**АНОТАЦІЯ**

Бакалаврська кваліфікаційна робота була спрямована на дослідження принципів роботи машинного зору за допомогою бібліотеки OpenCV та інтеграцію їх в спеціалізовану систему моніторингу трафіку автомобілів. Було спроектовано архітектуру спеціалізованої системи, проаналізовані алгоритми обробки відео потоку. Дане програмне забезпечення розроблене із метою дослідження інтенсивності руху автомобілів в містах та виявлення найбільш завантажених ділянок дороги. Для реалізації поставленої задачі було вирішено обрати технологію .NET Core, яка підтримує крос-платформну розробку і має в наявності усі інструменти для реалізації клієнт-серверної архітектури, для зберігання даних обрано систему керування базами даних Microsoft SQL Server.

**ANNOTATION**

Bachelor qualification work was aimed at studying the principles of the operation of the machine unit with the help of the OpenCV library and integrating them into a specialized system for monitoring traffic of cars. The architecture of the specialized system was designed, video stream processing algorithms analyzed. Offer help with engineering vehicle inspection in cities and discover large loaded sections of the road. To accomplish its tasks, it was decided to develop .NET Core technology that supports the development platform and has all the tools for implementing client-server architecture for storing data in the form of the Microsoft SQL Server database system.

**ЗМІСТ**

[**ВСТУП** 3](#_Toc9785391)

[1. АНАЛІЗ СЕЦІАЛІЗОВАНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ З ВІДЕО ПОТОКУ 5](#_Toc9785392)

[1.1. Огляд існуючих систем для розпізнавання об’єктів, транспортних засобів 5](#_Toc9785393)

[1.2. Загальні етапи вирішення задачі розпізнавання об’єктів з відео потоку 6](#_Toc9785394)

[1.3. Формування бінарних зображень з багатоканальних 7](#_Toc9785395)

[1.3.1. Граничне перетворення 7](#_Toc9785396)

[1.3.2. Виділення об'єкта за кольором 8](#_Toc9785397)

[1.4. Контурний аналіз 9](#_Toc9785398)

[1.5. ROI (Region Of Interest) 13](#_Toc9785399)

[1.6. Класифікатор Хаара 14](#_Toc9785400)

[2. ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ З ВІДЕО ПОТОКУ 17](#_Toc9785401)

[2.1. Вибір технологій розробки програмного забезпеченн 17](#_Toc9785402)

[2.2. Вибір середовища розробки 18](#_Toc9785403)

[2.3. Вибір бази даних 20](#_Toc9785404)

[2.4. Вибір програмних засобів 22](#_Toc9785405)

[2.5. Вибір апаратних засобів 24](#_Toc9785406)

**ВСТУП**

Сьогодні актуальним є проблеми завантаженості вулиць та доріг автомобільними транспортними засобами, тому актуальною є здача ефективного регулювання трафіку потоку автомобілів. Реалізації даних систем вирішують ряд складних завдань, зокрема, однією із головних таких проблем є автоматичне розпізнавання автомобілів. Ці системи мають певні недоліки, які полягають в обмеженій кількості автомобілів, які вони можуть розпізнати, мають прив’язку до певної платформи, тобто не універсальні, є закритими та дорогими. В зв’язку з цим є необхідність в розробленні більш ефективних, автономних, універсальних і доступніших систем автоматичної ідентифікації руху транспортних засобів.

Узагальнена структурна схема існуючих систем ідентифікації автомобілів практично однакова і включає відеокамеру, операційний блок і сервіс для відображення даних. Операційний блок реалізується на певному обчислювальному пристрої у вигляді сервера, який виконує всі обчислювальні операції, пов’язані з розпізнаванням транспортних засобів на вхідному зображенні з камери.

При ідентифікації об’єкту на вхідному зображенні, він реєструється у базі даних а на клієнтську частину транслюється вже оброблений відео ряд із статистичними даними. Частково подібні технології реалізовуються в системах автоматичного контролю швидкості руху автомобілів на дорогах, але зазвичай дані системи мають закритий доступ. Прийнято рішення розробити систему, яка б могла виконувати аналогічні функції, але була б більш доступною для загальної маси людей. Для вирішення цієї задачі було вирішено розробити програмне забезпечення, яке б давало доступ великій кількості людей до даного функціоналу. Користувач матиме змогу, в режимі реального часу відслідковувати трафік на визначеній ділянці дороги. Дана система розроблена на платформі .NET Core. Для отримання можливості роботи з об’єктами з відео потоку використовується бібліотека класів OpenCVSharp, що є обгорткою OpenCV написаною для розробки під платформу .NET. Ця бібліотека містить алгоритми для обробки зображень, розпізнавання образів, роботи з відео, для пошуку та слідкування за об’єктами та інше.

