**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Кафедра комп’ютеризованих систем управління**

**ДОПУСТИТИ ДО ЗАХИСТУ**

**Завідувач кафедри**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**“\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_2020 р.**

**ДИПЛОМНА РОБОТА**

**(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**

**“МАГІСТР”**

**Тема: « Програмний модуль ідентифікації рухомого об’єкта у просторі »**

**Виконавець: \_\_\_\_\_\_студентка групи СП-224М Тучина Валентина Валеріївна**

**Керівник: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н. доцент Нечипорук Олена Петрівна**

**Нормоконтролер: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тупота Євгеній Вікторович**

**Київ 2020**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет Кібербезпеки, комп'ютерної та програмної інженерії\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра комп’ютеризованих систем управління\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність 123 «Комп'ютерна інженерія»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр, найменування)

Освітньо-професійна програма «Системне програмування»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2020 р.

**ЗАВДАННЯ**

**на виконання дипломної роботи (проекту)**

Тучиної Валентини Валеріївни \_

(прізвище, ім'я, по батькові випускника в родовому відмінку)

**1.** **Тема дипломної роботи:** Програмний модуль ідентифікації рухомого об’єкту у просторі \_

затверджена наказом ректора від «27» серпня 2019 р. №\_ 1808/ст \_

**2. Термін виконання роботи:** з 27.08.2019 по 20.02.2020

**3. Вихідні дані до роботи:**\_\_\_\_завдання на дипломну роботу, технічна документація

**4. Зміст пояснювальної записки:** Огляд предметної області і постановка завдання дослідження. Програмний модуль ідентифікації рухомого об’єкта у просторі. Програмне забезпечення програмного модуля та підготовка даних. Розробка і тестування програмного модулю ідентифікації рухомого об’єкта у просторі.

**5. Перелік обов'язкового графічного (ілюстративного) матеріалу:**

1) Програмний модуль розпізнавання автомобільного номера (Схема структурна)

2) Робота модуля розпізнавання автомобільного номера (Схема алгоритму) \_

3) Робота методу розпізнавання автомобільного номера (Схема алгоритму) \_

4) Робота методу пошуку верхньої границі номера (Схема алгоритму) \_

5) Результат роботи програмного модуля з відеофайлом \_

6) Результат роботи програмного модуля з *IP*-камерою \_

**6. Календарний план-графік**

|  |  |  |  |
| --- | --- | --- | --- |
| №  пор. | Завдання | Термін  виконання | Відмітка про виконання |
| 1 | Ознайомлення з постановкою завдання дипломного проекту | 27.08.2019 –  27.09.2019 |  |
| 2 | Аналіз літературних джерел та інтернет ресурсів | 28.09.2019 –  10.10.2019 |  |
| 3 | Проектування інтерфейсу користувача | 11.10.2019 –  01.11.2019 |  |
| 4 | Програмування програмного модулю | 02.11.2019 –  25.12.2019 |  |
| 5 | Підготовка пояснювальної записки | 26.12.2019 –  20.01.2020 |  |
| 6 | Підготовка графічного матеріалу | 21.01.2020 –  03.02.2020 |  |
| 7 | Захист дипломного проекту | 04.02.2020 |  |

7. Дата видачі завдання: “27 " серпня 2019 р.

Керівник дипломної роботи (проекту) Нечипорук О.П.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання Тучина В.В.

(підпис випускника) (П.І.Б.)

**РЕФЕРАТ**

Пояснювальна записка до дипломної роботи «Програмний модуль ідентифікації рухомого об’єкта у просторі » складається зі вступу, чотирьох розділів, що містять 30 рисунків та 1 таблицю, висновків та списку використаних джерел, що містять 19 найменувань. Загальний обсяг роботи становить 89 аркушів.

Під час роботи над дипломною роботою, запропонованою до розгляду, розроблено програмний модуль для ідентифікаціїї рухомого об’єкта у просторі (розпізнавання автомобільного номера).

Об’єкт дослідження – ідентифікація рухомого об’єкта у просторі.

Предмет дослідження – програмний модуль ідентифікації рухомого об’єкта у просторі.

Мета роботи –розробити програмний модуль ідентифікації рухомого об’єкта у просторі.

Методи дослідження – застосування інструменту розробки *Qt*, мови програмування *C++,* та бібліотек *OpenCV* та *Teseract OCR* для створення програмного модуля розпізнавання автомобільних номерів.

Подальший розвиток програмного модулю можливий в напрямку застосування його для розпізнавання автомобільних номерів в системах автоматизованих парковок та системах контролю в’їзду на територію.

Результати дипломного проекту рекомендується використовувати при створенні систем для розпізнавання автомобільних номерів в цілях забезпечення безпеки компаній, або покращення існуючих програмних забезпечень.

ІДЕНТИФІКАЦІЯ, АВТОМОБІЛЬНІ НОМЕРИ, РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, МОДУЛЬ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ.

**ЗМІСТ**

[ВСТУП............... 6](#_Toc31048417)

[РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ  
 ДОСЛІДЖЕННЯ 12](#_Toc31048418)

[1.1. Методи ідентифікації 12](#_Toc31048419)

[1.2. Сучасні програмні засоби для ідентифікації об’єктів у просторі 15](#_Toc31048420)

[1.3. Постановка завдання дослідження 28](#_Toc31048421)

[1.4. Висновки до розділу 29](#_Toc31048422)

[РОЗДІЛ 2 СТРУКТУРА ПРОГРАМНОГО МОДУЛЯ ІДЕНТИФІКАЦІЇ  
 РУХОМОГО ОБ’ЄКТА ПРОСТОРІ .............................................…....30](#_Toc31048423)

[2.1. Метод Віоли-Джонса 30](#_Toc31048424)

[2.2. Пошук автомобільного номера на цифровому зображенні 33](#_Toc31048425)

[2.3. Розпізнавання автомобільного номера 34](#_Toc31048426)

[2.4. Висновки до розділу 47](#_Toc31048429)

[РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ ІДЕНТИФІКАЦІЇ  
 РУХОМОГО ОБ’ЄКТА У ПРОСТОРІ 48](#_Toc31048430)

[3.1. Огляд обраних інструментів розробки 48](#_Toc31048431)

[3.2. Програмні та апаратні вимоги до програмного модулю 58](#_Toc31048432)

[3.3. Навчання каскадів Харра для програмного модуля 58](#_Toc31048433)

[3.4. Висновки до розділу 61](#_Toc31048434)

[РОЗДІЛ 4 ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ІДЕНТИФІКАЦІЇ   
 РУХОМОГО ОБ’ЄКТА У ПРОСТОРІ 63](#_Toc31048435)

[4.1. Реалізація алгоритму ідентифікації рухомого об’єкта у просторі 63](#_Toc31048436)

[4.2. Тестування роботи програмного модуля 80](#_Toc31048437)

[4.3. Висновки до розділу....................................................................................................81](#_Toc31048438)

[ВИСНОВКИ…...................................................................................................................83](#_Toc31048439)

[СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ](#_Toc31048440) [ДЖЕРЕЛ..............88](#_Toc31048441)

ДОДАТОК А……………………………………………………………..….…….……...90

# **ВСТУП**

Розпізнавання образів – відносно нова галузь математичної кібернетики, але вона інтенсивно розвивається, і вже зараз є численні важливі програми що реалізують методи розпізнавання буквально у всіх сферах науки і техніки, в тому числі в економіці, техніці, медицині, фізиці, біології, соціології і так далі. Розпізнавання образів є основою штучного інтелекту [4].

Завдання розпізнавання образів зводиться до опису та класифікації (впізнавання) об’єктів різного типу.

Об’єкти одного класу мають бути подібні один до одного за певними критеріями, ознаками схожості. Об’єкти з різних класів повинні мати ознаки не характерні один одному.

Відповідно до цього образом є множина подібних об’єктів що мають спільні риси. Розпізнати образ означає вказати приналежність об’єкта до певного класу.

Із завданням розпізнавання будь-яка людина стикається щохвилини. Наприклад, ми впізнаємо людей і предмети, розпізнаємо букви і цифри, розуміємо мову, розпізнаємо з різним ступенем успішності небезпечні ситуації. Можна сказати, що живі організми змушені в процесі своєї життєдіяльності постійно вирішувати задачі розпізнавання [5].

Класифікація є однією з найпростіших логічних операцій, та є базовим елементом мислення людини. Як правило ми здійснюємо класифікацію і розпізнавання об’єктів автоматично, навіть не замислюючись над цим. Операція ідентифікації відбувається завдяки нейронам головного мозку, та триває лічені мілісекунди.

Розпізнавання образів є одним з актуальних напрямків інформатики, пов’язаних з автоматизацією процесів обробки і використання інформації.

За оцінкою *Markets And Markets,* опублікованій в 2017 році, обсяг ринку розпізнавання образів досягне 29,98 мільярдів доларів до 2021 року із середнім *CAGR* на рівні 19,1 відсотків. Технології розпізнавання образів містять в собі розпізнавання патернів, оптичних образів, коду, об’єктів і цифрових фотографій. Вони або окремо, або в інтегрованому вигляді використовуються в таких сферах, як безпека та спостереження, сканування і створення зображень, маркетинг і реклама, доповнена реальність і пошук зображень.

Ключовими фігурами цього ринку є такі великі корпорації, як *NEC, Google, Honeywell, Hitachi і Qualcomm*. Також є багато менших за розміром компаній, таких як *LTU Technologies, Attrasoft, Blippar* і *SLYCE,* і таких вендорів, як *Catchoom* і *Wikitude.*

Розпізнавання об’єктів широко застосовується у багатьох сферах. Прикладами сфер є медицина, де розпізнавання образів слугує для діагностики захворювань. Розвагах – велике різноманіття продуктів ігрової індустрії. Безпека – охоронні системи, автоматизовані пропускні пункти та інше.

В контексті даної роботи звернемо увагу безпосередньо на розпізнавання автомобільних номерів.

Область застосування систем автоматизованого розпізнавання автомобільних номерів досить різноманітна. Перш за все, розпізнавання номера автомобіля буде корисно на станціях технічного обслуговування, автомобільних заправних станціях, автомийках, складах, підприємствах, паркінгах.

