APURV KUMAR 14CS10006

CHESSBOARD

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i=1, j=1;

Mat img(500, 500, CV\_8UC1, 255);

for (i = 0; i < 500; i++){

for (j = 0; j < 500; j++){

if (((i/50)%2==0)&&(((j/50)%2==0))) img.at < uchar >(i,j) = 0;

if (((i / 50) % 2 == 1) && (((j / 50) % 2 == 1))) img.at < uchar >(i, j) = 0;

}

}

imshow("Chess Board",img);

waitKey(0);

}

FLAG

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

Mat img(300, 500, CV\_8UC1);

Mat img1(100, 500, CV\_8UC1, Scalar(255));

Mat img2(100, 500, CV\_8UC1, Scalar(100));

Mat img3(100, 500, CV\_8UC1, Scalar(0));

Mat img4(200, 500, CV\_8UC1);

vconcat(img1, img2, img4);

vconcat(img4, img3, img);

imshow("Blank", img);

waitKey(0);

return 0;

}

EROSION/DILATION

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i, j,max,i1,j1,min;

Point pt,pt1,pt2;

pt.x = 300;

pt.y = 300;

pt1.x = 0;

pt1.y = 150;

pt2.x = 600;

pt2.y = 450;

Mat img(600, 600, CV\_8SC1,Scalar(255));

Mat img1(600, 600, CV\_8SC1);

circle(img,pt,100,0, 4, 8, 0);

rectangle(img, pt1, pt2,0, 4, 8, 0);

img1 = img.clone();

imshow("Original", img);

for (i = 1; i < 600-1; i++){

for (j = 1; j < 600-1; j++){

max = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

if (img.at<uchar>(i1, j1)>max) {

max = img.at<uchar>(i1, j1);

}

}

}

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

}

}

imshow("Test1", img1);//EROSION

Mat img2;

img2 = img.clone();

for (i = 1; i < 600 - 1; i++){

for (j = 1; j < 600 - 1; j++){

min = 255;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

if (img.at<uchar>(i1, j1)<min) {

min = img.at<uchar>(i1, j1);

}

}

}

img2.at<uchar>(i, j) = min;

}

}

imshow("Test2", img2);//DILATION

waitKey(0);

return 0;

}

Erosion/Dilation of download.jpg

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i, j,max,i1,j1,min;

Point pt,pt1,pt2;

pt.x = 300;

pt.y = 300;

pt1.x = 0;

pt1.y = 150;

pt2.x = 600;

pt2.y = 450;

/\*Mat img(600, 600, CV\_8SC1,Scalar(255));

Mat img1(600, 600, CV\_8SC1);\*/

/\*circle(img,pt,100,0, 4, 8, 0);

rectangle(img, pt1, pt2,0, 4, 8, 0);\*/

Mat img = imread("download.jpg", 0);

Mat img1 = img.clone();

imshow("Original", img1);

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

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

max = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

if (img.at<uchar>(i1, j1)>max) {

max = img.at<uchar>(i1, j1);

}

}

}

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

}

}

imshow("EROSION", img1);//EROSION

Mat img2;

img2 = img.clone();

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

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

min = 255;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

if (img.at<uchar>(i1, j1)<min) {

min = img.at<uchar>(i1, j1);

}

}

}

img2.at<uchar>(i, j) = min;

}

}

imshow("DILATION", img2);//DILATION

waitKey(0);

return 0;

}

Filtering

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i, j,i1,j1,k,max=0,mode,mean,sum,m,n,t;

Point pt,pt1,pt2;

pt.x = 300;

pt.y = 300;

pt1.x = 0;

pt1.y = 150;

pt2.x = 600;

pt2.y = 450;

int a[255] = {0};

int b[9] = { 0 };

Mat img = imread("download.jpg", 0);

Mat img1 = img.clone();

imshow("Original", img1);

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

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

mode = 0;

max=0;

for (k = 0; k < 255; k++) a[k] = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

a[img.at<uchar>(i1, j1)]++;

}

}

for (k = 0; k < 255; k++) {

if (a[k]>max) {

max = a[k];

mode = k;

}

}

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

}

}

imshow("MODE", img1);//MODE

Mat img2;

img2 = img.clone();

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

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

mean = 0;

sum = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

sum = sum + img.at<uchar>(i1, j1);

}

}

mean = sum / 9;

img2.at<uchar>(i, j) = mean;

}

}

imshow("MEAN", img2);//MEAN

Mat img3;

img3 = img.clone();