# АНАЛІЗ СЕЦІАЛІЗОВАНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ З ВІДЕО ПОТОКУ

## 1.1. Огляд існуючих систем для розпізнавання об’єктів, транспортних засобів

В даний час існують різні рішення, що дозволяють розпізнавати ті чи інші візуальних об’єктів . Більшість з цих систем є системами контролю швидкості, які створені для забезпечення безпеки дорожнього руху. У зв'язку з призначенням, дані системи здатні розпізнавати тільки обмежений набір рухомих об’єктів.

## 1.2. Основні етапи алгоритму вирішення задачі розпізнавання об’єктів у відео потоці

Перший етап – отримання зображення з камери. В даній розробці використовуватиметься веб-камера, яка розміщується над ділянкою дороги, де ведеться спостереження. Вона зможе забезпечити достатню якість та частоту кадрів для ідентифікації автомобілів. Для операцій по визначенню меж об’єктів, їх відстеженню та підрахунку кожен кадр із відео потоку представляється у форматі цифрового зображення. Використовуючи засоби бібліотеки OpenCV отримується доступ до камери за допомогою, якої в подальшому проводиться обробка кадрів.

Другий етап – перетворення отриманого зображення. На даному етапі проводиться пошук образів. На зображення накладаються певні фільтри, які роблять чіткими контури об’єктів. Далі робота проводиться тільки із цими областями, що дозволяє підвищити продуктивність, адже система опрацьовуватиме лише окремі зони кадру.

Третій етап – виявлення замкнутих контурів. На цьому етапі фільтрується значна частина областей із попереднього етапу, залишаються лише об’єкти контури, яких будуть замкнуті та чіткі. Для того, щоб помилково не пропустити контури транспортних, які з якихось причин помилково були відсіяні, можна застосувати граничні фільтри, що роблять всі об’єкти, що залишились більш чіткими, та заповнять порожні проміжки між пікселями, які знаходяться близько один до одного.

Четвертий етап **–** вибір відповідного шаблона. Усі контури, які були знайдені на попередньому етапі будуть порівняні з спеціально підготовленими шаблонами автотранспорту для класифікації по формі (прямокутник, трикутник, круг), цим самим уже на цьому етапі робляться перші кроки до ідентифікації автомобіля.

П’ятий етап **–** ідентифікація дорожнього транспорту. Ідентифікація відбувається за допомогою спеціально навченого каскаду Хаара. Об’єкти з попереднього етапу були розбиті на декілька груп, що дозволило на цьому етапі не проводити порівняння з усіма наявними каскадами, а виконувати цю дію лише з конкретними із них. Даний етап є завершаючим, адже коли . каскад розпізнає образ, він автоматично виділяється на даному кадрі і відбувається процес відслідковування рух об’єкта.

## 1.3. Формування бінарних зображень з багатоканальних

### 1.3.1. Граничне перетворення

Зображення в програмування прийнято розглядати як матрицю (масив) пікселів. У завданнях часто потрібно вибрати пікселі вище / нижче певного граничного значення. Для таких цілей в бібліотеці OpenCV є функція: *threshold(Mat src, Mat dst, double threshold, double max\_value, int threshold\_type)*, яка виконує фіксоване граничне перетворення для елементів масиву (пікселів зображення), її аргументи:

1. src - вихідне зображення (одноканальне, 8-бітове або 32-бітове);

2. dst - цільове зображення того ж типу, що й src, або 8-бітове;

3. threshold - порогова величина;

4. max\_value - максимальне значення (використовується спільно з CV\_THRESH\_BINARY і CV\_THRESH\_BINARY\_INV);

5. threshold\_type - тип порогового перетворення:

* CV\_THRESH\_BINARY (value = value > threshold ? max\_value : 0);
* CV\_THRESH\_BINARY\_INV (value = value > threshold ? 0 : max\_value);
* CV\_THRESH\_TRUNC (value = value > threshold ? threshold : value);
* CV\_THRESH\_TOZERO (value = value > threshold ? value : 0);
* CV\_THRESH\_TOZERO\_INV (value = value > threshold ? 0 : value);
* CV\_THRESH\_MASK;
* CV\_THRESH\_OTSU.

