DIP Final Project : Stero Matching

Basic Task

1.Implement a test program to evaluate the quality of your disparity maps. Your program should output the percentage of bad pixels in each of your disparity maps, where bad pixels are those whose errors are greater than one when comparing to ground truths (without multiplication).

(You can see the implementation in the code evaluate.cpp)
Calculation Formula:

$$E(x,y) = |D(x,y) - G(x,y)| \ \ (G(x,y)! = 0)$$

(D(x,y) indicates a pixel in disparity map, G(x,y) indicates the corresponding pixel in GroundTruth, when G(x,y) = 0, the result is not defined)

This program is used to evaluate the quality of the disparity map later with inputs(disparitymap, groundtruth) and output(bad_pixel_percentage)
This is an example of its output:

```
[Rocks2]the percentage of bad pixels: 9.355803
[Rocks2]the percentage of bad pixels: 8.801272
[Wood2]the percentage of bad pixels: 18.744331
[Wood2]the percentage of bad pixels: 17.688102
```

The higher bad_pixel_percentage the disparity map has, the worse the disparity map is. So the best way for stero matching should be the way that result in a small bad percentage.

2. Implement a local stereo matching algorithm using "Sum of Squared Difference (SSD)" as matching cost. Compute both the left and the right disparity maps for all test cases, and visualize them by scaling their intensities by a factor of three. Save each disparity map to a file named testcasename disp1 SSD.png or testcasename disp5 SSD.png. Also write down the matching cost function and the quality of your disparity maps.
Matching cost function:

$$D(p,q) = \sum_{i,j \in P_p, Qq} (P_p(i,j) - Q_q(i,j))^2$$

(P,Q indicates the patch in Left and Right)

Experiment in patch size(5*5)
(You can see the result image in the SSD folder)

Evaluating result:

```
[Baby2]          86.623912
[Baby2]          86.485832
[Plastic]        95.403489
[Plastic]        95.960003
[Lampshade1]     90.939392
[Lampshade1]     90.484364
[Wood1]          84.996747
[Wood1]          85.702289
[Cloth2]         85.989014
[Cloth2]         86.743649
[Bowling2]       88.971997
[Bowling2]       89.059850
[Midd1]          90.292357
[Midd1]          91.058413
[Cloth1]         77.970056
[Cloth1]         77.347851
[Aloe]           82.254573
[Aloe]           82.334958
[Monopoly]       91.509975
[Monopoly]       91.359893
[Cloth3]         80.191847
[Cloth3]         79.908614
[Baby1]          79.717950
[Baby1]          79.548459
[Rocks1]         80.377742
[Rocks1]         80.512560
[Lampshade2]     91.407528
[Lampshade2]     91.998627
[Flowerpots]     93.149236
[Flowerpots]     93.285917
[Midd2]          92.732997
[Midd2]          92.919513
[Baby3]          84.986084
[Baby3]          84.874142
[Cloth4]         81.736471
[Cloth4]         82.141564
[Bowling1]       92.239938
[Bowling1]       92.373453
[Rocks2]         82.866137
[Rocks2]         82.267727
[Wood2]          85.840323
[Wood2]          85.942839
```

3. Implement a local stereo matching algorithm using “Normalized Cross Correlation (NCC)” as matching

cost. Visualize and save your estimated disparity maps to testcasename disp1 NCC.png or testcasename disp5 NCC.png. You are required to figure out the meaning of NCC by yourselves and write down the formula of the NCC matching cost. Also write down the quality of your disparity maps.

NCC explantion:

> In signal processing, cross-correlation is a measure of similarity of two series as a function of the lag of one relative to the other. This is also known as a sliding dot product or sliding inner-product. It is commonly used for searching a long signal for a shorter, known feature. It has applications in pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology.

From wikipedia

As an example, consider two real valued functions f and g differing only by an unknown shift along the x-axis. One can use the cross-correlation to find how much g must be shifted along the x-axis to make it identical to f. The formula essentially slides the g function along the x-axis, calculating the integral of their product at each position. When the functions match, the value of (f\star g) is maximized. This is because when peaks (positive areas) are aligned, they make a large contribution to the integral. Similarly, when troughs (negative areas) align, they also make a positive contribution to the integral because the product of two negative numbers is positive.