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

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

for (k = 0; k < 9; k++) b[k] = 0;

k = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

b[k] = img.at<uchar>(i1, j1);

k++;

}

}

for (m = 0; m < 9; m++){

t = 0;

for (n = m; n < 9; n++) {

if (b[n]>t) t = b[n];

}

t = b[m];

b[m] = b[n];

b[n] = t;

}

img3.at<uchar>(i, j) = b[4];

}

}

imshow("MEDIAN", img3);//MEDIAN

waitKey(0);

}

Filtering color

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i, j,i1,j1,k,max=0,mode,mean,sum,m,n,t,x;

Point pt,pt1,pt2;

pt.x = 300;

pt.y = 300;

pt1.x = 0;

pt1.y = 150;

pt2.x = 600;

pt2.y = 450;

int a[255] = {0};

int b[9] = { 0 };

Mat img = imread("download.jpg", 1);

Mat img1 = img.clone();

imshow("Original", img);

for (x = 0; x < 2; x++){

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

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

mode = 0;

max = 0;

for (k = 0; k < 255; k++) a[k] = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

a[img.at<Vec3b>(i1, j1)[x]]++;

}

}

for (k = 0; k < 255; k++) {

if (a[k]>max) {

max = a[k];

mode = k;

}

}

img1.at<Vec3b>(i, j)[x] = mode;

}

}

}

imshow("MODE", img1);//MODE

Mat img2;

img2 = img.clone();

for (x = 0; x < 2; x++){

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

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

mean = 0;

sum = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

sum = sum + img.at<Vec3b>(i1, j1)[x];

}

}

mean = sum / 9;

img2.at<Vec3b>(i, j)[x] = mean;

}

}

}

imshow("MEAN", img2);//MEAN

Mat img3;

img3 = img.clone();

for (x = 0; x < 2; x++){

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

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

for (k = 0; k < 9; k++) b[k] = 0;

k = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

b[k] = img.at<Vec3b>(i1, j1)[x];

k++;

}

}

for (m = 0; m < 9; m++){

t = 0;

for (n = m; n < 9; n++) {

if (b[n]>t) t = b[n];

}

t = b[m];

b[m] = b[n];

b[n] = t;

}

img3.at<Vec3b>(i, j)[x] = b[4];

}

}

}

imshow("MEDIAN", img3);//MEDIAN

waitKey(0);

}

Gausian filter

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

using namespace cv;

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

{

int i, j,i1,j1,k,max=0,mode,mean,sum,m,n,t,x;

Point pt,pt1,pt2;

pt.x = 300;

pt.y = 300;

pt1.x = 0;

pt1.y = 150;

pt2.x = 600;

pt2.y = 450;

int a[255] = {0};

int b[9] = { 0 };

int c[9] = { 1, 2, 1, 2, 4, 2, 1, 2, 1 };

Mat img = imread("download.jpg", 1);

Mat img1 = img.clone();

imshow("Original", img);

for (x = 0; x <= 2; x++){

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

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

mode = 0;

max = 0;

for (k = 0; k < 255; k++) a[k] = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

a[img.at<Vec3b>(i1, j1)[x]]++;

}

}

for (k = 0; k < 255; k++) {

if (a[k]>max) {

max = a[k];

mode = k;

}

}

img1.at<Vec3b>(i, j)[x] = mode;

}

}

}

imshow("MODE", img1);//MODE

Mat img2;

img2 = img.clone();

for (x = 0; x <= 2; x++){

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

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

mean = 0;

sum = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

sum = sum + img.at<Vec3b>(i1, j1)[x];

}

}

mean = sum / 9;

img2.at<Vec3b>(i, j)[x] = mean;

}

}

}

imshow("MEAN", img2);//MEAN

Mat img3;

img3 = img.clone();

for (x = 0; x <= 2; x++){

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

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

for (k = 0; k < 9; k++) b[k] = 0;

k = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

b[k] = img.at<Vec3b>(i1, j1)[x];

k++;

}

}

for (m = 0; m < 9; m++){

t = 0;

for (n = m; n < 9; n++) {

if (b[n]>t) t = b[n];

}

t = b[m];

b[m] = b[n];

b[n] = t;

}

img3.at<Vec3b>(i, j)[x] = b[4];

}

}

}

imshow("MEDIAN", img3);//MEDIAN

Mat img4 = img.clone();

for (x = 0; x <= 2; x++){

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

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

mean = 0;

sum = 0;

k = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

sum = sum + img.at<Vec3b>(i1, j1)[x]\*c[k];

k++;

