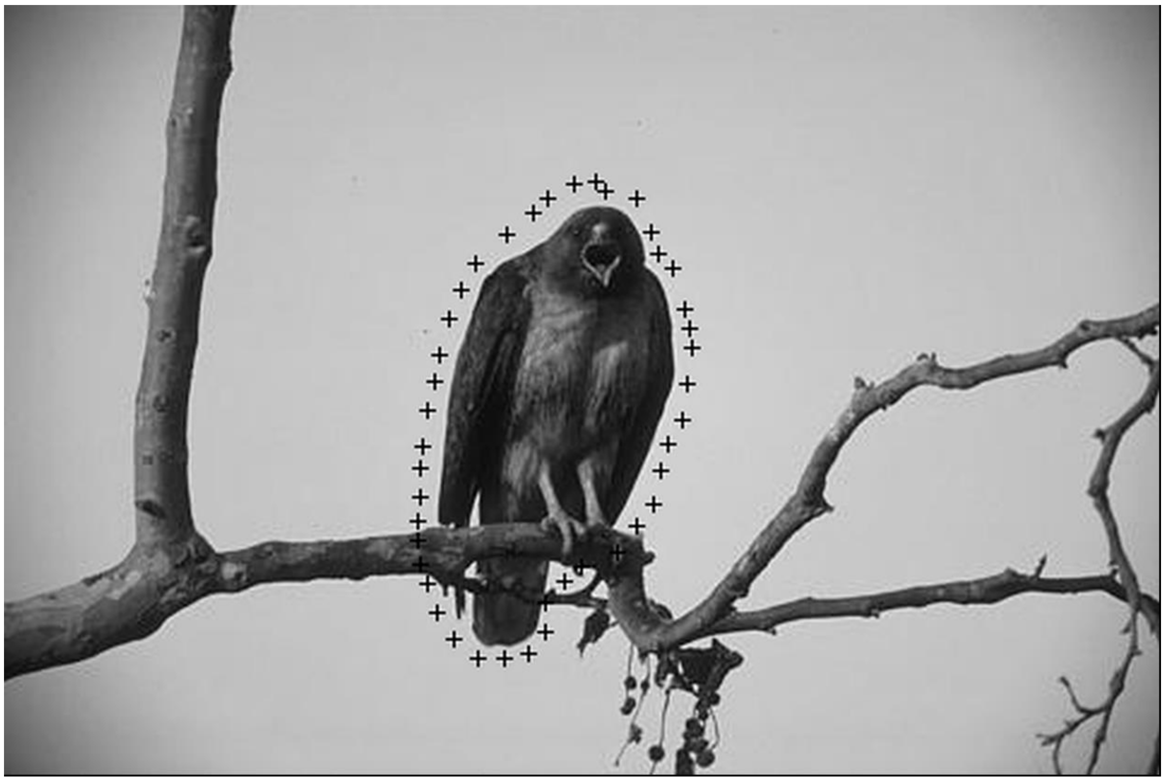
ECE 4310/6310 Introduction to Computer Vision

Lab #5 – contouring

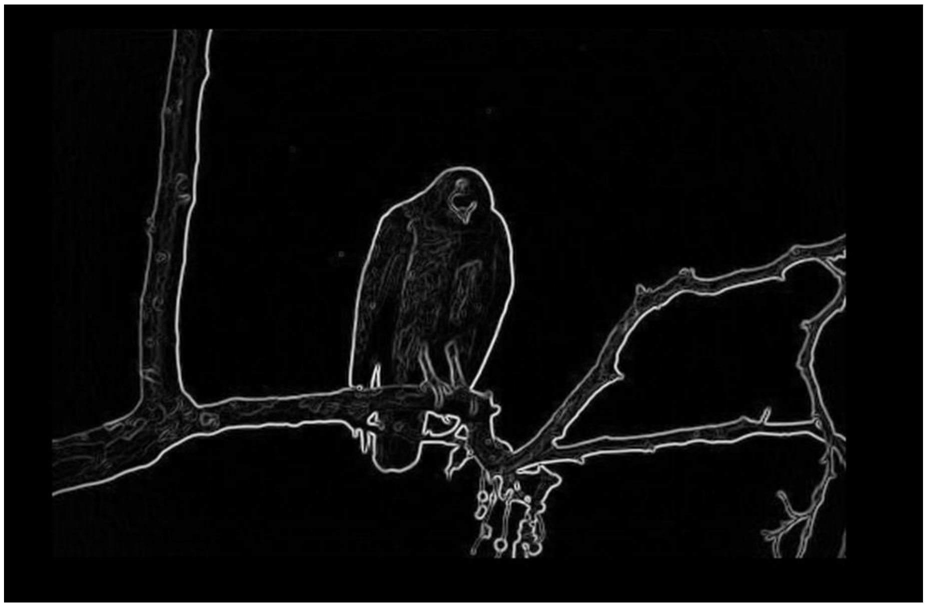
Name-Pinak Kelkar.

