

**FACULTY OF INFORMATION TECHNOLOGIES**

**DEPARTMENT OF INFORMATION SYSTEMS MANAGEMENT**

**MACHINE VISION**

*Report*

**Laboratory work #6,7**

|  |  |
| --- | --- |
|  | **Done by: Sailaubekova Zhansaya** |
|  | **Checked by: Auezova Anel** |

**Almaty 2015**

///1. Implement detecting image contours with the Canny operator.

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include "opencv2/imgproc/types\_c.h"

int main(){

cv::Mat image= cv::imread("/Users/Zhansaya/Desktop/Unknown.jpg");

cv::Mat contours;

cv::Mat gray\_image;

cvtColor( image, gray\_image, CV\_RGB2GRAY );

cv::Canny(image, // gray-level image

contours, // output contours

125, // low threshold

350); // high threshold

cv::namedWindow("Image");

cv::imshow("Image",image);

cv::namedWindow("Gray");

cv::imshow("Gray",gray\_image);

cv::namedWindow("Canny");

cv::imshow("Canny",contours);

cv::Mat contoursInv; // inverted image

cv::threshold(contours,contoursInv,

128, // values below this

255, // becomes this

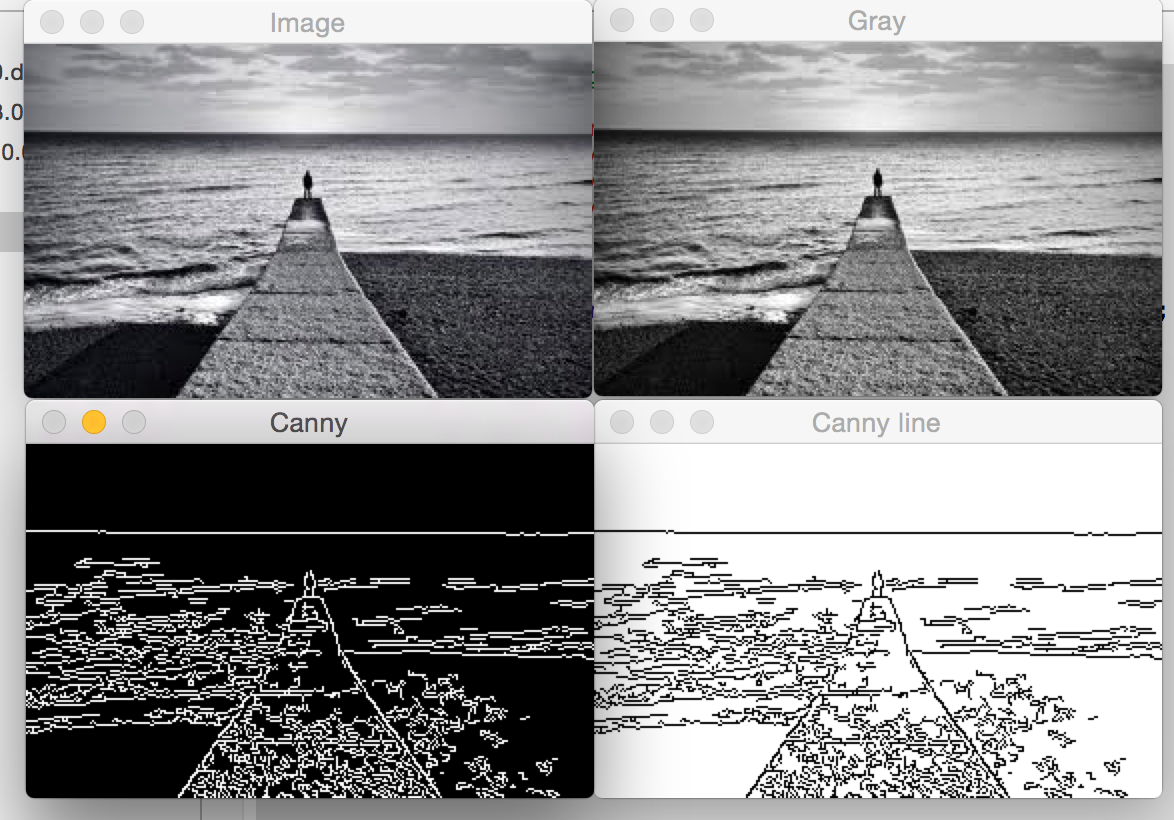
cv::THRESH\_BINARY\_INV);

