Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатика

Спеціальність 122 Комп’ютерні науки

Освітньо-професійна програма Інформатика

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи з дисципліни “ Об’єктно орієнтоване програмування ”

за темою

Розробка системи підрахунку рухомих автомобілів на відеозаписі

Виконав: Перевірила:

студент гр. IТIНФ-20-3 Кириченко I.Ю.

Сафонов О.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2021 р.

Харківський національний університет радіоелектроніки

Кафедра Інформатика

(повна назва)

Дисципліна Об’єктно орієнтоване програмування

(повна назва)

Спеціальність 122 Комп’ютерні науки

Освітньо-професійна програма Інформатика

(повна назва)

Курс I Група IТIНФ-20-3 Семестр 2 .

ЗАВДАННЯ

на курсовий проект (роботу) студента

Сафонов Олег Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) Розробка системи підрахунку рухомих автомобілів на

відеозаписі

2. Строк здачі студентом закінченого проекту (роботи) 15.06.2021 .

3. Вихідні данні проекту (роботи) *1. Відомості про предметну область*

*2. Методичні вказівки до курсового проектування*

*3 . Перелік використовуваних програмних засобів:*

*3.1 IDE MS Visual Studio 2017*

*3.2 Файли бібліотеки OpenCV*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

*1. Аналіз галузі обробки зображень та комп’ютерного зору.*

*2. Математична модель*

*3. Комп’ютерна модель*

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових   
креслень)

Додаток A – код програми

UML-діаграми

6. Дата видачі завдання 03.02.2021

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування етапів курсового проекту | Термін виконання етапів проекту | Примітки |
| 1 | Пошук необхідної літератури | 20.02.21 – 25.02.21 | Виконано |
| 2 | Пошук необхідного програмного забезпечення | 25.02.21 – 26.02.21 | Виконано |
| 3 | Ознайомлення з можливостями візуального програмування | 01.03.21 – 15.03.21 | Виконано |
| 4 | Розроблення алгоритму для вирішення поставленої задачі | 22.03.21 – 01.04.21 | Виконано |
| 5 | Створення комп’ютерної моделі по розробленому алгоритму | 01.05.21 – 10.05.21 | Виконано |
| 6 | Оформлення пояснювальної записки | 01.06.21 – 06.06.21 | Виконано |
| 7 | Захист курсового проекту | 15.06.21 |  |

Студент\_\_*Сафонов О.О*.\_\_\_\_ Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (підпис)

РЕФЕРАТ

Пояснювальна Записка: Обсяг 22 стр., 1 додаток, 4 джерела.

Мета курсової роботи полягала у вивчені та закріпленні на практиці принципів об’єктно-орієнтованого програмування. Також до мети курсової роботи входило самостійне ознайомлення з бібліотекою OpenCV та її функціоналом.

Курсова робота присвячена розробці системи розпізнавання рухомих об’єктів на відеозаписі із застосуванням мови програмування C++ у середовищі Microsoft Visual Studio.

Предмет дослiдження – методології розробки програм, такі як об’єктно-орієнтованого програмування.

Дана робота була проведена шляхом вивчення теоретичного матеріалу стосовно теми, розробки алгоритму, та реалізації у вигляді програми.

C++, OPECV, КОМП’ЮТЕРНИЙ ЗIР , ОБРОБКА ВIДЕО, РОЗПІЗНАВАННЯ ОБ’ЄКТIВ, ВИДАЛЕННЯ ФОНУ, ФIЛЬТРАЦIЯ ЗОБРАЖЕНЬ, ОБРОБКА БАГАТОКАНАЛЬНИХ МАТРИЦЬ, МАШИННЕ НАВЧАННЯ.