The Formula of NCC:

$$D(p,q) = \frac{1}{n} \cdot \sum_{i,j \in N_{pq}} \frac{(f_p(i,j) - \hat{f}_p) \cdot (g_q(i,j) - \hat{g}_q)}{\sigma f_p \sigma g_p}$$

(f hat and g hat indicates the mean value of the window, and sigma means the variance of the window)
Notice that we are supposed to use max value according to the definition of NCC, i.e.

$$D(x,y) = arg_{d \in \{0 \ldots dmax\}} max(F,G)$$

Evaluating result:

```
[Baby2]          86.623912
[Baby2]          86.485832
[Plastic]        95.403489
[Plastic]        95.960003
[Lampshade1]     90.939392
[Lampshade1]     90.484364
[Wood1]          84.996747
[Wood1]          85.702289
[Cloth2]         85.989014
[Cloth2]         86.743649
[Bowling2]       88.971997
[Bowling2]       89.059850
[Midd1]          90.292357
[Midd1]          91.058413
[Cloth1]         77.970056
```

```
[Cloth1]          77.347851
[Aloe]            82.254573
[Aloe]            82.334958
[Monopoly]        91.509975
[Monopoly]        91.359893
[Cloth3]          80.191847
[Cloth3]          79.908614
[Baby1]           79.717950
[Baby1]           79.548459
[Rocks1]          80.377742
[Rocks1]          80.512560
[Lampshade2]      91.407528
[Lampshade2]      91.998627
[Flowerpots]      93.149236
[Flowerpots]      93.285917
[Midd2]           92.732997
[Midd2]           92.919513
[Baby3]           84.986084
[Baby3]           84.874142
[Cloth4]          81.736471
[Cloth4]          82.141564
[Bowling1]        92.239938
[Bowling1]        92.373453
[Rocks2]          82.866137
[Rocks2]          82.267727
[Wood2]           85.840323
[Wood2]           85.942839
```

Add a small constant amount of intensity (e.g. 10) to all right eye images, and re-run the above two methods. Analyze how the intensity change affects the results (i.e. the quality) of the two methods. Explain in which ways that NCC is a better matching cost than SSD.

In my experiment, if add a small constant of intensity to the image, both NCC and SSD's disparity map will result in worse result. Specially, the SSD's disparity map will result even worse than NCC. SSD's disparity map loss its basic profile but NCC reserved.(see the result in CNCC and CSSD)

SSD add constant 10 result

```
[Baby2]           89.896604
[Baby2]           85.593220
[Plastic]         88.304262
[Plastic]         86.158073
[Lampshade1]      84.219462
[Lampshade1]      73.606516
[Wood1]           88.854456
[Wood1]           91.019575
[Cloth2]          94.607702
[Cloth2]          91.513014
[Bowling2]        84.290159
```

```
[Bowling2]        79.769386
[Midd1]           83.192677
[Midd1]           72.415577
[Cloth1]          97.745803
[Cloth1]          95.213559
[Aloe]            93.188176
[Aloe]            94.152794
[Monopoly]        96.091758
[Monopoly]        90.593618
[Cloth3]          94.215438
[Cloth3]          89.472422
[Baby1]           94.743799
[Baby1]           91.389961
[Rocks1]          94.054690
[Rocks1]          90.436248
[Lampshade2]      83.931090
[Lampshade2]      72.315711
[Flowerpots]      73.113365
[Flowerpots]      70.875750
[Midd2]           81.773686
[Midd2]           71.585982
[Baby3]           88.570103
[Baby3]           86.137052
[Cloth4]          95.223769
[Cloth4]          92.755758
[Bowling1]        83.129820
[Bowling1]        76.843606
[Rocks2]          95.115421
[Rocks2]          92.391097
[Wood2]           92.915191
[Wood2]           92.140416
```

