ECE 5470 Computer Vision Lab5 report

Image Sequence Processing

Weiran Wang

NetID: ww463

Section 2:

In fig1, there are three points being tracked and the points coordinates are given in file loc1. the small box is the location of a feature in the previous frame, the large box is the size of the search area, and the + sign indicates the location of the best match for the feature in the current frame.

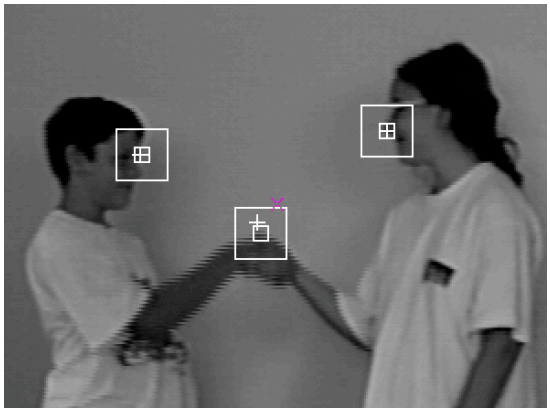

Fig1 feature tracking of the first frame of hand.vs

With vtile command, we are able to display the image sequence as a single image. The large image fig2 shown below is composed of 3*4 small images in the sequence

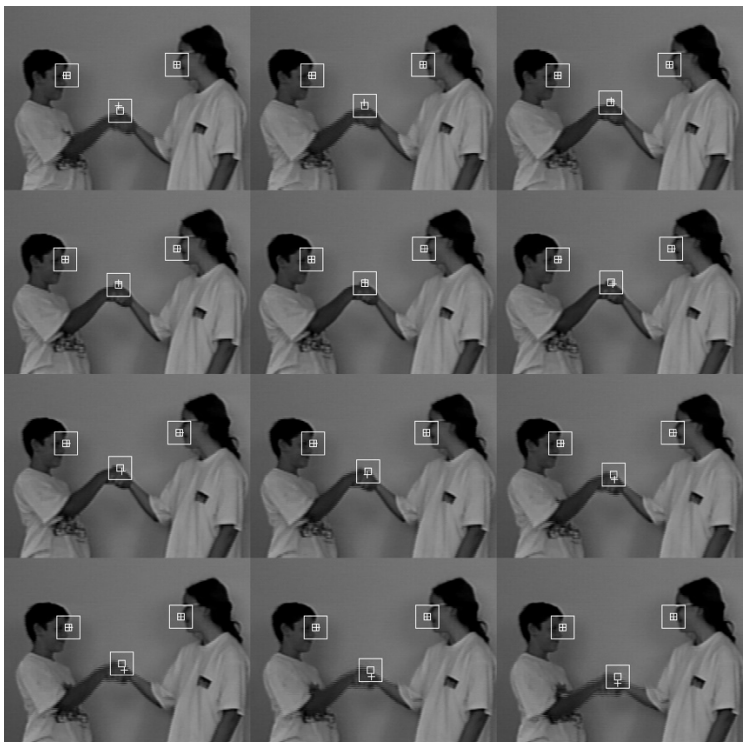

Fig2 feature tracking of hand.vs

We can also use vpr command to print out trk.g which has space-time trajectories for the patches. V3d command is another intuitive way to learn trajectories for the patches. And the outputs are shown below

```
ECE5470:~/lab5 [11:44pm] 423$vpr trk.g
[File Name      ]: [6] trk.g
[File Command   ]: [57] vtrack if=hand.vs pf=loc1 h=10 v=10 of=trk.vs gf=trk.g
[File Machine   ]: [6] linux
[File User ID   ]: 2022
[File Date      ]: 1571705595
[File History   ]: [96]
vtrim s=300,224 p=153,36 of=h if=hand
addvisx of=hand,f=100,y=15,x=-1
vcp hand.s -o hand.vs

[Bounding Box   ]: [6]
          0         299           0         223           0          14
[Object ID.     ]: 2
[3-D vector     ]: [45]
         141          96           0         139         102           1
         138         106           2         139         108           3
         139         110           4         139         111           5
         141         109           6         143         105           7
         142         101           8         143          95           9
         146          87          10         147          79          11
         147          71          12         148          61          13
         148          51          14
[Object ID.     ]: 1
[3-D vector     ]: [45]
         210         152           0         210         152           1
         210         152           2         210         152           3
         211         152           4         212         152           5
         213         152           6         214         152           7
         214         152           8         215         152           9
         215         152          10         215         152          11
         215         152          12         214         152          13
         212         152          14
[Object ID.     ]: 0
[3-D vector     ]: [45]
          76         139           0          75         139           1
          74         139           2          74         139           3
          74         139           4          74         139           5
          75         139           6          76         139           7
          77         139           8          78         139           9
          79         139          10          80         139          11
          80         139          12          80         139          13
          80         140          14
```

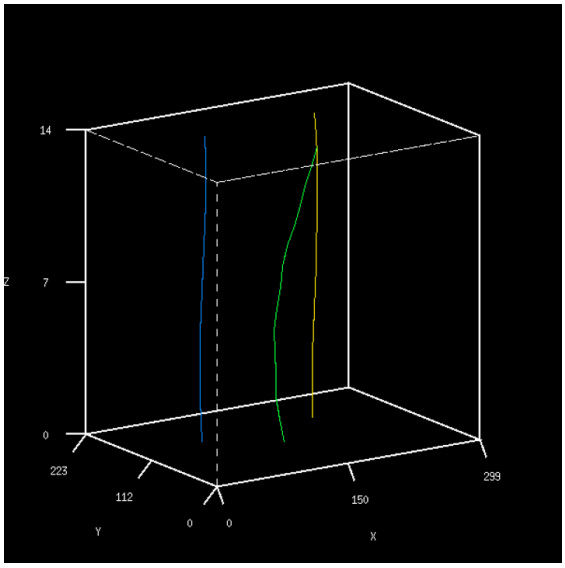

Fig3 two ways to learn trajectories for patches

To track features in taxi sequence, I spotted eight coordinates, and they are (35, 153) (27,57) (37,64) (87, 96) (120,70) (140, 80) (239, 44) (251,62). These eight points are important because these are dynamic areas in the image sequence. The head and the tail of the moving cars and the pedestrian are critical features in this image sequence.