ЗМІСТ

Введення 6

1. Аналiз галузi обробки зображень та Комп’ютерного зору 7

1.1 Основнi поняття в галузi комп’ютерного зору та обробки вiдеофайлiв 7

1.2 Сфери застосування технологiй комп’ютерного зору 8

1.3 Постановка Задачi 9

2. Математична модель системи пiдрахунку об’єктiв 10

2.1 Алгоритм програми 10

2.2 UML–Дiаграми 12

3. Комп’ютерна модель системи 14

3.1 Обґрунтування вибору середовища програмної реалізації 14

3.2 Опис Iнтерфейсу програми 15

3.3 Iнструкцiя користувача 15

3.4 Опис основних алгоритмiв та методiв програми 16

Висновки 17

Перелік посилань 18

Додаток А 19

## Введення

У вiк цифрових технологiй та штучного інтелекту люди часто зустрiчаються з проблемами, якi можна вирiшити автоматизацiею, це допомагає виконати потрiбнi дії не витрачаючи багато часу або зусилль. Однiєю з таких проблем можна назвати потребу людей автоматично вiдстежувати рухливi об’єкти та класифiкувати їх. Розробки останнiх 40 рокiв дали результат у виглядi продвинутих систем спостереження, якi покращили умови безпеки та вiдкрили новi можливостi у зборi важливоi статистичноi iнформацiї для аналiзу. Цю задачу можна легко вирiшити використовуючи технологiї штучного iнтелекту у поєднання з алгоритмами комп’ютерного зору.

Сьогоднi iснує безлiч бiблiотек, фреймворкiв та систем, пов’язаних з комп’ютерним зором. Гарним прикладом такоi бiблiотеки будуть бiблiотеки openCV, simpleCV, Matlab, libfacedetection та багато iнших.

# Аналiз галузi обробки зображень та Комп’ютерного зору

### Основнi поняття в галузi комп’ютерного зору та обробки вiдеофайлiв

Комп'ютерний зір - це науковий напрям в області штучного інтелекту і пов'язані з ним технології отримання зображень об'єктів реального світу, їх обробки і використання отриманих даних для вирішення різного роду прикладних задач без участі (повного або часткового) людини.

Обробка зображень – будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад, фотографіями або відеокадри. Обробка зображень може здійснюватися як для отримання зображення на виході (наприклад, підготовка до поліграфічного тиражування, до телетрансляції і т.п.)

Основне завдання напряму розпізнавання образів – одержання семантичного опису зображених об'єктів. Мета розпізнавання може бути різна: як виділення окремих елементів на зображенні, так і класифікація зображення в цілому. У якомусь сенсі завдання розпізнавання є зворотним стосовно завдання візуалізації. Області застосування – системи розпізнавання текстів, створення тривимірних моделей людини по фотографіях і т.п. Зазвичай розпiзнавання образiв потребує використання алгоритмiв машинного навчання.

Трекінг - це робота алгоритму обробки для пошуку в руху в кадрi, визначення і класифікацiя об'єка, що рухається, опис його характеристик (розмір, колір, швидкість). Варіацій трекінгу (відеодетектора) може бути досить багато [[1](#_Перелік_посилань)].

Матриця – це двувимiрний масив заповнений значеннями, вiдповiдно до положення пiкселiв на зображеннi.Матрицi бувають одноканальнi(чорно-бiлi) або трьохканальнi(RGB або BGR матрицi), де кiлькiсть каналiв це кiлькiсть кольорiв, що складають кожен її елемент [[1](#_Перелік_посилань)].

Машинне навчання - це програма штучного інтелекту (ШІ), що надає системам можливість автоматичного навчання та вдосконалення на основі досвіду без явного програмування. Машинне навчання фокусується на розробці комп’ютерних програм, які можуть отримати доступ до даних та використовувати їх для навчання самій. Машинне навчання та комп'ютерний зір - це дві галузі, які тісно пов’язані між собою. Машинне навчання покращило комп’ютерний зiр щодо розпізнавання та відстеження. Воно пропонує ефективні методи отримання, обробки зображень та фокусування об'єкта, які використовуються в комп'ютерному зорі. У свою чергу, комп’ютерний зір розширив сферу машинного навчання. Він включає в себе цифрове зображення або відео, пристрій зондування, пристрій інтерпретації та етап інтерпретації. Машинне навчання використовується в комп'ютерному зорі на етапі перекладу та на етапі інтерпретації.

