ECE 5470 Computer Vision Lab3 report

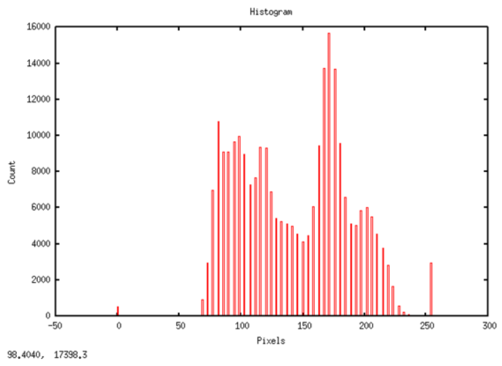
Segmentation Thresholding and Region Growing

Weiran Wang

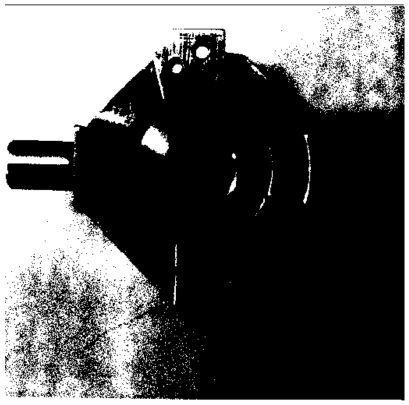
NetID: ww463

Peakiness Detection:

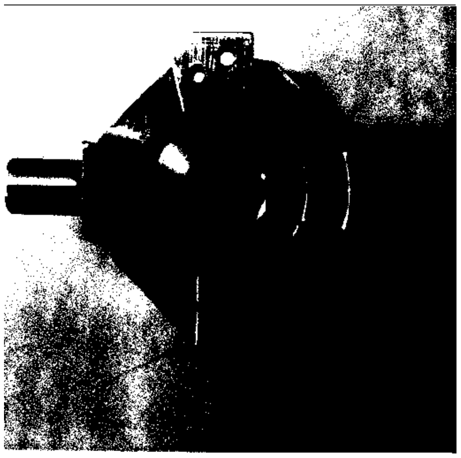
For the images mp, map, facsimile, shtl. I displayed histogram for each of them and tried several distance parameters and find the one gives the best image after thresholding.



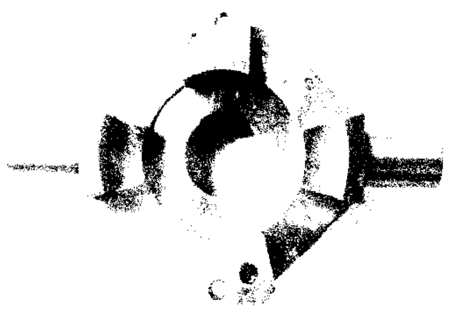
img1. Histogram of mp.vx



img2. mp threshold:169



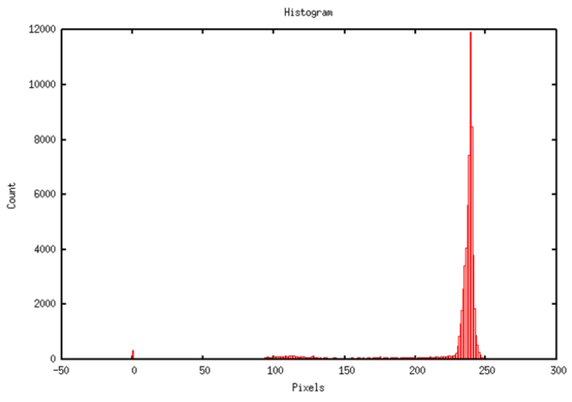
img3. mp threshold:173



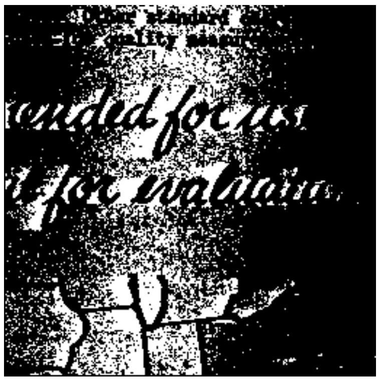
img4. mp threshold:83

Facsimile:

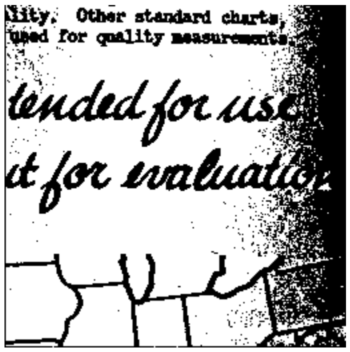
As we can see from the histogram and results, threshold 231 has the best effect