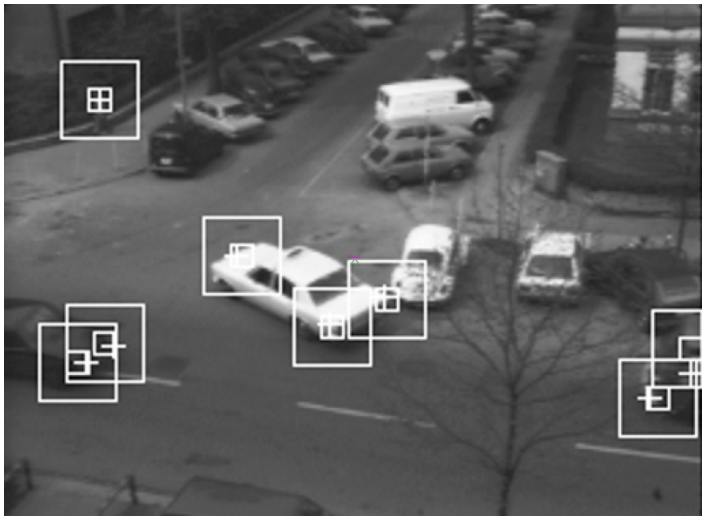

Fig4 feature tracking of the image taxi

By keep tracking these 8 features, the result is shown in one large image below.

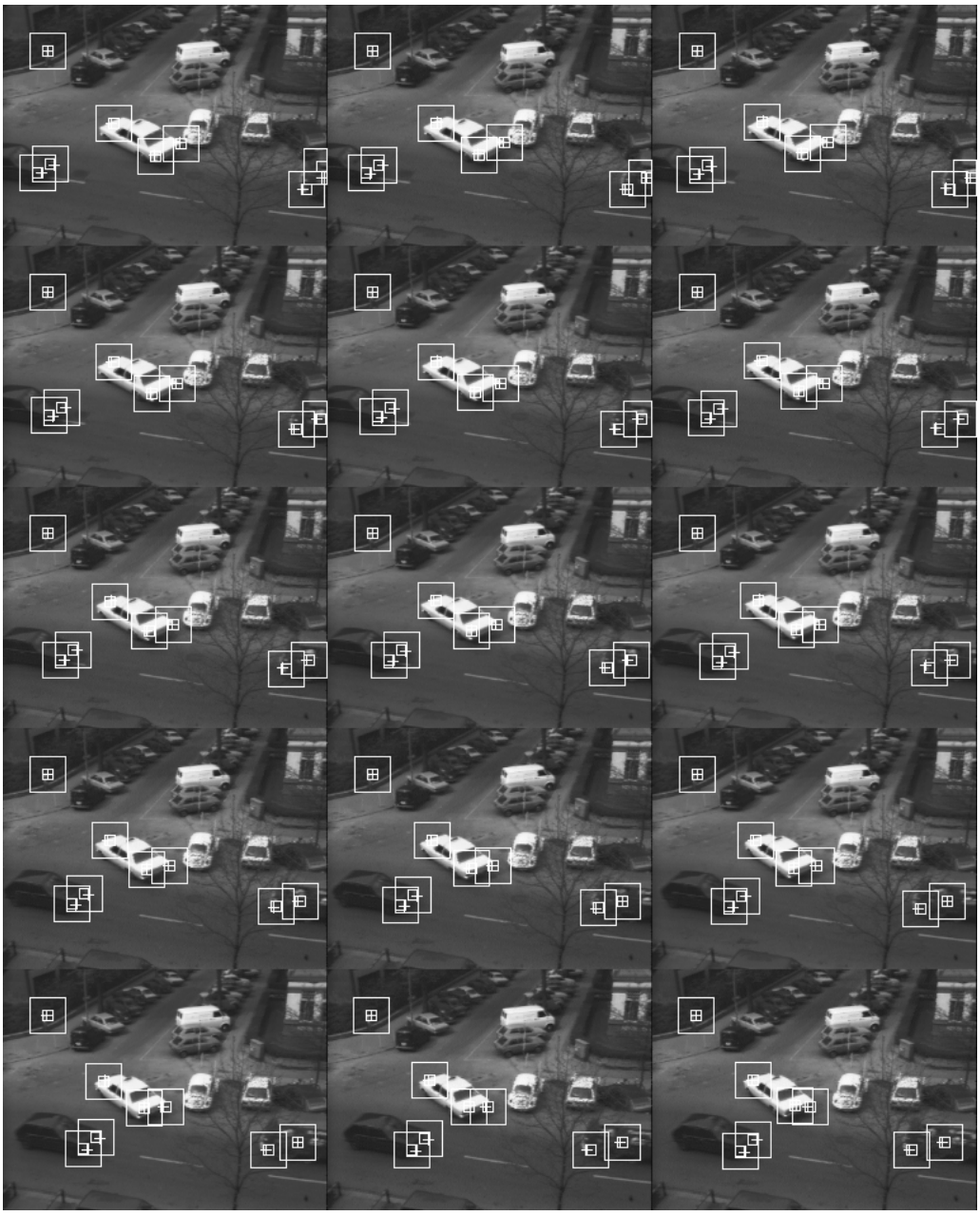

Fig5 feature tracking of the taxi.vs image sequence

The temporal mean filter computes the mean value of n consecutive images and use the mean value as the output. For example, when the window size n=3, it takes the first three images in the sequence as input. For each pixel on the image, we compute the mean value among three images and use it as the output. The second output image would be the mean value of the 2nd 3rd and 4th images in the original sequence. So the output sequence always has m-n+1 images. m refers to number of images in the original sequence, n is the window size. The result of applying temporal mean filter on hand.vs is

