

**FACULTY OF INFORMATION TECHNOLOGIES**

**DEPARTMENT OF INFORMATION SYSTEMS MANAGEMENT**

**MACHINE VISION**

*Report*

**Labaratory work #2**

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

**Almaty 2015**

**Source code: main.cpp**

#if !defined COLORDETECT

#define COLORDETECT

#include <opencv2/core/core.hpp>

#include <opencv2/core/mat.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/highgui/highgui.hpp>

using namespace cv;

class ColorDetector {

private:

static const ColorDetector \* instance;

// minimum acceptable distance

int minDist;

// target color

Vec3b target;

// image containing resulting binary map

Mat result;

// image containing color converted image

Mat converted;

// img src

Mat image;

// inline private member function

// Computes the distance from target color.

int getDistance(const cv::Vec3b& color) const {

// calculate Euclidian distance

float s = pow( (float) ( abs(color[0]-target[0])+ abs(color[1]-target[1]) + abs(color[2]-target[2] ) ), 2 );

return sqrt( s );

}

public:

static const ColorDetector& getInstance()

{

static ColorDetector instance;

return instance;

}

// empty constructor

ColorDetector() : minDist(100) {

// default parameter initialization here

target[0]= target[1]= target[2]= 0;

}

ColorDetector(const ColorDetector& root);

ColorDetector& operator=(const ColorDetector&);

// Getters and setters

// Sets the color distance threshold.

// Threshold must be positive, otherwise distance threshold

// is set to 0.

void setColorDistanceThreshold(int distance) {

if (distance<0)

distance=0;

minDist= distance;

}

// Gets the color distance threshold

int getColorDistanceThreshold() const {

return minDist;

}

// Sets the color to be detected

void setTargetColor(unsigned char red, unsigned char green, unsigned char blue) {

cv::Mat tmp(1,1,CV\_8UC3);

tmp.at<cv::Vec3b>(0,0)[0]= blue;

tmp.at<cv::Vec3b>(0,0)[1]= green;

tmp.at<cv::Vec3b>(0,0)[2]= red;

// Converting the target to Lab color space

cvtColor(tmp, tmp, COLOR\_BayerGR2GRAY);

target= tmp.at<cv::Vec3b>(0,0);

}

// Sets the color to be detected

void setTargetColor(cv::Vec3b color) {

cv::Mat tmp(1,1,CV\_8UC3);

tmp.at<cv::Vec3b>(0,0)= color;

// Converting the target to Lab color space

cvtColor(tmp, tmp, COLOR\_BayerGR2GRAY);

target= tmp.at<cv::Vec3b>(0,0);

}

// Gets the color to be detected

Vec3b getTargetColor() const {

return target;

}

// Processes the image. Returns a 1-channel binary image.

Mat process(const cv::Mat &image)

{

result.create(image.rows,image.cols,CV\_8U);

converted.create(image.rows,image.cols,image.type());

cv::cvtColor(image, converted, COLOR\_BayerGR2GRAY);

cv::Mat\_<cv::Vec3b>::iterator it=converted.begin<cv::Vec3b>();

cv::Mat\_<cv::Vec3b>::iterator itend=converted.end<cv::Vec3b>();

cv::Mat\_<uchar>::iterator itout= result.begin<uchar>();

for ( ; it!= itend; ++it, ++itout)

{

if (getDistance(\*it)<minDist)

{

\*itout= 255;

}

else

{

\*itout= 0;

}

}

return result;

}

bool setInputImage(std::string filename) {

image = cv::imread(filename);

if (!image.data)

return false;

else

return true;

}

// Returns the current input image.

const cv::Mat getInputImage() const {

return image;

}

};

#endif

**Source code: 2.cpp**

//

// 2.cpp

// opencv2

//

// Created by Zhansaya on 17.09.15.

// Copyright (c) 2015 Zhansaya. All rights reserved.

//

#include <stdio.h>

#include <opencv2/core/core.hpp>

#include <opencv2/core/mat.hpp>

#include <opencv2/imgproc/imgproc.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <iostream>

#include "main.cpp"

using namespace cv;

using namespace std;

int main()

{

ColorDetector cdetect;

Mat source = imread( "/Users/Zhansaya/Desktop/MachineVision/Unknown.jpg" );

if (!source.data) return 0;

cdetect.setTargetColor(130,190,230);

namedWindow("result");

imshow( "result", cdetect.process( source ) );

waitKey(0);

return 0;

}