NCC add constant 10

```
[Baby2]           87.540082
[Baby2]           20.133499
[Plastic]         71.748131
[Plastic]         25.738930
[Lampshade1]      71.480557
[Lampshade1]      16.047063
[Wood1]           90.974037
[Wood1]           22.077000
[Cloth2]          94.617065
[Cloth2]          20.848262
[Bowling2]        79.796230
[Bowling2]        31.101214
[Midd1]           67.438535
[Midd1]           29.808777
[Cloth1]          85.013935
```

```
[Cloth1]          11.200985
[Aloe]            90.169631
[Aloe]            18.427116
[Monopoly]        92.618510
[Monopoly]        37.651150
[Cloth3]          93.256854
[Cloth3]          41.552272
[Baby1]           86.632419
[Baby1]           15.481971
[Rocks1]          93.103975
[Rocks1]          44.372019
[Lampshade2]      65.176955
[Lampshade2]      12.695837
[Flowerpots]      90.497866
[Flowerpots]      47.730843
[Midd2]           66.247104
[Midd2]           30.431838
[Baby3]           88.249119
[Baby3]           33.716989
[Cloth4]          91.010549
[Cloth4]          15.795518
[Bowling1]        61.824486
[Bowling1]        17.068507
[Rocks2]          93.700477
[Rocks2]          37.656598
[Wood2]           67.785648
[Wood2]           11.588071
```

> SSD method has a higher computational complexity compared to SAD algorithm as it involves numerous multiplication operations. Normally, two areas which consist of exactly the same pixel values would yield a score of zero. However, these measures will no longer yield the correct results in the case of radiometric distortion, i.e., **where the pixel values in one image differ from those in the other image by a constant offset and/or gain factor.**
>
> …
>
> NCC algorithm is robust to the linear variation in the brightness due to different illumination conditions to the cameras. But it can be seen that due to more complex calculations of division, multiplication and square root its computational time is more than SAD, SSD. Hence it could only be used for real time application only if we are able to develop more efficient algorithm to speed up matching process.

From Papper :Comparison of Various Stereo Vision Cost Aggregation Methods

In a word, NCC perform better than SSD when one image differ from those in the other image by a constant offset but cost more time.

5.Cost aggregation using Adaptive Support Weight (ASW) is a powerful local stereo approach. You are required to implement this approach described in the CVPR'05 paper "Locally Adaptive Support-Weight Approach for Visual Correspondence Search" (to save time you are recommended to read mainly sec. 2.6 and sec. 3 of the paper). Visualize and save your estimated disparity maps to testcasename disp1 ASW.png

or testcasename disp5 ASW.png. Also write down the quality of your disparity maps. Note that this algorithm is in general a bit slow.

```
[Baby2]          20.133499
[Plastic]        25.738930
[Lampshade1]     16.047063
[Wood1]          22.077000
[Cloth2]         20.848262
[Bowling2]       31.101214
[Midd1]          29.808777
[Cloth1]         11.200985
[Aloe]           18.427116
[Monopoly]       37.651150
[Cloth3]         41.552272
[Baby1]          15.481971
[Rocks1]         44.372019
[Lampshade2]     12.695837
[Flowerpots]     47.730843
[Midd2]          30.431838
[Baby3]          33.716989
[Cloth4]         15.795518
[Bowling1]       17.068507
[Rocks2]         37.656598
[Wood2]          11.588071
```

6.6. Explain the idea of the ASW paper in question 5. Also explain in which ways ASW is better than NCC. Why?

The core idea of ASW(Adaptive Support-Weight Approach) is to give a weight value to each pixel in the matching window, which is based on the color difference and the distance between them and the center point of the window. In essence, an approximate image segmentation is completed.

According to the like between location and color difference of window in the original pixel cost Fu to different weighting, and to perform polymerization. Many studies show that the algorithm is all local stereo matching effect is the best, and the result of matching algorithm and the optimization results compared. But the algorithm of adaptive weighted algorithm is relatively slow, and the complexity of the algorithm is relatively high, and the weight of the storage space is very large.