Установка системи розпізнавання дозволяє контролювати присутність і переміщення транспортних засобів на території будь-якого розміру, а створення відеоархіву дає можливість швидко і ефективно знаходити порушників. Інтеграція системи з паркінгом зводить до мінімуму вплив людського фактора і унеможливлює зловживання персоналу своїми повноваженнями. Інтеграція з комплексами відеофіксації порушень правил дорожнього руху забезпечує надійний контроль над транспортними магістралями: ідентифікація порушників за номером дозволяє автоматично готувати квитанції для оплати штрафів, а також виявляти в потоці автомобілі, що представляють інтерес для правоохоронних органів.

Істотними перевагами, які надає система розпізнавання номерів автомобілів:

* значне підвищення рівня безпеки та контролю автомобільного транспорту на об’єкті;
* виключається можливість для третіх осіб проникнути на територію, що охороняється використовуючи підроблені або викрадені магнітні пропуску або електронні картки (автомобіль теж можна викрасти, але це набагато складніше);
* автоматичне ведення звітності про транспортні засоби з можливістю формування численних звітів;
* систему розпізнавання автомобільних номерів можна легко інтегрувати в загальну базу даних організації.

Функції, які може виконувати така система розпізнавання автомобільних номерів досить різноманітні:

* контроль в’їзду і виїзду на контрольовану територію;
* обмеження виїзду з території підприємства, наприклад, на автостанції, клієнта що не вчинив оплату;
* здійснення контролю завантаження сервісної зони.

У поєднанні з системами контролю доступу ідентифікація автомобільних номерів дає додаткові переваги. Перш за все, це повний контроль знаходження автомобільного транспорту в навантажувальної зоні підприємства. Це дає можливість відстежити ввезення сировини або вивезення готової продукції, перевірити ефективність вантажно-розвантажувальних робіт і запобігти розкрадання.

При цьому, перевіркою номера автомобіля не тільки на в’їзді, але і на виїзді виключається можливість вивезення вантажу за підробленими або помилковим супровідним документам.

Але найбільше переваг отримує власник паркінгу або автостоянки. Система автоматичного розпізнавання номерів дозволить проконтролювати заповнюваність території в реальному часі, що дасть можливість вжити заходів щодо підвищення ефективності.

Поєднання розпізнавання номерів автомобілів з системою оплат повністю виключить можливість зловживання або розкрадання з боку найманих працівників. А так же повністю виключить можливість помилок в підрахунку часу знаходження транспортного засобу на території автостоянки і дасть залізні докази в суперечках з недобросовісними клієнтами.

Вартості подібних систем різняться та лежать в діапазонах цін від 6 000 гривень до понад 250 000 гривень, тож продаж програмного забезпечення для розпізнавання автомобільних номерів приносить гарний прибуток власникам компаній розробників.

У першому розділі дипломної роботи розглядається сучасні системи для розпізнавання автомобільних номерів, їх характеристики, вартості, вимоги. В загальних рисах виконується огляд методів розпізнавання об’єктів. Також встановлюється що для досягнення мети дипломної роботи необхідно вирішити такі завдання:

* проаналізувати сучасні засоби розпізнавання автомобільних номерів;
  + розробити структуру програмного модулю;
* модифікувати алгоритм ідентифікації рухомого об’єкта;
  + програмно реалізувати модуль для розпізнавання автомобільних номерів як з *IP*-камери, так і з завантаженого файлу та протестувати роботу модулю.

У другому розділі розглядається алгоритм Віоли-Джонса, створюється алгоритм роботи модулю розпізнавання автомобільних номерів, та алгоритми роботи його фрагментів.

У третьому розділі розглянуті інструменти розробки за допомогою яких буде розроблюватись програмний модуль розпізнавання автомобільних номерів. До їх числа входить *Qt* , мова програмування *С++,* та бібліотеки *OpenCV* та *Teseract OCR.* Також виконується створення та навчання каскадів Харра для подальшого використання при розпізнаванні об’єктів, та розглядаються системні вимоги для використання розроблюваного програмного модулю.

У четвертому розділі даної роботи розроблено програмний модуль, відповідно до викладених в другом розділі алгоритмів:

* алгоритм нормалізації кута нахилу;
* алгоритм пошуку нижньої границі автомобільного номера;
* алгоритм пошуку верхньої границі автомобільного номера;
* алгоритм пошуку бічних границь автомобільного номера;
* алгоритм сегментації автомобільного номера на символи;
* алгоритм розпізнавання символів;
* алгоритм роботи модуля розпізнавання автомобільного номера;
* загальний алгоритм програмного модулю.

Відбувається тестування програми.

Задача ідентифікації автомобільних номерів є актуальною, адже це дозволяє вирішувати ряд практичних задач, в тому числі задач підвищення безпеки.

Об’єкт дослідження – ідентифікація рухомого об’єкта у просторі.

Предмет дослідження – програмний модуль ідентифікації рухомого об’єкта у просторі.

Мета роботи –розробити програмний модуль ідентифікації рухомого об’єкта у просторі.

Методи дослідження – застосування інструменту розробки *Qt*, мови програмування *C++,* та бібліотек *OpenCV* та *Teseract OCR* для створення програмного модуля розпізнавання автомобільних номерів.

Апробація результатів дослідження відбулася в вінницькій компанії «Токін». Компанією зроблено замовлення на програмний модуль розпізнавання автомобільних номерів що буде використовуватися з метою відстежити ввезення і вивезення інструментів та обладнання з території підприємства, та перевірки номера автомобіля для виключення можливість вивезення товару за підробленими або помилковим супровідним документам.

При використані програмного модулю отримані результати, які задовольнили очікування керівництва компанії.

Новизна отриманих результатів:

* реалізовано два варіанти розпізнавання автомобільного номера, перший являє собою розпізнавання у реальному часі (вимагає підключення до *IP*-камери), другий – розпізнавання з відеофайлу що знаходиться на машині користувача;
* реалізовано два методи розпізнавання верхньої границі автомобільного номера, для коректного розпізнавання автомобільних номерів.

Практичне значення одержаних результатів– програмний модуль ідентифікації рухомого об’єкта у просторі (модуль розпізнавання автомобільних номерів) надає змогу користувачам за допомогою персонального комп’ютера та *IP*-камери здійснювати розпізнавання автомобільних номерів з метою контролю в’їзду та виїзду транспортних засобів на певну територію.

# **РОЗДІЛ 1 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ**

Розпізнавання образів (об’єктів, сигналів, ситуацій, явищ або процесів) – завдання ідентифікації об’єкта або визначення будь-яких його властивостей за його зображенням (оптичне розпізнавання) або аудіозапису (акустичне розпізнавання) та іншими характеристиками [6].

Образ – класифікаційне угрупування в системі класифікації, що об’єднує (виділяє) певну групу об'єктів за деякою ознакою [6].

# **Методи ідентифікації**

Для оптичного розпізнавання образів можна застосувати наступні методи:

* метод порівняння шаблонів;
* метод кваріації;
* метод Віоли-Джонса;
* штучні нейронні мережі (вимагає або великої кількості прикладів завдання розпізнавання або спеціальної структури нейронної мережі, яка враховує специфіку даного завдання).
  + 1. Метод порівняння шаблонів

У методі порівняння шаблонів (*Temzona Matching*), для порівняння областей застосовуються найпростіші алгоритми на зразок попіксельного порівняння.

Їх основними принципами є виділення областей об’єкта на зображенні і їх порівняння для двох різних зображень.

Недоліки методу:

* ресурсоємність (зберігання проміжної інформації, еталонів   
  для порівняння, та самої обробка всієї наявної інформації);
* нестабільна робота в разі зміни освітлення;
* нестабільна робота при зміні масштабу або повороті зображення;
* нестабільна робота в разі зміни розміщення об’єкта;
* помилкові спрацьовування (у різних зображень можуть бути подібні деталі або області).

Кожна збіжна область збільшує міру схожості.

* + 1. Метод кваріації

Використовується коли потрібно просте порівняння двох досить схожих фрагментів зображення.

На прикладі даний метод можна представити таким чином: об’єкт, що містить шуканий зразок, пересувається по координатам *X*, *Y* по зображенню,   
в якому здійснюється пошук. Робота алгоритму вважається успішною, коли знаходиться така точка, де відмінність шуканого зразка від зображення, в якому відбувається пошук, досягає свого мінімуму [2].

Цей спосіб швидкий в реалізації і інтуїтивно зрозумілий.

Однак у нього є свої недоліки:

* низька швидкість роботи, особливо при обробці великих зображень;
* нестабільна робота при зміні освітлення;
* нестабільна робота при зміні масштабу або повороті зображення;
* нестабільна робота в разі, якщо частина зображення – фон що змінюється.
  + 1. Метод Виоли-Джонса

Метод Віоли-Джонса (англ. *Viola-Jones object detection*) – алгоритм, що дозволяє виявляти об’єкти на зображеннях в реальному часі.

Основні принципи, на яких базується даний метод, такі [8]:

* робота з об’єктами в інтегральному відображенні (для збільшення швидкості обчислення об’єктів);
* робота з ознаками Хаара, за допомогою яких відбувається пошук особи і його характеристик;
* використовується бустінг (від англ. *Boost* – поліпшення, посилення) з метою вибору найбільш відповідних вирішальних ознак для шуканого об’єкта на даній частині зображення;
* всі ознаки надходять на вхід класифікатора, який дає результат «істина» або «брехня»;
* використовуються каскади ознак для швидкого відкидання вікон, де не знайдено об’єкт.
  + 1. Нейронна мережа Хопфілда

Нейронна мережа Хопфілда  (*Hopfield neural network*) — це тип повнозв’язної, штучної нейронної мережі з симетричною матрицею зв’язків. У процесі роботи динаміка таких мереж сходиться до одного з положень рівноваги. Ці положення рівноваги є локальними мінімумами функціоналу, що називається енергія мережі.

Недоліки:

* запам’ятовувані образи не повинні бути сильно схожі;
* зображення не повинно бути зміщено або повернуто щодо його вихідного стану.

Алгоритм навчання цієї мережі істотно відрізняється від класичних алгоритмів навчання персептронов тим, що замість послідовного наближення до потрібного стану з обчисленням помилок все коефіцієнти вагової матриці розраховуються за однією формулою, за один цикл, після чого мережа відразу готова до роботи.