cv::namedWindow("Canny line");

cv::imshow("Canny line",contoursInv);

cv::waitKey(0);

}



///2. Implement detecting lines in images with the Hough transform

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include "opencv2/imgproc/types\_c.h"

#include <math.h>

using namespace std;

int main(){

cv::Mat image= cv::imread("/Users/Zhansaya/Desktop/Unknown.jpg");

// Apply Canny algorithm

cv::Mat contours;

cv::Canny(image,contours,125,350);

// Display the image

cv::namedWindow("Original Image");

cv::imshow("Original Image",image);

// Hough tranform for line detection

std::vector<cv::Vec2f> lines;

cv::HoughLines(contours,lines,

1,M\_PI/180, // step size

80); // minimum number of votes

// Draw the lines

cv::Mat result(contours.rows,contours.cols,CV\_8U,cv::Scalar(255));

image.copyTo(result);

std::vector<cv::Vec2f>::const\_iterator it= lines.begin();

while (it!=lines.end()) {

float rho= (\*it)[0]; // first element is distance rho

float theta= (\*it)[1]; // second element is angle theta

if (theta < M\_PI/4. || theta > 3.\*M\_PI/4.) { // ~vertical line

// point of intersection of the line with first row

cv::Point pt1(rho/cos(theta),0);

// point of intersection of the line with last row

cv::Point pt2((rho-result.rows\*sin(theta))/cos(theta),result.rows);

// draw a white line

cv::line( result, pt1, pt2, cv::Scalar(255), 1);

} else { // ~horizontal line

// point of intersection of the line with first column

cv::Point pt1(0,rho/sin(theta));

// point of intersection of the line with last column

cv::Point pt2(result.cols,(rho-result.cols\*cos(theta))/sin(theta));

// draw a white line

cv::line( result, pt1, pt2, cv::Scalar(255), 1);

}

++it;

}

cv::namedWindow("Detected Lines with Hough");

cv::imshow("Detected Lines with Hough",result);

cv::waitKey(0);

return 0;

}

///3. Show how to do extracting the components' contours with your own picture

#include <opencv2/core/core.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include "opencv2/imgproc/types\_c.h"

#include <vector>

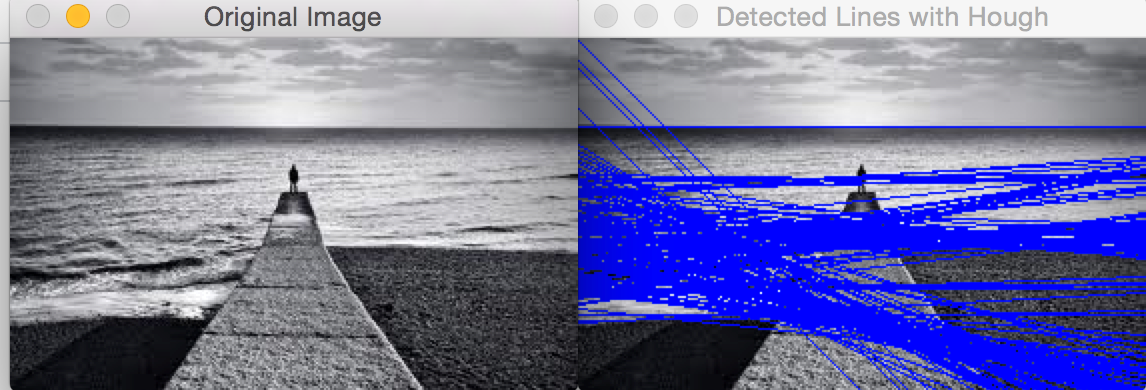
int main(){

cv::Mat image= cv::imread("/Users/Zhansaya/Desktop/Unknown.jpg");