shown below

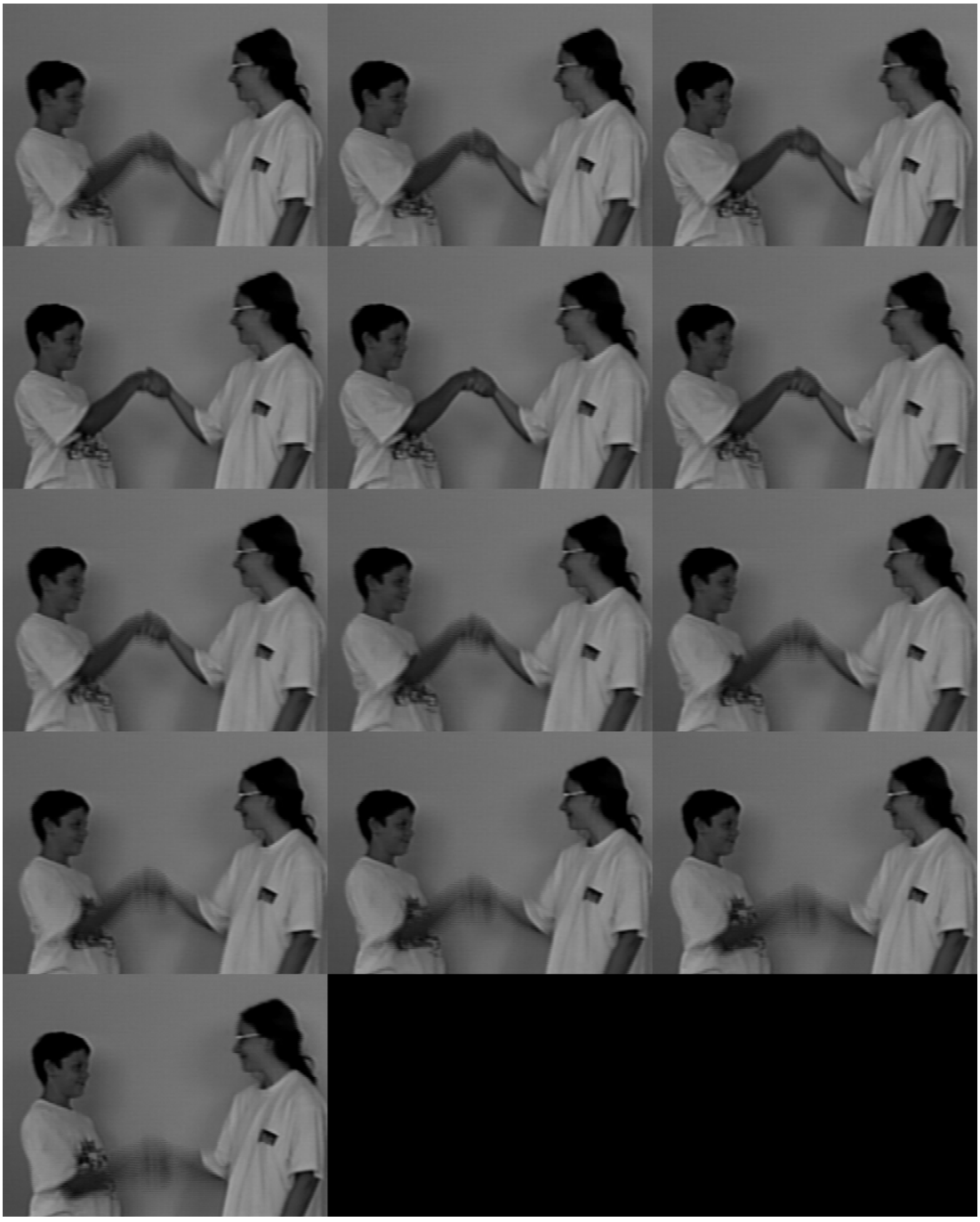

Fig6 mean filtering output of hand.vs with n=3

The result of applying temporal mean filter on hand.vs with window size n=5 is shown below