### Сфери застосування технологiй комп’ютерного зору

1. Автоматичне розпізнавання обличчя та інтерпретація його виразу
2. Візуальне керівництво автономними транспортними засобами
3. Автоматизований аналіз, інтерпретація та діагностика медичних зображень
4. Робототехнічне виробництво: маніпуляції, класифікація та складання деталей
5. Оптичне розпізнавання символів і цифр, таких як поштові індекси або номерні знаки.
6. OCR: розпізнавання друкованих або рукописних символів та слів
7. Сільськогосподарські роботи: візуальне сортування та збирання врожаю
8. Розумні офіси: відстеження людей та предметів; розуміння жестів
9. Біометрична візуальна ідентифікація осіб
10. Візуально наділені роботизовані помічники
11. Моніторинг та попередження безпеки; виявлення аномалії
12. Інтелектуальні “розумнi” протези для сліпих
13. Віртуальна реальність: дозволяє нам знати позицію користувача та позиції всіх об’єктів навколо.
14. Відстеження рухомих об'єктів; уникнення зіткнень;

### Постановка Задачi

Необхiдно реалiзувати систему знаходження та пiдрахунку рухомих об’єктiв на вiдеозаписi. Для виконання цього плану нам потрiбно:

1. Пiдiбрати вхiднi матерiали (Вiдеозапис та тип об’єктiв, якi ми будемо шукати).
2. Створити методи, якi дозволили би нам зчитати вiдеофайл з диску, вiдкрити його та вивести на екран.
3. Створити цикл, який нам дозволить аналiзувати кожен кадр вiдео як окреме зображення.
4. Використовуючи алгоритми та об’єкти видалення фону та видiлення переднього плану створити “маску” iз переднiм планом вiдео.
5. Проаналiзувавши оброблену карту переднього плану, використати методи пошуку контурiв на зображеннi, та занести їх в контейнер з урахуванням ієрархії контурiв.
6. Реалiзувати метод, який би видiляв контури, знайденi на попередньому кроцi i створював “рамки” на вихiдному вiдео.
7. Розробити метод, який би пiдраховував кiлькiсть об’єктiв, перетинаючих тонку лiнiю посерединi дороги.
8. Зробити лiчильники на iнтерфейci програми, якi будуть виводити потрiбнi нам данi.
9. Реалiзувати консольний iтнерфейс для вибору вiдеофайла.

# Математична Модель системи пiдрахунку об’єктiв

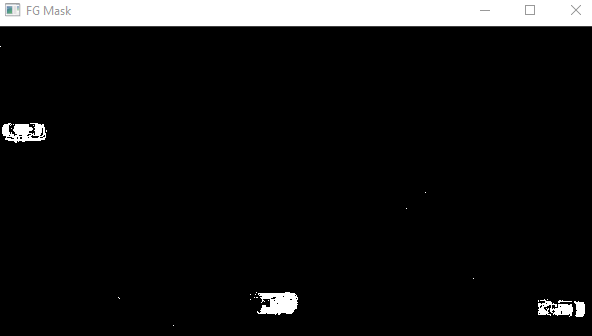
1. Алгоритм програми

Спочатку програма зчитує вiдеофайл з диска та розбивае його на кадри, якi вона буде використовувати як окремi зображення для подальшого аналiзу (малюнок 2.1.1).

Перед початком обробки зображень(кадрiв) потрiбно привести їх до зручного для алгоритмiв openCV вигляду. Для цього у програмi створюється об’єкт, який видаляе фон зображення, залишаючи лише “переднiй план.” Отримане зображення проходить подальшу обробку за допомогою операторiв фiльтрацiї, прибираючи зайвий “шум” у кадрi (малюнок 2.1.2).



Малюнок 2.1.1 – Оригiнальний кадр



Малюнок 2.1.2 – кадр пiсля видалення фону

Далi, аналiзуючи кадр за кадром програма викликає методи пошуку контурiв на обробленному кадрi, та знаходить серед цих контурiв лише тi, якi за розмiром перевищують 300 пікселів, та виділяє їх прямокутними рамками.

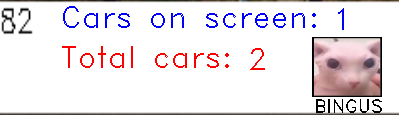
Враховуючи позицiю об’єкта в ієрархії контурiв, кожному об’єкту присвоюється порядковий номер, який разом iз рамкою видiляє кожен об’єкт (малюнок 2.1.3).



Малюнок 2.1.3 – приклад видiленого об’єкта

Основним елементом програми є тонка лiнiя посерединi дороги, яка рахуе кожен об’єкт, що її перетне, та збiльшуе загальну кiлькiсть автомобiлiв на один. Ця лiнiя реалiзується через створення двох невидимих границь, якi шукають об’єкти, чий номер опинився мiж ними.