}

}

mean = sum / 16;

img4.at<Vec3b>(i, j)[x] = mean;

}

}

}

imshow("GAUSSIAN MEAN", img4);//GAUSIAN MEAN

waitKey(0);

return 0;

}

Histogram Equalize

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;

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

{

Mat img;

img = imread("abc.jpg", 0);

imshow("original", img);

Mat imga(img.rows, img.cols, CV\_8SC1);

int a[256] = { 0 };

int i,j;

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

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

a[img.at<uchar>(i, j)]++;

}

}

Mat img1(660, 257, CV\_8SC1);

img1 = Scalar(255);

for (i=0; i < 255; i++){

for (j =1; j<= a[i]/2;j++) img1.at<uchar>(660-j,i+1)=0;

}

imshow("Histogram",img1);

Mat img2(660, 257, CV\_8SC1);

img2 = Scalar(255);

equalizeHist(img,imga);

imshow("Refined", imga);

for (i = 0; i < 255; i++) a[i] = 0;

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

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

a[imga.at<uchar>(i, j)]++;

}

}

for (i = 0; i < 255; i++){

for (j = 1; j <= a[i] / 2; j++) img2.at<uchar>(660 - j, i + 1) = 0;

}

imshow("Histogram Refined", img2);

waitKey(0);

return 0;

}

Trackbar

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;

Mat greytobin(int slidervalue){

int i, j,k;

Mat img;

img = imread("abc.jpg", 1);

for (k = 0; k < 3; k++){

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

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

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

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

}

}

}

return img;

}

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

{

int slidervalue = 300;

namedWindow("Trackbar", 1);

createTrackbar("Slider", "Trackbar", &slidervalue, 512);

while (1){

Mat img=greytobin(slidervalue);

imshow("Trackbar", img);

int ikey = waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Rotation

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;

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

{

Mat img;

img=imread("abc.jpg",1);

int slidervalue = 0;

namedWindow("Trackbar", 1);

Point pt;

pt.x = img.rows / 2;

pt.y = img.cols / 2;

createTrackbar("Rotation", "Trackbar", &slidervalue, 180);

while (1){

Mat matRotation = getRotationMatrix2D(pt,slidervalue, 1);

Mat imgRotated;

warpAffine(img, imgRotated, matRotation, img.size());

imshow("Trackbar", imgRotated);

int ikey = waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Rotation and binary simultaneous

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;

Mat greytobin(int slidervalue){

int i, j, k;

Mat img;

img = imread("abc.jpg", 1);

for (k = 0; k < 3; k++){

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

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

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

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

}

}

}

return img;

}

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

{

Mat img;

img=imread("abc.jpg",1);

int slidervalue = 0,slidervalue1=300;

namedWindow("Trackbar", 1);

Point pt;

pt.x = img.rows / 2;

pt.y = img.cols / 2;

createTrackbar("Rotation", "Trackbar", &slidervalue, 180);

createTrackbar("ToBinary", "Trackbar", &slidervalue1, 512);

while (1){

Mat matRotation = getRotationMatrix2D(pt,slidervalue, 1);

Mat imgRotated;

warpAffine(img, imgRotated, matRotation, img.size());

imshow("Trackbar", imgRotated);

waitKey(0);

imgRotated = greytobin(slidervalue1);

imshow("Trackbar", imgRotated);

img=imgRotated;

int ikey = waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Feature reading

#include "stdafx.h"

#include "opencv2/core/core.hpp"

#include "opencv2/highgui/highgui.hpp"

#include <stdio.h>

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

using namespace cv;

Mat feature(int value){

Mat img = imread("abc.jpg", 1);

Mat img1(img.rows, img.cols, CV\_8SC3);

img1 = Scalar(255,255,255);

int i, j,k,l,x, max, min;

for (x = 0; x < 3; x++){

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

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

max = 0;

min = 256;

for (k = i - 1; k <= i + 1; k++){

for (l = j - 1; l <= j + 1; l++){

if (img.at<Vec3b>(k, l)[x]>max) max = img.at<Vec3b>(k, l)[x];

if (img.at<Vec3b>(k, l)[x] < min) min = img.at<Vec3b>(k, l)[x];

}

}

if (max - min >= value) img1.at<Vec3b>(i, j)[x] = 0;

}

}

}

return img1;