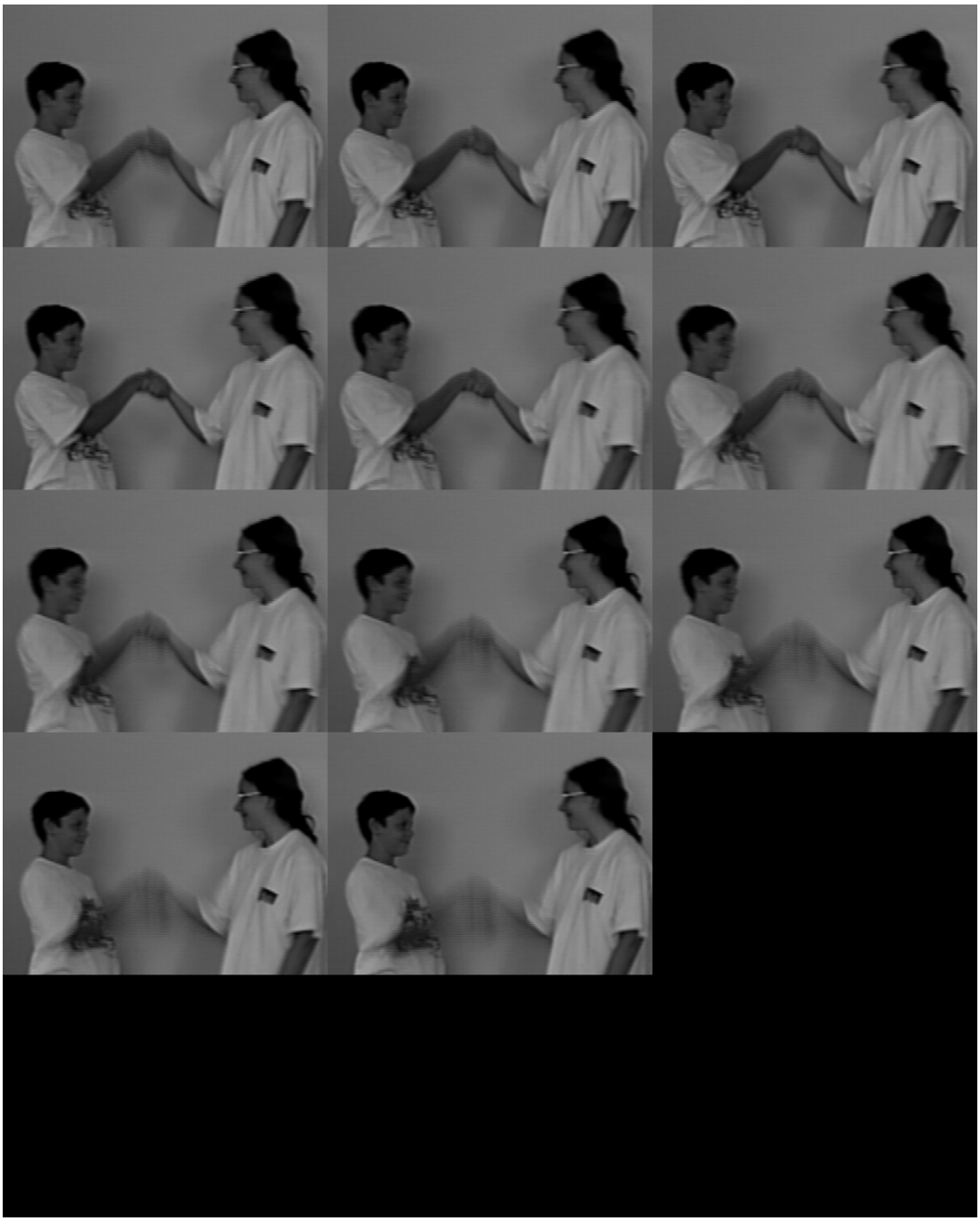

Fig7 mean filtering output of hand.vs with n=5

After applying temporal mean filter, the images get more blurry, and there some overlapped area around the children's hands area. When the window size is 5, the blurry and overlapped area is larger than n=3, which makes sense because the mean value is taken from more images. More images would have a larger overlapped area when hands are shaking.

Median filter programming

To implement the median filter with window size n=3, the only thing we need to change from the mean filter is taking the median value from three consecutive images rather than the mean value. To find the median, I compare the pixel value of three consecutive images and use the median as the output. The programming is provided below

```
 4
 5  #include "VisXV4.h"          /* VisionX structure include file       */
 6  #include "Vutil.h"           /* VisionX utility header files         */
 7
 8  VXparam_t par[] =            /* command line structure               */
 9  {
10  {    "if=",     0,   " input file vssum: compute temporal mean"},
11  {    "of=",     0,   " output file "},
12  {    "n=",      0,   " number of frames "},
13  {     0,        0,    0}
14  };
15  #define  IVAL   par[0].val
16  #define  OVAL   par[1].val
17  #define  NVAL   par[2].val
18
19  int
20  main(argc, argv)
21  int argc;
22  char *argv[];
23  {
24  V3fstruct (im);
25  V3fstruct (tm);
26  int        x,y,z;           /* index counters                  */
27  int        n;               /* Number of frames to average     */
28  int        sum;
29  int num1;
30  int num2;
31  int num3;
32  int res;
33
34      VXparse(&argc, &argv, par); /* parse the command line     */
35
36      n = (NVAL ? atoi(NVAL) : 1); /* read n, default is n=1    */
```

```
37
38      while (Vbfread( &im, IVAL, n)) {
39      if ( im.type != VX_PBYTE || im.chan != 1) { /* check format  */
40              fprintf (stderr, "image not byte type\n");
41              exit (1);
42          }
43          for (y = im.ylo; y <= im.yhi; y++) {
44              for (x = im.xlo; x <= im.xhi; x++) {
45                if(im.u[0][y][x] <= im.u[1][y][x]){
46                   num1 = im.u[0][y][x];
47                   num2 = im.u[1][y][x];
48          }
49           else{
50          num1 = im.u[1][y][x];
51          num2 = im.u[0][y][x];
52          }
53           if(im.u[2][y][x] <= num1)
54          res = num1;
55           else if(im.u[2][y][x] >= num2)
56          res = num2;
57           else
58          res = im.u[2][y][x];
59           im.u[0][y][x] = res;
60              }
61          }
62          V3fwrite (&im, OVAL); /* write the oldest frame */
63      }
64      exit(0);
65  }
```

Fig8 Code for median filtering

The figures below is the median filtering result of the sequence hand.vs and lb5.vs. As you can see in fig11, the images are blurry than original image sequence but less blurry than the result of mean filter. The overlapped arear between the children's hands are also smaller. For lb5.vs sequence, median filter performed much better than the mean filter in terms of filtering out salt and pepper noise. Also, I noticed that there is a cartoon character in the image sequence which should be considered as an error or noise. With mean filter, we can still see the shadow of the cartoon character, which is not very optimal. With median filter, the cartoon character can barely been seen in the output sequence. So median filter is better at filtering noise than mean filter.