This is the initial position of the contour points. To make sure that the contour points are moving equidistant from each other we use one internal energy from i+1 contour point and the second internal energy calculated from the average distance between all contour points. The external energy is the negative of the edge detection using the sobel filter. Both the internal energy and the external energy have been normalized to 0-1.

Starting point of contour point



Edge map using sobel filter



Final image of contour points

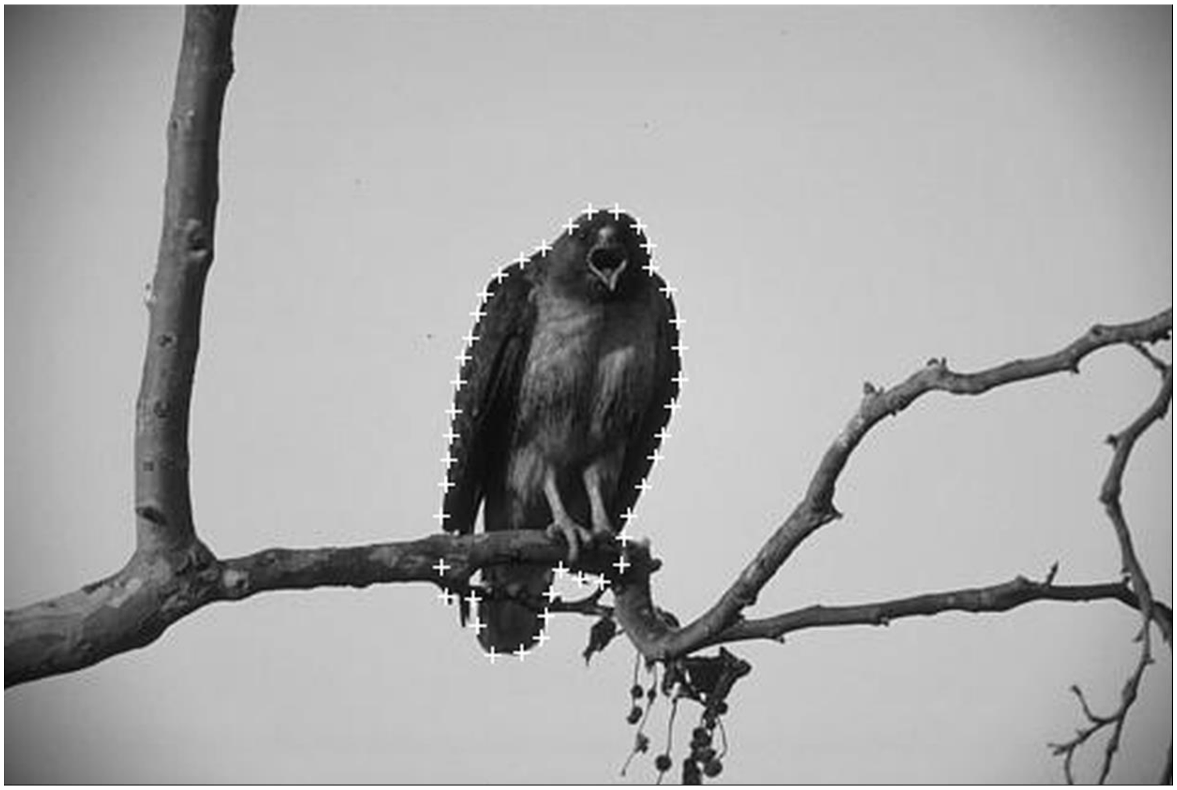
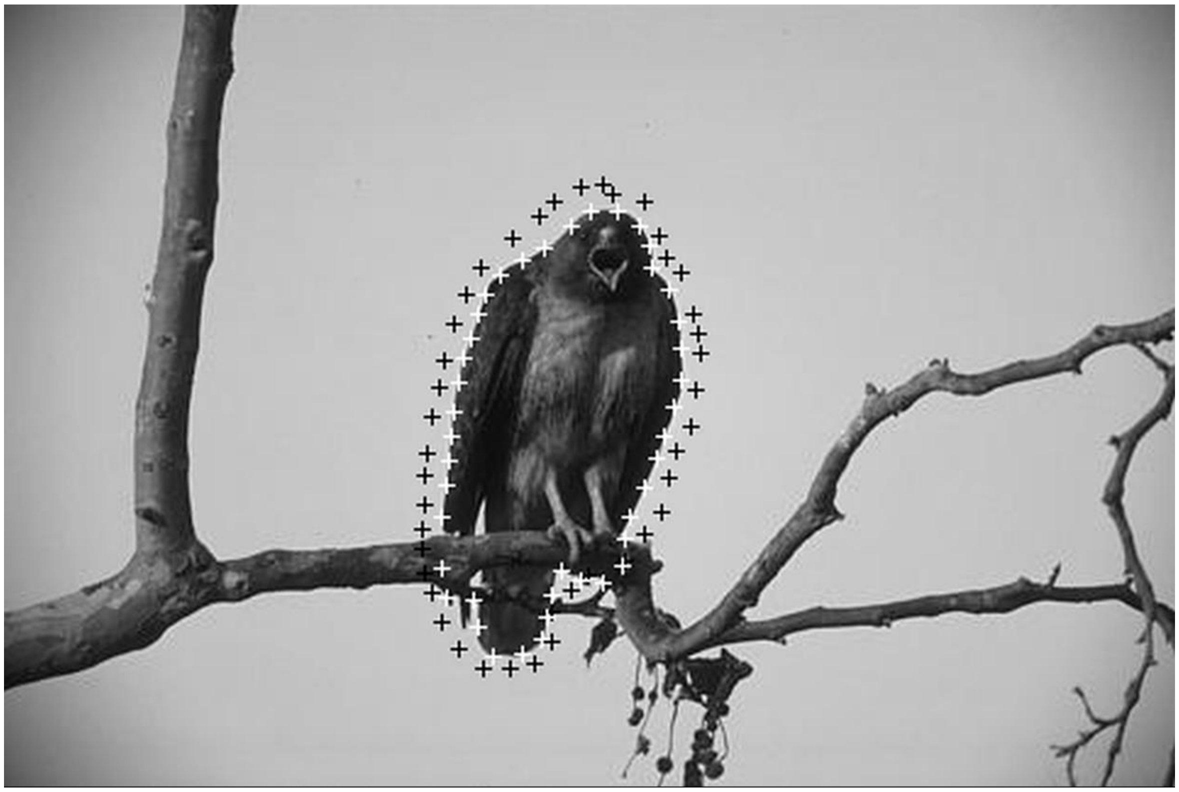
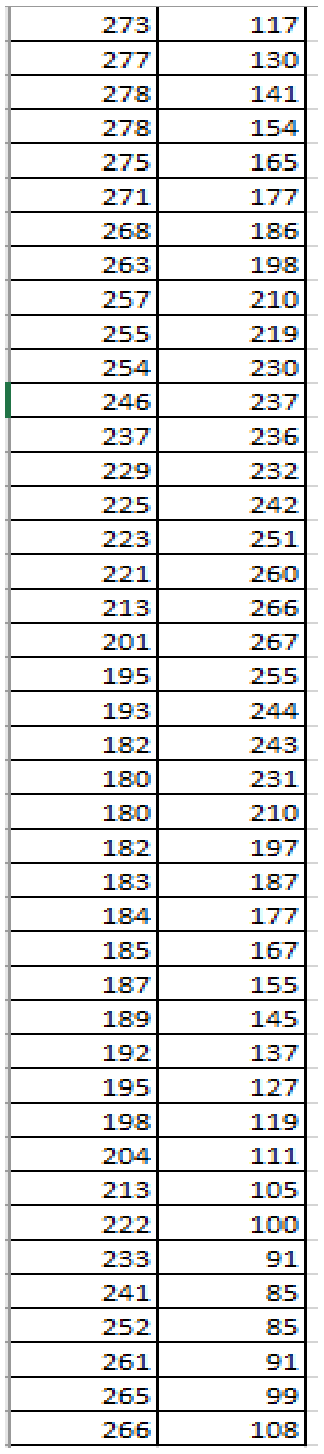


Image of start and final contour points black points are the starting and the white cross are the end contour points



Final contour points locations.