* + 1. Нейронні мережі Кохонена

Багатошарові нейронні мережі прямого поширення, для навчання яких, як правило, застосовується алгоритм зворотного поширення помилки, можна охарактеризувати як базові нейронні мережі. Вони успішно застосовуються при вирішенні великого кола завдань, але в деяких областях або є недоступними, або просто неефективні через дуже тривалий час навчання.

Для застосування нейронних мереж Кохонена в задачах класифікації потрібна деяка формалізація. Кожен об’єкт, який потрібно класифікувати, представляється у вигляді деякого вектора, що подається на вхід нейронної мережі. Кількість нейронів у вхідному шарі визначається кількістю компонентів цього вхідного вектора. Кількість же виходів визначається кількістю класів, тобто якщо всього існує *M* класів, то кількість нейронів у вихідному шарі теж буде *M*.

Таким чином, кожен нейрон у вихідному шарі «відповідає» за свій клас. Значення, які приймають нейрони в вихідному шарі, відображають наскільки вектор об’єкта що класифікується на вході близький, на думку нейронної мережі Кохонена, до того чи іншого класу. Чим більше «впевненість», що об’єкт належить до того чи іншого класу, тим більше значення приймає нейрон відповідного класу. Іноді застосовують спеціальну функцію активацію, яка робить суму виходів зі всіх нейронів рівними одиниці. У такому випадку кожен вихід можна трактувати, як ймовірність того, що об’єкт належить до даного класу.

У своїй найпростішій формі прошарок Кохонена функціонує за принципом «переможець забирає все», тобто для даного вхідного вектора один і тільки один нейрон Кохонена видає на виході логічну одиницю, всі інші видають нуль. Нейрони Кохонена можна сприймати як набір електричних лампочок, так що для будь-якого вхідного вектора загоряється одна з них [9].

# **Сучасні програмні засоби для ідентифікації об’єктів у просторі**

З кожним днем все більше і більше галузей використовують у своїй діяльності ідентифікацію тих чи інших об’єктів.

Розпізнавання об’єктів широко використовується у програмах доповненої реальності (*AR – augmented reality*), у віртуальній реальності (*VR –virtual realyty*) у медицині та у безпеці.

Оберемо об’єктом для ідентифікації – автомобільний номер.

* + 1. Застосування програм розпізнавання авто номерів

Сучасні програмні засоби розпізнавання автомобільних номерів справляються з наступними задачами.

Обмеження доступу. Програмні засоби для розпізнавання автомобільних номерів в цьому випадку використовуються для контролю над в’їздом транспорту на територію. Існує можливість гнучкого налаштовування рівнів доступу.

Організація платного доступу для автомобілів. Використання програмного забезпечення для подібних задач, дозволяє автоматизувати процес заїзду і оплати на платні парковки а також визначити час перебування транспортного засобу на ній.

Збір статистичних даних. Використовуються для отримання статистичних даних, що згодом можуть бути використані для оптимізації руху автотранспорту або у комерційних цілях.

* + 1. Програмно-апаратний комплекс *NumberOK*

Програмно-апаратний комплекс «*NumberOK*» призначений для розпізнавання автомобільних номерів і управління виконавчими пристроями.

Розробником проекту є компанія *FF-group* що знаходиться в Україні, Київ.

Як зазначає розробник модуль дозволяє з імовірністю до 95відсотків розпізнавати автомобільні номерні знаки. Програма призначена для розпізнавання номерів України, Російської Федерації, Республіки Білорусь, Ізраїлю та деяких європейських країн.

У компанії існує декілька продуктів в основі яких лежить розпізнавання автомобільних номерів, розглянемо їх.

*NumberOK Lite* (рис. 1.1) є найпростішою версією програмного забезпечення *SW NumberOK*. Вона містить детектор номерних знаків з функцією визначення напрямку руху автомобіля. Лінійка *NumberOK Lite* включає модифікації, що підтримують від 1 до 12 каналів. Програма забезпечить автоматизований доступ транспорту на території логістичних комплексів, транспортних підприємств.

Вартість такого програмного забезпечення становить від 8 904 гривень для одного каналу до 64 512 гривень для чотирьох.

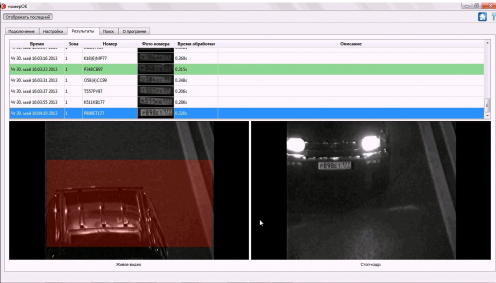


Рис. 1.1. Вікно програми *NumberOK Lite*

*NumberOK SMB* (рис. 1.2) розроблений для управління бізнес-процесами підприємств малого та середнього бізнесу через систему розпізнавання автомобільних номерів. Програмне забезпечення має ряд вбудованих   
бізнес-логік "КПП" (система контролю і управління доступом автотранспорту) і "Парковка" (підрахунок кількості автомобілів і розрахунок часу перебування машини на території), а також гнучкий модуль реакцій на події. *NumberOK SMB* визначає напрямки руху транспорту.

Система зручно і просто інтегрується з зовнішніми системами розпізнавання і клієнтськими додатками. Лінійка *NumberOK SMB* включає модифікації, що підтримують від 1 до 12 каналів. Програма забезпечить автоматизований доступ транспорту та управління бізнес-процесами логістичних комплексів, транспортних підприємств.

Вартість програмного забезпечення лежить в діапазоні від 12 432 гривень до   
276 242 гривень (в залежності від кількості точок доступу).

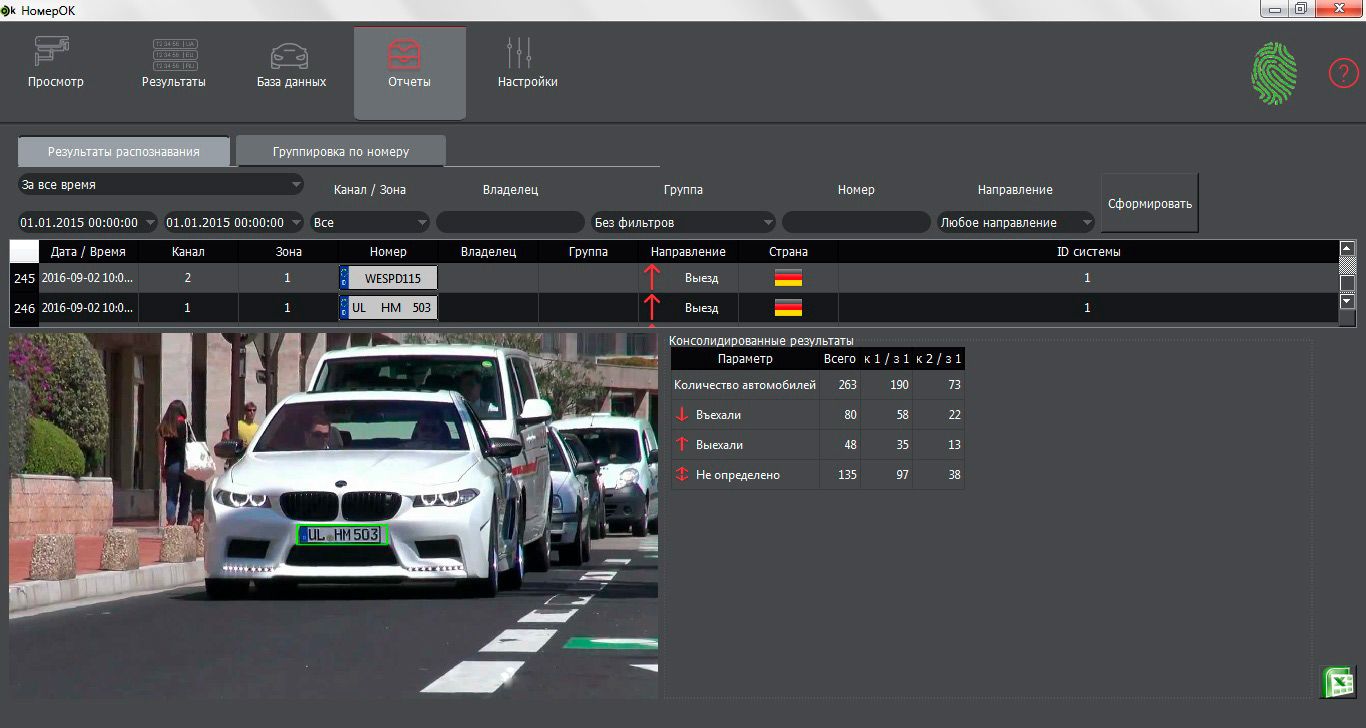


Рис. 1.2. Вікно програми *NumberOK SMB*

*NumberOK Enterprise* (рис. 1.3) розроблений для забезпечення цілодобового моніторингу ситуації на автошляхах, контролю руху транспорту і дотримання правил

дорожнього руху.

Програмне забезпечення має вбудовані модулі контролю правил дорожнього руху (ПДР): проїзд на червоне світло, перевищення дозволеної швидкості, контроль середньої швидкості руху автомобіля, виїзд на смугу громадського транспорту, фіксація автомобілів, що знаходяться в угоні або розшуку.

Отримані з *NumberOK Enterprise* дані можуть бути інтегровані в будь-яку систему для подальшого аналізу, або для виписки протоколу порушення.

Лінійка *NumberOK* *Enterprise* включає модифікації, що підтримують від 1 до 12 каналів.

Рекомендовані системні вимоги (на прикладі 4-х каналів розпізнавання):

* операційна система*: Microsoft Windows* 7/8/10;
* процесор: *Intel Core i*5-6400 та вище;
* оперативна пам’ять: *RAM* 8 *Гб*;
* відеокарта: *Radeon HD* 7750;
* запам’ятовуючий пристрій: *SSD* 120 *Гб*, *HDD* 4 *Тб*.