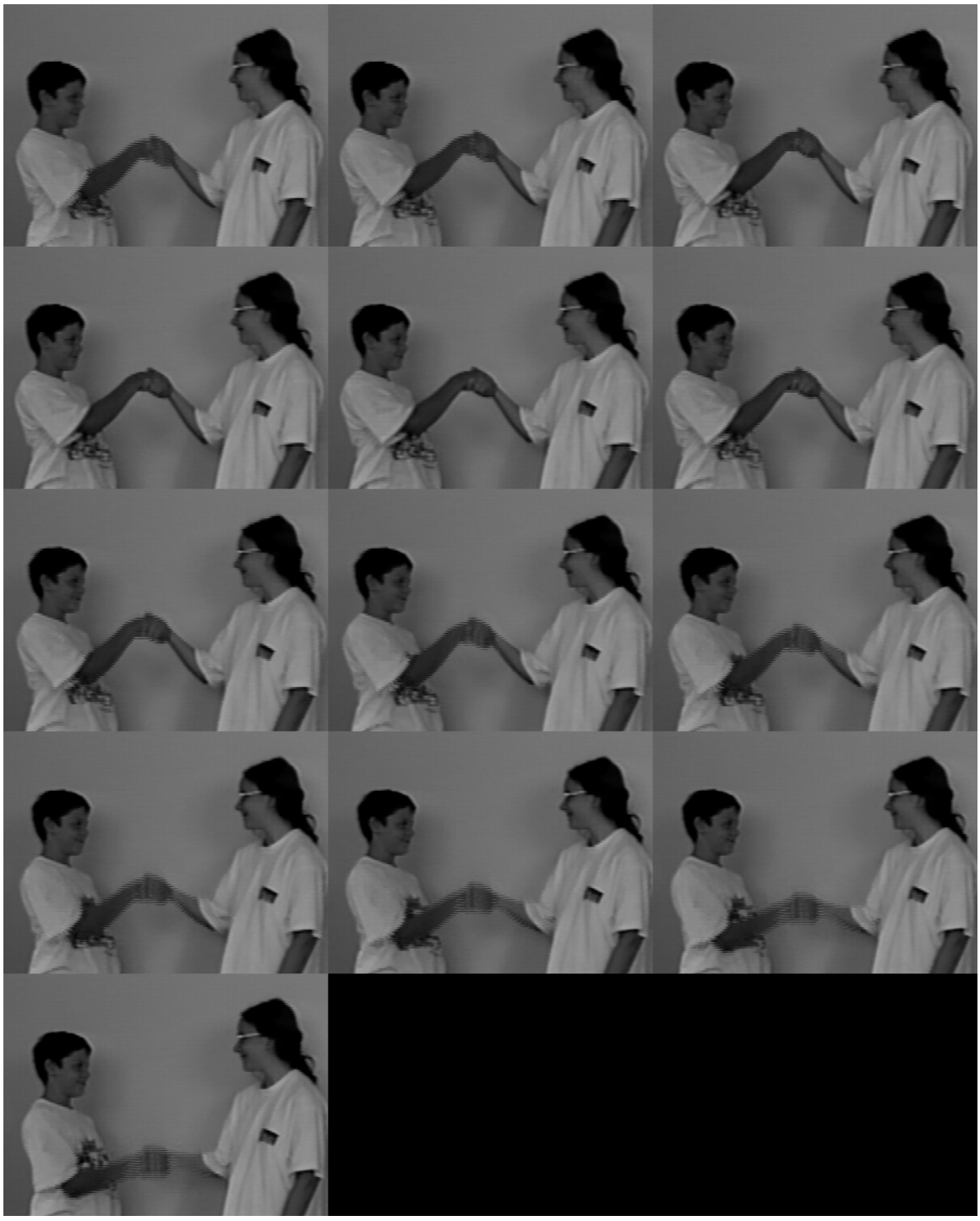

Fig9 median filtering output of hand.vs with n=3

Fig10 mean filtering output of lb5.vs

Fig11 median filtering output of lb5.vs

Change Detection Programming

To implement the difference filter, I started by reading in two consecutive images in the sequence can compute the difference between them. If the value difference exceeds the given threshold, I set the pixel value to be 255, if the difference is smaller than the threshold, the pixel value will be set to 128. The details the program is shown below.

```
 5  #include "VisXV4.h"          /* VisionX structure include file       */
 6  #include "Vutil.h"           /* VisionX utility header files         */
 7
 8  VXparam_t par[] =            /* command line structure               */
 9  {
10  {    "if=",     0,    " input file vssum: compute temporal mean"},
11  {    "of=",     0,    " output file "},
12  {    "n=",      0,    " number of frames "},
13  {    "th=",     0,    " threshold"},
14  {     0,        0,     0}
15  };
16  #define  IVAL   par[0].val
17  #define  OVAL   par[1].val
18  #define  NVAL   par[2].val
19  #define  THRESH par[3].val
20
21  int
22  main(argc, argv)
23  int argc;
24  char *argv[];
25  {
26  V3fstruct (im);
27  V3fstruct (tm);
28  int        x,y,z;           /* index counters                  */
29  int        n;               /* Number of frames to average     */
30  int        thresh;
```

```
31
32      VXparse(&argc, &argv, par); /* parse the command line    */
33
34      n = (NVAL ? atoi(NVAL) : 2); /* read n, default is n=1   */
35      if(THRESH)
36      thresh = atoi(THRESH);
37
38      while (Vbfread( &im, IVAL, n)) {
39      if ( im.type != VX_PBYTE || im.chan != 1) { /* check format  */
40              fprintf (stderr, "image not byte type\n");
41              exit (1);
42          }
43    for(z = im.zlo; z < im.zhi; z++){
44      for (y = im.ylo; y <= im.yhi; y++) {
45         for (x = im.xlo; x <= im.xhi; x++) {
46           if(abs(im.u[z+1][y][x] - im.u[z][y][x]) > thresh){
47          im.u[z][y][x] = 255;
48              }
49           else{
50          im.u[z][y][x] = 128;
51           }
52          }
53         }
54       }
55          V3fwrite (&im, OVAL); /* write the oldest frame */
56      }
57      exit(0);
58  }
```

Fig12 Code for change detection programming

By applying difference filter to taxi.vs sequence, I have output sequence when the threshold are 5, 8, 10, 13, 15, 20. The purpose of difference filter should be detecting image changes compared with the previous frame. The effect of difference filter is that the changed areas are outlined and the unchanged areas become the background, in this case, the background should be 128. It's not hard to see that when the threshold is 15, the output sequence has the best effect. It outlines the movements of three cars. When the threshold is too low, even some noise is considered as changes and be outlined. This can be seen in fig13, fig14 and fig15. When the threshold is too large, the filter is too unsensitive. Even the movement of the car is not considered as changes. fig28 doesn't show the movement of the car on the left bottom corner. So when the threshold is 15, the filter gives the best overall effect.