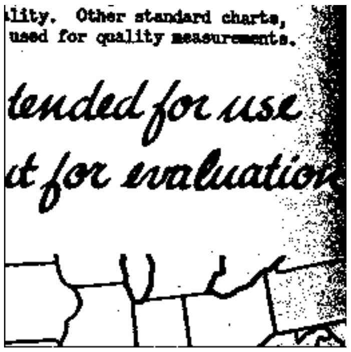
img5. Histogram of facsimile



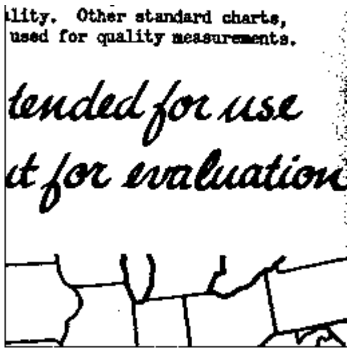
img6. facsimile threshold: 241



img7. facsimile threshold: 237



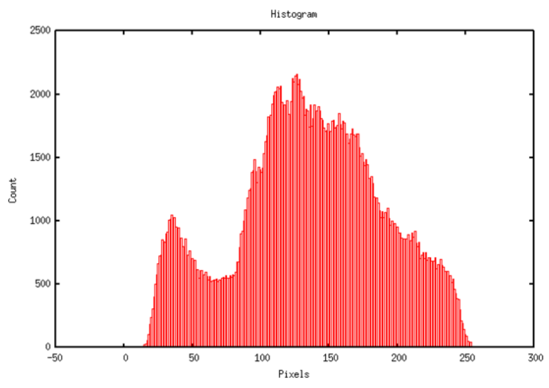
img8. facsimile threshold: 235



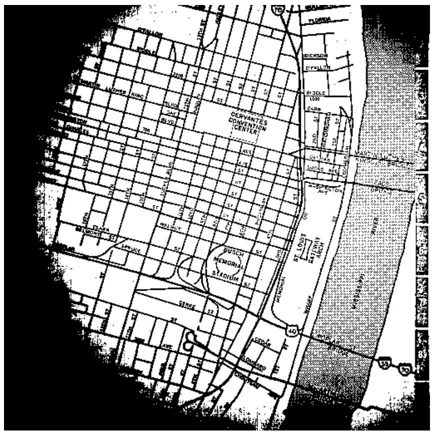
img9. facsimile threshold: 231

Map:

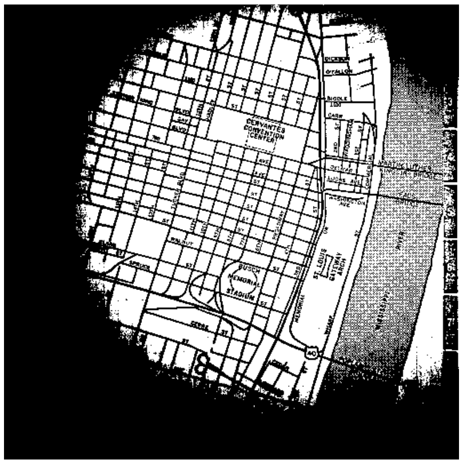
As we can see from the histogram and results, threshold 64 has the best effect



