**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



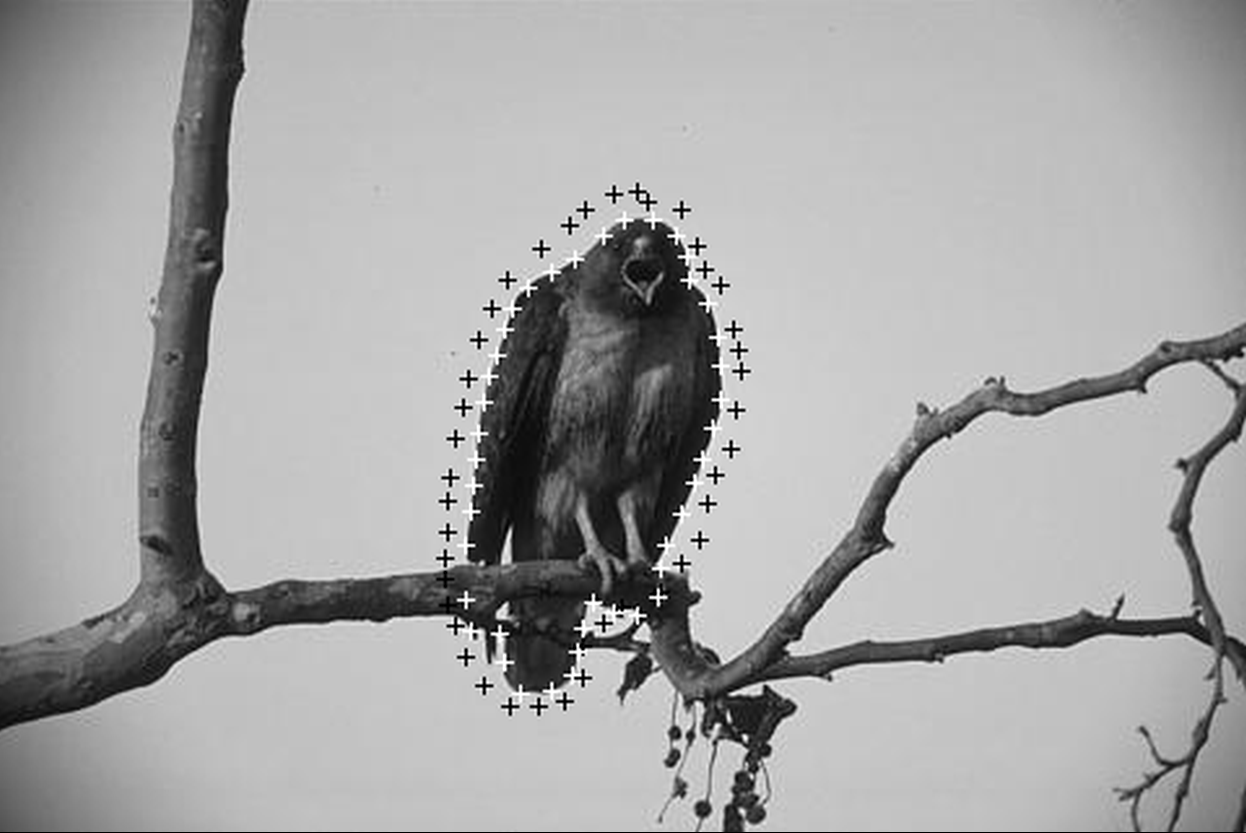
Edge map using sobel filter



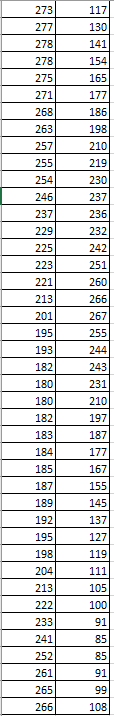
Final image of contour points



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.



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);

}