}

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

{

namedWindow("Feature", 1);

int slidervalue = 100;

createTrackbar("Slider", "Feature", &slidervalue, 255);

while (1){

Mat img1=feature(slidervalue);

imshow("Feature", img1);

int ikey=waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Prewitt And Sobel

#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>

using namespace cv;

Mat feature(int value){

Mat img = imread("abc.jpg", 1);

Mat img1(img.rows, img.cols, CV\_8SC3);

img1 = Scalar(255,255,255);

int gx[9] = { -1, 0, 1, -1, 0, 1, -1, 0, 1 };

int gy[9] = { -1, -1, -1, 0, 0, 0, 1, 1, 1 };

int i, j,k,l,x,y, Gx, Gy,G;

for (x = 0; x < 3; x++){

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

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

Gx=0;

Gy = 0;

G = 0;

y = 0;

for (k = i - 1; k <= i + 1; k++){

for (l = j - 1; l <= j + 1; l++){

Gx = Gx + img.at<Vec3b>(k, l)[x] \* gx[y];

Gy = Gy + img.at<Vec3b>(k, l)[x] \* gy[y];

y++;

}

}

G =sqrt(Gx\*Gx+Gy\*Gy);

if (G > value) img1.at<Vec3b>(i, j)[x] = 0;

}

}

}

return img1;

}

Mat feature1(int value){

Mat img = imread("abc.jpg", 1);

Mat img1(img.rows, img.cols, CV\_8SC3);

img1 = Scalar(255, 255, 255);

int gx[9] = { -1, 0, 1, -2, 0, 2, -1, 0, 1 };

int gy[9] = { -1, -2, -1, 0, 0, 0, 1, 2, 1 };

int i, j, k, l, x, y, Gx, Gy, G;

for (x = 0; x < 3; x++){

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

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

Gx = 0;

Gy = 0;

G = 0;

y = 0;

for (k = i - 1; k <= i + 1; k++){

for (l = j - 1; l <= j + 1; l++){

Gx = Gx + img.at<Vec3b>(k, l)[x] \* gx[y];

Gy = Gy + img.at<Vec3b>(k, l)[x] \* gy[y];

y++;

}

}

G = sqrt(Gx\*Gx + Gy\*Gy);

if (G > value) img1.at<Vec3b>(i, j)[x] = 0;

}

}

}

return img1;

}

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

{

namedWindow("Prewitt", 1);

int slidervalue = 100;

createTrackbar("Slider", "Prewitt", &slidervalue, 255);

while (1){

Mat img1=feature(slidervalue);

imshow("Prewitt", img1);

Mat img2 = feature1(slidervalue);

imshow("Sobel", img2);

int ikey=waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Hysterisis (Canny)

#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>

using namespace cv;

Mat feature2(int value,int value1){

Mat img = imread("abc.jpg", 1);

int gx[9] = { -1, 0, 1, -2, 0, 2, -1, 0, 1 };

int gy[9] = { -1, -2, -1, 0, 0, 0, 1, 2, 1 };

int i, j, k, l, y, Gx, Gy, G,sum,mean,i1,j1,x;

int c[9] = { 1, 2, 1, 2, 4, 2, 1, 2, 1 };

for (x = 0; x < 3; x++){

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

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

mean = 0;

sum = 0;

k = 0;

for (i1 = i - 1; i1 <= i + 1; i1++){

for (j1 = j - 1; j1 <= j + 1; j1++){

sum = sum + img.at<Vec3b>(i1, j1)[x] \* c[k];

k++;

}

}

mean = sum / 16;

img.at<Vec3b>(i, j)[x] = mean;

}

}

}

for (x = 0; x < 3; x++){

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

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

Gx = 0;

Gy = 0;

G = 0;

y = 0;

for (k = i - 1; k <= i + 1; k++){

for (l = j - 1; l <= j + 1; l++){

Gx = Gx + img.at<Vec3b>(k, l)[x] \* gx[y];

Gy = Gy + img.at<Vec3b>(k, l)[x] \* gy[y];

y++;

}

}

G = sqrt(Gx\*Gx + Gy\*Gy);

if (G > value) img.at<Vec3b>(i, j)[x] = 0;

}

}

}

Mat img2;

Canny(img,img2,value1, value, 3, true);

return img2;

}

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

{

namedWindow("Sobel", 1);

int slidervalue = 200;

int slidervalue1 = 50;

createTrackbar("SliderUp", "Sobel", &slidervalue, 255);

createTrackbar("SliderLo", "Sobel", &slidervalue1, 255);

while (1){

Mat img3 = feature2(slidervalue,slidervalue1);

imshow("Sobel", img3);

int ikey = waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Corner 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>

using namespace cv;

