Lab 5: Image Sequence Processing

ECE 5470

Computer Vision

Name: Yanling Wu

NetID: yw996

Section 2: Feature Tracking

In this section, we firstly got familiar with how to use the tracking program to follow a set of image feature. Then, we marked different eight points in taxi.vs image set to track the image.

In hand.vs image set, this program did very well. It did track the movement of marked points. The square of marked point changed as the boy's and girl's movement in the image.

Also, we do the similar process to the taxi images set. I cannot decide how to mark eight points in the image because there are only four objects to move in the images set. So, I choose two ways to mark. One is to mark these four objects and then choose other four objects that might move but they are still in the image. Another way is only to mark these four objects and mark two points at the same moving car. Besides, I used different patch size. The first way, I choose "h=10, v=10" and I use "h=5, v=5" in second way.

In the process of processing the taxi images set, this program can also track the movement of the objects. But I also notice that if the points that are marked at the edge of the objects, the sensitivity of tracking will be better than those points that are marked at the middle of objects, which means the points should be marked at the area where the gradient is sharp, and the accuracy will be bigger.

Besides, I put the data of location that I marked in the appendix. The first one is for figure 2.1 and the second one is for figure 2.2.

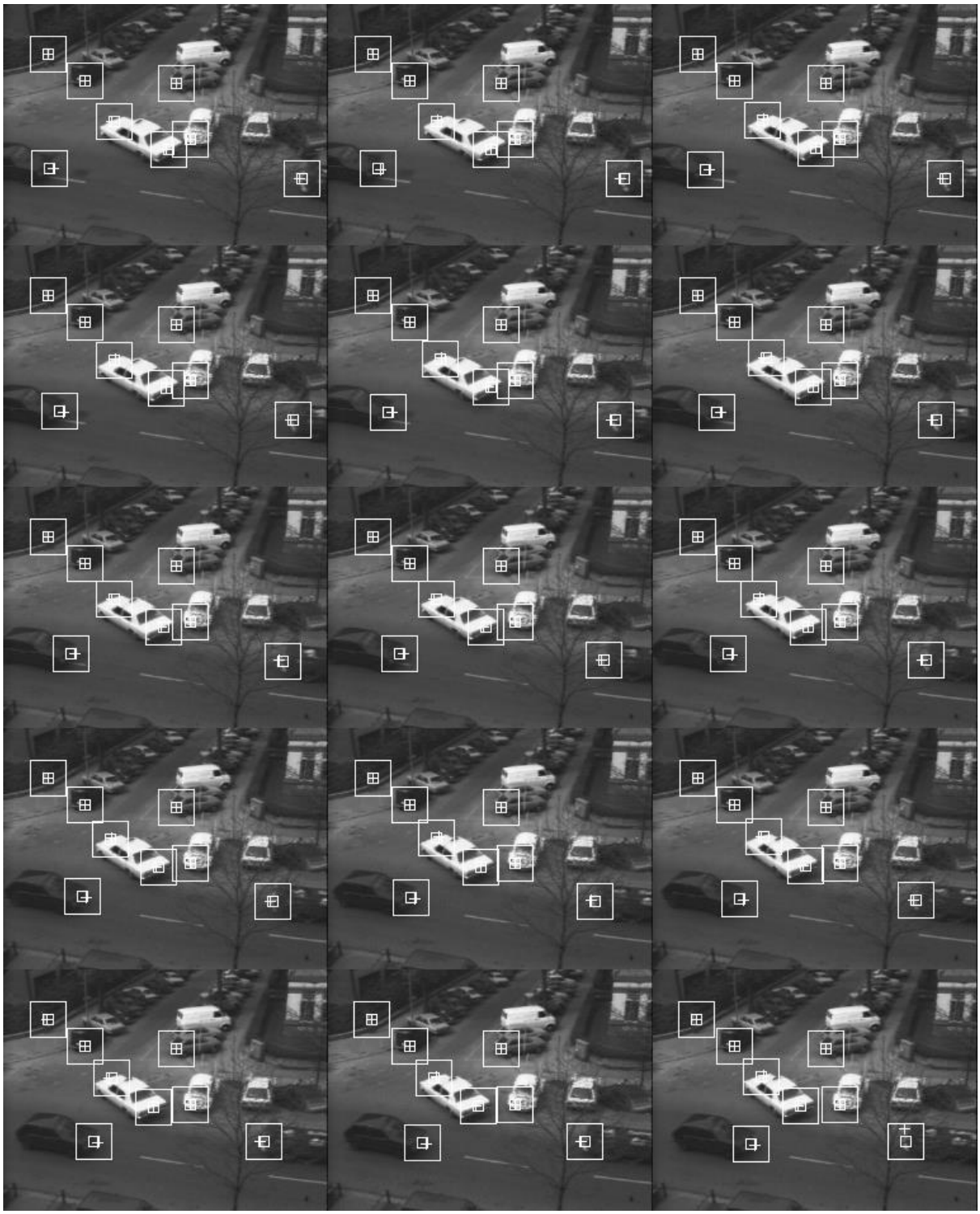

Figure 2.1 the first way to mark the points

Figure 2.2 the second way to mark the points

Section 3: Temporal Domain Filtering

Q1: What does the temporal mean filter do?

This temporal mean filter calculates the average pixel value of specific number of pictures in every location of the image and make this average as the pixel value of this location.

