssd disp1 : 0.675487
The percentage of bad pixels in my right eyed ssd disp5 : 0.602276
The percentage of bad pixels in my right eyed ncc disp1 : 0.503703
Cloth1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.307466
The percentage of bad pixels in my right eyed ssd disp5 : 0.0132413
The percentage of bad pixels in my right eyed ncc disp1 : 0.304375
Cloth2:
The percentage of bad pixels in my right eyed ssd disp1 : 0.623276
The percentage of bad pixels in my right eyed ssd disp5 : 0.418863
The percentage of bad pixels in my right eyed ncc disp1 : 0.501685
Cloth3:
The percentage of bad pixels in my right eyed ssd disp1 : 0.470795
The percentage of bad pixels in my right eyed ssd disp5 : 0.168579
The percentage of bad pixels in my right eyed ncc disp1 : 0.35666
Cloth4:
微软拼音 半 :
```

```
Cloth4:
The percentage of bad pixels in my right eyed ssd disp1 : 0.635597
The percentage of bad pixels in my right eyed ssd disp5 : 0.365321
The percentage of bad pixels in my right eyed ncc disp1 : 0.446389
Flowerpots:
The percentage of bad pixels in my right eyed ssd disp1 : 0.926798
The percentage of bad pixels in my right eyed ssd disp5 : 0.920014
The percentage of bad pixels in my right eyed ncc disp1 : 0.545927
Lampshade1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.815867
The percentage of bad pixels in my right eyed ssd disp5 : 0.825105
The percentage of bad pixels in my right eyed ncc disp1 : 0.553861
Lampshade2:
The percentage of bad pixels in my right eyed ssd disp1 : 0.868654
The percentage of bad pixels in my right eyed ssd disp5 : 0.886742
The percentage of bad pixels in my right eyed ncc disp1 : 0.482417
Midd1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.769677
The percentage of bad pixels in my right eyed ssd disp5 : 0.712119
The percentage of bad pixels in my right eyed ncc disp1 : 0.615728
Midd2:
The percentage of bad pixels in my right eyed ssd disp1 : 0.782198
The percentage of bad pixels in my right eyed ssd disp5 : 0.743855
The percentage of bad pixels in my right eyed ncc disp1 : 0.614921
Monopoly:
The percentage of bad pixels in my right eyed ssd disp1 : 0.591556
The percentage of bad pixels in my right eyed ssd disp5 : 0.410134
The percentage of bad pixels in my right eyed ncc disp1 : 0.605643
Plastic:
The percentage of bad pixels in my right eyed ssd disp1 : 0.831723
The percentage of bad pixels in my right eyed ssd disp5 : 0.776174
The percentage of bad pixels in my right eyed ncc disp1 : 0.50092
Rocks1:
The percentage of bad pixels in my right eyed ssd disp1 : 0.411351
The percentage of bad pixels in my right eyed ssd disp5 : 0.227828
The percentage of bad pixels in my right eyed ncc disp1 : 0.332922
Rocks2:
微软拼音 半 :
```



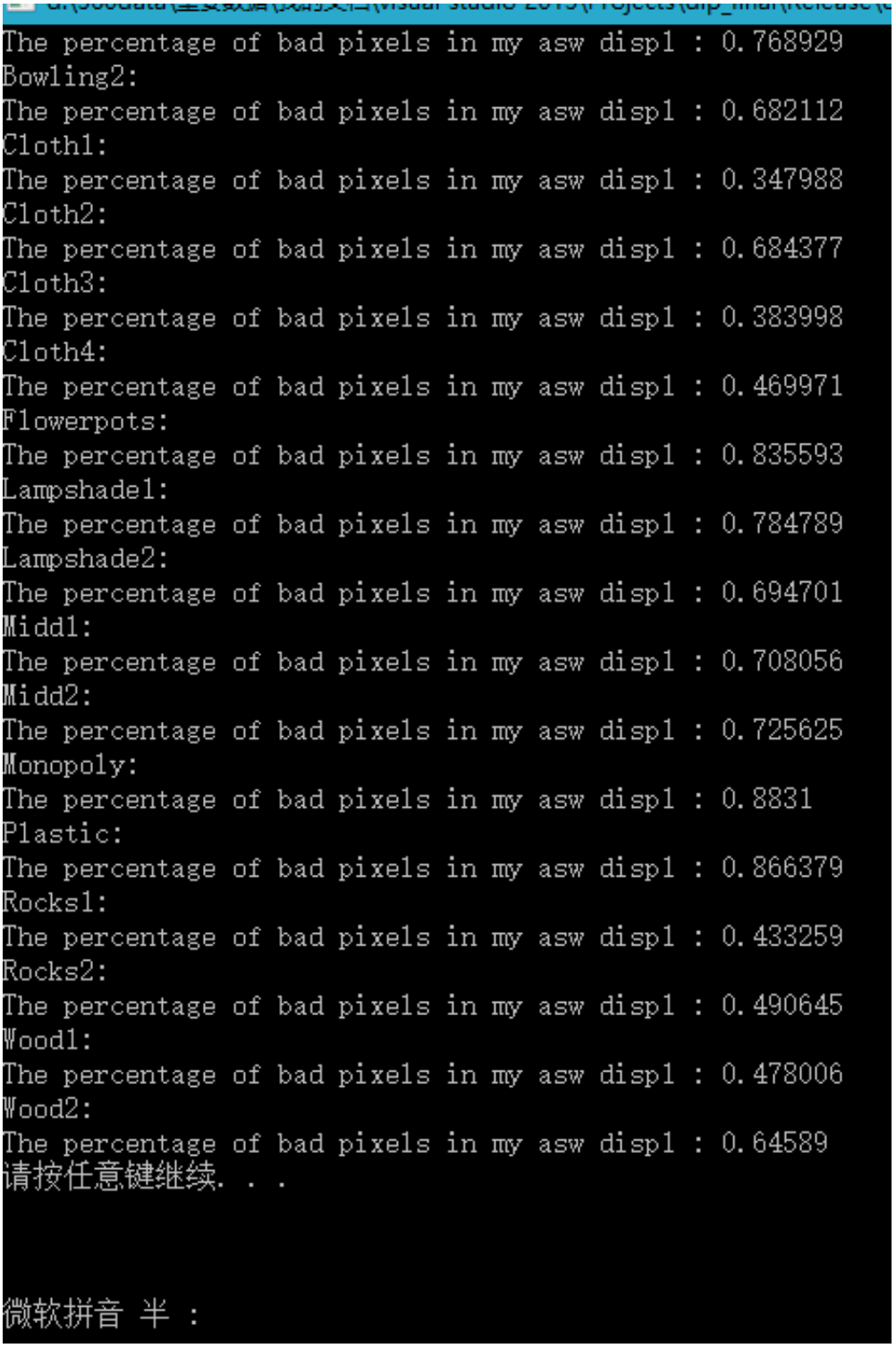
As for the **SSD**, the absolute value between the pixels on two images are increased, then the sum of squared difference are also increased, which might results in confusion while matching.
As for the **NCC**, the averages are also increase but the standard differences on self image stay the same.

