**DAY 6**

Blob detection by mouse click

Blob with center detection

Problem statement 1

Particular blob detection

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

#include <vector>

#include <deque>

using namespace std;

using namespace cv;

int q=0,w=0,a=0,b=0;

void Call(int event, int x, int y, int flags, void\* userdata){

if (event == EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

void Call1(int event, int x, int y, int flags, void\* userdata){

if (event == EVENT\_LBUTTONDOWN){

a = x;

b = y;

}

}

int main(int argc, \_TCHAR\* argv[])

{

int value1 = 100,value2=100,value3=100,val1=100,val2=100,val3=100;

int i, j, a1 = 0, a2 = 25, b1 = 0, b2 = 250,c1 = 0, c2 = 250;

int x1 = 0, x2 = 250, y1 = 0, y2 = 250, z1 = 0, z2 = 250;

namedWindow("binary", 1);

namedWindow("graph", 1);

while (1){

setMouseCallback("binary", Call, NULL);

Mat imgo = imread("def.jpg", 1);

Mat img;

blur(imgo, imgo, Size(3, 3));

cvtColor(imgo, img, CV\_BGR2HLS);

vector<Mat> channel;

split(img, channel);

if (q != 0 && w != 0){

value1 = channel[0].at<uchar>(w,q);

value2 = channel[1].at<uchar>(w, q);

value3 = channel[2].at<uchar>(w, q);

}

Mat img1(img.rows, img.cols, CV\_8UC1);

img1 = Scalar(0);

for (i = 0; i < img1.rows; i++){

for (j = 0; j < img1.cols; j++){

a1 = value1 - 15;

a2 = value1 + 15;

if (a1 < 0) a1 = 0;

if (a2 > 255) a2 = 255;

b1 = value2 - 15;

b2 = value2 + 15;

if (b1 < 0) b1 = 0;

if (b2 > 255) b2 = 255;

c1 = value3 - 15;

c2 = value3 + 15;

if (c1 < 0) c1 = 0;

if (c2 > 255) c2 = 255;

if (channel[0].at<uchar>(i, j) >= a1&&channel[0].at<uchar>(i, j)<=a2&&channel[1].at<uchar>(i, j) >= b1&&channel[1].at<uchar>(i, j)<=b2&&channel[2].at<uchar>(i, j) >= c1&&channel[2].at<uchar>(i, j)<=c2) img1.at<uchar>(i, j) = 255;

else img1.at<uchar>(i, j) = 0;

}

}

Mat img2(img.rows, img.cols, CV\_8UC3);

img2 = Scalar(255,255,255);

deque<Point> q;

int k, l, x[3] = { 175 }, y = 2,z=0, i1, j1;

Point pt;

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (img1.at<uchar>(i, j) == 255 ) {

q.push\_front(Point(j, i));

y--;

if (y < 0) y = 2;

z--;

if (z < 0) z = 2;

x[y] = x[y] - 37;

if (x[y] <= 0) x[y] = 230;

while (!q.empty()){

pt=q.back();

q.pop\_back();

i1 = pt.y;

j1 = pt.x;

for (k = i1 + 1; k > i1 - 2; k--){

for (l = j1 + 1; l > j1 - 2; l--){

if (img1.at<uchar>(k, l) == 255 ) {

q.push\_front(Point(l, k));

img2.at<Vec3b>(k, l)[y] = x[y];

img2.at<Vec3b>(k, l)[z] = 255-x[y];

img2.at<Vec3b>(k, l)[(y+z)/2] =( x[y]+255)/2;

img1.at<uchar>(k, l) = 0;

}

}

}

}

}

}

}

setMouseCallback("graph", Call1, NULL);

Mat imgx;

cvtColor(img2, imgx, CV\_BGR2HLS);

vector<Mat> cha;

split(imgx, cha);

if (a != 0 && b != 0){

val1 = cha[0].at<uchar>(b, a);

val2 = cha[1].at<uchar>(b, a);

val3 = cha[2].at<uchar>(b, a);

}

Mat img3(imgx.rows, imgx.cols, CV\_8UC1);

img3 = Scalar(0);

for (i = 0; i < imgx.rows; i++){

for (j = 0; j < imgx.cols; j++){

x1 = val1 - 15;

x2 = val1 + 15;

if (x1 < 0) x1 = 0;

if (x2 > 255) x2 = 255;

y1 = val2 - 15;

y2 = val2 + 15;

if (y1 < 0) y1 = 0;

if (y2 > 255) y2 = 255;

z1 = val3 - 15;

z2 = val3 + 15;

if (z1 < 0) z1 = 0;

if (z2 > 255) z2 = 255;

if (cha[0].at<uchar>(i, j) >= x1&&cha[0].at<uchar>(i, j) <= x2&&cha[1].at<uchar>(i, j) >= y1&&cha[1].at<uchar>(i, j) <= y2&&cha[2].at<uchar>(i, j) >= z1&&cha[2].at<uchar>(i, j) <= z2) img3.at<uchar>(i, j) = 255;

else img3.at<uchar>(i, j) = 0;