Mat feature( int m,int q){

if (m == 0) m = 1;

if (q == 0) q = 1;

double n = (double)q / 1000.0;

vector<Point2f> corners;

int i;

Mat img = imread("abc.jpg", 0);

goodFeaturesToTrack(img, corners,m,n,10, Mat(), 3, false, 0.04);

Mat img1 = imread("abc.jpg", 1);

for (i = 0; i < corners.size(); i++){

circle(img1, corners[i], 1, 0, 5, 8, 0);

}

return img1;

}

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

{

namedWindow("Corners", 1);

int slidervalue = 0;

int slidervalue1 = 0;

createTrackbar("MaxCorners", "Corners", &slidervalue,3000);

createTrackbar("Quality", "Corners", &slidervalue1, 1000);

while (1){

Mat img1 = feature(slidervalue,slidervalue1);

imshow("Corners", img1);

int ikey = waitKey(50);

if (ikey == 27) break;

}

return 0;

}

Color Splitting

#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>

using namespace cv;

Mat feature(int b1, int b2, int g1, int g2, int r1, int r2){

Mat img = imread("abc.jpg", 1);

Mat img1;

cvtColor(img, img1, CV\_BGR2HLS);

vector<Mat> channel;

split(img1, channel);

int i, j;

Mat img2(img1.rows, img1.cols, CV\_8UC3, Scalar(0, 0, 0));

Mat img3(img1.rows, img1.cols, CV\_8UC3, Scalar(0, 0, 0));

Mat img4(img1.rows, img1.cols, CV\_8UC3, Scalar(0, 0, 0));

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

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

if (channel[0].at<uchar>(i, j)>b1&&channel[0].at<uchar>(i, j)<b2) img2.at<Vec3b>(i, j)[0] = 255;

if (channel[1].at<uchar>(i, j)>g1&&channel[1].at<uchar>(i, j)<g2) img3.at<Vec3b>(i, j)[1] = 255;

if (channel[2].at<uchar>(i, j)>r1&&channel[2].at<uchar>(i, j)<r2) img4.at<Vec3b>(i, j)[2] = 255;

}

}

imshow("Woa Blue", img2);

imshow("Woa Green", img3);

imshow("Woa Red", img4);

Mat img5 = img2 + img3 + img4;

imshow("Woa Sum of HSL split", img5);

Mat img6;

cvtColor(img5, img6, CV\_HLS2BGR);

return img6;

}

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

{

namedWindow("Controls", 1);

int b1 = 10;

createTrackbar("B1", "Controls", &b1,255);

int b2 = 10;

createTrackbar("B2", "Controls", &b2, 255);

int g1 = 10;

createTrackbar("G1", "Controls", &g1, 255);

int g2 = 10;

createTrackbar("G2", "Controls", &g2, 255);

int r1 = 10;

createTrackbar("R1", "Controls", &r1, 255);

int r2 = 10;

createTrackbar("R2", "Controls", &r2, 255);

while (1){

Mat ans = feature(b1, b2, g1, g2, r1, r2);

imshow("Controls", ans);

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

}

return 0;

}

Rotated Video

#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>

using namespace cv;

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

{

Mat frame;

namedWindow("Say Hi", 1);

VideoCapture cap(0);

cap >> frame;

Mat out;

int slidervalue = 0;

Point pt;

pt.x = frame.rows / 2;

pt.y = frame.cols / 2;

createTrackbar("Rotation", "Say Hi", &slidervalue, 360);

while (1){

cap >> frame;

Mat matRotation = getRotationMatrix2D(pt, slidervalue, 1);

Mat imgRotated;

warpAffine(frame, imgRotated, matRotation, frame.size());

out = imgRotated.clone();

imshow("Say Hi", out);

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

}

return 0;

}

Video Fun

#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>

using namespace cv;

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

{

Mat frame;

namedWindow("Say Hi", 1);

VideoCapture cap(0);

Mat out;

int slidervalue = 0;

Point pt;

createTrackbar("Rotation", "Say Hi", &slidervalue, 360);

int x = 0, i, j,y=0;

int temp;

while (1){

cap >> frame;

out = frame.clone();

for (x = 0; x < 3; x++){

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

for (j = 0; j < frame.cols / 2; j++) {

temp = out.at<Vec3b>(i, j)[x];

out.at<Vec3b>(i, j)[x] = out.at<Vec3b>(i, frame.cols - j)[x];

out.at<Vec3b>(i, frame.cols - j)[x] = temp;