ASW is better than NCC, but NCC is very slow. In the case that the images are colorful(3 channels), ASW is good. But for gray images, ASW are not necessary because color images do not have the definition of color difference(cq).

implementation details

Code sturcture:

```
bat.py #the program to process all the images
```

```
stero0.cpp #the program to process SSD and NCC
stero1.cpp #the program to process CNCC and CSSD
stero2.cpp #the program to process ASW
```

For basic SSD and NCC and also ASW, the main algorithm is as follow:
1. Read the image into memory
2. Traversal the image, process each point and calculate d for the destination image.
2.1. for each of the point, calculate a patch from the left(right) image and then travel from 0 - dmax and calculate another patch from the right(left) image.
2.2 use the setting patch to calculate match cost and then choose the best match cost to be arg d.
3. Save the image

For SSD:
The matching cost image is calculate by the following fomula:

$$C(x,y) = \sum_{i,j \in N} (N_p(i,j) - N_q(i,j))^2$$

i.e. sum the absolute square value of the corresponding patches.

```
for(int i = 0; i < P.rows; i++) {
  for(int j = 0; j < P.cols; j++) {
    int difference = P.at<int>(i,j) - Q.at<int>(i,j);
    sum += difference * difference;
  }
}
```

For NCC:

$$D(p,q) = \frac{1}{n} \cdot \sum_{i,j \in N_{pq}} \frac{(f_p(i,j) - \hat{f}_p) \cdot (g_q(i,j) - \hat{g}_q)}{\sigma f_p \sigma g_p}$$

For ASW:(it will be more diffcult)
In the paper, we can conclude the several formulas:
Matching Value:
$d_p = arg_{d \in \{d_{min}....d_{max}\}}$E(p,pd)
Matching Cost:
$E(p,pd) = \frac{\sum_{q \in W}w(p,q)w(p_d,q_d)e_0(q,qd)}{\sum_{q\in W}w(p,q)w(pd,qd)}$

```
double Eppd(const Mat & P, const Mat & Q, int d) {
  Vec3b p = P.at<Vec3b>(P.rows / 2, P.cols / 2);
  Vec3b pd = Q.at<Vec3b>(Q.rows / 2, Q.cols / 2);

  double sum1 = 0;
  double sum2 = 0;
```

```
    for(int i = 0; i < P.rows; i++) {
      for(int j = 0; j < P.cols; j++) {
        sum1 += wpq(p, P.at<Vec3b>(i,j), d) * wpq(pd, Q.at<Vec3b>(i,j), d) *
  e0(P.at<Vec3b>(i,j), Q.at<Vec3b>(i,j));
        sum2 += wpq(p, P.at<Vec3b>(i,j), d) * wpq(pd, Q.at<Vec3b>(i,j), d);
      }
    }
    return sum2 != 0 ? sum1 / sum2 : 0;
  }
```

Adptive Weight:
$w(p,q)=k\cdot exp(-(\frac{\Delta cpq}{\gamma c} + \frac{\Delta gpq}{\gamma p}))$

```
  double wpq(const Vec3b & p, const Vec3b & q, int d) {
    double k = 100;
    double gamac = 40;
    double gamap = 7;
    return k*exp(-(deltaCpq(p,q) / gamac + d / gamap));
  }
```

Error modification:
$e0(q,qd) = \sum_{c \in \{R,G,B\}}|I_c(q)-I_c(qd)|$

```
  double e0(const Vec3b & q, const Vec3b & qd) {
    return abs(q[0] - qd[0]) + abs(q[1] - qd[1]) + abs(q[2] - qd[2]);
  }
```

