**Assignment 3**

**5140379043 盛思远**

**Part 1 source code**

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <iostream>

using namespace cv;

using namespace std;

const int MAX\_GRAY\_VALUE = 256;

const int MIN\_GRAY = 90;//remove the effect of background

int main(int argc, char\*\* argv)

{

IplImage\* image = cvLoadImage("cherry.png", 0);//load image with gray model

//draw histogram and show it

int bins = 256;

int hist\_size[] = { bins };

float range[] = { 0, 256 };

const float\* ranges[] = { range };

MatND hist;//hist data

int channels[] = { 0 };

Mat gray(image, 0);

//calculate hist and store the data into dist

calcHist(&gray, 1, channels, Mat(), // do not use mask

hist, 1, hist\_size, ranges,

true, // the histogram is uniform

false);

double max\_val = 0;

//minMaxLoc(hist, 0, &max\_val, 0, 0);//calculate the max of histogram(max num of occurences)

for (int i = MIN\_GRAY; i < MAX\_GRAY\_VALUE; i++)

{

if (hist.at<float>(i) > max\_val)

{

max\_val = hist.at<float>(i);

}

}

int scale = 2;//width of hist

int hist\_height = 256;//height of hist

Mat hist\_img = Mat::zeros(hist\_height, bins\*scale, CV\_8UC3);//image of histogram

//load the data from hist to image

for (int i = MIN\_GRAY; i<bins; i++)

{

float bin\_val = hist.at<float>(i);

int intensity = cvRound(bin\_val\*hist\_height / max\_val); //要绘制的高度

rectangle(hist\_img, Point(i\*scale, hist\_height - 1),

Point((i + 1)\*scale - 1, hist\_height - intensity),

CV\_RGB(255, 255, 255));

}

double histogram[MAX\_GRAY\_VALUE];//histogram for probability of gray value

for (int i = 0; i < MAX\_GRAY\_VALUE; i++)//initialize the histogram

{

histogram[i] = 0;

}

//update the histogram

for (int row = 0; row < image->height; row++)

{

uchar\* ptr = (uchar\*)image->imageData + row\*image->widthStep;

for (int col = 0; col < image->width; col++)

{

int temp\_gray\_value = ptr[col];

histogram[temp\_gray\_value]++;

}

}

//normalization

int totalpoints = 0;

for (int i = MIN\_GRAY; i < MAX\_GRAY\_VALUE; i++)

{

totalpoints += histogram[i];

}

for (int i = MIN\_GRAY; i < MAX\_GRAY\_VALUE; i++)

{

histogram[i] = histogram[i] / totalpoints;

}

//w0:percentage of foreground

//w1:percentage of background

//u0:average gray value of foreground

//u1:average gray value of background

//u:global average gray value

//tips:w0 + w1 = 1 and u0 + u1 = u = sum(histogram[i]\*i)

double w0 = 0, w1 = 0, u0 = 0, u1 = 0, u = 0;;

for (int i = MIN\_GRAY; i < MAX\_GRAY\_VALUE; i++)//initialize u

{

u = u + i\*histogram[i];

}

//search optimal threshold values

int threshold = -1;

double max\_variance = -1;

for (int index = MIN\_GRAY; index < MAX\_GRAY\_VALUE; index++)

{

w0 = w0 + histogram[index];//update percentage of foreground

w1 = 1 - w0;//update percentage of background

u0 = u0 + histogram[index] \* index;//update average gray value of foreground

u1 = u - u0;//update average gray value of background

double temp\_variance = w0\*(u0 - u)\*(u0 - u) + w1\*(u1 - u)\*(u1 - u);

if (temp\_variance > max\_variance)

{

max\_variance = temp\_variance;

threshold = index;

}

}

//create foreground and background pictures

IplImage\* foreground\_image = cvCloneImage(image);

IplImage\* background\_image = cvCloneImage(image);

for (int row = 0; row < foreground\_image->height; row++)

{

uchar\* foreground\_ptr = (uchar\*)foreground\_image->imageData + row\*foreground\_image->widthStep;

uchar\* background\_ptr = (uchar\*)background\_image->imageData + row\*background\_image->widthStep;

for (int col = 0; col < foreground\_image->width; col++)

{

int temp\_gray\_value = foreground\_ptr[col];

if (temp\_gray\_value > MIN\_GRAY)

{

if (temp\_gray\_value > threshold)

{

foreground\_ptr[col] = 255;

background\_ptr[col] = 0;

}

else

{

foreground\_ptr[col] = 0;

background\_ptr[col] = 255;

}

}

}

}

//output the threshold

cout << "Threshold: " << threshold << endl;

cvSaveImage("foreground.png", foreground\_image);

cvSaveImage("background.png", background\_image);

cvSaveImage("histogram.png", &IplImage(hist\_img));

cvReleaseImage(&image);

cvReleaseImage(&foreground\_image);

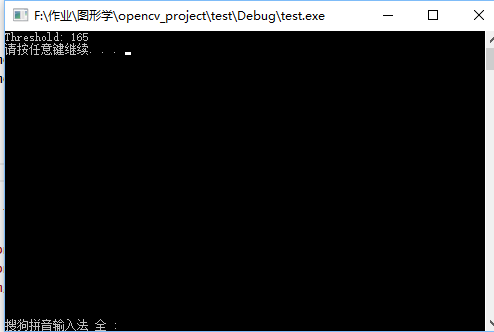
cvReleaseImage(&background\_image);

system("pause");

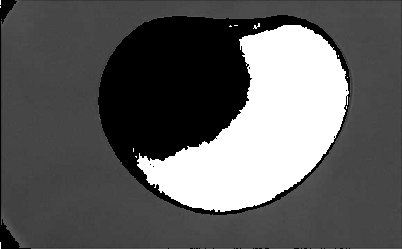
}

**Part 2 output**

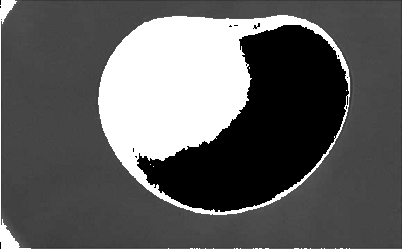
1. Threshold



1. Foreground



1. Background



1. Histogram