Результатом такого перетворення є чорно-біле зображення, білим кольором на якому представлені відповідні під умови пошуку області, які необхідно виділити, тобто шукані об'єкти.

### 1.3.2. Виділення об'єкта за кольором

Потрібний результат можна отримати, використовуючи функцію для вибірки тих пікселів зображення, які лежать в заданому інтервалі значень. Це функція inRange(Mat src, Mat / Scalar lower, Mat / Scalar upper, Mat dst), вона проводить поелементну перевірку елементів масиву (пікселів) і перевіряє, чи лежать значення масиву між значеннями елементів двох інших масивів або скалярів. Формула:

|  |  |
| --- | --- |
| 𝑑𝑠𝑡(𝑖𝑑𝑥) = 𝑙𝑜𝑤𝑒𝑟(𝑖𝑑𝑥) ≤ 𝑠𝑟𝑐(𝑖𝑑𝑥) < 𝑢𝑝𝑝𝑒𝑟(𝑖𝑑𝑥) | (1) |

Де:

1. src - вихідний масив;

2. lower - масив або скаляр з нижньою межею (включаючи), якщо це скаляр (Scalar (x, y, z)), то пікселі, значення яких нижче, ніж x, y і z для HUE, SATURATION і VALUE вважаються чорними пікселями в результуючому зображенні dst;

3. upper - масив або скаляр з верхньою межею (не включаючи), якщо це скаляр (Scalar (x, y, z)), то пікселі, значення яких вище або дорівнюють x, y і z для HUE, SATURATION і VALUE вважаються чорними пікселями в результуючому зображенні dst;

4. dst - масив для зберігання результату.

Результат цього перетворення такий же ж, як і в попередньому пункті, чорно-біле зображення з виділеними об'єктами, які підходять під задані в аргументах умови.

Відмінність цього методу в тому, що він здатний задавати обмеження, як знизу, так і зверху, що робить його більш універсальним.

## 1.4. Контурний аналіз

Контурний аналіз - це один з важливих і дуже корисних методів опису, розпізнавання, порівняння і пошуку графічних образів (об'єктів).

Контур - це зовнішні обриси об'єкта.

При проведенні контурного аналізу:

* припускається, що контур містить достатню інформацію про форму об'єкта;
* внутрішні точки об'єкта, до уваги не беруться.

Тому накладається ряд обмежень на область застосування контурного аналізу, які в основному, пов'язані з проблемами виділення контуру на зображеннях:

* через однакову яскравість з фоном, об'єкт може не мати чіткої межі, або може бути зашумлений перешкодами, що призводить до неможливості виділення контуру (успішне застосування тільки при чітко вираженому об'єкті на контрастному тлі і відсутності перешкод);
* перекриття об'єктів або їх угруповання призводить до того, що контур виділяється неправильно і не відповідає границям об'єкту.

Серед методів отримання довічного зображення можна виділити, наприклад, граничне перетворення або виділення елементу за кольором, які були описані раніше.

Після будь-якого з перетворень виходить бінарне зображення, яке однозначно задає границі об'єкта. Сукупність пікселів, що становлять границю об'єкта і є контуром об'єкта.

Щоб оперувати отриманим контуром, його необхідно якось представити. Наприклад, вказувати вершини відрізків, що становлять контур.

Бібліотека OpenCV реалізує зручні методи для детектування і маніпуляції з контурами зображення.

Для пошуку контурів на бінарному зображенні використовується функція: *findContours(Mat image, vector <vector <Point >> contours, int mode CV\_DEFAULT (CV\_RETR\_LIST), int method CV\_DEFAULT (CV\_CHAIN\_APPROX\_SIMPLE), Point offset CV\_DEFAULT (Point (0,0)))*, аргументи якої:

1. image - вихідне 8-бітове одноканальне зображення (ненульові пікселі обробляються як 1, а нульові - 0. Для отримання такого зображення з градацій сірого можна, наприклад, використовувати функції threshold () або inRange ();

2. contours - вектор векторів знайдених контурів;

3. mode - режим пошуку:

* CV\_RETR\_EXTERNAL - знайти тільки крайні зовнішні контури;
* CV\_RETR\_LIST - знайти всі контури і розмістити їх списком;
* CV\_RETR\_CCOMP - знайти всі контури і розмістити їх у вигляді 2-рівневої ієрархії;
* CV\_RETR\_TREE - знайти всі контури і розмістити їх в ієрархії вкладених контурів.