273	117
277	130
278	141
278	154
275	165
271	177
268	186
263	198
257	210
255	219
254	230
246	237
237	236
229	232
225	242
223	251
221	260
213	266
201	267
195	255
193	244
182	243
180	231
180	210
182	197
183	187
184	177
185	167
187	155
189	145
192	137
195	127
198	119
204	111
213	105
222	100
233	91
241	85
252	85
261	91
265	99
266	108

Code

```
#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <string.h>

#include <math.h>


double min,max;

void findminmax(double *array,int size){

                                        min=array[0];max=0;

                                                    for(int i=0;i<size;i++){

                                                            if(min>array[i]){

                                                                    min=array[i];

                                                            }

                                                            if(max<array[i]){

                                                                    max=array[i];

}

}

}

int main ()

{

FILE *fpt,*ftr;

unsigned char *image;

unsigned char *image1;

double *external_energy;

unsigned char *sobe;

int *image_inverse;

int contour_points=0;

int window_size=49;

int *cc_points,*cr_points;
int *moved_cc_points,*moved_cr_points;

char T_header[80];

int ROWS,COLS,T_BYTES;

char * line = NULL;

size_t len = 0;

ssize_t read;

            int i=0;


fpt=fopen("hawk.ppm","r");

ftr=fopen("ac.txt","r");

i=fscanf(fpt,"%s %d %d %d",T_header,&COLS,&ROWS,&T_BYTES);

image=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));

image1=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));

sobe=(unsigned char *)calloc(ROWS*COLS,sizeof(unsigned char));

external_energy=(double *)calloc(ROWS*COLS,sizeof(double));

image_inverse=(int *)calloc(ROWS*COLS,sizeof(int));

cc_points=(int *)calloc(window_size,sizeof(int));

cr_points=(int *)calloc(window_size,sizeof(int));

moved_cc_points=(int *)calloc(window_size,sizeof(int));

moved_cr_points=(int *)calloc(window_size,sizeof(int));

            //sobel

int sobelx[]={-1,0,1,

                                        -2,0,2,

                                        -1,0,1};


int sobely[]={-1,-2,-1,

                          0,0,0,

                           1,2,1};


T_header[0]=fgetc(fpt);

fread(image,1,ROWS*COLS,fpt);
```

```
fclose(fpt);
        for(int i=0;i<ROWS*COLS;i++){
        image1[i]=image[i];
        }
        //Read Contour Points
        int index=0;
 while ((read = getline(&line, &len, ftr)) != -1) {
        char *token = strtok(line, " ");
          cc_points[index]=atoi(token);   //c
        token = strtok(NULL, " ");
          cr_points[index]=atoi(token); //r
                index++;
 }
        contour_points=index;
        int Smin=1000,Smax=0;
                        for(int r=1;r<(ROWS-1);r++){
                                for(int c=1;c<(COLS-1);c++){
                                        int gx=0,gy=0;
                                        int index=0;
                                                for(int r1=-1;r1<=1;r1++){
                                                        for(int c1=-1;c1<=1;c1++){
                                                                gx+=image[(r+r1)*COLS+(c+c1)]*sobelx[index];
                                                                gy+=image[(r+r1)*COLS+(c+c1)]*sobely[index];
                                                                index++;
                                                        }
                                                }//sobel;
                                        external_energy[r*COLS+c]=sqrt(pow(gx,2)+pow(gy,2));
                                        if(Smax<external_energy[r*COLS+c]) Smax=external_energy[r*COLS+c];
                                        if(Smin>external_energy[r*COLS+c]) Smin=external_energy[r*COLS+c];
                                }
                        }
                        for(int r=1;r<ROWS*COLS;r++){
                                sobe[r]=((external_energy[r]-Smin)*255/(Smax-Smin));
                                external_energy[r]=-((external_energy[r]-Smin)/(Smax-Smin));
                        }


                        for(int i=0;i<contour_points;i++){
                        for(int r=-3;r<=3;r++){
                                image[((cr_points[i]+r)*COLS)+cc_points[i]]=0;
                                image[(cr_points[i]*COLS)+(cc_points[i]+r)]=0;
                         }
                        }
        int run=0;
        while(run<30){
                double* internal_enegry=(double *)calloc(window_size,sizeof(double));
                double* internal_enegry2=(double *)calloc(window_size,sizeof(double));