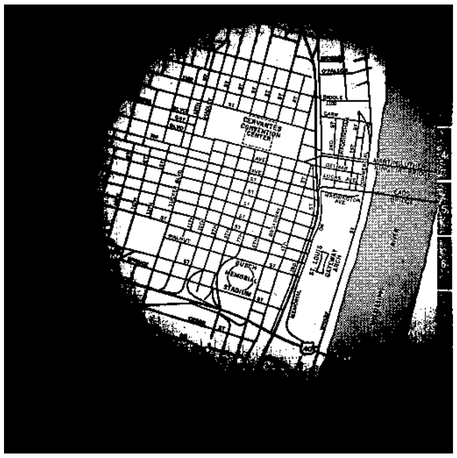
img10. Histogram of facsimile



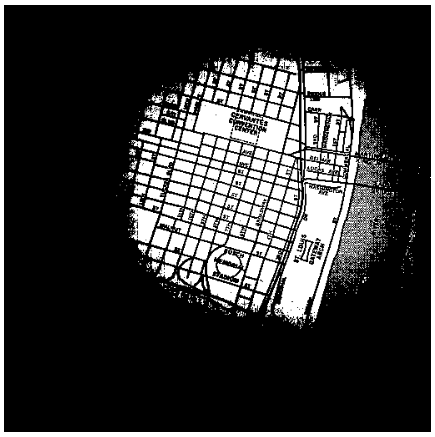
img11. map threshold: 126



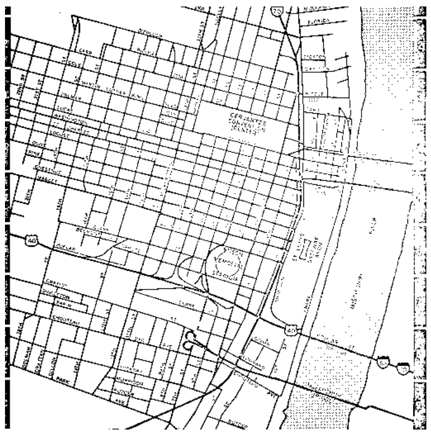
img12. map threshold: 148



img13. map threshold: 165

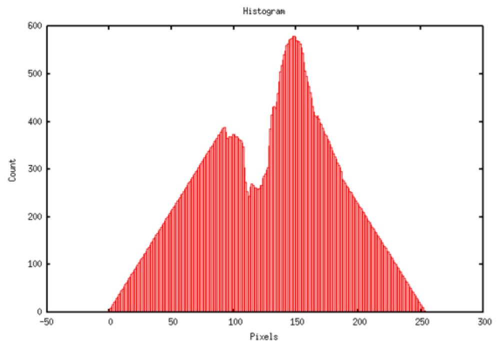


Img14. map threshold: 185

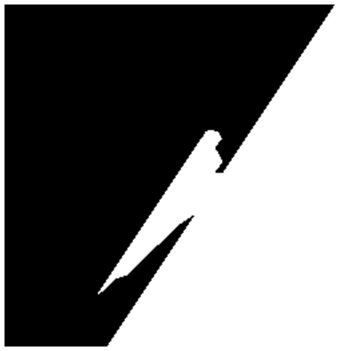


img15. map threshold: 64

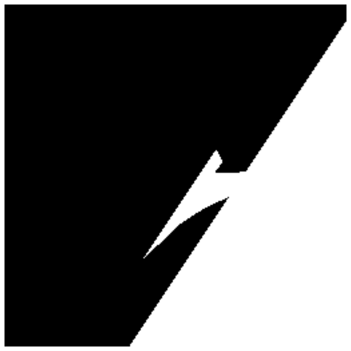
Shtl:



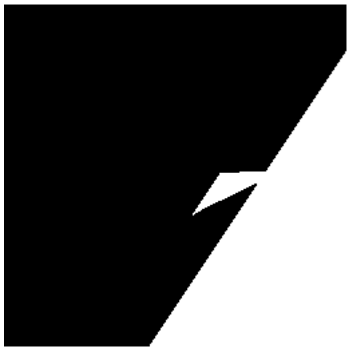
Img16. Histogram of shtl



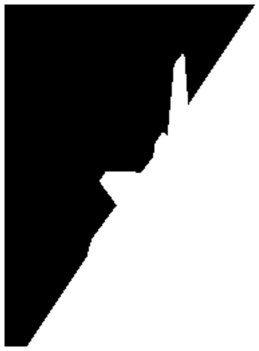
img17.shtl threshold:148



Img18. shtl threshold:158



Img19.shtl threshold:167



Img20.shtl threshold:112

Analysis: From the results shown above, we can see the weakness is that peakiness algorithm is very sensitive of noise, especially when the noise creates peaks in the histogram, another weakness is when there are multiple peaks on the histogram, the performance of the algorithm is too dependent on the parameter such as maximum distance in this case. Of course, when the peaks are clear on the histogram, peakiness algorithm is caple of giving us almost the best result.

Iterative Threshold(vits)

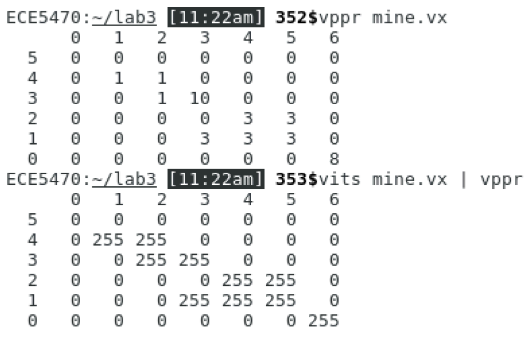
Description:
The images setup is the same as before, we have the new image im and old image tm. To find the threshold iteratively, I started by computing average pixel value of the whole image. Once I have the initial threshold, I partition the image to two parts. One part has all the pixels whose value are smaller than the threshold, the other has all the pixels whose value are greater than the threshold. In the program, I used sum1 and sum2 to keep track of the total pixel value for each region. Num1 and num2 are used to count number of pixels in for each region. After going through all pixels in the image, I computed average pixel value for these two regions by using sum1, sum2, num1, num2. The new threshold is calculated by threshold = (avg1 + avg2)/2. We can then use the new threshold to partition the image until avg1 and avg2 remain the same as the previous iteration. Otherwise, we keep partitioning and computing new threshold. When the iteration terminates, the threshold value is optimal.

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

VXparam_t par[] =                      /* command line structure                            */
{
{    "if=",      0,      " input file, vtpeak: threshold between hgram peaks"},
{    "of=",      0,      " output file "},
{    "d=",       0,      " min dist between hgram peaks (default 10)"},
{    "-v",       0,      "(verbose) print threshold information"},
{      0,        0,      0} /* list termination */
};
#define    IVAL      par[0].val
#define    OVAL      par[1].val
#define    DVAL      par[2].val
#define    VFLAG     par[3].val

main(argc, argv)
int argc;
char *argv[];
{

        Vfstruct (im);                          /* input image structure                  */
        int y,x;                                /* index counters                            */
        int i;
        int j;

        int hist[256];                          /* histogram bins                              */
        int thresh;                             /* threshold                                   */
        int maxbin;                             /* maximum histogram bin                       */
        int nxtbin;                             /* second maximum bin                          */
        int minbin;                             /* minumim histogram bin                       */
        int maxa, maxb;            /* second maximum bin above/below maxbin      */
        int dist;                          /* minimum distance between maxima      */
        int cycle = 100000;
        int sum = 0;
        int sum1 = 0;
        int sum2 = 0;
        int num1 = 0;
        int num2 = 0;
        int avg1=0;
        int avg2 = 0;