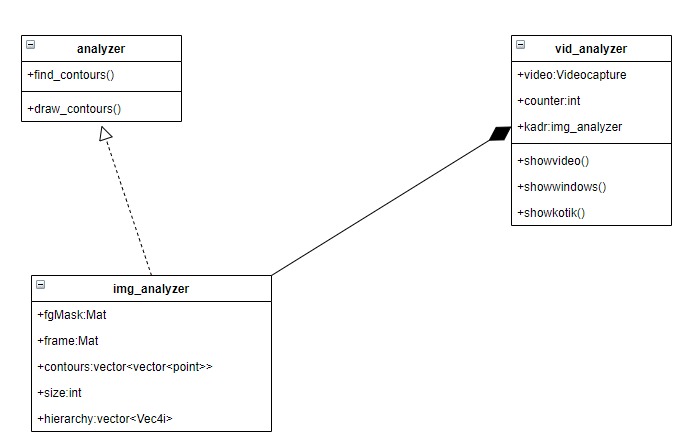
Наприкiнцi роботи програми викликаються алгоритми бiблiотеки, вiдповiдальнi за додавання геометричних об’єктiв у кадр, та створюється iнтерфейс програми(лiчильник кадрiв, лiчильник об’єктiв на екранi, зображення з файлу i т.п.) (малюнок 2.1.4) Пiсля додавання iнтерфейсу на вихiдне вiдео, програма змiнюе розмiр вiкна, та виводить результати на екран.



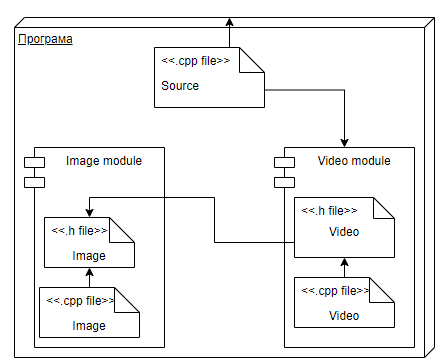
Малюнок 2.1.4 – вiкно iнтерфейсу програми

1. UML–Діаграми

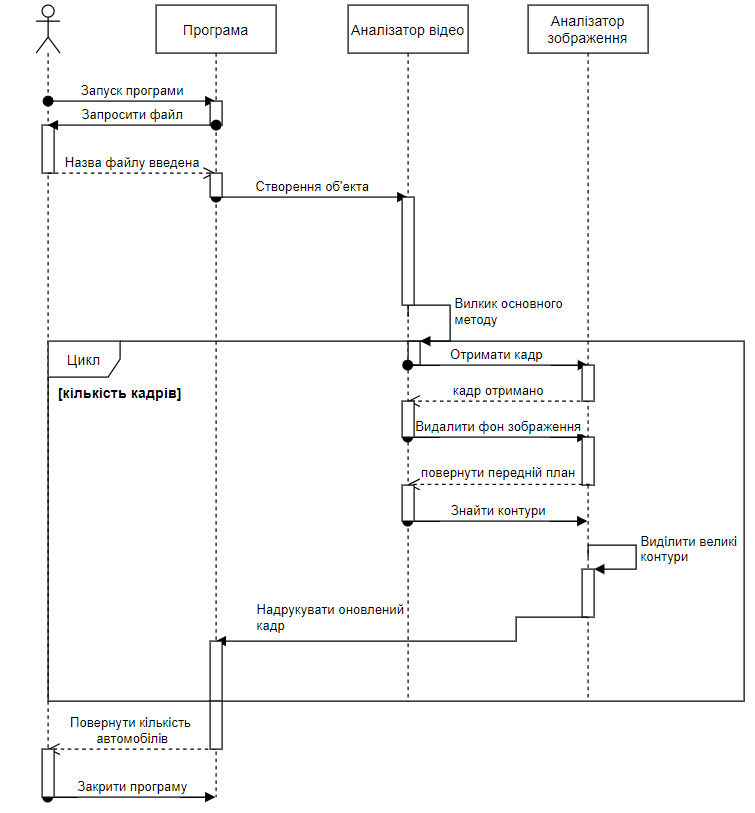
UML - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення [3].