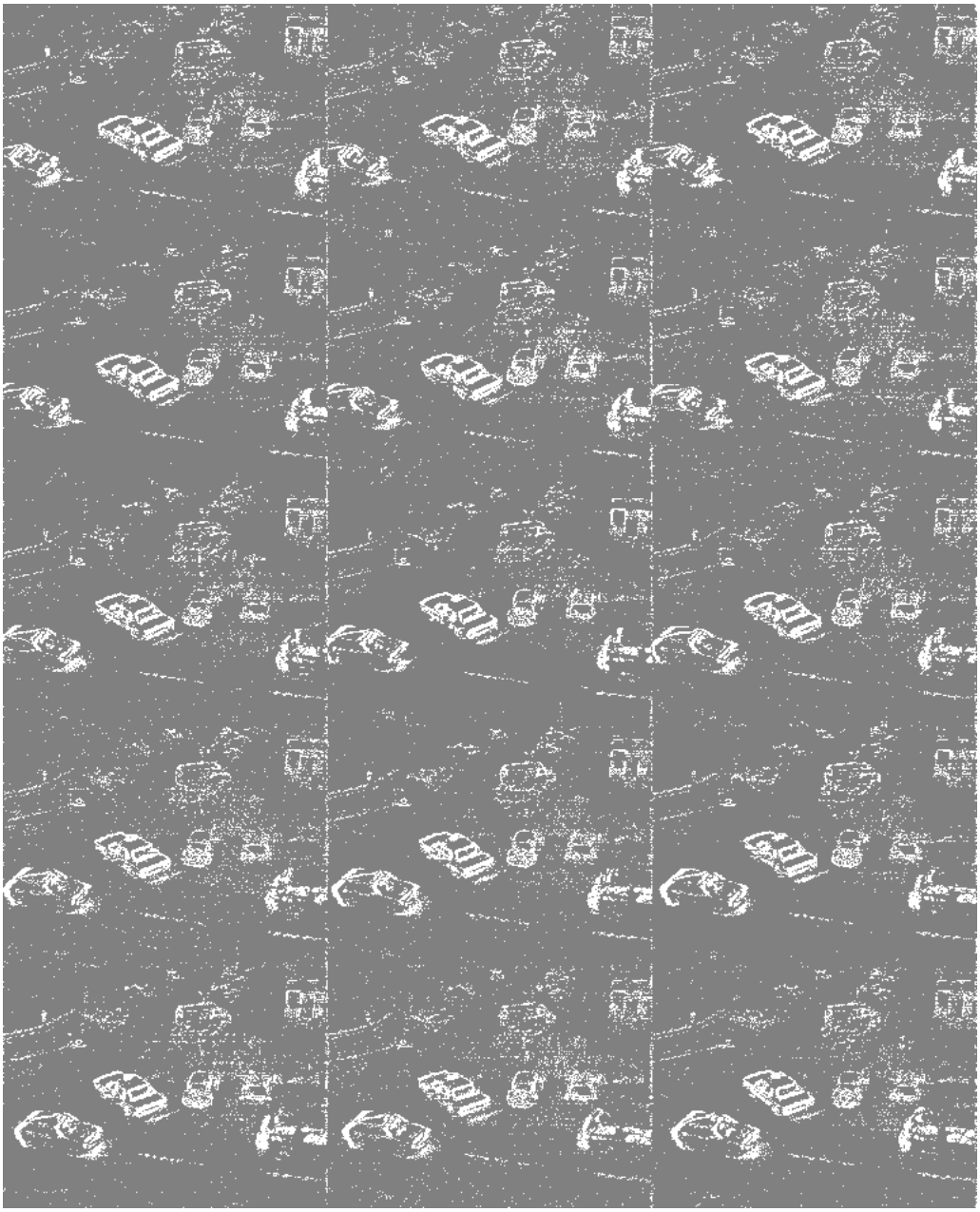

Fig13 Output of difference filtering when threshold = 5

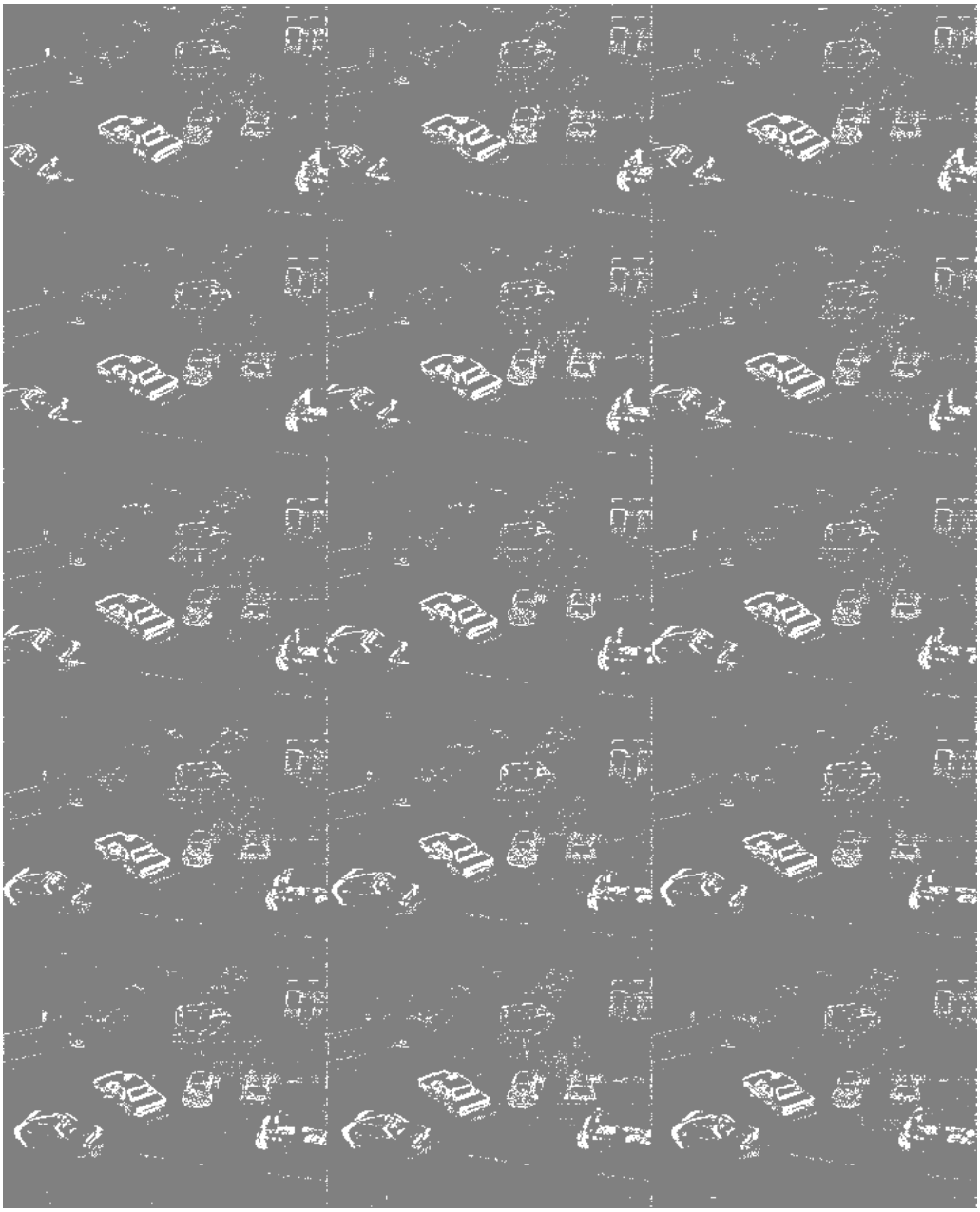

Fig14 Output of difference filtering when threshold = 8