}

}

imshow("Result", img3);

imshow("binary", imgo);

imshow("graph", img2);

if (waitKey(30) == 27) break;

}

return 0;

}

Blob with center detection

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

#include <vector>

#include <deque>

using namespace std;

using namespace cv;

int q=0,w=0,a=0,b=0;

void Call(int event, int x, int y, int flags, void\* userdata){

if (event == EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

void Call1(int event, int x, int y, int flags, void\* userdata){

if (event == EVENT\_LBUTTONDOWN){

a = x;

b = y;

}

}

int main(int argc, \_TCHAR\* argv[])

{

/\*VideoCapture cap(0);\*/

int value1 = 100,value2=100,value3=100,val1=100,val2=100,val3=100;

int i, j, a1 = 0, a2 = 25, b1 = 0, b2 = 250,c1 = 0, c2 = 250;

int x1 = 0, x2 = 250, y1 = 0, y2 = 250, z1 = 0, z2 = 250;

namedWindow("binary", 1);

namedWindow("graph", 1);

while (1){

/\*Mat imgo;

cap >> imgo;\*/

setMouseCallback("binary", Call, NULL);

Mat imgo = imread("def.jpg", 1);

Mat img;

blur(imgo, imgo, Size(3, 3));

cvtColor(imgo, img, CV\_BGR2HLS);

vector<Mat> channel;

split(img, channel);

if (q != 0 && w != 0){

value1 = channel[0].at<uchar>(w,q);

value2 = channel[1].at<uchar>(w, q);

value3 = channel[2].at<uchar>(w, q);

}

Mat img1(img.rows, img.cols, CV\_8UC1);

img1 = Scalar(0);

for (i = 0; i < img1.rows; i++){

for (j = 0; j < img1.cols; j++){

a1 = value1 - 15;

a2 = value1 + 15;

if (a1 < 0) a1 = 0;

if (a2 > 255) a2 = 255;

b1 = value2 - 15;

b2 = value2 + 15;

if (b1 < 0) b1 = 0;

if (b2 > 255) b2 = 255;

c1 = value3 - 15;

c2 = value3 + 15;

if (c1 < 0) c1 = 0;

if (c2 > 255) c2 = 255;

if (channel[0].at<uchar>(i, j) >= a1&&channel[0].at<uchar>(i, j)<=a2&&channel[1].at<uchar>(i, j) >= b1&&channel[1].at<uchar>(i, j)<=b2&&channel[2].at<uchar>(i, j) >= c1&&channel[2].at<uchar>(i, j)<=c2) img1.at<uchar>(i, j) = 255;

else img1.at<uchar>(i, j) = 0;

}

}

Mat img2(img.rows, img.cols, CV\_8UC3);

img2 = Scalar(255,255,255);

deque<Point> q;

int k, l, x[3] = { 175 }, y = 2,z=0, i1, j1;

Point pt;

for (i = 0; i < img.rows; i++){

for (j = 0; j < img.cols; j++){

if (img1.at<uchar>(i, j) == 255 ) {

q.push\_front(Point(j, i));

y--;

if (y < 0) y = 2;

z--;

if (z < 0) z = 2;

x[y] = x[y] - 37;

if (x[y] <= 0) x[y] = 230;

while (!q.empty()){

pt=q.back();

q.pop\_back();

i1 = pt.y;

j1 = pt.x;

for (k = i1 + 1; k > i1 - 2; k--){

for (l = j1 + 1; l > j1 - 2; l--){

if (img1.at<uchar>(k, l) == 255 ) {

q.push\_front(Point(l, k));

img2.at<Vec3b>(k, l)[y] = x[y];

img2.at<Vec3b>(k, l)[z] = 255-x[y];

img2.at<Vec3b>(k, l)[(y+z)/2] =( x[y]+255)/2;

img1.at<uchar>(k, l) = 0;

}

}

}

}

}

}

}

setMouseCallback("graph", Call1, NULL);

Mat imgx;

cvtColor(img2, imgx, CV\_BGR2HLS);

vector<Mat> cha;

split(imgx, cha);

if (a != 0 && b != 0){

val1 = cha[0].at<uchar>(b, a);

val2 = cha[1].at<uchar>(b, a);

val3 = cha[2].at<uchar>(b, a);