Q2: What is the effect of a larger window size?

Larger window size will affect the number of images that are used to calculate the average of pixel value. The blurred area of edge becomes larger.

Test the program with hand.vs

Process the *hand.vs* sequence and get the compared image as below shown. The whole image is the figure 3.1. The left image is the one that was processed by mean filtered program and the right one was processed by median filter. From this image, we can some edge of the picture become less blur and cannot see the tendency of moving clearly in median filtered image. After amplifying some part of the image, we can clearly see the change of boundary in image as shown in figure 3.2 and 3.3. In median filtered image, the boundaries of the arms and the boy's head become clear.

The reason of this might be the way of calculating mean of the set of images consider all the images' pixel value and the objects in the image are to move or have the tendency to move. As a result, the area of edge will become blur after mean filtered. Although the median filter considers all three images' pixel value to sort and get the median, there is bigger possibility to offset the effect of move and get the pixel value of middle image out of three image set.

Figure 3.1 The comparison images of mean filtered and median filtered images

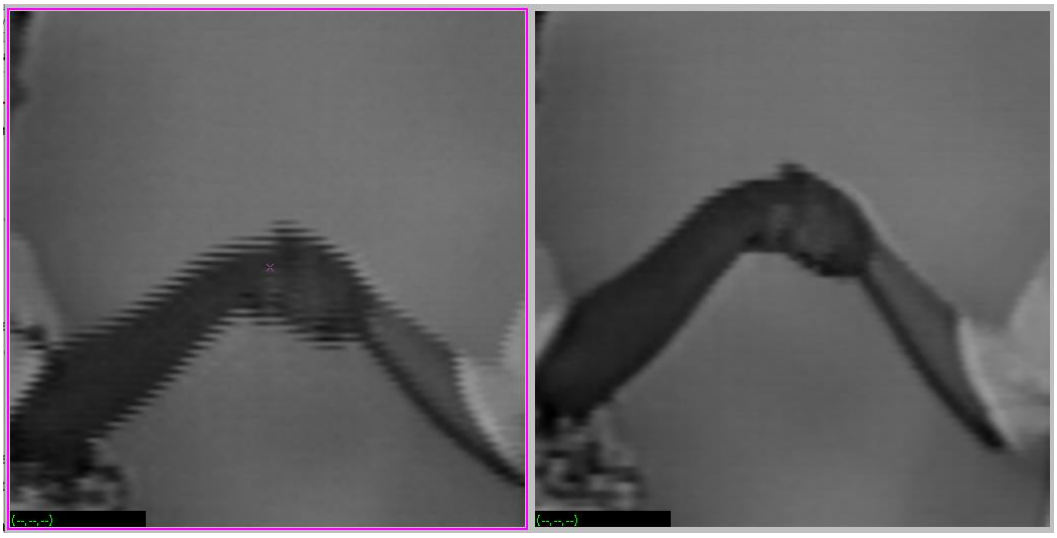

Figure 3.2 The amplified image in arms' part

Figure 3.3 The amplified image in the boy's head part

The compared result 0f lab5.vs

The compared result images show below processed by vssum.c and vsmed.c.

Firstly, the original image set consist of one normal image, one picture with salt and pepper noise and one normal image, one bear image, and three normal images but are moved to left and down.