        VXparse(&argc, &argv, par);        /* parse the command line                  */

        dist = 10;                              /* default dist */
        if (DVAL) dist = atoi(DVAL);    /* if d= was specified, get value */
        if (dist < 0 || dist > 255) {
        fprintf(stderr, "d= must be between 0 and 255\nUsing d=10\n");
                dist = 10;
        }

        while ( Vfread( &im, IVAL) ) {
                if ( im.type != VX_PBYTE ) {
                        fprintf (stderr, "error: image not byte type\n");
                        exit (1);
                }

                /* clear the histogram */
                for (i = 0; i < 256; i++) hist[i] = 0;

                /* compute the histogram */
                for (y = im.ylo; y <= im.yhi; y++){
                        for (x = im.xlo; x <= im.xhi; x++){
                                hist[im.u[y][x]]++;
                                sum += im.u[y][x];
```

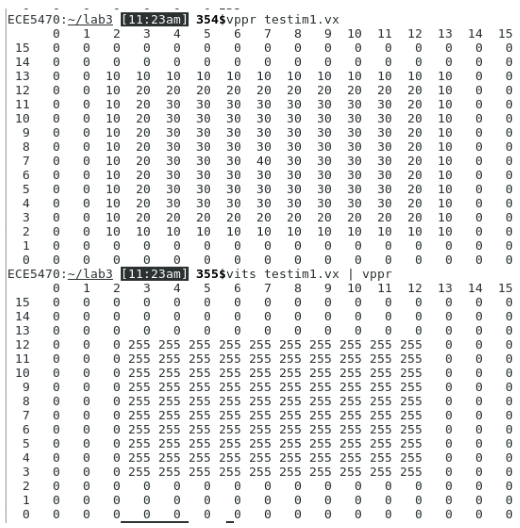
```
}
}
thresh = sum /(im.yhi*im.xhi);

                  /* compute the threshold */
for(i=1; i<cycle; i++){
sum1 = 0;
sum2 = 0;
num1 = 0;
num2 = 0;
for(j=0; j<thresh; j++){
sum1 += j*hist[j];
num1 += hist[j];
}

for(j=thresh; j<=255; j++){
sum2 += j*hist[j];
num2 += hist[j];
}

if(num1 == 0)
avg1 = 0;
else
avg1 = sum1/num1;

if(num2 == 0)
avg2 = 0;
else
avg2 = sum2/num2;

if(thresh == (avg1+avg2)/2)
break;
else
thresh == (avg1+avg2)/2;

}
                /* apply the threshold */
                for (y = im.ylo; y <= im.yhi; y++) {
                        for (x = im.xlo; x <= im.xhi; x++) {
                                if (im.u[y][x] >= thresh) im.u[y][x] = 255;
                                else                         im.u[y][x] = 0;
                }
            }

            Vfwrite( &im, OVAL);
        } /* end of every frame section */
        exit(0);
}
```

Small images:

As we can see below, vits.c works for small images.



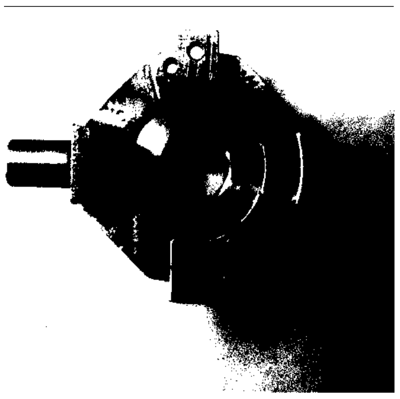
```
ECE5470:~/lab3 [11:22am] 352$vppr mine.vx
     0   1   2   3   4   5   6
  5  0   0   0   0   0   0   0
  4  0   1   1   0   0   0   0
  3  0   0   1  10   0   0   0
  2  0   0   0   0   3   3   0
  1  0   0   0   3   3   3   0
  0  0   0   0   0   0   0   8
ECE5470:~/lab3 [11:22am] 353$vits mine.vx | vppr
     0   1   2   3   4   5   6
  5  0   0   0   0   0   0   0
  4  0 255 255   0   0   0   0
  3  0   0 255 255   0   0   0
  2  0   0   0   0 255 255   0
  1  0   0   0 255 255 255   0
  0  0   0   0   0   0   0 255
```



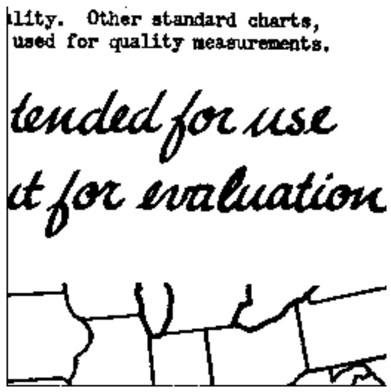
```
ECE5470:~/lab3 [11:23am] 354$vppr testim1.vx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
 12   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
 11   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
 10   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  9   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  8   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  7   0   0  10  20  30  30  30  40  30  30  30  30  20  10   0   0
  6   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  5   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  4   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  3   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
  2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
ECE5470:~/lab3 [11:23am] 355$vits testim1.vx | vppr
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 12   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
 11   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
 10   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  9   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  8   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  7   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  6   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  5   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  4   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  3   0   0   0 255 255 255 255 255 255 255 255 255 255   0   0   0
  2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Img21. Small image results