4. method - метод апроксимації:

* CV\_CHAIN\_CODE - ланцюговий код Фрідмана;
* CV\_CHAIN\_APPROX\_NONE - всі крапки ланцюгового коду переводяться в точки;
* CV\_CHAIN\_APPROX\_SIMPLE - стискає горизонтальні, вертикальні і діагональні сегменти і залишає тільки їх кінцеві точки;
* інші методи;

5. offset - зміщення, на яке зрушуються точки контуру (корисно, якщо контури витягуються з області інтересів і потім повинні аналізуватися в контексті цілого зображення).

Функція findContours() може знаходити зовнішні і вкладені контури і визначати їх ієрархію вкладення. Відобразити знайдені контури можна за допомогою функції: *drawContours(Mat img, vector <vector <Point >> contours, int contourId, Scalar external\_color, const Scalar & color, int thickness CV\_DEFAULT (1), int line\_type = 8, Array hierarchy = noArray (), int max\_level = INT\_MAX, Point offset = Point (0, 0))*, аргументи якої:

1. img – зображення, на якому будуть намальовані контури;

2. contours - всі контури;

3. contourId - визначає контур для малювання;

4. color - колір контуру;

5. thickness - товщина лінії для відображення контурів (якщо величина

негативна, то область, обмежена контуром заливається вибраним кольором);

6. line\_type - тип лінії;

7. hierarchy - інформація про ієрархію;

8. max\_level - максимальний рівень для відображення контурів (0 - тільки даний контур, 1 - даний і всі наступні на даному рівні, 2 - всі наступні контури і всі контури на наступному рівні і т.д.);

9. offset - параметр зсуву контуру.

Звичайна послідовність дій при розпізнаванні об'єктів методом контурного аналізу:

1. попередня обробка зображення (згладжування, фільтрація перешкод, збільшення контрасту);

2. бінаризація зображення;

3. виділення контурів об'єктів;

4. первинна фільтрація контурів (по периметру, площі і т.п.);

5. еквалізація контурів (приведення до єдиного розміру), щоб домогтися незалежності від масштабу;

6. перебір всіх знайдених контурів (об'єктів) та пошук шаблону, максимально схожого на даний (або ж сортування по будь-якою ознакою).

Додаткові корисні функції:

* boundingRect(vector <Point> points, int update CV\_DEFAULT (0)) - повертає прямокутник, яким можна обвести контур;

1. points - набір 2D-точок (послідовність точок контуру);

2. update - прапорець обновлення:

* + 0 - прямокутник розраховується і повертається;
  + 1 - помилка виконання.

Функція повертає прямокутник, у якого сторони строго вертикальні і горизонтальні.

Існують також і інші функції, які часто використовуються, наприклад, minAreaRect2(), яка повертає мінімально можливий прямокутник, яким можна обвести контур, але який може бути повернений, щодо системи координат зображення на певний кут, і minEnclosingCircle(), яка знаходить коло мінімальної площі, що містить даний контур, однак при вирішенні поставленого в цій роботі завдання вони не потрібні.

## 1.5. ROI (Region Of Interest)

ROI (Region Of Interest - регіон інтересів) - один з фундаментів OpenCV. ROI дозволяє користувачеві задати певну прямокутну область зображення (тобто ROI завжди повинна знаходиться всередині вихідного зображення). Майже всі функції повинні підтримувати роботу з ROI, тобто роботу з виділеною областю зображення, що корисно для прискорення роботи алгоритмів. Якщо нас цікавить тільки певна область зображення - можна її виділити і працювати тільки з нею, не зачіпаючи все зображення цілком.

Ось які функції для роботи з ROI надає OpenCV:

* CVAPI(void) cvSetImageROI(IplImage \* image, CvRect rect); - встановлення регіону інтересів;

image - вказівник на зображення;

rect - прямокутник, що цікавить.