Besides, the method of calculating mean of three images will take all the information of these three images into account. For example, the second image with salt and pepper noise, which often has extreme pixel value such as 0 or 255, will have huge influence on the first two results and this is the reason why the first tow result image both have salt and pepper noise Also, because the third image in original image sequence is the bear image, so, if we need to use this image to calculate the mean of the pixel value, the bear will affect the final result. Hence, we can see the bear from the three result images in the middle. What's more, because the last three images of the women are moved in left and down direction, we can see the blur in the result image and the tendency of moving. To sum up, the effect of mean filter will show all image information that are calculated.

However, the median filter can remove the effect of some extreme pixel values. For instance, if there is only one image with the extreme pixel values in three images that are calculated, the extreme value will be sorted as maximum or minimum value, which is no effect to the final pixel value. That is why the image processed by vsmed.c have fewer result images with salt and pepper noise. The most interesting but tricky part is the

second result image. Because this image is generated by three different images. One is the women image with salt and pepper noise, the second one is the normal woman image, and the third image is the bear one. Because there are two same woman images, though one with noise, the median after sorted still is the pixel value of the woman's image. However, the exist of bear image could help to remove the effect of salt and pepper noise (extreme pixel value).

Figure 3.4 The image sequence of lb5.vs processed by vssum.c

Figure 3.5 The image sequence of lb5.vs processed by vsmed.c

The code of vsmed.c and vdif.c are listed in Appendix.

Section 4: Change Detection

Q1: What effect does the threshold have on the filtered sequence?

The processed image by threshold could make the pixel value change clearer between two consecutive images and make images becomes binary images. So, It is easy for us to see the moving objects and study the sequence image.

Q2: What is a good threshold for this image set?

A good threshold could help us to tell which object moved and which objects stayed still from the image set. So, for this image set, a good threshold is the one that could mark the moving cars in image sequence. When the threshold is big, only the part that change sharply could be marked, which means it will lose those that moved but did not move sharply and vice versa. Figure 4.1 shows the two bad threshold values. The threshold of left image is 5, which is small and a lot of objects that did not move showed in the picture and the right one is 30, which is big and the other two cars that moved but did not move so much obviously did not show. Hence, I choose the threshold value is 15. And the result shows in the figure 4.2. We can see all three cars in the image set.