}

}

}

pt.x = out.cols/2;

pt.y = out.rows/2;

Mat matRotation = getRotationMatrix2D(pt, slidervalue, 1);

Mat imgRotated;

warpAffine(out, imgRotated, matRotation, frame.size());

out = imgRotated.clone();

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

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

out.at<Vec3b>(i, j)[y]=200;

}

}

y++;

if (y == 3) y = 0;

Mat out1;

Canny(out, out1, 100, 150, 3, false);

imshow("Say Hi", out1);

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

}

return 0;

}

Transition of video to image

#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>

using namespace cv;

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

{

namedWindow("See the Change to Be change", 1);

int value = 0;

int i, j, x;

VideoCapture cap(0);

createTrackbar("Changer", "See the Change to Be change", &value, 100);

while (1){

Mat img3;

cap >> img3;

Mat img1;

Mat img2 = imread("def.jpg", 1);

resize(img3, img1, img2.size());

Mat img = img2.clone();

for (x = 0; x < 3; x++){

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

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

img.at<Vec3b>(i, j)[x] = img2.at<Vec3b>(i, j)[x] \* (value / 100.0) + img1.at<Vec3b>(i, j)[x] \* ((100-value) / 100.0);

}

}

}

imshow("See the Change to Be change", img);

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

}

return 0;

}

Longest Path

#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>

using namespace cv;

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

{

VideoCapture cap(0);

namedWindow("canny", 1);

int value = 100;

createTrackbar("Threshold", "canny", &value, 255);

int i, j, k,i1,j1;

while (1){

Mat img;

cap >> img;

Mat img1;

Canny(img, img1, 120, 200, 3, false);

Mat plot(10000,400,CV\_8UC1,Scalar(0));

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

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

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

for (k = 0; k < 360; k++) plot.at<uchar>((1000+(int)(j\*cos(k\*3.14 / 180.0) + i\*sin(k\*3.14 / 180.0)) ), k)++;

}

}

}

k = 0;

Mat img2(img1.rows, img1.cols, CV\_8UC1);

img2 = Scalar(0);

for (i = 0; i < 1400; i++){

for (j = 0; j < 180; j++){

if (plot.at<uchar>(i, j) > k && plot.at<uchar>(i,j)<value) {

k = plot.at<uchar>(i, j);

i1 = i;

j1 = j;

}

}

}

for (i = 0; i < img2.rows; i++){

for (j = 0; j < img2.cols; j++){

if (i1 == (1000 + (int)(j\*cos(j1\*3.14 / 180.0) + i\*sin(j1\*3.14 / 180.0)))) {

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

img2.at<uchar>(i, j) == 255;

circle(img2, Point(j, i), 1, 255, 2, 8, 0);

}

}

}

}

imshow("Longest path", img2);

imshow("canny", plot);

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

}

return 0;

}

Hough circles

#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>

using namespace cv;

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

{

VideoCapture cap(0);

while (1){

vector<Vec3f> circles;

Mat img;

Mat img1;

cap >> img1;

cvtColor(img1, img1, CV\_RGB2GRAY);

GaussianBlur(img1, img, Size(9, 9), 2, 2);

HoughCircles(img, circles, CV\_HOUGH\_GRADIENT, 1, img.rows / 16, 200, 30, 0, 0);

for (size\_t i = 0; i < circles.size(); i++)

{

Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));

int radius = cvRound(circles[i][2]);

circle(img, center, 3, Scalar(0, 255, 0), -1, 8, 0);

circle(img, center, radius, Scalar(0, 0, 255), 3, 8, 0);

}

namedWindow("Hough Circle Transform Demo", CV\_WINDOW\_AUTOSIZE);

imshow("Hough Circle Transform Demo", img);

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

}

return 0;

}

Video Color 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>

using namespace cv;

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

{

int i,a=0,j,b=0,c=0,a1=255,b1=255,c1=255;

VideoCapture cap(0);

namedWindow("Contour", 1);

createTrackbar("Color\_blue1", "Contour", &a, 255);

createTrackbar("Color\_green1", "Contour", &b, 255);

createTrackbar("Color\_red1", "Contour", &c, 255);

createTrackbar("Color\_blue2", "Contour", &a1, 255);

createTrackbar("Color\_green2", "Contour", &b1, 255);

createTrackbar("Color\_red2", "Contour", &c1, 255);

while (1){

Mat img;

cap >> img;