Малюнок 2.2.1 – Діаграма класів



Малюнок 2.2.2 – Діаграма розгортання



Малюнок 2.2.3 – Діаграма послідовності

# Комп’ютерна модель Системи

### Обґрунтування вибору середовища програмної реалізації

Програма реалізована на платформі Microsoft Windows iз використанням IDE Microsoft Visual Studio 2017, мови С++ та бiблiотеки алгоритмiв комп’ютерного зору openCV.

Мова С++ - процедурний та об’єкто орiентований iнструмент розробки сильнонавантаженних програм з широким вибором високоякiсних бiблiотек для вирiщення рiзних задач.[[2](#_Перелік_посилань)]. Обрана мова програмування дозволяє напряму використовувати методи з бiблiотеки openCV не потберуючи додаткових iнтерфейсiв та операторiв доступу. Також мова С++ є низькорiвневою мовою, що вiдкриває великий простiр для оптимiзацiї та розширення коду й функцiоналу програми.

Середа розробки була обрана за її легкiсть, простий iнтерфейс та можливiсть розширювати функцiонал за допомогою додаткiв(Resharper C++,Github…), що суттєво спрощує процес розробки програм. Також Visual Studio дозволяє вказувати залежностi вiд стороннiх бiблiотек компiлятору, завдяки чому можна швидко встановлювати новi модулi та одразу починати ними користуватись.

Для реалiзацiї функцiоналу програми була обрана бiблiотека комп’ютерного зору openCV, розроблена компанiєю Itseez, яка є лiдуючою технологiєю у галузi комп’ютерного зору для академiчних та коммерцiйних цiлей. Код бiблiотеки знаходиться у вiдкритому доступi, що дозволяє збирати новi пакети за допомогою Cmake, а завдяки аппаратному прискоренню вiд Intel, програми виходять швидкодiйними та потребують небагато ресурсiв комп’ютера. Бiблiотека включае в себе безлiч алгоритмiв для обробки та аналiзу зображень, та для роботи з простою графiкою.

### Опис Iнтерфейсу програми

Iнтерфейс програми складаеться з вiкна з основною iнформацiею та з обробленого вiдеозапису (малюнок 3.2.1).



Малюнок 3.2.1 – Iнтерфейс програми

### Iнструкцiя користувача

Для початку роботи нашоi програми, потрiбно запустити .exe файл у консольному iнтерфейсi (малюнок 3.3.1). та передати йому в якостi параметра назву файлу або шлях до файлу у виглядi строки (малюнок 3.3.2).

Якщо файл було знайдено, програма почне виконувати основний алгоритм.



Малюнок 3.3.1 – Запуск програми через командну строку



Малюнок 3.3.2 – передача вхiдного параметра

Для керування програмою передбачено двi команди: вийти з програми та поставити/зняти паузу.

Щоб вийти з програми користувачу потрiбно натиснути клавiшу ‘q’, або ‘escape’.

Для того, щоб поставити поток кадрiв на паузу потрiбно натиснути клавiшу “space”, щоб зняти паузу, потрiбно натиснути ту ж саму клавiшу повторно.

### Опис основних алгоритмiв та методiв програми

1. Клас img\_analyzer мiстить поле классу Mat, який являє собою багатоканальну(за замовчуванням трьохканальну) матрицю та всi методи, для обробки пiкселiв цiєi матрицi [[4](#_Перелік_посилань)].
2. Метод find\_contours(): шукае окремi контури на чорно-бiлому зображеннi, та заносить їх у контейнер зi збереженням iєрархiї контурiв.
3. Метод draw\_contours(vector<vector<Point>>): малюе контури та порядковi номери навколо об’єктiв на екранi, та повертае кiлькiсть контурiв, що перетнули центральну лiнiю
4. Метод showimage(): зчитує з диску зображення, зменуше його, та заносить до трьохканальноi матрицi. Використовуючи цикл, переносить зображення з матрицi на кожний кадр вiдео.
5. Метод showwindows(): виводить на екран у двох окремих вiкнах карту переднього плану зображення та вихiднi кадри з контурами та лiчильниками.
6. Showvideo() – головний метод програми, який реалiзує ранiше описаний алгоритм за допомогою базових методiв та повертае загальну кiлькiсть автомобiлiв, яку програмi вдалось знайти.