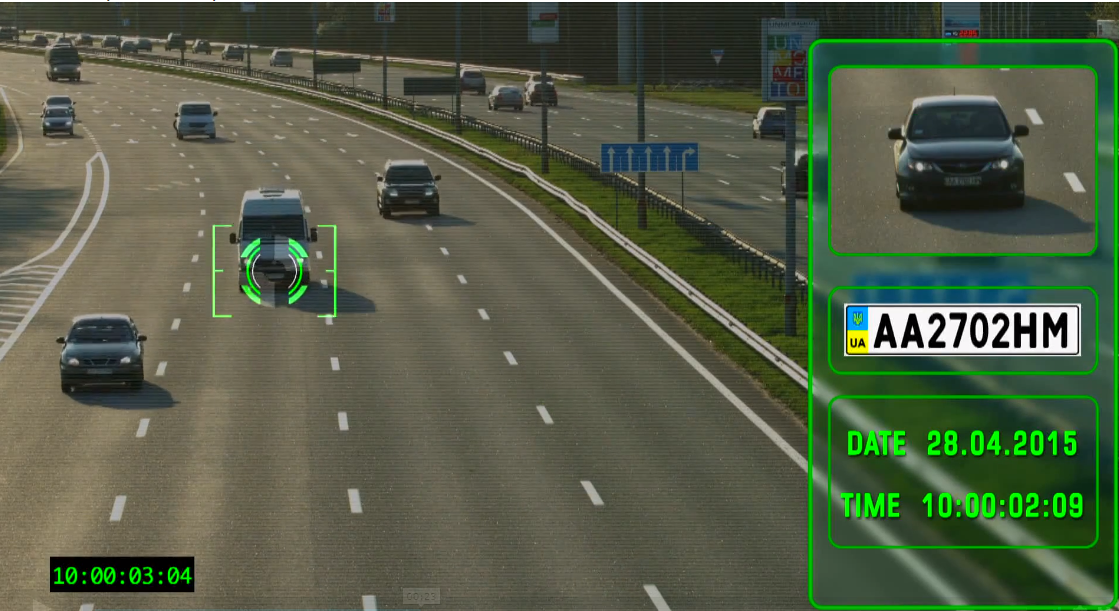


Рис. 1.3. Вікно програми *NumberOK Enterprise*

Вартість програмного забезпечення лежить в діапазоні від 20 345 гривень до   
304 293 гривень (в залежності від кількості точок доступу).

* + 1. Автоматизована система *CarGo Enterprise*

Автоматизована система управління контрольно-транспортним пунктом   
«*CarGo Enterprise*» (рис. 1.4) призначена для організації контролю проїзду автомобільного транспорту через точки контролю з організацією необхідного алгоритму проїзду і збору даних про автомобіль.

У число даних можуть входити такі:

* державний реєстраційний номер автомобіля (номерний знак);
* напрямок руху (в’їзд/виїзд);
* повноваження автомобіля (приналежність групі доступу);
* дата/час проходження точки контролю.

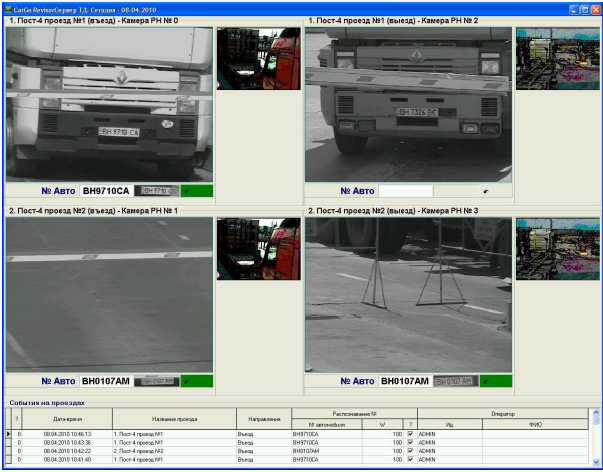


Рис. 1.4. Вікно програми *CarGo Enterprise*

Підтримуються основні формати однорядкових і дворядкових номерних знаків багатьох країн Європи, Азії, Південної та Північної Америки (як загальногромадянські, так і спеціалізовані – дипломатичні, військові, поліцейські, транзитні і т.п.), розпізнаються арабські цифри, а також латинські та кириличні символи алфавітів.

Модуль функціонує, в основному, в автоматичному режимі.

При появі автомобіля на в’їзді розпізнає державний номер автомобіля і виконує відкриття шлагбауму для в’їзду на територію, якщо це вимагається правилами функціонування:

* якщо автомобіль належить групі доступу, якої дозволений проїзд через дану точку контролю в даний час;
* якщо автомобіль належить до категорії *VIP*;
* якщо, при активованому режимі доступу по заявкам, автомобіль належить до категорії автомобілів партнерів або гостей і на нього в даний час діє дозволена заявка.

При появі автомобіля на виїзді розпізнає державний номер автомобіля і виконує відкриття шлагбауму для виїзду з території, якщо це передбачено правилами функціонування:

* якщо автомобіль належить групі доступу, якої дозволений проїзд через дану точку контролю в даний час;
* якщо, при активованому режимі доступу по заявкам, автомобіль належить до категорії автомобілів партнерів або гостей і в’їжджав на територію по заявці.

Програмою допускається одночасне керування до 16 (шістнадцять) односпрямованими точками доступу або, з урахуванням того, що кожна точка доступу може використовуватися в двох напрямках – для в’їзду і для виїзду – до 8 (восьми) двонапрямлених проїздів.

Вартість програмного забезпечення лежить в діапазоні від 6 000 гривень до   
36 000 гривень (в залежності від кількості точок доступу).

Системні вимоги (для 4-х каналів розпізнавання):

1. операційна система*: Microsoft Windows* 7/8;
2. процесор: *Intel Core i*3 та вище;
3. оперативна пам’ять: *RAM* 4 *Гб*;
4. відеокарта: *Radeon HD* 7750;
5. запам’ятовуючий пристрій: *HDD* 4 *Тб*.

Розробник *Intteks*, Україна, місто Київ. Точність розпізнавання 94 відсотки.

* + 1. Система *CVS*-Авто

Система *CVS*-Авто (рис. 1.5) розроблена компанією «Нові Технології» (Росія, Московська область), є клієнтським додатком для основної програми системи відеоспостереження *CVSCenter* і призначена для автоматичного розпізнавання і фіксації автомобільних номерів, які потрапили в поле зору відеокамери.

Для роботи модуля потрібно встановлене програмне забезпечення сервера відеоспостереження. В разі необхідності весь комплект програм може встановлюватися на один комп’ютер.



Рис. 1.5. Вікно програми *CVS*-Авто

Функціональні можливості системи:

* подача сигналів на блокування /розблокування виконавчі пристроїв;
* формування бази даних розпізнаних державних реєстраційних знаків із зазначенням номера транспортного засобу, дати і часу його фіксації, напрямки руху, фотографії транспортного засобу, типу реєстраційного знака;
* автоматичний пошук інформації за нижченаведеними параметрами і їх комбінацій: по частково або повністю розпізнаного реєстраційному знаку; за датою проїзду; з тимчасового діапазону; у напрямку; за типом реєстраційного знака;
* автоматичний збір за номером транспортного засобу всієї інформації що зберігається в базі – фотографія автомобіля, дані про власника і т.д.

У модулі реалізована можливість ідентифікації транспортного засобу – при внесенні номера автомобіля в базу даних додається його фото (використовується лише один знімок). Додаткова текстова інформація про автомобіль може містити: марку, модель, колір автомобіля, ім’я власника, контактну інформацію.

Точність розпізнавання 96 відсотків.

Системні вимоги (для 4-х каналів розпізнавання):

* операційна система*: Microsoft Windows* 7/8/10;
* процесор: *Intel Core i*5-6400 та вище;
* оперативна пам’ять: *RAM* 8 *Гб*;
* відеокарта: *Radeon HD* 7750;
* запам’ятовуючий пристрій: *SSD* 120 *Гб*, *HDD* 4 *Тб*.

Вартість програмного забезпечення становить близько 6278 гривень.

* + 1. *iPera EX-LPR*

Модуль розпізнавання автомобільних номерів *iPera EX-LPR* (рис. 1.6), розроблений спільно з компанією *iPera*, дозволяє сформувати базу даних по всім транспортним засобам, які потрапили в область моніторингу системи відеоспостереження.

Система розпізнавання автомобільних номерів *EX-LPR*, переваги та особливості:

* + ймовірність розпізнавання автономера – 96 відсотків в реальних умовах (при дотриманні рекомендацій по встановленню та налагодженню камер);
  + підтримка номерних знаків наступних країн: Росія, Білорусь, Казахстан, Йорданія, ОАЕ;
  + визначення наявності автотранспортного номера в поле зору камери і його розпізнавання;
  + збереження розпізнаного автомобільного номера з фіксованою датою і часом;
  + можливість присвоєння ключових номерів для зв’язку з подією;
  + можливість запису відеоролика;
  + пошук відеозаписів за текстовим даними або їх фрагментами, а також за часом реєстрації;



Рис. 1.6. Вікно програми *iPera EX-LPR*

Додатково додаток *EX-LPR* забезпечує цілодобове спостереження в режимі реального часу і запис того, що відбувається в зоні парковки.

Системні вимоги :

* операційна система*: Microsoft Windows* 7/8/10;
* процесор: *Intel Core i*3 та вище;
* оперативна пам’ять: *RAM* 4 *Гб*;
* відеокарта: *NVidia*;
* запам’ятовуючий пристрій: *HDD* 4 *Тб*.

Вартість програмного забезпечення становить близько 5278 гривень.

* + 1. Програма *IPVideoRecord*

Програмне забезпечення *IPVideoRecord* (рис. 1.7) призначене для моніторингу та запису відео і аудіоінформації, що надходить з *IP*-камер відеоспостереження.