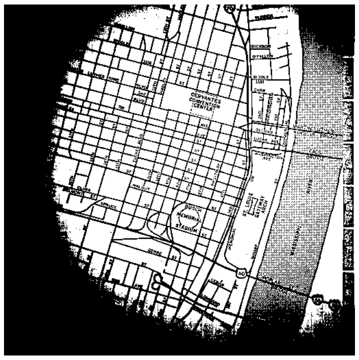
Large images:

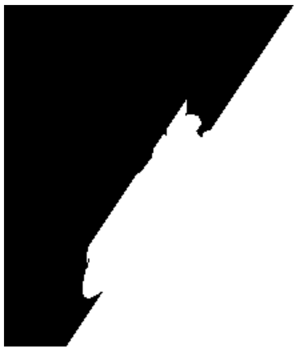


img22. mp iterative thresholding result


lity. Other standard charts,
used for quality measurements.
tended for use
t for evaluation

img23.facsimile iterative thresholding result

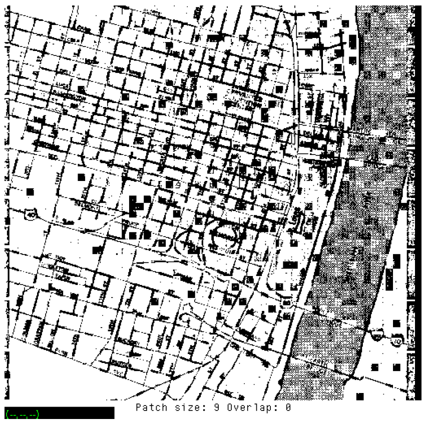

img24. map iterative thresholding result


img25. shtl iterative thresholding result

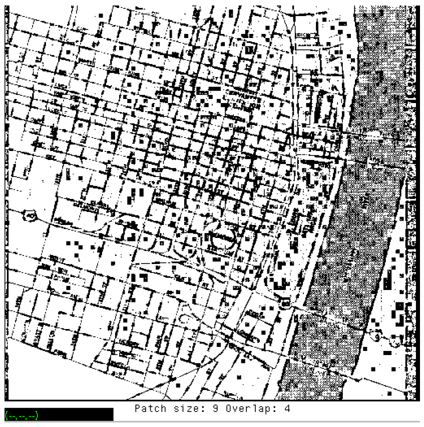
For large images, iterative algorithm works really good on img23 and img24. For image 25, the effect will be better if the algorithm gives a lower threshold.

Adaptive Thresholding

After running patchval script with 9 different configuration, I observed the image sequence from mpa and mp and choose two best thresholding images from each of them.
map.seq:
img 26 has clearer boundary on the map whereas img27 contains more details

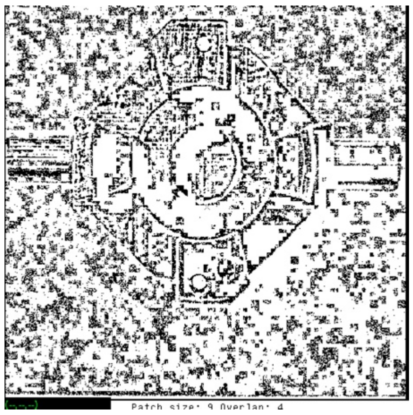


img26. Patch:9 Overlap:0

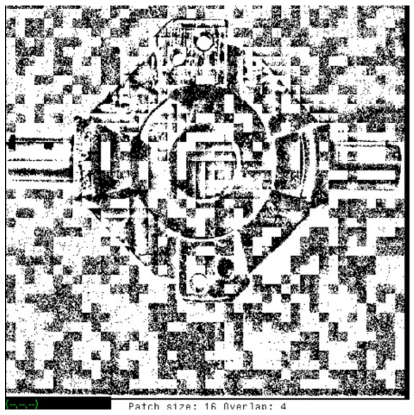


Img27. Patch:9 Overlap:0

mp.vx.seq:
img 28 has more details and img 29 has more clearer general view of the image



Img28. Patch:9 Overlap:4



Img29. Patch:16 Overlap:4

Region Growing

Description:
Program vgrow.c has new parameters -p and r as required. -p is a flag that controls labeling scheme. When this flag is set, the label value will be the value of the first pixel in the region. Otherwise the regions are numbered sequentially. r is the maximum range within a region, if the pixel value difference is greater than r, we are going to label a new region. Based on program cclabel.c, region growing can also be achieved by recursion. For each pixel, we are going to recursively call the function setlabel to label four connected pixels(up down left right). If the original pixel value is not 0, and it hasn't not been labeled, setlabel function is going to assign a label to the current pixel and recursively label connected pixels until the whole region is complete.