* СVAPI(void) cvResetImageROI(IplImage \* image); - скидає область інтересів;

image - вказівник на зображення;

* CVAPI(CvRect) cvGetImageROI(const IplImage \* image); - повертає область інтересів зображення;

image - покажчик на зображення;

Якщо ROI не встановлена, ​​функція поверне cvRect(0,0, image-> width, image-> height)

* CVAPI(void) cvAddS(const CvArr \* src, CvScalar value, CvArr \* dst, const CvArr \* mask CV\_DEFAULT (NULL)); - розрахунок суми масиву і скаляра (скаляр додається до кожного елементу масиву);

src - вихідний масив;

value - скаляр для підсумовування;

dst - масив для збереження результату;

mask - маска (8-бітний 1-канальний масив, що визначає, які елементи масиву підлягають зміні).

ROI можна використовувати для вирізання частини зображення або, навпаки, додавання зображення. Потім викликається функція cvAddS(), яка додає до елементів масиву задану скалярну величину. Результатом є виділення кольором квадрата ROI.

## 1.6. Класифікатор Хаара

Виявлення об'єктів за допомогою каскадних класифікаторів Хаара є ефективним методом виявлення об'єктів, запропонованим Полом Віолою та Майклом Джонсом. Функція каскаду навчається на безлічі позитивних і негативних зображень. Позитивними називаються зображення, де чітко виділений образ об’єкту, що моє бути розпізнаний. Потім вона використовується для виявлення об'єктів в інших зображеннях.

Класифікатор формується на примітивах Хаара шляхом розрахунку значень ознак. Для навчання на вхід класифікатора спочатку подається набір позитивних зображень, з попередньо виділеною областю на зображенні. Далі проходить порівняння примітивів і обчислення значення ознаки. Результати заздалегідь розрахованих значень зберігаються в XML-файлі.

Оскільки ознаки Хаара мало підходять для навчання або класифікації, для опису об'єкта з достатньою точністю необхідна більша кількість ознак. Тому ознаки Хаара надходять в каскадний класифікатор, слугуючий для швидкого відкидання виділених областей, де відсутній необхідний об'єкт, і видачі результатів щодо знаходження об'єкта.

Класифікатор побудований на основі алгоритму бустингу (від англ.boost-покращення, посилення), для вибору найбільш підходящих ознак для шуканого об'єкта на даній частині зображення. У загальному випадку бустинг - це комплекс методів, що сприяють підвищенню точності аналітичних моделей. Ефективна модель, яка припускає мало помилок класифікації, називається "сильною". "Слабка" ж, навпаки, не дозволяє надійно розділяти класи або давати точні прогнози, робить велику кількість помилок. Тому бустинг означає "посилення" "слабких" моделей і є процедурою послідовного побудови композиції алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів.

В результаті роботи алгоритму бустингу на кожній ітерації формується простий класифікатор виду:



де p\_j- напрямок знака нерівності, θ\_j- значення порога, f\_j (z) - обчислене значення ознаки, z - вікно зображення розміром 24 × 24 пікселів.

Отриманий класифікатор має мінімальну помилку по відношенню до поточних значень ваг, задіяним в процедурі навчання для визначення помилки.

Для пошуку об'єкта на цифровому зображенні використовується навчений класифікатор, представлений в форматі xml. Класифікатор формується на примітивах Хаара.

На рис. 1.1 зображено структуру класифікатора.

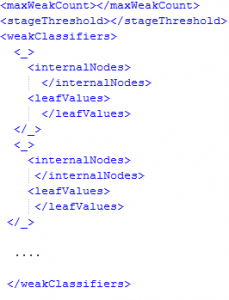


Рис. 1.1. Структура класифікатора

де maxWeakCount - кількість слабких класифікаторів; stageThereshold - максимальний поріг яскравості; weakClassifiers - набір слабких класифікаторів, на основі, яких виноситься рішення про те, чи знаходиться об'єкт на зображенні чи ні; internalNodes і leafValues ​​- параметри конкретного слабкого класифікатора.

Перші два значення в internalNodes не використовуються, третє - номер ознаки в загальній таблиці ознак (вона розташовується в XML-файлі під тегом features), четверте - порогове значення слабкого класифікатора. Якщо значення ознаки Хаара менше порога слабкого класифікатора, вибирається перше значення leafValues, якщо більше – друге.

На основі даного алгоритму будується каскад класифікаторів, які приймають рішення про те, розпізнаний об'єкт на зображенні, чи ні. Наявність або відсутність предмета на ділянці зображення, що досліджується, визначається різницею між значенням ознаки і порогом, отриманим під час навчання.