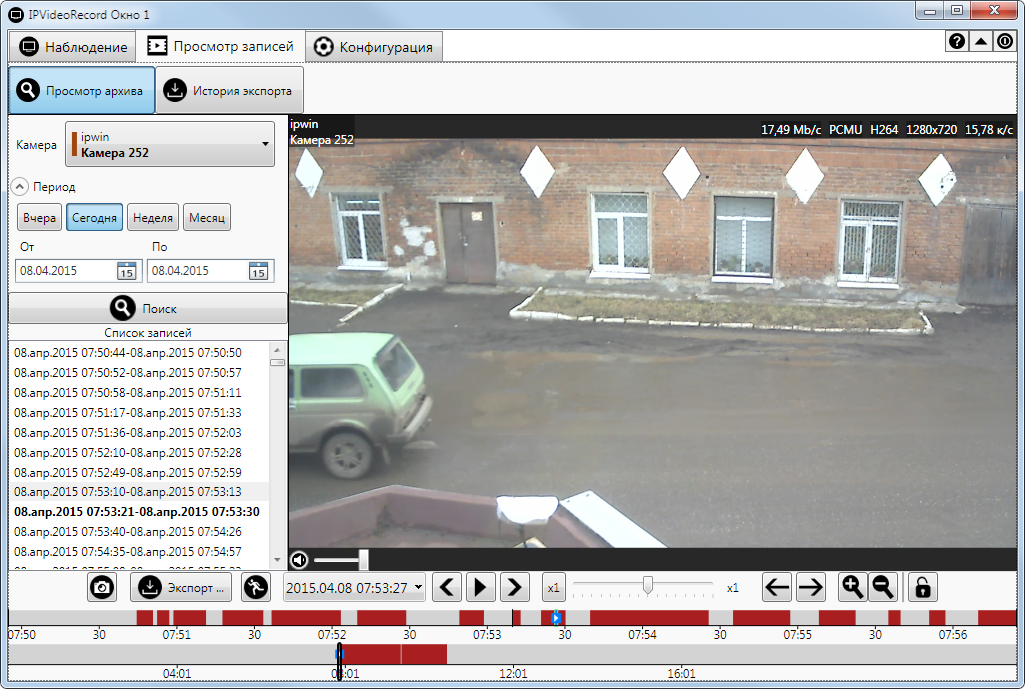


Рис. 1.7. Вікно програми *IPVideoRecord*

Вартість програмного забезпечення становить близько 6479 гривень.

Програма складається з серверного та клієнтського додатків.

Сервер призначений для отримання відео і аудіо з *IP*-камер, записи архіву, роздачі даних клієнта.

Клієнт призначений для спостереження в режимі реального часу і перегляду записаного архіву з камер, а також для адміністрування сервера, в тому числі віддалено як по локальній мережі, так і через інтернет.

Системні вимоги :

* операційна система*: Microsoft Windows* 7;
* процесор: *Intel Core i3-2100* та вище;
* оперативна пам’ять: *RAM* 4 *Гб*;
* відеокарта: *NVidia*;
* запам’ятовуючий пристрій: *HDD* 4 *Тб*.

Основні можливості:

* підтримка кодеків: *H.265, H.264, MJPEG*;
* протокол передачі відеопотоку: *RTSP over UDP, RTSP over TCP*;
* запис відео і аудіо з *IP*-камер;
* віддалене відеоспостереження і аудіопрослушування в режимі реального часу;
* підтримка другого потоку з камери;
* віддалений перегляд і прослуховування записаного відео з архіву;
* повнофункціональний віддалене адміністрування;
* розмежування прав користувачів і груп;
* запуск/зупинка запису за розкладом;
* запуск/зупинка по детектору руху;
* експорт записів з архіву в стандартний формат (\* .*mp4*);
* цифрове збільшення при перегляді відео з камери або з архіву;
* журнал подій;
* підтримка *IP*-камер різних виробників, інтеграція з вашої камерою в короткі терміни;
* пошук і підключення камер з підтримкою *ONVIF* і *UPnP* в локальній мережі.
  + 1. Система розпізнавання автомобільних номерів *Hunter LPR*

Система розпізнавання автомобільних номерів *Hunter LPR* (рис. 1.8) орієнтована на розпізнавання автомобільних номерів як в режимі реального часу так і при перегляді записаного архіву.

Клієнтська частина програмного забезпечення *Hunter LPR Client* створена для контролю службою безпеки проїзду автотранспорту, безпосередньо на пропускному пункті. Інформація про розпізнаний номері автомобіля і його фото, відправляються на віддалений клієнт для верифікації авто і видачі дозволу на проїзд співробітником служби безпеки.



Рис. 1.8. Вікно програми *Hunter LPR*

Крім розпізнавання номера, система в автоматичному режимі реєструє напрямок руху автомобіля, його фото, фото номера, час проїзду через контрольований ділянку дороги.

Таким чином, співробітник служби безпеки може приймати рішення на пропуск автомобіля, при появі відповідної інформації про автомобіль з сервера бази даних системи розпізнавання.

Крім пропуску автомобіля, оператором може бути введена додаткова інформація в базу даних про автомобіль, його водія, співробітника відповідальному

за пропуск.

*Hunter LPR Client* адаптований для роботи на планшетному ПК, у разі якщо співробітник служби безпеки проводить контроль проїзду авто безпосередньо біля автомобіля.

Системні вимоги :

* операційна система*: Microsoft Windows* *XP/*7;
* процесор: *Intel Core i3-2100* та вище;
* оперативна пам’ять: *RAM* 4 *Гб*;
* відеокарта: *NVidia*;
* запам’ятовуючий пристрій: *HDD* 2 *Тб*.

Вартість програмного забезпечення становить близько 6000 гривень.

## **Постановка завдання дослідження**

Метою даного дослідження є розробка програмного модуля для ідентифікації рухомого об’єкта у просторі. З оглядом на те що існує велика кількість задач розпізнавання об’єктів що дуже сильно варіюються, як мета дослідження була обрана розробка модуля ідентифікації (розпізнавання) автомобільного номера.

Програмний модуль для розпізнавання автомобільних номерів буде отримувати на вхід зображення з *IP*-камери (розпізнавання у реальному часі) або з відеофайлу (обробка вже існуючого матеріалу). На виході будемо отримувати розпізнаний автомобільний номер.

Розроблений програмний модуль надасть можливість в подальшому інтегрувати його у систему автоматизації на зразок автоматизованої парковки, або системи контролю доступу. У своєму першочерговому вигляді надає можливість розпізнавати Українські номерні знаки за умов часткового забруднення, недостатньої освітленості, нахилі номера.

У відповідності до сформульованої мети у роботі поставлено наступні завдання:

* проаналізувати сучасні засоби розпізнавання автомобільних номерів;
  + розробити структуру програмного модулю;
* модифікувати алгоритм ідентифікації рухомого об’єкта;
  + програмно реалізувати модуль для розпізнавання автомобільних номерів як з *IP*-камери, так і з завантаженого файлу та протестувати роботу модулю.

## **Висновки до розділу**

У першому розділі розглянуто сучасні засоби для ідентифікації автомобільних номерів.

З огляду існуючого програмного забезпечення можна зробити висновок що недоліками існуючих засобів є:

– занадто висока вартість;

– високі апаратні вимоги;

– залежність від операційної системи *Microsoft Windows*.

Також розглянуто методи для ідентифікації об’єкта у просторі. Серед методів, необхідно обрати один, що буде використано для розробки програмного модулю. Беручи до уваги те, що метод Віоли-Джонса створений спеціально для вирішення задач розпізнавання об’єктів у реальному часі, виберемо його для вирішення поставлених задач.

В кінцевому результаті огляду предметної області сформовано завдання дослідження.

# **РОЗДІЛ 2 СТРУКТУРА ПРОГРАМНОГО МОДУЛЯ ІДЕНТИФІКАЦІЇ РУХОМОГО ОБ’ЄКТА У ПРОСТОРІ**

Для розробки модуля ідентифікації рухомого об’єкта у просторі необхідно розглянути метод за яким буде здійснюватися ідентифікація та побудувати структурну схему модуля.

## **2.1. Метод Віоли-Джонса**

Метод Віоли-Джонса виконує пошук об’єктів в режимі реального часу.

В основу метода лежать:

* інтегральне представлення зображення по ознакам Харра;
* побудова класифікатору на основі алгоритму адаптивного бустінга;
* комбінування класифікаторів в каскадну структуру.

Розглянемо це більш детально.

2.1.1. Інтегральне представлення зображення за ознаками Харра

Інтегральне представлення зображення – це матриця, однакова за розмірами з вхідним зображенням. У кожному елементі матриці зберігається сума інтенсивності всіх пікселів, що знаходяться лівіше і вище даного елемента – правого нижнього кута прямокутної області (0,0) до (*х, у*).

Елементи матриці L можна розрахувати за формулою:

де *I (i, j)* – яскравість пікселя вихідного зображення.

Розрахунок значень елементів матриці проходить за час, пропорційне числу пікселів в оригінальному документі, тому інтегральне зображення прораховується за один прохід.

Елементи матриці розраховуються за формулою:

де *x,y* – координати.

На етапі виявлення об’єкта в методі Віоли-Джонса використовується вікно певного розміру, яке рухається по зображенню. Для кожної області зображення, над якою проходить вікно, розраховується ознака Хаара, за допомогою якого відбувається пошук потрібного об’єкта.

Якщо задані ознаки *f1, ..., fn*, то вектор ознак *x = (f1 (x), ..., fn (x))* називається ознаковим описом об’єкта *x*. Ознаковий опис допустимо зіставляти з самими об’єктами. При цьому множину *X = Df1 \* ... \* Dfn* називають простором ознак.

Ознака Хаара обчислюється по суміжним прямокутним областям. У стандартному методі Віоли-Джонса використовуються прямокутні примітиви, зображені на рис. 2.1.

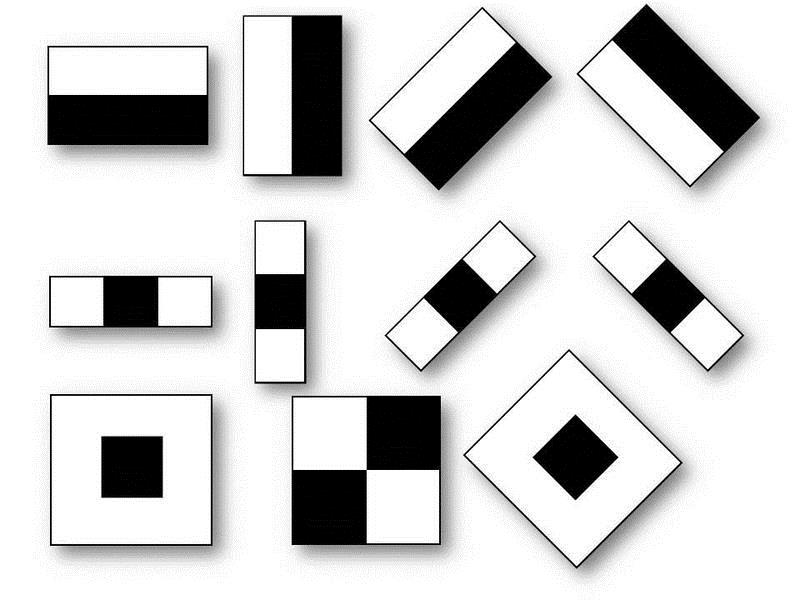


Рис. 2.1. Примітиви Харра

Ознаки поділяються на такі типи:

* бінарна ознака, *Df*  = {0,1};
* номінальна ознака: *Df* – кінцева множина;
* порядкова ознака: *Df* – кінцева впорядкована множина;
* кількісна ознака: *Df* – множина дійсних чисел.

