**Задание для магистрантов Программной инженерии.**

1. **Характеристика дисциплины** Распознавание и интеллектуальная обработка данных.

**Цель изучения дисциплины:** заключается в обеспечении магистров базовыми знаниями в области современных методов обработки и интеллектуального анализа данных, в том числе сигналов и изображений, с использованием программного и аппаратного обеспечения, характеризующегося цифровым представлением и численными методами обработки данных.

**Формируемые компетенции.**

|  |  |  |
| --- | --- | --- |
| Формируемые компетенции | | Осваиваемые  знания, умения, владения |
| Код | Наименование |
|  | профессиональные компетенции (ПК) по видам профессиональной деятельности | |
|  | Проектная деятельность | |
| ПК-12 | способностью проектировать вспомогательные и специализированные языки программирования и языки представления данных | З математические основы преобразований данных в виде сигналов и изображений |
| У выбирать и разрабатывать эффективные алгоритмы интеллектуального анализа данных, исходя из условий, поставленных в рамках конкретной задачи и доступных вычислительных ресурсов. |
| В методами математического описания алгоритмов преобразования цифровых сигналов и изображений |
| ПК-15 | владением навыками создания программного обеспечения для анализа, распознавания и обработки информации, систем цифровой обработки сигналов | З основные методы и алгоритмы распознавания изображений |
| У проектировать, разрабатывать и отлаживать эффективные алгоритмы и программы распознавания и интеллектуальной обработки данных с использованием современных технологий |
| В практическими навыками реализации методов и алгоритмов распознавания и обработки цифровых сигналов и изображений |

1. **Описание разработанного задания и учебно-методического сопровождения к нему.**

Задание посвящено распознаванию образов, конкретней – распознаванию объектов по цвету, так как цвет является самым распространённым и близким человеку способом выделить объект.

Цель задания состоит в знакомстве с библиотекой OpenCV (**OpenCV** (англ. Open Source Computer Vision Library, библиотека компьютерного зрения с открытым исходным кодом) и предлагаемыми ею возможностями для решения задач распознавания объектов по цвету.

В процессе выполнения задания студенты знакомятся с содержанием, возможностями и особенностями библиотеки OpenCV, работой таких функций как:

* cvSplit: разделение многоканальноно массива на одноканальные массивы или выделение отдельных цветовых плоскостей.
* cvMerge: объединение нескольких одноканальных массивов в один многоканальный массив.
* cvAnd: выполнение по-элементой операции конъюнкции (логического И (AND)) над элементами двух массивов.
* cvMinMaxLoc: определение минимального и максимального значения массива, а так же их местоположения.
* cvMoveWindow: перемещение окна.
* cvResizeWindow: изменение размера окна.

Студенты решают практическую задачу системы машинного зрения.

Формулируется задача следующим образом. Над колесом игорной рулетки установлена камера. Требуется программа распознавания номера, в который попал шарик.

Исходные данные:

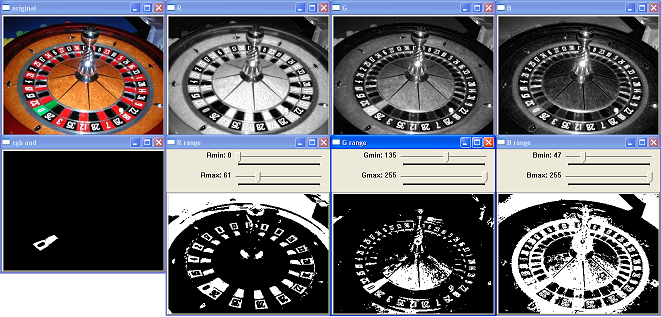
\* шарик лежит во внешнем углу ячейки.  
\* номера на всех колесах располагаются одинаково.  
\* цвет номеров строго чередуется (красное-чёрное)  
\* номер 0 (Zero) зелёного цвета.

Один из возможных алгоритмов решения задачи — это найти зелёный сектор Zero и отсчитать от него сектор в котором детектируется белый шарик.  
Т.о., в первом приближении, задача состоит в нахождении цветных объектов на картинке.



Вариант возможного решения задачи представляет собой программу, которая считывает картинку (в качестве первого параметра), разбивает её на слои (**cvSplit()**), над каждым из которых можно проделать пороговое преобразование (**[cvInRangeS()](http://robocraft.ru/blog/computervision/357.html)**), причём значения интервалов удобным образом изменяются с помощью ползунков.  
  
Результирующее значение выводится в окошке «rgb and» и представляет собой логическое И — **cvAnd()** между получившимися пороговыми картинками.