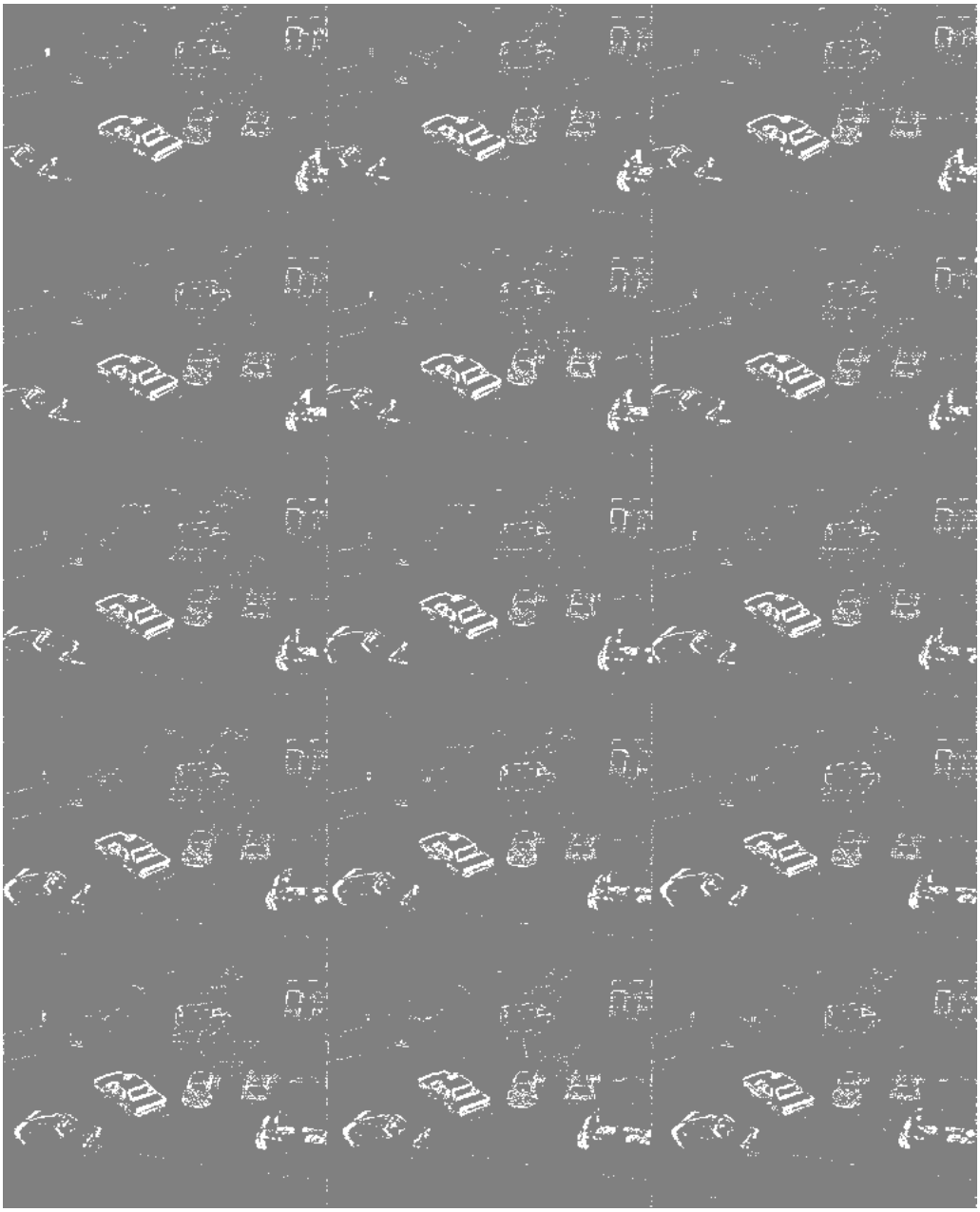

Fig15 Output of difference filtering when threshold = 10

Fig16 Output of difference filtering when threshold = 13