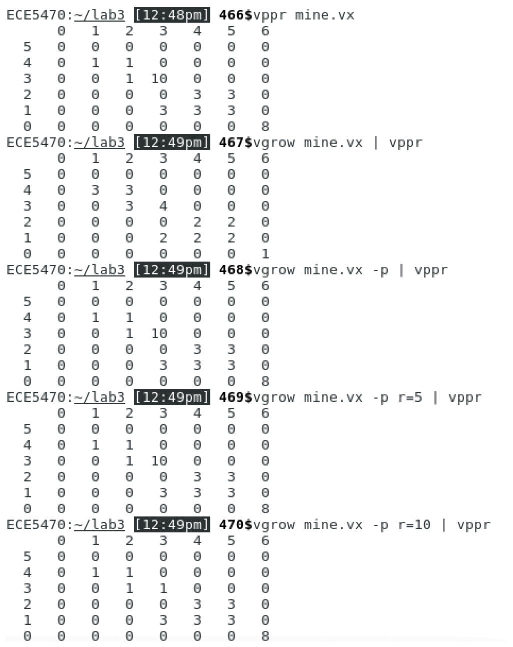
Program:
```
#include "VisXV4.h"          /* VisionX structure include file       */
#include "Vutil.h"           /* VisionX utility header files         */

VXparam_t par[] =            /* command line structure               */
{ /* prefix, value,   description                        */
{    "if=",    0,   " input file    vtemp: local max filter "},
{    "of=",    0,   " output file "},
{    "r=",     0,   " region pixel range "},
{    "-p",     0,   " set first pixel value in the region as label value "},
{     0,       0,    0}   /* list termination */
};
#define   IVAL    par[0].val
#define   OVAL    par[1].val
#define   RANGE    par[2].val
#define   FLAG   par[3].val

void setlabel(int, int, int);
Vfstruct (im);                      /* i/o image structure          */
Vfstruct (tm);                      /* temp image structure         */
int first;
int range;


main(argc, argv)
int argc;
char *argv[];
{
```

```
int             y,x;                                /* index counters                    */
int label = 1;
    VXparse(&argc, &argv, par);             /* parse the command line              */

    range = 8;                                          /* default range                          */
    if (RANGE) range = atoi(RANGE);         /* assign input value to range    */

    Vfread(&im, IVAL);                             /* read image file                          */
    Vfembed(&tm, &im, 1,1,1,1);              /* image structure with border    */
    if ( im.type != VX_PBYTE ) {            /* check image format                   */
          fprintf(stderr, "vtemp: no byte image data in input file\n");
          exit(-1);
    }


    for (y = im.ylo ; y <= im.yhi ; y++) {
          for (x = im.xlo; x <= im.xhi; x++)    {
                      im.u[y][x] = 0;
          }
      }

      for (y = im.ylo ; y<= im.yhi ; y++) {
            for (x = im.xlo ; x <= im.xhi ; x++) {
                if (tm.u[y][x] != 0 && im.u[y][x] == 0) {
                        first = tm.u[y][x];
                        if (FLAG) setlabel(x, y, first);
                        else {
                                setlabel(x, y, label);
                                if (label == 255) label = 1;
                                else label++;
                        }
                }
            }
      }

      Vfwrite(&im, OVAL);                        /* write image file                         */
      exit(0);
}

/* function to compute the local maximum */
void setlabel(int x, int y, int label)
```

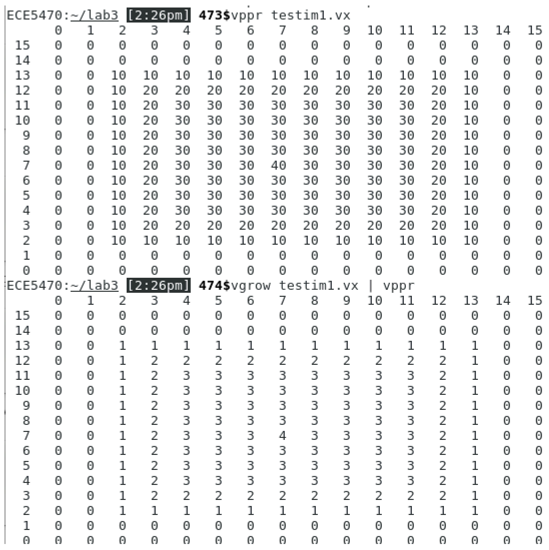
```
{
    im.u[y][x] = label;
    if (tm.u[y][x+1] != 0 && im.u[y][x+1] == 0 && abs(tm.u[y][x+1] - first) < range)
        setlabel(x+1, y, label);
    if (tm.u[y][x-1] != 0 && im.u[y][x-1] == 0 && abs(tm.u[y][x-1] - first) < range)
        setlabel(x-1, y, label);
    if (tm.u[y+1][x] != 0 && im.u[y+1][x] == 0 && abs(tm.u[y+1][x] - first) < range)
        setlabel(x, y+1, label);
    if (tm.u[y-1][x] != 0 && im.u[y-1][x] == 0 && abs(tm.u[y-1][x] - first) < range)
        setlabel(x, y-1, label);
}
```