Обчислюваним значенням F ознаки Хаара буде:

де *X* – сума значень яскравості точок, які закриваються світлою частиною примітиву;

*Y* – сума значень яскравості точок, які закриваються темною частиною.

Використання ознак Хаара дає значення перепаду яскравості по осі *X* і *Y* відповідно для кожної точки.

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, що слугує для швидкого відкидання вікон, де не знайдено необхідний об’єкт, і видачі результату «істина» або «брехня» щодо знаходження об’єкта.

2.1.2. Побудова класифікатору на основі алгоритму адаптивного бустінга

Класифікатор будується на основі алгоритму бустінга для вибору найсуттєвіших ознак для шуканого об’єкта на даній частині зображення. У загальному визначенні бустінг – це совокупність 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 бустінг 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 бустінга на кожній ітерації формується простий класифікатор виду:

де *pj* – напрямок знаку нерівності;

*θj* – значення порога;

*fj* (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.

## **2.2. Пошук автомобільного номера на цифровому зображенні**

Для пошуку об’єкта на цифровому зображенні використовується навчений класифікатор, представлений в форматі *xml*.

Класифікатор (рис. 2.2) формується на примітивах Хаара.

Згідно заданої структури *maxWeakCount* це 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, четверте – порогове значення слабкого класифікатора.

Якщо значення ознаки Хаара менше порога слабкого класифікатора, вибирається перше значення *leafValues*, якщо більше – друге.

На основі цього базису будується каскад класифікаторів, які приймають рішення про те, розпізнаний об’єкт на зображенні чи ні. Наявність або відсутність предмета в вікні визначається різницею між значенням ознаки і порогом, отриманим в результаті навчання.

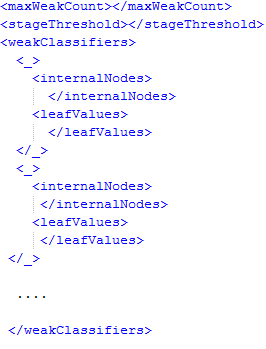


Рис. 2.2. Структура класифікатора

## **2.3. Розпізнавання автомобільного номера**

Для розпізнавання автомобільного номера необхідно підготувати вхідне зображення.

2.3.1. Алгоритм нормалізації кута нахилу

Припустимо що номер може мати діапазон нахилу від мінус 10 до плюс 10 градусів. З кожним кадром необхідно працювати окремо.

Побудуємо схему алгоритму пошуку кута нахилу (рис. 2.3).



Рис.2.3. Схема алгоритму нормалізації кута нахилу

Для кожного кадру розраховується нижня границя зображення. Після роботи алгоритму обирається кут для якого буда розрахована верхня границя.

2.3.2. Алгоритм пошуку нижньої границі автомобільного номера

Пошук нижньої границі автомобільного номера будується на аналізі гістограми яскравості.

Для початку зображення обробляється за допомогою певного правила, що задає умову розділення пікселів на чорні та білі. Ця операція називається бінаризацією зображення. Після бінаризації ми рахуємо кількість чорних пікселів у кожному стовпці і на основі отриманих даних будуємо гістограму зображення.

Принцип побудови гістограми зображання зображено на рис. 2.4.

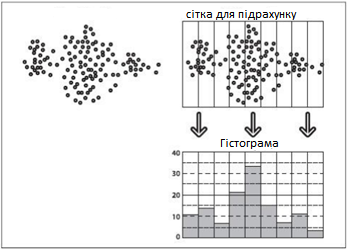


Рис. 2.4. Принцип побудови гістограми

Схему алгоритму пошуку нижньої границі автомобільного номера приведено на рис. 2.5.



Рис. 2.5. Схема алгоритму пошуку нижньої границі

2.3.3. Алгоритм пошуку верхньої границі автомобільного номера

Схему алгоритму пошуку верхньої границі номера приведено на рис. 2.6.



Рис. 2.6. Схема алгоритму пошуку верхньої границі автомобільного номера

Гіпотеза з використанням гітограми яскравості для знаходження верхньої границі працює лише 50 відсотків випадків, що не є допустимим для нашого модулю.  
З оглядом на це прийнято рішення натренувати каскад Харра на кожну літеру проаналізувати верхні границі літер, в наслідок чого ми зможемо отримати верхню границю автомобільного номера.

В деяких випадках, алгоритм з використанням каскада Харра, може не дати результату, що характерно для зображень з дуже низьким розширенням. Для таких зображень як альтернативний алгоритм використовується пошук границі з використанням гістограми яскравості.

2.3.4. Алгоритм пошуку бічних границь автомобільного номера

Після виконання двох попередніх алгоритмів, зображення автомобільного номера обрізано по верхній і нижній границях, тепер залишається визначити бічні границі. Для цього будемо використовувати метод побудови гістограм яскравості.

Необхідно прийняти до уваги що якщо колір автомобіля білий, то після бінаризації зображення по краям буде білий колір, а якщо чорний, то краї будуть чорні, а сам номер білий (рис. 2.7). Отже необхідно використовувати по дві гіпотези на кожен бік, одна для пошуку границь на білій машині, інша для пошуку границь на чорній машині. Результатом обирається та гіпотеза що має результат ближчій к центру автомобільного номера.



Рис. 2.7. Бінаризовані зображення чорного та білого автомобіля

Для покращення якості пошуку бічних границь, перед побудовою гістограм яскравості необхідно провести розмивання (операція звуження) та розтягнення (операція розширення).

Схему алгоритму пошуку бічних границь приведено на Рис.2.8.



Рис. 2.8. Схема алгоритму пошуку бічних границь автомобільного номера

2.3.5. Сегментація автомобільного номера на символи

Схему алгоритму сегментації приведено на рис. 2.9.



Рис. 2.9. Схема алгоритму сегментації автомобільного номера

Після обрізання зображення по границям, необхідно виконати сегментацію автомобільного номера на символи. Цей крок необхідний для подальшого розпізнавання символів що зображені на автомобільному номері.

Спершу необхідно перевести зображення в чорно-білий формат і бінаризувати його. Після бінаризації необхідно обробити зображення фільтром середніх часто, щоб прибрати перешкоди. Наступним кроком буде застосування розмиття зображення методом Гаусса. Розмиття необхідно для сгладжування країв символів, що підвищує вірогідність розпізнавання символів.

Наступним кроком буде використання детектору границь Кенні, що є розповсюдженим методом для виділення границь. Алгоритм виглядає наступним чином:

* прибрати шум і зайві деталі з зображення;
* розрахувати градієнт зображення;
* зробити краї тонкими;
* зв’язати краї в контури.

Кінцевим етапом пошуку зв’язаних областей є порівняння знайденої області з шаблоном символу автомобільного номера і відкиданням некоректних областей. Цей спосіб дозволяє вірно визначити символи автомобільного номера на зображенні з підвищеною кількістю завад.

Після того як знайдено контури усіх символів (для українського номера 8) алгоритм завершує свою роботу та повертає флаг успіху або невдачі сегментації.

2.3.6. Розпізнавання символів автомобільного номера

Заключним етапом роботи модулю є розпізнавання символів автомобільного номера. Для розпізнавання символів прийнято рішення використовувати готову бібліотеку *Tesseract OCR*.

Алгоритм має пройти по всім символам в контейнері та після ряду перетворень (перетворення в ч\б формат, бінаризація, приведення до потрібного розміру, фільтр Гауса) та порівняння з шаблоном, видати на вихід результат «істина» або «брехня» в залежності від того, вдалось розпізнати автомобільний номер чи ні, а також контейнер символів що були розпізнані.

Схему алгоритму розпізнавання символів з використанням *Tesseract OCR* наведено на рис. 2.10.



Рис. 2.10. Схема алгоритму розпізнавання символів

## 2.3.8. Загальна схема алгоритму роботи модуля ідентифікації рухомого об’єкта у просторі

Враховуючи алгоритми описані вище, схема загального алгоритму буде виглядати наступним чином (рис. 2.11).



Рис. 2.11. Схема загального алгоритму роботи модуля ідентифікації рухомого об’єкта у просторі

Схема алгоритму розпізнавання автомобільного номера (рис. 2.12).



Рис. 2.12. Схема алгоритму роботи модуля розпізнавання автомобільного номера

## 2.3.9. Побудова структурної схеми

Побудуємо структурну схему програмного модуля (рис. 2.13).



Рис. 2.13. Структурна схема програмного модулю ідентифікації автомобільного номера

# **2.4. Висновки до розділу**

В другому розділі розглянуто метод Віоли-Джонса та його складові.

Також побудовано схеми наступних алгоритмів для програмного модулю ідентифікації автомобільного номера:

* алгоритм нормалізації кута нахилу;
* алгоритм пошуку нижньої границі автомобільного номера;
* алгоритм пошуку верхньої границі автомобільного номера;
* алгоритм пошуку бічних границь автомобільного номера;
* алгоритм сегментації автомобільного номера на символи;
* алгоритм розпізнавання символів;
* алгоритм роботи модуля розпізнавання автомобільного номера;
* загальний алгоритм програмного модулю.

Після чого побудовано структурну схему програмного модуля розпізнавання автомобільних номерів.

# **РОЗДІЛ 3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ МОДУЛЯ ІДЕНТИФІКАЦІЇ РУХОМОГО ОБ’ЄКТА У ПРОСТОРІ**

## **3.1. Огляд обраних інструментів розробки**

3.1.1. Інструмент розробки *Qt*

Для розробки програмного модулю ідентифікації рухомого об’єкта у просторі обрано *Qt* (рис. 3.1).