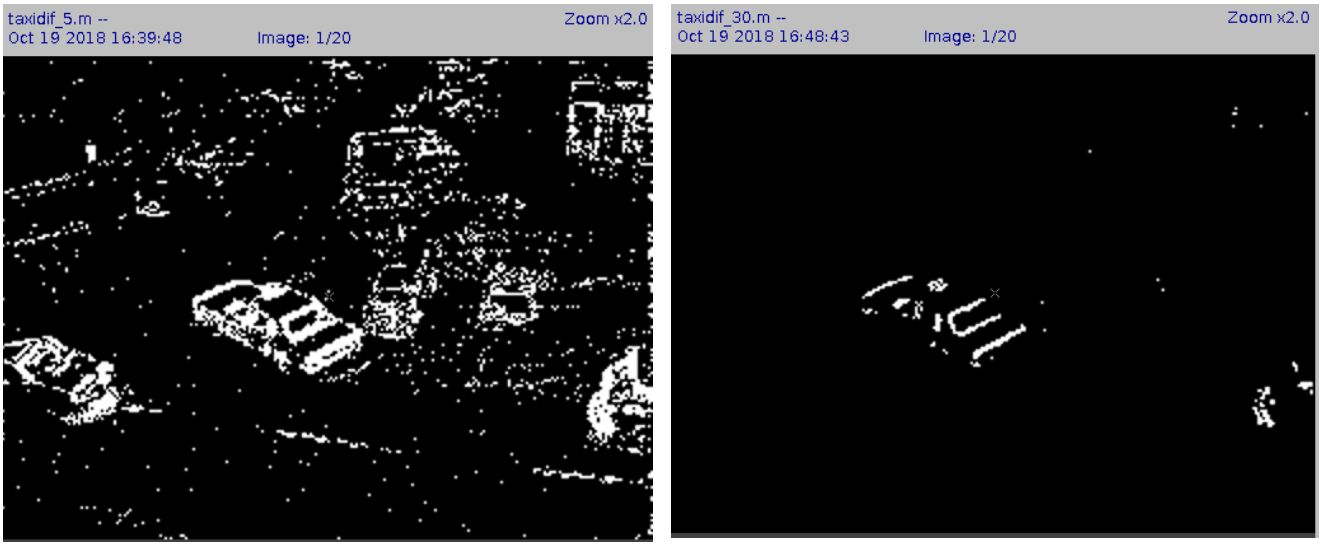

Figure 4.1 the examples of bad threshold

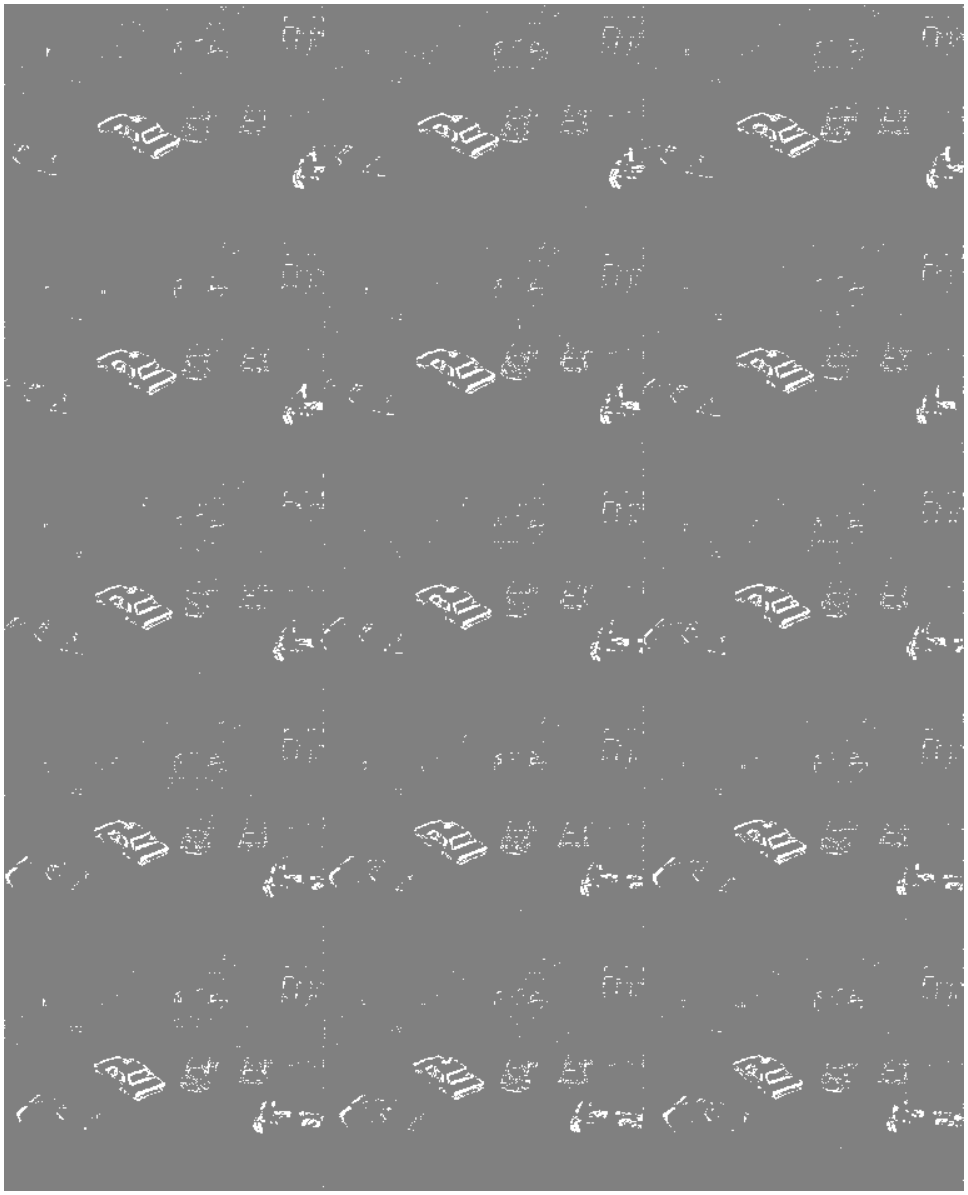

Figure 4.2 The image set of taxi.vs using th=15

Code Appendix:

Location1

```
8
36 67
35 153
87 97
128 75
235 52
155 83
136 126
61 133
```

Location2

```
8
5 78
33 62
35 153
88 97
108 91
130 75
235 52
250 68
```

vmed.c

```
/*************************************************************************/
/* vsmed.c   Compute the median of the set of images                     */
/* Yanling Wu                                                            */
/* NetID: yw996                                                          */
/* Lab5                                                                  */
/*************************************************************************/

#include "VisXV4.h"          /* VisionX structure include file       */
#include "Vutil.h"           /* VisionX utility header files         */

VXparam_t par[] =            /* command line structure               */
{
{    "if=",    0,   " input file vssum: compute temporal mean"},
{    "of=",    0,   " output file "},
{    "n=",     0,   " number of frames "},
{     0,       0,    0}
};
#define  IVAL   par[0].val
```

```
#define  OVAL   par[1].val
#define  NVAL   par[2].val

Int main(argc, argv)
int argc;
char *argv[];
{
V3fstruct (im);
V3fstruct (tm);
int        x,y,z;           /* index counters                */
int        n;               /* Number of frames to average    */
int        val1, val2;

    VXparse(&argc, &argv, par); /* parse the command line    */
    n = (NVAL ? atoi(NVAL) : 3); /* read n, default is n=1   */

    while (Vbfread( &im, IVAL, n)) {
       if ( im.type != VX_PBYTE || im.chan != 1) { /* check format  */
           fprintf (stderr, "image not byte type\n");
           exit (1);
        }
        for (y = im.ylo; y <= im.yhi; y++) {
           for (x = im.xlo; x <= im.xhi; x++) {
                 val1 = 0;
                 val2 = 0;
                 z = im.zlo;
                 if(im.u[z][y][x] >= im.u[z+1][y][x]){
                    val1 = im.u[z][y][x];
                    val2 = im.u[z+1][y][x];
                 }
                 else{
                    val1 = im.u[z+1][y][x];
                    val2 = im.u[z][y][x];
                 }
                 if(val1 > im.u[z+2][y][x]){
                     if(val2 >= im.u[z+2][y][x])
                        im.u[0][y][x] = val2;
                    else
                        im.u[0][y][x] = im.u[z+2][y][x];
                 }
                 else{
                    im.u[0][y][x] = val1;
                 }
            }
```

```
        }
        V3fwrite (&im, OVAL); /* write the oldest frame */
    }
    exit(0);
}
```

vsdif.c

```
1.  /*********************************************************************/
2.  /* vsdif   Set the threshold of the set of imagesize             */
3.  /* Yanling Wu                                                        */
4.  /* NetID: yw996                                                      */
5.  /* Lab5                                                              */
6.  /*********************************************************************/
7.  #include "VisXV4.h"          /* VisionX structure include file       */
8.  #include "Vutil.h"           /* VisionX utility header files         */
9.
10. VXparam_t par[] =            /* command line structure               */
11. {
12. {    "if=",    0,   " input file vssum: compute temporal mean"},
13. {    "of=",    0,   " output file "},
14. {    "n=",     0,   " number of frames "},
15. {    "th=",     0,   " value of threshold "},
16. {     0,       0,    0}
17. };
18. #define  IVAL   par[0].val
19. #define  OVAL   par[1].val
20. #define  NVAL   par[2].val
21. #define  TVAL   par[3].val
22.
23. Int main(argc, argv)
24. int argc;
25. char *argv[];
26. {
27. V3fstruct (im);
28. V3fstruct (tm);
29. int        x,y,z;           /* index counters                 */
30. int        n,th;               /* Number of frames to average    */
31. int        sum;
32.     VXparse(&argc, &argv, par); /* parse the command line    */
33.
34.     n = (NVAL ? atoi(NVAL) : 2); /* read n, default is n=1   */
35.  th = (TVAL ? atoi(TVAL): 15);/*read th, default is th=128*/
```

```
36.
37.     while (Vbfread( &im, IVAL, n)) {
38.  if ( im.type != VX_PBYTE || im.chan != 1) { /* check format  */
39.            fprintf (stderr, "image not byte type\n");
40.            exit (1);
41.         }
42.         for (y = im.ylo; y <= im.yhi; y++) {
43.            for (x = im.xlo; x <= im.xhi; x++) {
44.               z = im.zlo;
45.               if(abs(im.u[z][y][x] - im.u[z+1][y][x])>= th)
46.                     im.u[z][y][x] = 255;
47.               else
48.                     im.u[z][y][x] = 128;
49.               }
50.             }
51.         }
52.         V3fwrite (&im, OVAL); /* write the oldest frame */
53.     }
54.     exit(0);
55. }
```