Color difference:
RGB calculation:
$C_{pq} = \sqrt{\omega_r(p_R-q_R)^2+\omega_g(p_G-q_G)^2+\omega_b(p_B-q_B)^2}$
with($(\omega_r,\omega_g,\omega_b)=(3,4,2)$)
Lab calculation:
$C_{pq} = \sqrt{(L_1^*-L_2^*)^2+(a_1-a2)^2+(c_1-c_2)^2}$

```
  // Lab space color difference calculation
  double deltaCpq(const Vec3b & p, const Vec3b & q) {
    double pXYZ[3] = {0};
    double qXYZ[3] = {0};
    double pLab[3] = {0};
    double qLab[3] = {0};

    pXYZ[0] = p[0] * 0.4124 + p[1] * 0.3576 + p[2] * 0.1805;
    pXYZ[1] = p[0] * 0.2126 + p[1] * 0.7152 + p[2] * 0.0722;
    pXYZ[2] = p[0] * 0.0193 + p[1] * 0.1192 + p[2] * 0.9505;

    qXYZ[0] = q[0] * 0.4124 + q[1] * 0.3576 + q[2] * 0.1805;
```

```
    qXYZ[1] = q[0] * 0.2126 + q[1] * 0.7152 + q[2] * 0.0722;
    qXYZ[2] = q[0] * 0.0193 + q[1] * 0.1192 + q[2] * 0.9505;

    const double Xn = 95.047;
    const double Yn = 100.0;
    const double Zn = 108.883;

    qLab[0] = 116 * f(qXYZ[1] / Yn) - 16;
    qLab[1] = 500 * (f(qXYZ[0]) / Xn - (f(qXYZ[1]) / Yn));
    qLab[2] = 200 * (f(qXYZ[1]) / Yn - (f(qXYZ[2]) / Zn));

    return sqrt((pLab[0] - qLab[0]) * (pLab[0] - qLab[0]) + (pLab[1] - qLab[1]) *
(pLab[1] - qLab[1]) + (pLab[2] - qLab[2]) * (pLab[2] - qLab[2]));
}

//RGB space color difference calculation
 double deltaCpq(const Vec3b & p, const Vec3b & q) {
    const int wr = 3;
    const int wg = 4;
    const int wb = 2;
    return sqrt(wr*(p[0] - q[0]) * (p[0] - q[0]) + wg*(p[1] - q[1]) * (p[1] -
q[1]) + wb*(p[2] - q[2]) * (p[2] - q[2]));
 }
```

discussions of the disadvantages of the paper

Advantages:

1. The matching method can make a better result than both SSD and NCC. It can use a detail method to define the matching cost in a good way. Using the definition of color difference and the “adptive weight” is truly a good approach.
2. It’s a landmark works in local stereo matching algorithm(maybe but i think it’s not good, maybe it’s my problem). From CVPR and PAMI you can see its component. AdaptiveWeight method has been proposed, formally declared in the matching accuracy, the adaptive Adaptive out algorithm (Window)(This method is unscientific at all i think). And later became popular.

Disadvantage:

1. Too many formulas :<. They are supposed to give an example program but didn’t. :<
2. The algorithm is so slow that i can not stand it, and my computer can not afford a argument of window size of 33. :<
3. Locally approach can not do as well as global one. But there is so many projects for me to finish, i do not have time to do it. :<

Conclusion:

This project is in fact very boring and seems that it's irrelevant to what we have learnt. I have many unsolved questions:

1. Why my program is so slow(not a bit slow)? How can I improve it?

I used python as program language for the last 4 homework project. But this time i use c++ for programming, I think it should be quicker because c++ is a basic high level language and is known to be 10 times faster than python. But thep processing speed is stiil very slow.

2. Why my picture is so ugly compare to the ground truth?

Many of my fellows also have this problem that the output image is very ugly compare to the groud truth, especially for the edge of the image. We check it for several times and discuss it for a better way but failed. We try to ask TA for help, but no responds :<. We also ask professor Chao for help but she tells us to ask TA for help.

3. How to calculate the error pixels since the error rate in the paper is less than 5% when window size is set to 5, but my error rate is greater than 70% for average?

We should calculate absolute error in the paper that error < 1. But, the error rate is so high. I also ignore a border of 10 in the image but also very high.

In a word, we have still some problems. My be we can do better with more time. :<