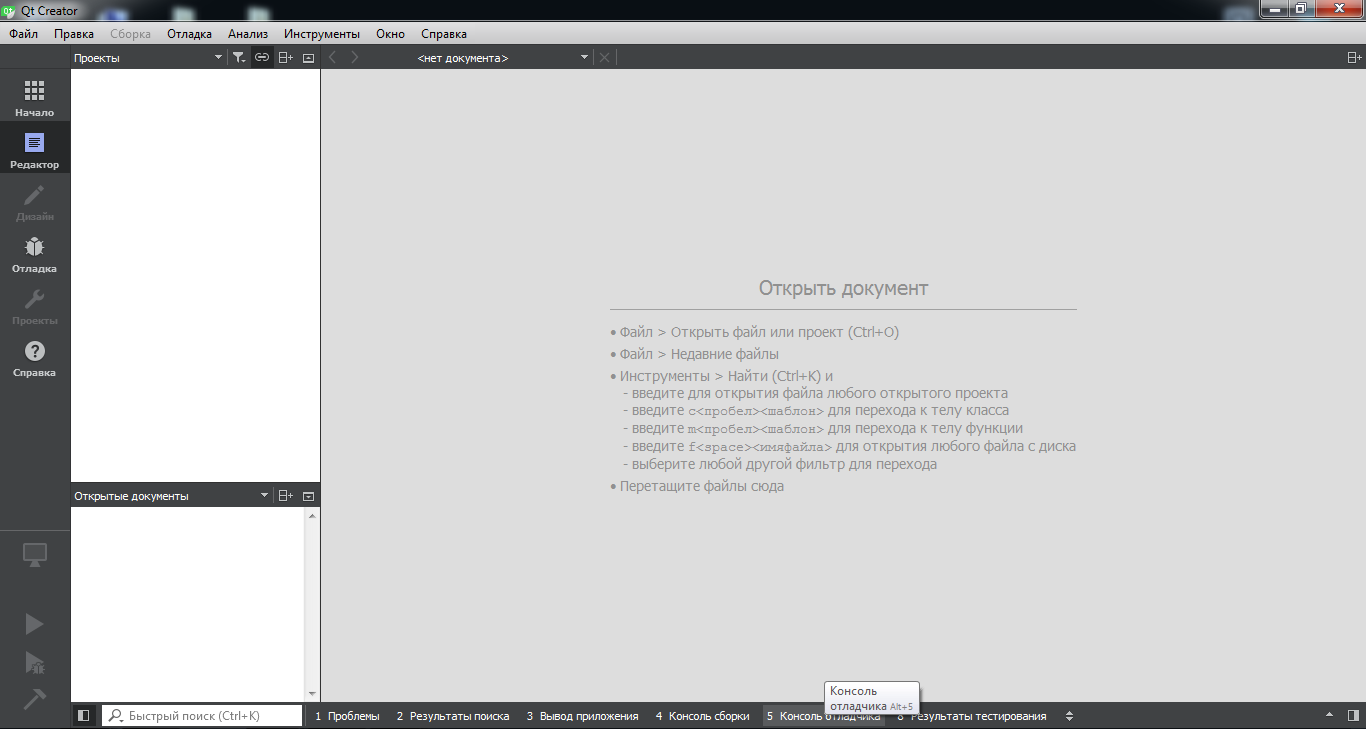


Рис. 3.1. Стартове вікно *QtCreator*

Цей інстументарій підтримується великою кількістю операційних систем, таких   
як: *Mac OS X*, *Linux*, *Solaris, Microsoft Windows*, *AIX*, *Irix*, *NetBSD*, *OpenBSD*, *HP-UX*, *FreeBSD* та інших *UNIX* систем. Окрім комп’ютерних операційних систем *Qt* підтримує *iOS*, *Android*, *Windows Phone*, *Windows RT* та *BlackBarry*. Це дозволяє з необхідності побудувати одне й те саме програмне забезпечення як мобільний додаток, так і як програмне забезпечення комп’ютерів. *Qt* використовує інтерфейс *API* низького рівня, що дозволяє додаткам працювати так само ефективно, як і додаткам, що були розроблені спеціально для певної платформи [10].

Сьогодні безліч компаній використовує *Qt* для написання своїх проектів як з графічним інтерфейсом так і без, адже є об’єктно-орієнованою бібліотекою. Цей інструмент розробки використовує відмінну від *Visual Studio* (модель зворотніх викликів) концепцію міжоб’єктних комунікацій.

Спершу бібліотека створювалась для мови програмування *C++,* проте згодом була розширена модулями роботи з іншими популярними мовами програмування (*C#, Python, Java).*

Написання програм у *QtCreator* включає використання його бібліотек. Розробниками середовища заявлено існування понад п’ятисота класів, що орієнтовані на використання як модулями для певних платформ та мов, так і кросплатформеними модулями. Усі модулі є окремими бібліотеками що забезпечую можливість вибіркового підключення, але мають деякі залежності один від одного [10].

Найбільш часто використовуються модулі приведені таблиці 3.1.

Таблиця 3.1

Модулі *Qt*

|  |  |  |
| --- | --- | --- |
| Бібліотека | Позначення в проектному файлі | Призначення |
| *QtCore* | *Core* | Основоположний модуль, що складається з класів, не пов’язаних з графічним інтерфейсом |
| *QtGui* | *Gui* | Модуль базових класів для програмування графічного інтерфейсу |
| *QtWidgets* | *Widgets* | Модуль, що доповнює *QtGui* елементами для графічного інтерфейсу у вигляді віджетів на *С++* |
| *QtQuick* | *Quick* | Модуль, що містить описовий фреймворк для швидкого створення графічного інтерфейсу |
| *Qt Quick Controls* | *Quick* | Модуль забезпечує легкі типи *QML* для створення ефективних інтерфейсів користувача для настільних, вбудованих та мобільних пристроїв. Ці типи використовують просту архітектуру стилів і дуже ефективні |

Закінчення таблиці 3.1.

|  |  |  |
| --- | --- | --- |
| *QtNetwork* | *Network* | Модуль для програмування мережі |
| *Qt Quick Dialogs* | *Quick* | Типи створення та взаємодії із системними діалогами із додатку *Qt Quick.* |
| *Qt Quick Test* | *Quick* | Тестова основа для програм *QML*, де тестові випадки записуються як функції *JavaScript* |
| *Qt Quick Layouts* | *Quick* | Елементи, які використовуються для упорядкування елементів на основі *Qt Quick* 2 в інтерфейсі користувача |
| *QtQML* | *Qml* | Модуль, що містить двигун для мови *QML* та *JavaScript*. Підтримка *QML* для розробки динамічних візуальних інтерфейсів |
| *QtOpenGL* | *Opengl* | Модуль для програмування графіки *OpenGL* |
| *QtSql* | *Sql* | Модуль для програмування баз даних |
| *QtSvg* | *Svg* | Модуль для роботи с *SVG* (*Scalable vector Graphics*,масштабована векторна графіка) |
| *QtXml* | *Xml* | Модуль підтримки *XML*,класи що відносяться до *SAX* та *DOM* |
| *QtXmlPatterns* | *Xmlpatterns* | Модуль підтримки *XPath*, *XQuerty*, *XSLT* та *XmlShemaValidator* |
| *QtScript* | *Script* | Модуль підтримки мови сценаріїв |
| *QtSkriptTools* | *Scripttools* | Модуль додаткових можливостей підтримки мови сценаріїв |
| *QtMultimedia* | *Multimedia* | Модуль мультимедіа |
| *QtMultimedia- Widgets* | *Multimediawidgets* | Модуль с віджетами для *QtMultimedia* |

Основними трьома модулями бібліотеки є *QtCore*, *QtGui*, та *QtWidgets*. Розглянемо модулі *Qt* більш детально.

Модуль *QtCore.* Основний модуль, що є базовим по відношенню до інших. Без цього модуля не можлива реалізація як консольних програм так і програм з графічним інтерфейсом (Включає більш ніж 200 класів).

Модуль *QtGui.* Цей модуль надає можливість створювати графічний інтерфейс, має класи *OpenGL* і *OpenGLES*. Також включає клас *QWindow*, який відповідає за обробку та отримання подій на зразок зміни розміру вікна, відстежування курсору, малювання.

Модуль *QtWidgets.* Цей модуль містить в собі класи віджетів що являють собою реалізацію віконної системи (*QMainWindow, QDialog* та інші).

Також *QtCreator* надає зручний інтерфейс для створення графічного інтерфейсу, за це відповідає вбудований елемент *QtDesigner.* Вигляд цього інструменту представлено на рис.3.2.