//  
// позволяет подобрать параметры  
// Rmin, Rmax, Gmin, Gmax, Bmin, Bmax  
// для выделения нужного объекта по цвету  
//  
// robocraft.ru  
//  
  
#include <cv.h>  
#include <highgui.h>  
#include <stdlib.h>  
#include <stdio.h>  
  
IplImage\* image = 0;  
IplImage\* dst = 0;  
  
// для хранения каналов RGB  
IplImage\* rgb = 0;  
IplImage\* r\_plane = 0;  
IplImage\* g\_plane = 0;  
IplImage\* b\_plane = 0;  
// для хранения каналов RGB после преобразования  
IplImage\* r\_range = 0;  
IplImage\* g\_range = 0;  
IplImage\* b\_range = 0;  
// для хранения суммарной картинки  
IplImage\* rgb\_and = 0;  
  
int Rmin = 0;  
int Rmax = 256;  
  
int Gmin = 0;  
int Gmax = 256;  
  
int Bmin = 0;  
int Bmax = 256;  
  
int RGBmax = 256;  
  
//  
// функции-обработчики ползунка  
//  
void myTrackbarRmin(int pos) {  
        Rmin = pos;  
        cvInRangeS(r\_plane, cvScalar(Rmin), cvScalar(Rmax), r\_range);  
}  
  
void myTrackbarRmax(int pos) {  
        Rmax = pos;  
        cvInRangeS(r\_plane, cvScalar(Rmin), cvScalar(Rmax), r\_range);  
}  
  
void myTrackbarGmin(int pos) {  
        Gmin = pos;  
        cvInRangeS(g\_plane, cvScalar(Gmin), cvScalar(Gmax), g\_range);  
}  
  
void myTrackbarGmax(int pos) {  
        Gmax = pos;  
        cvInRangeS(g\_plane, cvScalar(Gmin), cvScalar(Gmax), g\_range);  
}  
  
void myTrackbarBmin(int pos) {  
        Bmin = pos;  
        cvInRangeS(b\_plane, cvScalar(Bmin), cvScalar(Bmax), b\_range);  
}  
  
void myTrackbarBmax(int pos) {  
        Bmax = pos;  
        cvInRangeS(b\_plane, cvScalar(Bmin), cvScalar(Bmax), b\_range);  
}  
  
int main(int argc, char\* argv[])  
{  
        // имя картинки задаётся первым параметром  
        char\* filename = argc == 2 ? argv[1] : "Image0.jpg";  
        // получаем картинку  
        image = cvLoadImage(filename,1);  
  
        printf("[i] image: %s\n", filename);  
        assert( image != 0 );  
  
        // создаём картинки  
        rgb = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 3 );  
        r\_plane = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        g\_plane = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        b\_plane = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        r\_range = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        g\_range = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        b\_range = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        rgb\_and = cvCreateImage( cvGetSize(image), IPL\_DEPTH\_8U, 1 );  
        //  копируем  
        cvCopyImage(image, rgb);  
        // разбиваем на отельные каналы  
        cvSplit( rgb, b\_plane, g\_plane, r\_plane, 0 );  
  
        //  
        // определяем минимальное и максимальное значение  
        // у каналов HSV  
        double framemin=0;  
        double framemax=0;  
  
        cvMinMaxLoc(r\_plane, &framemin, &framemax);  
        printf("[R] %f x %f\n", framemin, framemax );  
        Rmin = framemin;  
        Rmax = framemax;  
        cvMinMaxLoc(g\_plane, &framemin, &framemax);  
        printf("[G] %f x %f\n", framemin, framemax );  
        Gmin = framemin;  
        Gmax = framemax;  
        cvMinMaxLoc(b\_plane, &framemin, &framemax);  
        printf("[B] %f x %f\n", framemin, framemax );  
        Bmin = framemin;  
        Bmax = framemax;  
  
        // окна для отображения картинки  
        cvNamedWindow("original",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("R",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("G",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("B",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("R range",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("G range",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("B range",CV\_WINDOW\_AUTOSIZE);  
        cvNamedWindow("rgb and",CV\_WINDOW\_AUTOSIZE);  
  
        cvCreateTrackbar("Rmin", "R range", &Rmin, RGBmax, myTrackbarRmin);  
        cvCreateTrackbar("Rmax", "R range", &Rmax, RGBmax, myTrackbarRmax);  
        cvCreateTrackbar("Gmin", "G range", &Gmin, RGBmax, myTrackbarGmin);  
        cvCreateTrackbar("Gmax", "G range", &Gmax, RGBmax, myTrackbarGmax);  
        cvCreateTrackbar("Bmin", "B range", &Gmin, RGBmax, myTrackbarBmin);  
        cvCreateTrackbar("Bmax", "B range", &Gmax, RGBmax, myTrackbarBmax);  
  
        //  
        // разместим окна по рабочему столу  
        //  
        if(image->width <1920/4 && image->height<1080/2){  
                cvMoveWindow("original", 0, 0);  
                cvMoveWindow("R", image->width+10, 0);  
                cvMoveWindow("G", (image->width+10)\*2, 0);  
                cvMoveWindow("B", (image->width+10)\*3, 0);  
                cvMoveWindow("rgb and", 0, image->height+30);  
                cvMoveWindow("R range", image->width+10, image->height+30);  
                cvMoveWindow("G range", (image->width+10)\*2, image->height+30);  
                cvMoveWindow("B range", (image->width+10)\*3, image->height+30);  
        }  
  
        while(true){  
  
                // показываем картинку  
                cvShowImage("original",image);  
  