# Висновки

У рамках даної курсової роботи була створена система розпiзнавання на пiдрахунку рухомих об’єктiв на вiдеозаписi. Було проведено дослiдження компонентiв середовища розробки Microsoft Visual Studio 2017 та бiблiотеки комп’ютерного зору openCV, якi були використанi у процесi розробки програми. В результатi ориманий продукт мав наступнi якостi:

Недолiки програми:

1. Потрiбне додаткове налаштування для роботи з новим файлом
2. Неможливiсть роботи програми з вiдео, знятими на нестатичну камеру
3. Вiдсутнiсть можливостi класифiкувати знайденi об’єкти за рiзними групами

Переваги програми:

1. Простий та легко розширюємий код
2. Iснує великий простiр для оптимiзацiї
3. Можливiсть додавання елементiв машинного навчання для подальшоi iдентифiкацiї об’єктiв на екранi

# Перелік посилань

1. Келер, А. Бредскi, Г. Вивчаемо OpenCV 3 – O’reilly, 2017. – 823с.
2. Страуструп, Б. Програмування: Принципи та практика використання C++. – Litres, 2021. – 1213с.
3. Градi Буч, Мова UML. Посібник користувача. – M.:ДМК Пресс, 2006 – 496c.
4. [Електронний ресурс] Документацiя бiблiотеки OpenCV –   
   URL: <https://docs.opencv.org/master/index.html> (дата звернення 01.05.2021).

# Додаток А

Вихідний текст програми

Файл Source.cpp

#include "Video.h"

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

{

string input;

cout << "Input file name: ";

cin >> input;

vid\_analyzer Program(input); // инициализация обьекта с помощью введённого пути

int i = Program.showvideo();

cout<<"Total amount of cars that crossed the line: "<<i;

return 0;

}

Файл Image.h

#ifndef IMAGE\_H

#define IMAGE\_H

#include <iostream>

#include <opencv2/imgcodecs.hpp>

#include <opencv2/imgproc.hpp>

#include <opencv2/videoio.hpp>

#include <opencv2/highgui.hpp>

#include <opencv2/video.hpp>

using namespace cv;

using namespace std;

class analyzer //базовый класс для анализа данных

{

protected:

virtual void find\_contours() {}

virtual int draw\_contours(vector<vector<Point>> conturi) { return 0; }

};

class img\_analyzer :public analyzer // наследник, анализирующий изображение

{

public:

img\_analyzer() {}

img\_analyzer(Mat img) { this->frame = img; }

Mat frame, fgMask; // матрицы для хранения текущего кадра, и кадра с удалённым фоном

vector<vector<Point>> contours; // вектор контуров, которые подлежат выделению

int size = 0;

vector<Vec4i> hierarchy; // иерархия контуров

void find\_contours()override;

int draw\_contours(vector<vector<Point>> conturi) override; //переопределение метода базового класса для выделения контуров на изображении

};

#endif

Файл Image.cpp

#include "Image.h"

void img\_analyzer::find\_contours()

{

findContours(fgMask, contours, RETR\_TREE, CHAIN\_APPROX\_SIMPLE);// вызов алгоритма для поиска контуров на переднем плане

}

int img\_analyzer::draw\_contours(vector<vector<Point>> conturi)

{

{

vector<Rect>boxes;

size = 0;

int local\_counter = 0;

for (auto cnt : contours)// проходимся по массиву контуров, и выделяем все большие обьектыв кадре

{

auto area = contourArea(cnt);

if (area > 300)

{

Rect box = boundingRect(cnt);//создание прямоугольника вокруг контура

size++;

boxes.push\_back(box);

rectangle(frame, box.tl(), box.br(), Scalar(212, 255, 127), 1, LINE\_4, 0);

double numX = (box.x + box.x + box.width) / 2; //определяем центр

double numY = (box.y + box.y + box.height) / 2;

putText(frame, to\_string(size), Point(numX, numY + 2), FONT\_HERSHEY\_DUPLEX, 0.5, Scalar(0, 0, 255));

if (numX > 300 && numX < 307) local\_counter++; // если обьект находится в окресностях прямой, он её пересёк и мы его считаем

}

}

return local\_counter;//возвращение счетчика, содержащего кол-во проехавших машин

}

}

Файл Video.h

#ifndef VIDEO\_H

#define VIDEO\_H

#include "Image.h"

using namespace cv;

using namespace std;

class vid\_analyzer // основной класс программы, ответственный за обработку видео