Small images:

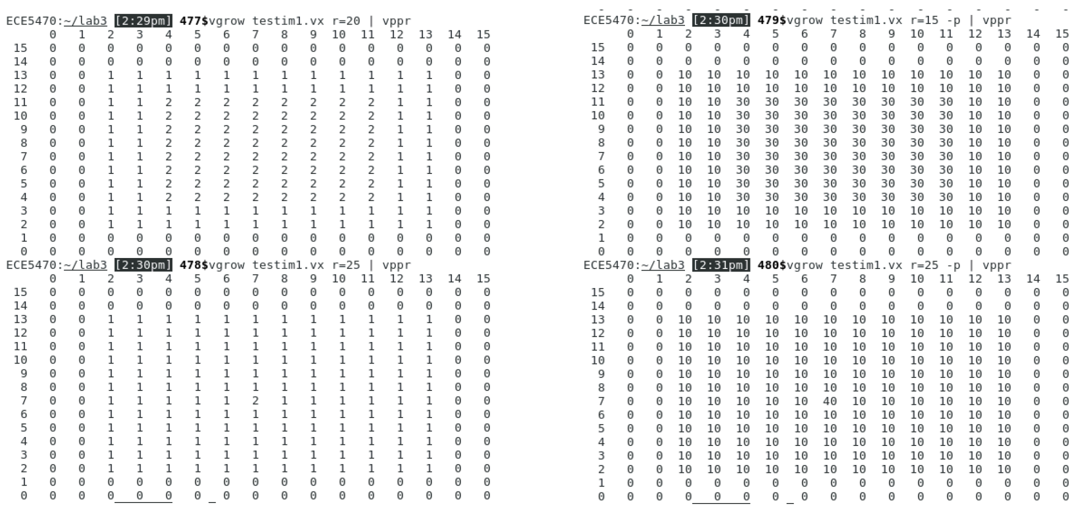
As we can see vgrow.c works for small images with different parameters setup



```
ECE5470:~/lab3 [12:48pm] 466$vppr mine.vx
       0   1   2   3   4   5   6
  5    0   0   0   0   0   0   0
  4    0   1   1   0   0   0   0
  3    0   0   1  10   0   0   0
  2    0   0   0   0   3   3   0
  1    0   0   0   3   3   3   0
  0    0   0   0   0   0   0   8
ECE5470:~/lab3 [12:49pm] 467$vgrow mine.vx | vppr
       0   1   2   3   4   5   6
  5    0   0   0   0   0   0   0
  4    0   3   3   0   0   0   0
  3    0   0   3   4   0   0   0
  2    0   0   0   0   2   2   0
  1    0   0   0   2   2   2   0
  0    0   0   0   0   0   0   1
ECE5470:~/lab3 [12:49pm] 468$vgrow mine.vx -p | vppr
       0   1   2   3   4   5   6
  5    0   0   0   0   0   0   0
  4    0   1   1   0   0   0   0
  3    0   0   1  10   0   0   0
  2    0   0   0   0   3   3   0
  1    0   0   0   3   3   3   0
  0    0   0   0   0   0   0   8
ECE5470:~/lab3 [12:49pm] 469$vgrow mine.vx -p r=5 | vppr
       0   1   2   3   4   5   6
  5    0   0   0   0   0   0   0
  4    0   1   1   0   0   0   0
  3    0   0   1  10   0   0   0
  2    0   0   0   0   3   3   0
  1    0   0   0   3   3   3   0
  0    0   0   0   0   0   0   8
ECE5470:~/lab3 [12:49pm] 470$vgrow mine.vx -p r=10 | vppr
       0   1   2   3   4   5   6
  5    0   0   0   0   0   0   0
  4    0   1   1   0   0   0   0
  3    0   0   1   1   0   0   0
  2    0   0   0   0   3   3   0
  1    0   0   0   3   3   3   0
  0    0   0   0   0   0   0   8
```



```
ECE5470:~/lab3 [2:26pm] 473$vppr testim1.vx
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
 12   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
 11   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
 10   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  9   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  8   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  7   0   0  10  20  30  30  30  40  30  30  30  30  20  10   0   0
  6   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  5   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  4   0   0  10  20  30  30  30  30  30  30  30  30  20  10   0   0
  3   0   0  10  20  20  20  20  20  20  20  20  20  20  10   0   0
  2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
ECE5470:~/lab3 [2:26pm] 474$vgrow testim1.vx | vppr
      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
 15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 13   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
 12   0   0   1   2   2   2   2   2   2   2   2   2   2   1   0   0
 11   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
 10   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  9   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  8   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  7   0   0   1   2   3   3   3   4   3   3   3   3   2   1   0   0
  6   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  5   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  4   0   0   1   2   3   3   3   3   3   3   3   3   2   1   0   0
  3   0   0   1   2   2   2   2   2   2   2   2   2   2   1   0   0
  2   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```



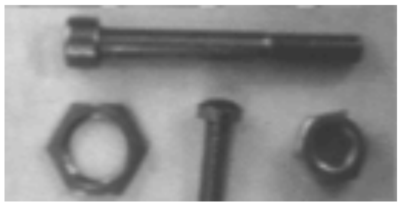
```
ECE5470:~/lab3 [2:29pm] 477$vgrow testim1.vx r=20 | vppr
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  13   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  12   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  11   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
  10   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   9   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   8   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   7   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   6   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   5   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   4   0   0   1   1   2   2   2   2   2   2   2   2   1   1   0   0
   3   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   2   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
ECE5470:~/lab3 [2:30pm] 478$vgrow testim1.vx r=25 | vppr
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  13   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  12   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  11   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
  10   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   9   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   8   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   7   0   0   1   1   1   1   1   2   1   1   1   1   1   1   0   0
   6   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   5   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   4   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   3   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   2   0   0   1   1   1   1   1   1   1   1   1   1   1   1   0   0
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
ECE5470:~/lab3 [2:30pm] 479$vgrow testim1.vx r=15 -p | vppr
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  12   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  11   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
  10   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   9   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   8   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   7   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   6   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   5   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   4   0   0  10  10  30  30  30  30  30  30  30  30  10  10   0   0
   3   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
ECE5470:~/lab3 [2:31pm] 480$vgrow testim1.vx r=25 -p | vppr
       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
  15   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  14   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  13   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  12   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  11   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
  10   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   9   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   8   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   7   0   0  10  10  10  10  10  40  10  10  10  10  10  10   0   0
   6   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   5   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   4   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   3   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   2   0   0  10  10  10  10  10  10  10  10  10  10  10  10   0   0
   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