                //Average Distance of the point
                        double dist=0;
                for(int i=0;i<contour_points;i++){
                        if(i!=41){
                        dist=dist+sqrt(pow((cc_points[i]-cc_points[i+1]),2)+pow(cr_points[i]-cr_points[i+1],2));
                        }else{
                                dist=dist+sqrt(pow((cc_points[i]-cc_points[0]),2)+pow(cr_points[i]-cr_points[0],2));
                        }
                }
                int avg_distance=dist/42;
                        int iterate=0;
                while(iterate<42){
                                int getCcon=cc_points[iterate];
                                int getRcon=cr_points[iterate];
                                int getCcon1;
                                int getRcon1;
```

```
            moved_cc_points[iterate]=cc_points[iterate];
            moved_cr_points[iterate]=cr_points[iterate];
            if(iterate!=41){
             getCcon1=cc_points[iterate+1];
             getRcon1=cr_points[iterate+1];
            }else{
             getCcon1=cc_points[0];
             getRcon1=cr_points[0];
            }
            int distance=sqrt(pow((getCcon-getCcon1),2)+pow(getRcon-getRcon1,2));
            index=0;
            for(int r=-3;r<=3;r++){
                        for(int c=-3;c<=3;c++){
                        int cols=getCcon+c;
                        int rows=getRcon+r;
            internal_enegry[index]=pow((cols-getCcon1),2)+pow((rows-getRcon1),2);
            internal_enegry2[index]= pow((sqrt(internal_enegry[index]))-avg_distance,2);
                                    index++;
                        }
            }
findminmax(internal_enegry,window_size);
//printf("%lf %lf \n",min,max);
//normallize
            for(int i=0;i<window_size;i++){
                        internal_enegry[i]=(internal_enegry[i]-min)/(max-min);
            //          printf("%lf ",internal_enegry[i]);
            }
findminmax(internal_enegry2,window_size);
//printf("\n%lf %lf \n",min,max);
//normallize
            for(int i=0;i<window_size;i++){
```

```
                                        internal_enegry2[i]=(internal_enegry2[i]-min)/(max-min);
                                //      printf("%lf ",internal_enegry2[i]);
                                }
                        index=0;
                        double min1=1000;
                        int locr,locc;
                        for(int r=-3;r<=3;r++){
                                                for(int c=-3;c<=3;c++){
                                                double
dummy=internal_enegry2[index]+internal_enegry[index]+external_energy[(r+getRcon)*COLS+(c+getCcon)];
                                                        index++;
                                                        if(min1>dummy)
                                                        {
                                                                min1=dummy;
                                                                locr=r+getRcon;
                                                                locc=c+getCcon;
                                                        }
                                                }
                        }
                        moved_cr_points[iterate]=locr;
                        moved_cc_points[iterate]=locc;
                                iterate++;
                }//end contour
                for(int i=0;i<contour_points;i++){
                        cc_points[i]=moved_cc_points[i];
                        cr_points[i]=moved_cr_points[i];
                        }
        run++;
        }
        fpt=fopen("start.ppm","w");
fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);
```

```
fwrite(image,1,ROWS*COLS,fpt);

fclose(fpt);

        for(int i=0;i<contour_points;i++){

                        for(int r=-3;r<=3;r++){

                            image1[((cr_points[i]+r)*COLS)+cc_points[i]]=255;

                            image1[(cr_points[i]*COLS)+(cc_points[i]+r)]=255;

                            image[((cr_points[i]+r)*COLS)+cc_points[i]]=255;

                            image[(cr_points[i]*COLS)+(cc_points[i]+r)]=255;


                         }

                printf("%d %d\n",cc_points[i],cr_points[i]);

                        }

fpt=fopen("sobel.ppm","w");

fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);

fwrite(sobe,1,ROWS*COLS,fpt);

fclose(fpt);

fpt=fopen("final1.ppm","w");

fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);

fwrite(image,1,ROWS*COLS,fpt);

fclose(fpt);

fpt=fopen("final.ppm","w");

fprintf(fpt,"P5 %d %d 255\n",COLS,ROWS);

fwrite(image1,1,ROWS*COLS,fpt);

fclose(fpt);

}
```