}

Mat img3(imgx.rows, imgx.cols, CV\_8UC1);

img3 = Scalar(0);

int sumx = 0, sumy = 0, cx = 0, cy = 0, count = 0;;

for (i = 0; i < imgx.rows; i++){

for (j = 0; j < imgx.cols; j++){

x1 = val1 - 10;

x2 = val1 + 10;

if (x1 < 0) x1 = 0;

if (x2 > 255) x2 = 255;

y1 = val2 - 10;

y2 = val2 + 10;

if (y1 < 0) y1 = 0;

if (y2 > 255) y2 = 255;

z1 = val3 - 10;

z2 = val3 + 10;

if (z1 < 0) z1 = 0;

if (z2 > 255) z2 = 255;

if (cha[0].at<uchar>(i, j) >= x1&&cha[0].at<uchar>(i, j) <= x2&&cha[1].at<uchar>(i, j) >= y1&&cha[1].at<uchar>(i, j) <= y2&&cha[2].at<uchar>(i, j) >= z1&&cha[2].at<uchar>(i, j) <= z2) {

img3.at<uchar>(i, j) = 255;

sumx = sumx + i;

sumy = sumy + j;

count++;

}

else img3.at<uchar>(i, j) = 0;

}

}

if (count <= 0) count = 1;

cx = sumx / count;

cy = sumy / count;

if(count!=1) circle(img3, Point(cy, cx), 2, 0, 5, 8, 0);

imshow("Result", img3);

imshow("binary", imgo);

imshow("graph", img2);

if (waitKey(30) == 27) break;

}

return 0;

}

Ps1

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <math.h>

#include <vector>

#include <deque>

using namespace std;

using namespace cv;

int main(int argc, \_TCHAR\* argv[])

{

VideoCapture cap("ps1.mp4");

namedWindow("original", 1);

int i, j;

while (1){

Mat imgo;

cap >> imgo;

Mat img = imgo.clone();

cvtColor(imgo, imgo, CV\_BGR2GRAY);

Mat imgx(imgo.rows, img.cols, CV\_8UC1);

imgx = Scalar(0);

for (i = 0; i < imgo.rows; i++){

for (j = 0; j < imgo.cols; j++){

if (imgo.at<uchar>(i, j)>5) {

imgx.at<uchar>(i, j) = 255;

}

}

}

Mat img1(imgx.rows, imgx.cols, CV\_8UC1);

img1 = Scalar(0);

Mat img2(imgx.rows, imgx.cols, CV\_8UC1);

img2 = Scalar(0);

deque<Point> q;

int k, l, x = 200, i1, j1,y,m,n,w;

Point pt;

for (i = 1; i < imgx.rows - 1; i++){

for (j = 1; j < imgx.cols - 1; j++){

if (imgx.at<uchar>(i, j) == 255) {

q.push\_front(Point(j, i));

x = x - 50;

if (x <= 0) x = 250;

y = 0;

m = 0;

n = 10000;

w = 10000;

while (!q.empty()){

pt = q.back();

q.pop\_back();

i1 = pt.y;

j1 = pt.x;

for (k = i1 + 1; k > i1 - 2; k--){

for (l = j1 + 1; l > j1 - 2; l--){

if (imgx.at<uchar>(k, l) == 255 && k<img.rows&&k>0 && l<img.cols&&l>0) {

q.push\_front(Point(l, k));

if (k > y) y = k;

if (l > m) m = l;

if (k < w) w = k;

if (l < n) n = l;

img2.at<uchar>(k, l) = x;

imgx.at<uchar>(k, l) = 0;

}

}

}

}

if(i>10&&j>20) rectangle(imgx, Point(n, w ), Point(m, y), Scalar(200), CV\_FILLED, 8, 0);

}

}

}

for (i = 0; i < imgx.rows; i++){

for (j = 4 \* img.cols / 6; j <= 4 \* img.cols / 6 + 10; j++){

if (imgx.at<uchar>(i, j) == 0) {

img.at<Vec3b>(i, j)[0] = 255;

img.at<Vec3b>(i, j)[1] = 255;

img.at<Vec3b>(i, j)[2] = 255;

}

}

}

for (i = 0; i < imgx.rows; i++){

for (j = 2 \* img.cols / 6; j <= 2 \* img.cols / 6 + 10; j++){

if (imgx.at<uchar>(i, j) == 0) {

img.at<Vec3b>(i, j)[0] = 255;

img.at<Vec3b>(i, j)[1] = 255;

img.at<Vec3b>(i, j)[2] = 255;

}

}

}

imshow("original", img);

if (waitKey(50) == 27) break;

}

return 0;

}