{

public:

vid\_analyzer(string str);

int showvideo(); // публичный метод, служащий для вызова приватных методов

private:

VideoCapture video{}; //класс видеопотока, из которого мы извлекаем поочередно кадры

int total\_counter;

img\_analyzer kadr; // объект класса img\_analyzer, нужный для обработки отдельных кадров

void showwindows(); // вывод окон с изображениями на экран

void showimage(); // метод, переносящий одно изображение на другое

};Файл Video.cpp

#include "Video.h"

vid\_analyzer::vid\_analyzer(string str)

{

VideoCapture Wideo{ str };

this->video = Wideo;

}

void vid\_analyzer::showwindows()

{

// вывод интерфейса программы и всех счётчиков

line(kadr.frame, Point(310, 0), Point(310, 550), Scalar(200, 150, 0), 1);

rectangle(kadr.frame, cv::Point(10, 2), Point(250, 50),

Scalar(255, 255, 255), -1);

stringstream ss;

ss << video.get(CAP\_PROP\_POS\_FRAMES); //счетчик кадров

string frameNumberString = ss.str();

putText(kadr.frame, frameNumberString, Point(10, 15),

FONT\_HERSHEY\_SIMPLEX, 0.5, Scalar(0, 0, 0));

resize(kadr.frame, kadr.frame, Size(1000, 720), 0, 0, INTER\_LINEAR\_EXACT);

this->showimage();

putText(kadr.frame, "Cars on screen: ", Point(78, 30), FONT\_HERSHEY\_DUPLEX, 1, Scalar(255, 0, 0));

putText(kadr.frame, to\_string(kadr.size), Point(350, 32), FONT\_HERSHEY\_DUPLEX, 1, Scalar(255, 0, 0));

putText(kadr.frame, "Total cars: ", Point(78, 70), FONT\_HERSHEY\_DUPLEX, 1, Scalar(0, 0, 255));

putText(kadr.frame, to\_string(total\_counter), Point(265, 72), FONT\_HERSHEY\_DUPLEX, 1, Scalar(0, 0, 255));

putText(kadr.frame,"q - exit program space - pause", Point(20, 110), FONT\_HERSHEY\_DUPLEX, 0.5, Scalar(0, 0, 0));

// создание окон с двумя изображениями

imshow("Frame", kadr.frame);

imshow("FG Mask", kadr.fgMask);

}

int vid\_analyzer::showvideo()

{

Ptr<BackgroundSubtractor> pBackSub;

pBackSub = createBackgroundSubtractorMOG2(100, 200);// инициализация указателя на объект, удаляющий статический фон

while (true) {// цикл нужный для обработки каждого кадра по-отдельности

video >> kadr.frame; // заносим новый кадр в матрицу подкласса

if (kadr.frame.empty())

break;

pBackSub->apply(kadr.frame, kadr.fgMask);

threshold(kadr.fgMask, kadr.fgMask, 0, 255, THRESH\_BINARY);// применение алгоритма вычитания фона

kadr.find\_contours();

total\_counter += kadr.draw\_contours(kadr.contours);// прибавляем кол-во обьектов, пересекающих линию к общему числу

this->showwindows(); // выводим окна

int keyboard = waitKey(30);

if (keyboard == 'q' || keyboard == 27) {// выход из программы при нажатии q

break;

}else if (keyboard == 'spc' || keyboard == 32) {// пауза при нажатии на space

while (true)

{

keyboard = waitKey();

if (keyboard == 'spc' || keyboard == 32)break;

}

}

}

return total\_counter;

}

void vid\_analyzer::showimage()

{

Mat kotik = imread("bingus.png"); // открытие изображения с диска

string kotik\_name = "BINGUS";

resize(kotik, kotik, Size(70, 60), 0, 0, INTER\_LINEAR\_EXACT); // масштабируем изображение

for (int i = 0; i < kotik.rows; i++)

{

for (int j = 0; j < kotik.cols; j++)

{

kadr.frame.at<Vec3b>(i + 40, j + 330) = kotik.at<Vec3b>(i, j);// замена области кадра на нужное изображение по пикселю за раз

}

}

putText(kadr.frame, kotik\_name, Point(332, 115), FONT\_HERSHEY\_DUPLEX, 0.6, Scalar(0, 0, 0));

}