With larger patches, NCC can occasionally outperform SSD. Additionally, when there are smaller/less areas where the shade is near flat, accuracy grows as the patch size increases. This makes sense since correspondences between patches are more likely to be unique. But we see that when handling images with more/larger areas with flat shading, larger patches could mean less accuracy.

Adaptive Support Weight (ASW)

Here are the results:

```
Aloe:
The percentage of bad pixels in my asw disp1 : 0.222812
Baby1:
The percentage of bad pixels in my asw disp1 : 0.466756
Baby2:
The percentage of bad pixels in my asw disp1 : 0.591912
Baby3:
The percentage of bad pixels in my asw disp1 : 0.696765
Bowling1:
The percentage of bad pixels in my asw disp1 : 0.768929
Bowling2:
The percentage of bad pixels in my asw disp1 : 0.682112
Cloth1:
The percentage of bad pixels in my asw disp1 : 0.347988
Cloth2:
The percentage of bad pixels in my asw disp1 : 0.684377
Cloth3:
The percentage of bad pixels in my asw disp1 : 0.383998
Cloth4:
The percentage of bad pixels in my asw disp1 : 0.469971
Flowerpots:
The percentage of bad pixels in my asw disp1 : 0.835593
Lampshade1:
The percentage of bad pixels in my asw disp1 : 0.784789
Lampshade2:
The percentage of bad pixels in my asw disp1 : 0.694701
Midd1:
The percentage of bad pixels in my asw disp1 : 0.708056
Midd2:
The percentage of bad pixels in my asw disp1 : 0.725625
Monopoly:
The percentage of bad pixels in my asw disp1 : 0.8831
Plastic:
The percentage of bad pixels in my asw disp1 : 0.866379
Rocks1:
The percentage of bad pixels in my asw disp1 : 0.433259
Rocks2:
微软拼音 半 :
```



The max offset is 79, and the window size is 7*7. First, read two input images and convert them to CIELab color space. Build a vector called **min_asw** and make some initialization. Build a new Mat called **disp** to load my disparity map.

Second, according to every offset, build a Mat called **tmp** to load the right-eyed image or the left-eyed image after shifting and build a Mat called **rgb5** to convert **tmp** to RGB color space.

Third, for every window to every pixel on the Mat, use the formula to calculate the parameters and comes up with a result called **sum_asw**. If the result is less than the according value on the **min_asw**, replace it and change the according value on **disp** to the **current offset * 3**.

The idea of the ASW paper is how to implement ASW stereo matching. Similarity and proximity are

the two main grouping concepts in the classic gestalt theory: similarity refers to what items look like and how that affects grouping, and proximity refers to where items are and how that affects grouping. The gestalt principles of similarity and proximity are also used to compute support-weights. The more similar the color of a pixel, the larger the support-weight of the pixel. In addition, the closer the pixel is, the larger the supportweight. The former is related to the grouping by similarity, and the latter is related to the grouping by proximity.

With larger patches, ASW can occasionally outperform NCC since the accurancy has been improved a lot.