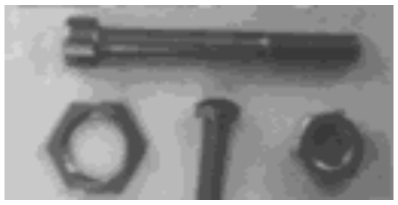
Img30. Small images region growing results

Large images:

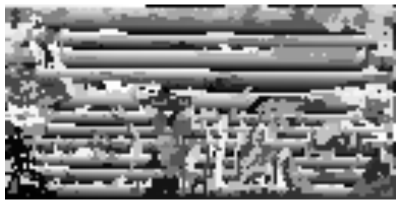
As we can see from the images, the results are really blurry when the flag is not set. It would make a lot of sense to label the region with the first pixel value, because these are real-life objects images. Labeling the region sequentially would cause confusion. When the range is too large, objects in the image has a 'fading' effect. Generally, when flag p is set and range=5 nb.vx has better effect.



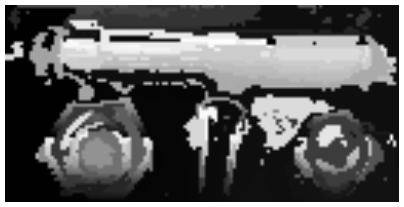
img31. Region growing -p, r=5



img32. Region growing -p, r=10

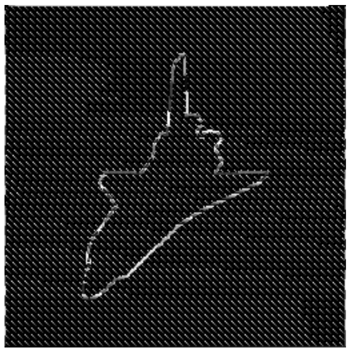


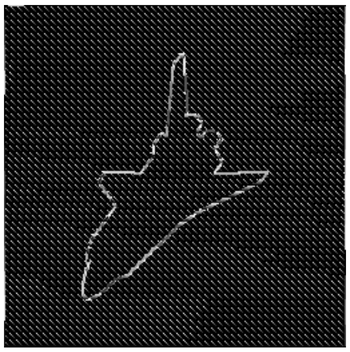
img33. Region growing r=5

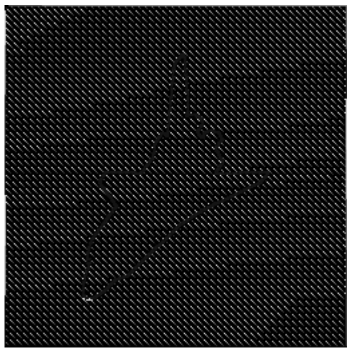


img34. Region growing r=10

As we can see from the images, when range=5 edge.vx has better effect


img35.Range = 2


img36.range = 5


img37.range = 10