# 2. ВИБІР ЗАСОБІВ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ З ВІДЕО ПОТОКУ

Завданням даної бакалаврської кваліфікаційної роботи є розробка інформаційної системи розпізнавання об’єктів з відео потоку, тобто пошуку та ідентифікації транспортних засобів із відео потоку. Дана система повинна стати загальнодоступним аналогом систем контролю швидкості чи руху автомобілів.

## 2.1. Вибір технологій розробки програмного забезпечення

Технологія WPF (Windows Presentation Foundation) є частиною стеку технологій платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Якщо при створенні традиційних додатків на основі WinForms за графічне відображення елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF використовують DirectX. У цьому полягає ключова особливість рендеринга графіки у WPF: використовуючи WPF, значна частина роботи по відображенню графіки, як найпростіших кнопочок, так і складних 3D-моделей, покладається на графічний процесор відеокарти, що також дозволяє скористатися апаратним прискоренням графіки.

Однією із важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, побудованого на XML: розробник можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

Переваги WPF

Технологія WPF пропонує розробнику:

* Використання традиційних мов .NET-платформи - C # і VB.NET для створення логіки додатка
* Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, на базі XML, що є альтернативою програмному створенню графіки та елементів управління, а також можливість комбінування XAML і C # / VB.NET
* Незалежність від розширення екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, проекти написані на WPF легко масштабуються під різні екрани з різним розширення.
* Нові можливості, яких складно було досягти в WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін.
* Хороша взаємодія із WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління WinForms.
* Великі можливості для створення різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самостійного створення нових елементів, анімацій, прив'язки даних, стилів, шаблонів, тем і багато іншого.
* Апаратне прискорення графіки - незалежно від того, чи працюєте розробник із 2D або 3D, графікою або текстом, усі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіку більш плавною.
* Створення додатків під безліч ОС сімейства Windows - від Windows XP до Windows 10

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення додатків з великою кількістю тривимірних зображень, перш за все ігор, краще використовувати інші засоби - DirectX або спеціальні фреймворки, такі як Monogame або Unity.

Також варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище. Але це компенсується більш широкими графічними можливостями і підвищеною продуктивністю при відображенні графіки.

## 2.2. Вибір середовища розробки

Visual Studio

Microsoft Visual Studio – інтегроване середовище розробки (IDE) створене корпорацією Microsoft. Дане середовище використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-служб та мобільних додатків. Visual Studio використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store та Microsoft Silverlight. Також воно може продукувати як і нативний код, так і керований.

Visual Studio включає в себе редактор коду, який підтримує технологію IntelliSense (компонент завершення коду), а також рефакторинг коду. Інтегрований відлагджувач працює як відлагоджувач верхнього рівня, так і на рівні машини. Інші вбудовані інструменти включають в себе профайлер коду, дизайнер форм для створення GUI-додатків, веб-дизайнер, дизайнер класів і дизайнер схеми бази даних. Середовище приймає плагіни, що підвищують функціональність практично на кожному рівні, включаючи додавання підтримки систем керування джерелами (наприклад, Subversion і Git) та додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменних мов або наборів інструментів для інших аспектів розробки програмного забезпечення життєвого циклу (наприклад, клієнт Team Foundation Server: Team Explorer).

Visual Studio підтримує близько 30 різних мов програмування і дозволяє редактору коду та відлагоджувачу підтримувати (в різній степені) практично будь-яку мову програмування, за умови наявності спеціальної мовної служби. До вбудованих відносяться такі мови як: C, C ++, C ++ / CLI, Visual Basic .NET, C #, F #, JavaScript, TypeScript, XML, XSLT, HTML і CSS. Підтримка інших мов, таких як Python, Ruby, Node.js доступна через плагіни.

Для розробки було використано базове видання Visual Studio 2017 Community edition, доступне безкоштовно.

SQL Server Management Studio

SQL Server Management Studio (SSMS) – це програмне забезпечення, що було випущено разом із Microsoft SQL Server 2005, який використовується для налаштування, керування та адміністрування всіх компонентів Microsoft SQL Server. Даний додаток став аналогом Enterprise Manager в SQL 2000, який застосовувався до нього. Інструмент включає в себе як редактори сценаріїв, так і графічні інструменти, що працюють з об'єктами та функціями сервера.