Mat img2 = img.clone();

Mat img1;

cvtColor(img, img1, CV\_BGR2HLS);

vector<Mat> channel;

split(img1, channel);

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

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

if (channel[0].at<uchar>(i, j)<a1 && channel[0].at<uchar>(i, j)>a) {

if (channel[1].at<uchar>(i, j)<b1 && channel[1].at<uchar>(i, j)>b) {

if (channel[2].at<uchar>(i, j)<c1 && channel[2].at<uchar>(i, j)>c) {

img.at<Vec3b>(i, j) = { 255, 255, 255 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

}

Mat imga = img.clone();

GaussianBlur(img, img, Size(3, 3), 0, 0);

Canny(img, img, 50, 150,3);

GaussianBlur(img,img,Size(3,3),0,0);

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(img, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

int j,max=0;

for (i = 0; i < contours.size(); i++){

if (contours[i].size()>max) {

max = contours[i].size();

j = i;

}

}

drawContours(img2, contours, j, Scalar(0, 0, 0), 2, 8, hierarchy, 0, Point());

imshow("Contour", img2);

imshow("Canny", img);

imshow("Binary", imga);

if (waitKey(30) == 27) {

break;

}

}

return 0;

}

Color detection by mouse click

#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>

using namespace std;

using namespace cv;

int q=0, w=0;

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

if (event ==EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

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

{

int i, a = 0, j, b = 0, c = 0, a1 = 255, b1 = 255, c1 = 255;

VideoCapture cap(0);

namedWindow("Contour", 1);

while (1){

setMouseCallback("Contour",Call,NULL);

Mat img;

cap >> img;

Mat img2 = img.clone();

Mat img1;

cvtColor(img, img1, CV\_BGR2HLS);

vector<Mat> channel;

split(img1, channel);

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

j = q;

i = w;

a = channel[0].at<uchar>(i, j) - 25;

a1 = channel[0].at<uchar>(i, j) + 25;

b = channel[1].at<uchar>(i, j) - 25;

b1 = channel[1].at<uchar>(i, j) + 25;

c = channel[2].at<uchar>(i, j) - 25;

c1 = channel[2].at<uchar>(i, j) + 25;

if (a <= 0) a = 0;

if (b <= 0) b = 0;

if (c <= 0) c = 0;

if (a1 >= 255) a1 = 255;

if (b1 >= 255) b1 = 255;

if (c1 >= 255) c1 = 255;

}

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

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

if (channel[0].at<uchar>(i, j)<a1 && channel[0].at<uchar>(i, j)>a) {

if (channel[1].at<uchar>(i, j)<b1 && channel[1].at<uchar>(i, j)>b) {

if (channel[2].at<uchar>(i, j)<c1 && channel[2].at<uchar>(i, j)>c) {

img.at<Vec3b>(i, j) = { 255, 255, 255 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

else img.at<Vec3b>(i, j) = { 0, 0, 0 };

}

}

Mat imga = img.clone();

GaussianBlur(img, img, Size(3, 3), 0, 0);

Canny(img, img, 50, 150,3);

GaussianBlur(img, img, Size(3, 3), 0, 0);

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(img, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

int j,max=0;

for (i = 0; i < contours.size(); i++){

if (contours[i].size()>max) {

max = contours[i].size();

j = i;

}

}

drawContours(img2, contours, j, Scalar((a+a1)/2, (b+b1)/2, (c+c1)/2), 2, 8, hierarchy, 0, Point());

imshow("Contour", img2);

imshow("Canny", img);

imshow("Binary", imga);

if (waitKey(30) == 27) {

break;

}

}

return 0;

}

Shape 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>

using namespace std;

using namespace cv;

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

{

int value = 100;

namedWindow("shape", 1);

createTrackbar("thresh", "shape", &value, 255);

VideoCapture cap(0);

while (1){

Mat img;

cap >> img;

cvtColor(img, img, CV\_RGB2GRAY);

Mat img1(img.rows, img.cols, CV\_8UC1, Scalar(0));

Mat img2 = img.clone();

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

int i, j;

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

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

if (img.at<uchar>(i, j)>value) img.at<uchar>(i, j) = 255;

