# Работа программы

При старте в командной строке необходимо указать название видеофайла, с которым будет работать приложение. Например, имя файла filename.avi. Исходя из имени файла приложение считает, что конфигурационный файл будет называться filename.cfg. Для одного видеофайла нет смысла создавать несколько конфигурационных файлов, поэтому на данном этапе в приложении отсутствует возможность указать имя файла конфигурации. Кроме того, должна быть установлена переменная окружения TESSDATA\_PREFIX. Она должна указывать на каталог, где хранятся актуальные файлы необходимые библиотеке tesseract для распознавания русских символов.

Приложение не запустится и выдаст ошибку в случаях:

-не указано имя видеофайла;

-видеофайл отсутствует, поврежден;

-отсутствует видеокодек, поддерживающий указанный видеофайл;

-отсутствует или поврежден файл конфигурации;

-ошибки в файле конфигурации;

-отсутствует переменная окружения TESSDATA\_PREFIX

-отсутствют файлы необходимые библиотеке tesseract.

В остальных случаях приложение запустится и создаст окно, пример которого указан на рис.1. Заголовок окна задается в переменной WindowName и добавляет в окно TaskBar, для того, чтобы можно было регулировать параметр fix\_threshold для лучшей бинаризации.

Далее запускается бесконечный цикл, в котором из видеопотока берутся кадры. Все кадры инвертируются и бинаризируются командой

cv::threshold(frame, dst, double(fixed\_threshold), 255, inv);

По нажатию клавиши «пробел» текущий кадр распознается функцией recog\_Mat(api, params, dst, samples);

Из изображения вырезаются фрагменты, которые были распознаны, и добавляются в map в соответствие распознанным символам.

По нажатию клавиши «ESC» или по завершении Видео-файла приложение завершается и записывает все изображения в файлы.



Рис.1. Пример окна приложения

# Конфигурационный файл

//Коментарий

#Коментарий

num\_x = 31

num\_y = 15

delta\_x = 21.5

delta\_y = 33.0

start\_x = 28

start\_y = 23

1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 1 0 0 1 1 1 0

1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1

1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1

1 0 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 1 1 1 1

1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0

0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0

0 0 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0

Рис.2. Пример конфигурационного файла

На рисунке 2 приведен пример конфигурационного файла. Все строки, которые начинаются ни с цифры и не с буквы считаются комментариями и игнорируются.

Все строки, которые начинаются с 0 или 1 считаются строками, описывающими шаблон. То есть места на экране, где могут встречаться символы. Символ «1» - на этом месте может появляться символ. «0» - символ не может встретиться. Остальные конфигурационные параметры описаны в разделе «Структура parameters».

# Структура «parameters»

struct parameters {

int num\_x = 0; // Кол-во знакомест по горизонтали

int num\_y = 0; // Кол-во знакомест по вертикали

double delta\_x = 21.5; // Ширина одного знакоместа

double delta\_y = 33.0; // Длина одного знакоместа

int start\_x = 28; // Координата x точки начала сетки знакомест

int start\_y = 23; // Координата y точки начала сетки знакомест

vector<vector<bool>> positions;

};

# Описание функций

void clear\_spaces(string& s)

Удаляет из переданной строки все пробельные символы.

bool parse\_config(parameters& params, string filename)

Заполняет переданную структуру param содержанием конфигурационного файла filename. Причем в качестве filename можно передавать название видеофайла. Функция сама подставит расширение cfg. Функция выбрасывает runtime\_error, поэтому при вызове ее необходимо оборачивать в try/catch блок.

void draw\_greed(parameters params, cv::Mat frame,

const cv::Scalar Color)

Рисует сетку знакомест на переданном изображении frame цветом, заданным переменной Color.

pair<char,char> UTF2OEM(char\* src = NULL)

принимает на вход указатель на символ в кодировке UTF8 и возвращает пару символов – first в кодировке ansi, second в кодировке cp1251. Первый символ необходим для отображения на экране. Второй символ для формирования имени файла.

pair<char,char> recog\_char(

tesseract::TessBaseAPI\* api,

parameters params,

cv::Mat dst,

int x,

int y)

Функция принимает на вход изображение, ссылку на экземпляр класса TessBaseAPI , структуру с конфигурацией видеофайла и столбец и строку с символом, который необходимо распознать. Функция возвращает возвращает пару символов – first в кодировке ansi, second в кодировке cp1251.

void dump\_images(map<char, cv::Mat> samples,string extention)

Функция принимает на вход map<char, cv::Mat>. Где ключом является распознаваемый символ а значением массив склеенных изображений, на которых были распознаны данные символы.

int recog\_Mat(tesseract::TessBaseAPI\* api,

parameters params,

cv::Mat dst,

map<char, cv::Mat> samples);

Функция добавляет распознанные символы в соответствующие этим символам изображения. Возвращает количество распознанных символов.