Основною особливістю SSMS є Object Explorer, який дозволяє розробнику переглядати, вибирати та керувати будь-якими з об'єктами розміщеними на сервері. Даний продукт також має окреме видання Express, яке можна вільно завантажувати, однак останні версії SSMS повністю здатні підключатися до будь-якого екземпляра SQL Server Express і керувати ним. Корпорація Майкрософт також додала підтримку старих версій SQL Server, що дозволило новій версії SSMS підключатися до старших версій SQL Server.

У червні 2015 року Microsoft оголосила про свій намір випустити майбутні версії SSMS незалежно від версій SQL Server

## 2.3. Вибір бази даних

SQL Server є однією з найбільш популярних систем управління базами даних (СУБД) в світі. Дана СУБД підходить для самих різних систем: від невеликих програм до проектів із складною архітектурою. SQL Server був створений корпорацією Microsoft. Перша версія вийшла в 1989 році. А поточною є версія SQL Server 2017, реліз якої відбувся в жовтні 2017.

SQL Server тривалий час був виключно системою управління базами даних для Windows, однак починаючи із версії SQL Server 2016 система доступна і на платформі Linux.

SQL Server характеризується такими особливостями, як:

* Продуктивність. SQL Server працює дуже швидко.
* Надійність та безпека. SQL Server підтримує шифрування даних.
* Простота. Із даної СУБД відносно легко працювати і вести адміністрування.

Центральний аспектом в MS SQL Server, як і в будь-якій СУБД, є база даних. База даних являє собою сховище даних, організованих певним способом. Нерідко фізична база даних представляє файл на жорсткому диску, хоча таке співвідношення є необов’язковим. Для зберігання та адміністрування баз даних застосовуються системи управління базами даних (СУБД).

Для організації баз даних MS SQL Server використовує реляційну модель. Ця модель баз даних була розроблена ще в 1970 році. А на сьогоднішній день вона фактично є стандартом для організації даних.

Microsoft SQL Server як мову запитів використовує версію SQL, що отримала назву Transact-SQL (скорочено T-SQL), яка є реалізацією SQL-92 (стандарт ISO для SQL) з багатьма розширеннями. T-SQL дозволяє використовувати додатковий синтаксис процедур, що зберігаються і забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream (TDS, протокол передачі табличних даних).

MS SQL Server доступний в різних варіантах. Перед усім, це MS SQL Server Enterprise - повний випуск, створений для використання в реальних проектах. Він використовується на різних хостингах і серверах баз даних. Проте він доступний тільки в платній. Для даного програмного забезпечення достатньо і випуску Express: він безкоштовний. Його можна поставити в якості реального сервера і використовувати в реальних завданнях, однак він має урізаний функціонал в порівнянні з повною версією.

Microsoft та інші компанії пропонують велику кількість програмних засобів розробки, які дозволяють розробляти застосунки для бізнесу з використанням баз даних Microsoft SQL Server. Microsoft SQL Server 2017 включає також Common Language Runtime (CLR) Microsoft .NET, що дозволяє застосункам, розробленим на мовах платформи .ΝΕΤ (наприклад, VB.NET або C#), реалізовувати спеціальні процедури, що є розширенням мови T-SQL

## 2.4. Вибір програмних засобів

Для реалізації роботи із зображенням була обрана бібліотека OpenCV.

OpenCV – це бібліотека написана на мові C / C++, що містить засоби комп'ютерного зору і є розміщена у відкритому доступі (Open Source Computer Vision Library). Вона містить набір готових класів, функцій, спеціальних типів даних та алгоритми для обробки зображень, розпізнавання образів, роботи з відео, стеження за об'єктами та ін.

OpenCV підтримуються такими операційними системами як Windows, OS X, Linux. Також існують так звані обгортки (wrappers) для різних мов програмування, в тому числі C#, Python, Java, Ruby та ін. Бібліотека містить наступні модулі:

* ядро, що містить базові структури даних, та реалізації операцій над багатовимірними масивами, алгоритми роботи з пам'яттю, алгоритми генерації випадкових чисел, функції для серіалізації в XML-файл, функції для роботи з 2D графікою;
* модуль обробки зображень, він включає функції, що реалізують базові операції для роботи зі зображеннями (перетворення, фільтрація і т.д.), функції для аналізу зображень (пошук контурів, гістограми і т.д.), алгоритми аналізу рухів, стеження за об'єктами, алгоритми розпізнавання об'єктів (осіб, предметів);
* модуль машинного навчання, цей модуль містить функції для класифікації та аналізу даних;
* модуль для створення призначеного для користувача інтерфейсу, який відповідає за створення вікон, вивід зображень, захоплення відео з файлів і камер, читання / запис зображень;