                // показываем слои  
                cvShowImage( "R", r\_plane );  
                cvShowImage( "G", g\_plane );  
                cvShowImage( "B", b\_plane );  
  
                // показываем результат порогового преобразования  
                cvShowImage( "R range", r\_range );  
                cvShowImage( "G range", g\_range );  
                cvShowImage( "B range", b\_range );  
  
                // складываем   
                cvAnd(r\_range, g\_range, rgb\_and);  
                cvAnd(rgb\_and, b\_range, rgb\_and);  
  
                // показываем результат  
                cvShowImage( "rgb and", rgb\_and );  
  
                char c = cvWaitKey(33);  
                if (c == 27) { // если нажата ESC - выходим  
                        break;  
                }  
        }  
        printf("\n[i] Results:\n" );  
        printf("[i][R] %d : %d\n", Rmin, Rmax );  
        printf("[i][G] %d : %d\n", Gmin, Gmax );  
        printf("[i][B] %d : %d\n", Bmin, Bmax );  
  
        // освобождаем ресурсы  
        cvReleaseImage(& image);  
        cvReleaseImage(&rgb);  
        cvReleaseImage(&r\_plane);  
        cvReleaseImage(&g\_plane);  
        cvReleaseImage(&b\_plane);  
        cvReleaseImage(&r\_range);  
        cvReleaseImage(&g\_range);  
        cvReleaseImage(&b\_range);  
        cvReleaseImage(&rgb\_and);  
        // удаляем окна  
        cvDestroyAllWindows();  
        return 0;  
}



Итак, сектор Zero выделяется хорошо, но попытка выделить по цвету шарик не даст ожидаемого результата. Разумеется, можно добиться, чтобы шарик был виден, но так же будут видны белые числа номеров, а ещё блики (на оси рулетки и металлической ленте).

Эти ложные срабатывания можно устранить, если работать только с заданной частью изображения, где их нет. Т.е. сначала мы ждём, когда в заданном секторе будет задетектировано Zero, а далее мы отсчитываем сектора рулетки, до тех пор, пока не будет задетектирован шарик.

После выполнения задания студентам необходимо проанализировать результаты работы программы, дать её краткую характеристику по производительности и объёму требуемой памяти.

Материалы, используемые при выполнении задания:

* документация библиотеки OpenCV.
* Материалы сайта <http://robocraft.ru/>, посвящённые библиотеке OpenCV.

Инструментальное средство реализации задания выбирается студентом самостоятельно, так как магистранты уже имеют достаточный опыт в разработке ПО, чтобы выработать собственные предпочтения. Как вариант: Microsoft Visual Studio.

Методические рекомендации. Разбить задачу на следующие:

* считывание картинки,
* разбиение её на слои (**cvSplit()**),
* над каждым слоем произвести пороговое преобразование (**[cvInRangeS()](http://robocraft.ru/blog/computervision/357.html)**), значения интервалов изменять с помощью ползунков.
* результирующее значение представляет собой логическое И — **cvAnd()** между получившимися пороговыми картинками.

1. **Особенности организации и руководства деятельностью студентов по выполнению задания и оценке его результатов.**

Перед выдачей задания необходимо провести краткий обзор библиотеки OpenCV, рассказать об основных её возможностях по распознаванию цвета, а также в общих чертах описать идею возможного решения задачи и основные его шаги. В ходе выполнения студентами задания лишь отвечать на неизбежно возникающие вопросы, не рассказывая целиком и полностью, что и как им делать. Иными словами, дать им возможность самостоятельного поиска решения.

В ходе оценки учитывать качество выполненного задания и степень самостоятельности принятых решений, а также креативные идеи и решения, предлагаемые студентами, то есть степень оригинальности выполнения задачи. Следует также учитывать культуру программирования, рациональность принятых решений. Точной количественной методики оценки не предлагается, так как решения могут обладать большой степенью индивидуальности. Задание считается выполненным, если написана программа, в какой-то степени реализующая любой работающий метод решения задачи.

1. **Доказательство целесообразности задания и соответствия формируемым профессиональным компетенциям.**

В рамках формирования профессиональных компетенций задание способствует выработке:

* умения выбирать и разрабатывать эффективные алгоритмы интеллектуального анализа данных, исходя из условий, поставленных в рамках конкретной задачи и доступных вычислительных ресурсов;
* умения проектировать, разрабатывать и отлаживать эффективные алгоритмы и программы распознавания и интеллектуальной обработки данных с использованием современных технологий;
* владения практическими навыками реализации методов и алгоритмов распознавания и обработки цифровых сигналов и изображений.