cv::namedWindow("Original Image");

cv::imshow("Original Image",image);

cv::Mat gray(image.size(),CV\_8U);

cvtColor(image,gray,CV\_BGR2GRAY);

threshold(gray,gray,145,255,cv::THRESH\_BINARY\_INV);

cv::namedWindow("Binary Image");

cv::imshow("Binary Image",gray);

// Get the contours of the connected components

std::vector<std::vector<cv::Point>> contours;

cv::findContours(gray,

contours, // a vector of contours

CV\_RETR\_EXTERNAL , // retrieve the external contours

CV\_CHAIN\_APPROX\_NONE); // retrieve all pixels of each contours

// draw black contours on white image

cv::Mat result(image.size(),CV\_8U,cv::Scalar(255));

cv::drawContours(result,contours,

-1, // draw all contours

cv::Scalar(0), // in black

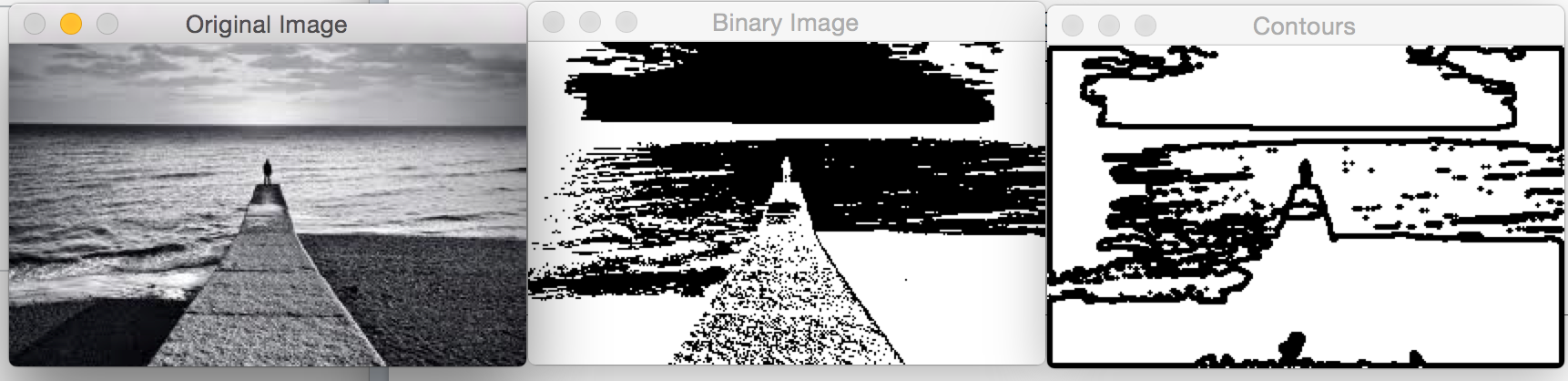
2); // with a thickness of 2

cv::namedWindow("Contours");

cv::imshow("Contours",result);

cv::waitKey(0);

}



///4. Realize detecting Harris corners

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

using namespace cv;

int main()

{

Mat cornerStrength;

Mat image = imread("/Users/Zhansaya/Desktop/Unknown.jpg",0);

cornerHarris(image,cornerStrength,

3,//neighborhood size

3,//aperture size

0.01);//harris parameter

//threshold the corner strengths

Mat harrisCorners;

double threshold = 0.0001;

cv::threshold(cornerStrength,harrisCorners,

threshold,255,THRESH\_BINARY);

imshow("Original\_image",image);

imshow("Harris\_image",harrisCorners);

waitKey(0);

system("pause");

return 0;

}