Для застосування бібліотека OpenCV на базі платформи .NET використовується бібліотека Emgu CV. Дана бібліотека є крос-платформною .Net-обгорткою для бібліотеки обробки зображень OpenCV. Вона дозволяє використовувати повний функціонал OpenCV у мовах сімейства .NET, таких як C #, VB, VC ++, IronPython і т.д. Обгортка може бути скомпільована Visual Studio, Xamarin Studio і Unity, вона може працювати на Windows, Linux, Mac OS X, iOS та Android.

Emgu CV написана повністю на C #. Перевага полягає в тому, що вона може бути скомпільована в Mono і тому може працювати на будь-якій платформі Mono, включаючи iOS, Android, Windows Phone, Mac OS X і Linux. Багато зусиль було витрачено розробниками на повноцінну реалізацію C#, адже заголовні файли та файли реалізації, що використовуються в C++ повинні були бути повністю перенесені, порівняно з керованою реалізацією C++, де файли заголовків можуть бути просто включені. Крім того, це дає змогу кодові бути крос-платформним.

Так як Emgu CV є програмною обгорткою OpenCV, вона повністю відображає її структуру. Emgu CV має два шари обгортки, як показано нижче:

* Базовий рівень (рівень 1) містить відображення функцій, класів, типів даних та перерахувань відображень, які безпосередньо представляють ті, що містяться в OpenCV.
* Другий рівень (рівень 2) містить класи, які розроблені та використовують переваги платформи .NET.

На рис. 2.1 зображено структуру бібліотеки Emgu CV.

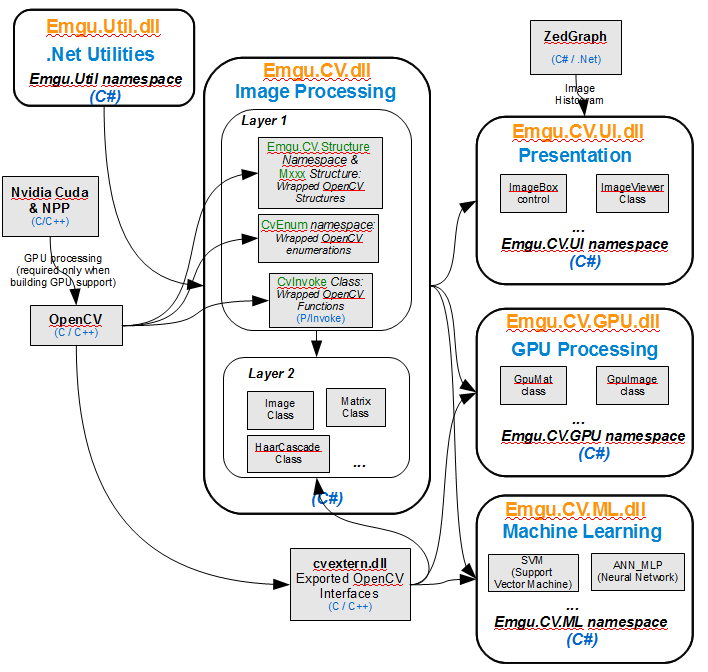


Рис. 2.1. Структура бібліотеки Emgu CV

2.5. Вибір апаратних засобів

Технологічна платформа, яка використовується для створення і розробки системи розпізнавання об’єктів з відео потоку на серверній частині:

* ОС: Windows, Linux чи OS X
* Процесор: Intel Core i5 CPU 2.7 GHz
* Операційна пам’ять: 8 ГБ
* Використовується веб камера пеоеферійного пристрою з 8Мп модулем, апаратурою f/2.2 . Роздільна здатність запису відео потоку – 120 fps.

Як бачимо, дана технологічна платформа більш ніж може слугувати архітектурним рішенням для даної мети створення і роботи з системою.

Для даної системи клієнтській частині також потрібен пристрій, що на платформі .NET Core, для тестування системи розпізнавання об’єктів з відео потоку, в результаті вибрано:

* Asus
* ОС: Windows 10 Pro.
* Використовується процесор Intel Pentium з 64-х бітною архітектурою, що забезпечує швидкодію при роботі із відео та фото.