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

}

}

findContours(img, contours, hierarchy, CV\_RETR\_CCOMP, CV\_CHAIN\_APPROX\_NONE);

for (i = 0; i < contours.size(); i++){

vector<vector<Point>> result(contours.size());

approxPolyDP(contours[i], result[i], 3, true);

if (result[i].size() == 3){

line(img2, result[i][0], result[i][1],255, 4, 8, 0);

line(img2, result[i][1], result[i][2],255, 4, 8, 0);

line(img2, result[i][2], result[i][0],255, 4, 8, 0);

}

if (result[i].size() == 4){

line(img2, result[i][0], result[i][1], 255, 4, 8, 0);

line(img2, result[i][1], result[i][2], 255, 4, 8, 0);

line(img2, result[i][2], result[i][3], 255, 4, 8, 0);

line(img2, result[i][3], result[i][0], 255, 4, 8, 0);

}

if (result[i].size() > 4){

for(j = 0; j < result[i].size(); j++){

if (j == result[i].size() - 1) {

line(img2, result[i][j], result[i][0], 255, 4, 8, 0);

break;

}

line(img2, result[i][j], result[i][j+1], 255, 4, 8, 0);

}

}

}

imshow("shape", img2);

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

}

return 0;

}

Blob detection using bfs

#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[])

{

int value = 150,i,j;

namedWindow("binary", 1);

createTrackbar("thresh", "binary", &value, 255);

while (1){

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

cvtColor(img, img, CV\_RGB2GRAY);

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++){

if (img.at<uchar>(i, j) >= value) img1.at<uchar>(i, j) = 0;

else img1.at<uchar>(i, j) = 255;

}

}

Mat imgo = img1.clone();

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

img2 = Scalar(255,255,255);

deque<Point> q;

int k, l, x[3] = { 150 }, y = 2 ,z=1, 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;

}

}

}

}

}

}

}

imshow("binary", img2);

imshow("gray", imgo);

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

}

return 0;

}

Blob detection by mouse click

#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;

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

if (event == EVENT\_LBUTTONDOWN){

q = x;

w = y;

}

}

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

{

int value = 100;

int i,j,v1=0,v2=250;

namedWindow("binary", 1);

//createTrackbar("thresh", "binary", &value, 255);

while (1){

setMouseCallback("binary", Call, NULL);

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

Mat imgo = img.clone();

cvtColor(img, img, CV\_RGB2GRAY);

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

value = img.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++){

v1 = value - 5;

v2 = value + 5;

if (v1 < 0) v1 = 0;

if (v2 > 255) v2 = 255;

if (img.at<uchar>(i, j) > v1&&img.at<uchar>(i,j)<v2) 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;

}

}

}

}

}

}

}

imshow("binary", imgo);

imshow("graph", img2);

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

}

return 0;

}

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;

}

Ps1 image

#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[])

{

Mat img;

img = imread("ps-2.jpg",1);

Mat img1=imread("ps-2.jpg",0);

int i, j;

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

img2 = Scalar(0);

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

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

if (img1.at<uchar>(i, j)>10 && img1.at<uchar>(i, j) < 240) img2.at<uchar>(i, j) = 255;

}

}

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

img3 = Scalar(0);

vector<vector<Point> > contours1;

vector<Vec4i> hierarchy1;

findContours(img2, contours1, hierarchy1, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

for (int i = 0; i< contours1.size(); i++)

{

drawContours(img2, contours1, i, 255,-1, 8, hierarchy1, 0, Point());

}

int temp1,temp2,k=-40;

for (i = img.rows; i > 0; i--){

for (j = img.cols; j > 0; j--){

if (img2.at<uchar>(i, j) == 255) {

temp1 = img1.at<uchar>(i, j)-10;

temp2 = img1.at<uchar>(i, j) +10;

if (temp1 < 0)temp1 = 0;

if (temp2 > 255)temp2 = 250;

k = k + 70;

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

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

if (img1.at<uchar>(i, j) >= temp1 && img1.at<uchar>(i, j) <= temp2 && img2.at<uchar>(i,j)==255) {

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

img2.at<uchar>(i, j) = 0;

}

}

}

}

}

}

k = 30;

Mat img4 = img3.clone();

vector<vector<Point>> contours;

vector<Vec4i> hierarchy;

findContours(img4, contours, hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, Point(0, 0));

for (int i = 0; i< contours.size(); i++)

{

if (contours[i].size() == 4 && img3.at<uchar>(contours[i][0].y, contours[i][0].x) == k){

drawContours(img3, contours, i, 60, -1, 8, hierarchy, 0, Point());

}

}

imshow("binary",img3);

waitKey(0);

return 0;

}