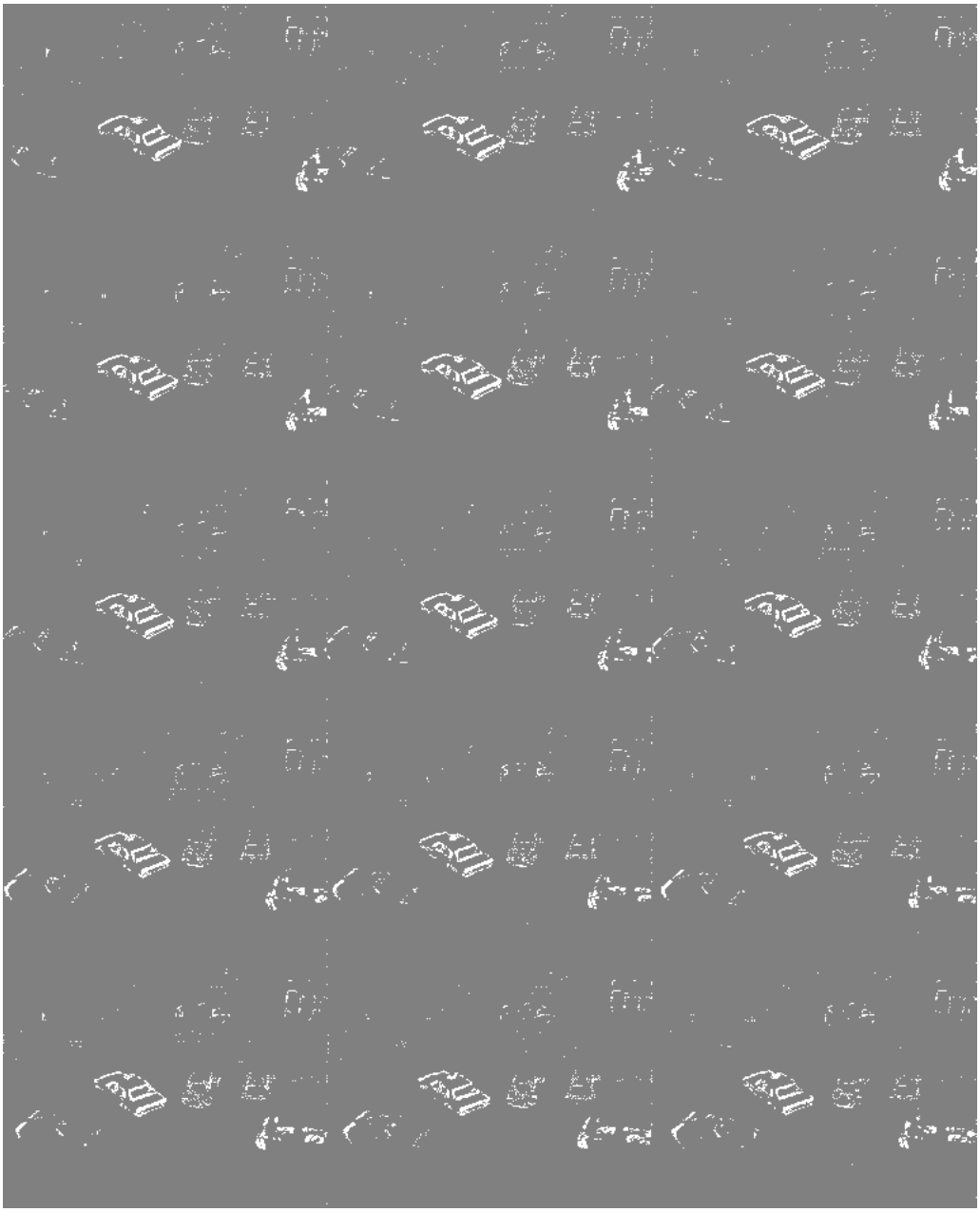

Fig17 Output of difference filtering when threshold = 15

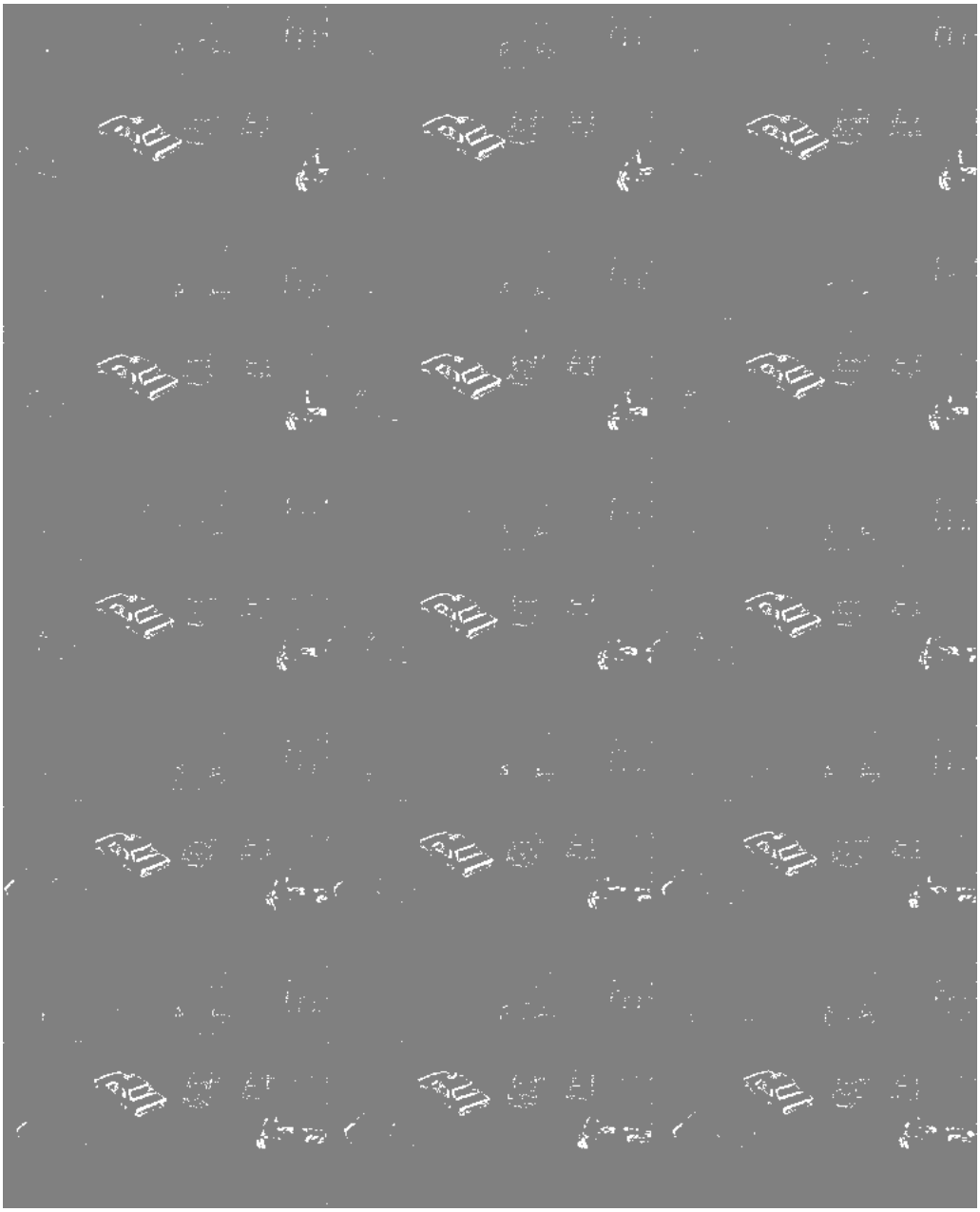

Fig18 Output of difference filtering when threshold = 20