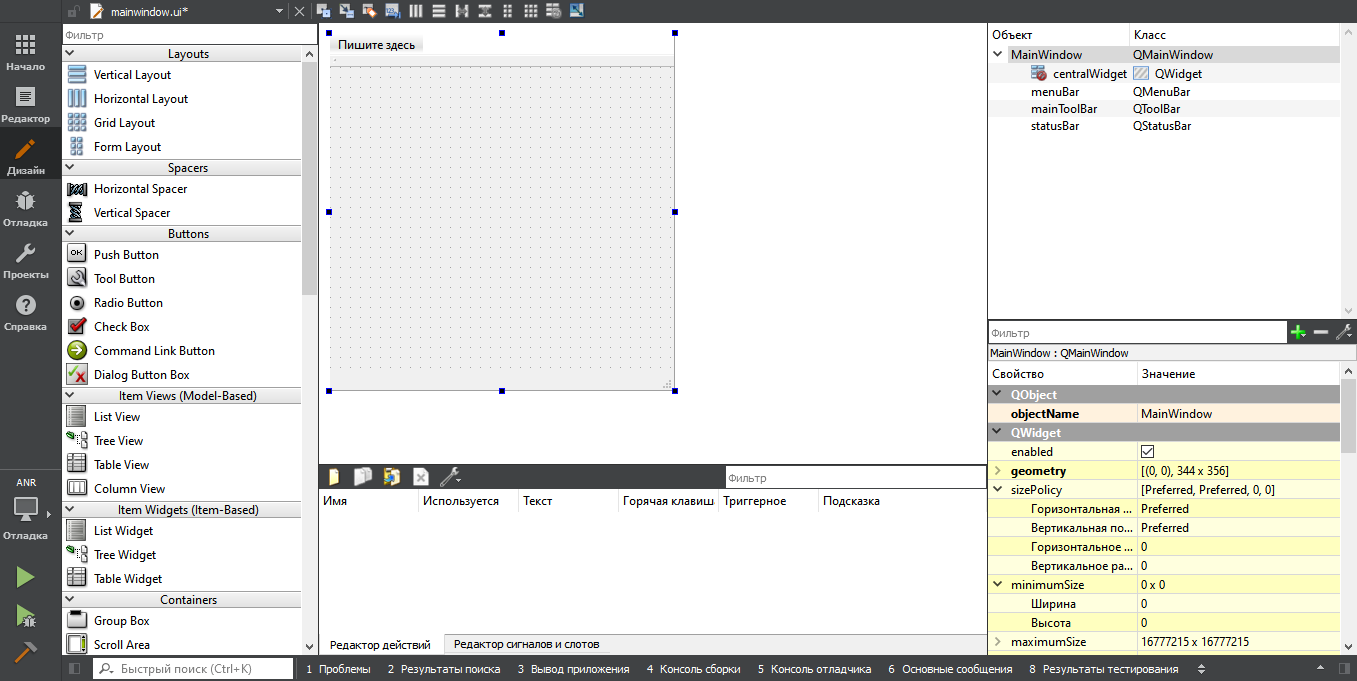


Рис. 3.2. Інтерфейс Q*tDesigner*

3.1.2. Мова програмування *С++*

*C ++ –* мова програмування високого рівня з підтримкою декількох парадигм програмування: об’єктно-орієнтованої, узагальненої та процедурної. Мова була розроблена Б’ярном Страуструпом (англ. *Bjarne Stroustrup*) в 1979 році та спочатку отримала назву «С з класами». Згодом була перейменована в *C++* у 1983 р. Базується на мові *С* [11].

Цю мову використовують для розробки комп’ютерних ігор, розробки програмного забезпечення, написання драйверів, створення серверних та клієнтських програм, а також для системного програмування.

Вибір цієї мови програмування для написання модуля пошуку шляху маршрутизації ґрунтується на її кросплатформеності та ефективності.

Основними перевагами цієї мови є кросплатформеність (існує велика кількість інструментів розробки для цієї мови), ефективність (дозволяє вільно реалізовувати безліч концептів), підтримка (можливості мови постійно розширюються за рахунок створення нових бібліотек і оновленню вже існуючих, на зразок *STL*).

Мова *C ++* містить два види бібліотек. У першій бібліотеці зберігаються стандартні універсальні функції, які не належать ні одного класу. Ця бібліотека успадкована у мови *С*. Друга бібліотека містить класи і є об’єктно-орієнтованої [11].

Головним недоліком мови є її складність порівняно з іншими високорівневими мовами програмування.

3.1.3. Стандартна бібліотека шаблонів *STL*

Стандартна бібліотека шаблонів (*STL*) – це набір шаблонів класів C ++ для забезпечення загальних структур даних та функцій програмування, таких як списки, стеки, масиви тощо. Це бібліотека контейнерних класів, алгоритмів та ітераторів. Це узагальнена бібліотека, і тому її компоненти параметризовані.

Контейнер це спосіб зберігання даних, незалежно від того, складаються вони з вбудованих типів, таких як *int* та *float*, або об’єктів класу. *STL* надає сім основних видів контейнерів, а також три додаткові. Крім того, можливе стоврення власних контейнерів.

Контейнери в *STL* поділяються на дві основні категорії: послідовні та асоціативні.

Послідовні контейнери – це вектор, список (*vector, list, deque*) Ассоціативні   
контейнери – множина, мультмножина, мапа та мультімапа (*set, multiset, map, multimap*). Також, декілька спеціалізованих контейнерів отримують з послідовних контейнерів. Це стек, черга та пріоритетна черга *(stack, queue, priority queue).*

Алгоритми в *STL* – це процедури, які застосовуються до контейнерів для обробки даних різними способами. Наприклад, існують алгоритми для сортування, копіювання, пошуку та об’єднання даних. Алгоритми представлені функціями шаблону. Ці функції не є членами класів контейнерів. Вони є окремими функціями.

Ітератори – це узагальнення поняття вказівників: вони вказують на елементи в контейнері. Ітератори є ключовою частиною *STL*, оскільки вони зєднують алгоритми з контейнерами.

3.1.4. Бібліотека *OpenCV*

Tехнологія комп’ютерного зору є важливою частиною технології штучного інтелекту, в той час як бібліотека *OpenCV* є однією з найбільш всебічних і зрілих бібліотек для технології комп’ютерного зору. *OpenCV* виходить за рамки традиційних технологій комп’ютерного зору; він включає багато інших технологій, таких як *DNN, CUDA, OpenGL* та *OpenCL*, і перетворюється на більш потужну бібліотеку. Але, водночас, його графічний інтерфейс, який не є основною особливістю бібліотеки, розвивається не дуже [12].

*OpenCV (Open Source Computer Vision Library)* – це бібліотека програмного забезпечення з відкритим кодом що має можливості для машинного навчання [9].

Бібліотека розроблена *Intel* і нині підтримується *Willow Garage* та *Itseez*. Вихідний код бібліотеки написаний мовою *C++* і поширюється під ліцензією *BSD*. *OpenCV* суміжна з *C ++, Python, Java* та *MATLAB* та підтримується такими операційними системами: *Windows, Linux, Android* та *Mac OS*.

Цю бібліотеку використовують чимало як великих компаній (*Google, Yahoo,  
 Microsoft, Intel, IBM, Sony, Honda, Toyota*) так і стартапів (*Applied Minds, VideoSurf* і *Zeitera).*

Бібліотека включає велику кількість алгоритмів машинного зору, машинного навчання, обробки зображень та інших.

*OpenCV* має близбко 30 основних модулів та близько 33 допоміжних модулів. Розглянемо основні модулі.

*Сore*. Відповідає за основну функціональність ядра.

*Imgproc*. Модуль включає функції обробки зображень такі як фільтрування зображень, геометричні перетворення зображень, функції малювання, робота з кольорами, створення гістограм, аналіз руху та відстеження об’єктів, виявлення об’єктів та інші.

Часто використовуваними є функції основних морфологічних операцій: звуження (*erosion*) та розширення (*dilation*). Вони застосовуються для: видалення шуму, виділення окремих елементів та приєднання до зображення різних фрагментів, знаходження рівнів інтенсивності у зображенні

*Іmgcodecs..*Читання та запис файлів зображень.

*Video I/O.* Читання та запис послідовності відео чи зображень за допомогою *OpenCV.* Включає у себе основні прапори для вводу-виводу відео, додаткові прапори для резервних файлів *API*, *C API* для вводу-виводу, а також механізми вводу-виводу для  *iOS* та *WinRT.*

*Highgui.* Забезпечує простий інтерфейс для:

* створення та маніпулювання вікнами, які можуть відображати зображення та “запам’ятовувати” їх вміст;
* відстеження події миші, а також робота з командами клавіатури;
* читання та запис зображень на/з диска чи пам’яті;
* читання відео з камери чи файлу та запис його у файл.

*Video.* Робота з відео, така як вимкнення відтворення, вилучення переднього плану та інші.

*Calib3d.* Калібрування камери та реконструкція 3*D*. Більшість наших зображень у 2*D*-форматі походять з 3*D*-світу, за допомогою цього модулю ми можемо отримати

інформацію з 2*D*-зображень про 3*D*-світ.

*Features2d.* Фреймворк *2D* функцій*.* Включає виявлення та опис особливостей 2*D* зображень, функція малювання ключових точок, ярус прискорення обладнання.

*Objdetect*. Виявлення об’єктів за допомогою каскадного класифікатору*.*

*Dnn.* Цей модуль містить: *API* для створення нових шарів, побудова базових шарів нейронних мереж; набір вбудованих найбільш корисних шарів;

*API* для побудови та модифікації комплексних нейронних мереж із шарів; функціональність для завантаження моделей серіалізованих мереж з різних фреймворків.

Функціональність цього модуля призначена лише для обчислень прямого проходу (тобто мережевого тестування). Мережеве навчання в принципі не підтримується.

*Ml.* Бібліотека машинного навчання (*ML*) – це набір класів і функцій для статистичної класифікації, регресії та кластеризації даних.

Більшість алгоритмів класифікації та регресії реалізовані у вигляді класів *C ++.* Оскільки алгоритми мають різні набори функцій (наприклад, здатність обробляти пропущені вимірювання або категоричні вхідні змінні), між класами існує невелика спільна залежність.

*Flann.* Це бібліотека, яка містить набір алгоритмів, оптимізованих для швидкого пошуку найближчого сусіда у великих наборах даних та для функцій з високими розмірами.

*Photo*. Цей модуль включає алгоритми обробки фотографій.

Набір модулів *Cuda* :

* *cudaarithm*. Операції на матрицях;
* *cudabgsegm.* Фонова сегментація;
* *cudacodec*. Кодування декодування відео;
* *cudafeatures2d.* Виявлення та опис особливостей;
* *cudafilters.* Фільтрування зображень;
* *cudaimgproc*. Обробка зображення;
* *cudalegacy.* Підтримка наслідування;
* *cudaobjdetect.* Виявлення об’єктів;
* *cudaoptflow*. Оптичний потік;
* *cudev*. Шар пристроїв.

*Reg.* Модуль реєстрації здійснює параметричну реєстрацію зображень. Реалізований метод – це пряме вирівнювання, тобто він використовує безпосередньо значення пікселів для обчислення реєстрації між парою зображень, на відміну від реєстрації на основі функцій.

Методи на основі функцій мають деякі переваги перед методами на основі пікселів, коли ми намагаємось реєструвати фотографії, які були зняті за різних умов освітлення чи експозиції, або коли зображення перекриваються лише частково. З іншого боку, головна перевага методів на основі пікселів у порівнянні з методами на основі функцій полягає в їх кращій точності для деяких знімків.

*Superres.* Модуль *Super Resolution* містить набір функцій та класів, які можуть бути використані для вирішення проблеми розширення роздільної здатності.

*Tracking. API* довготривалого оптичного відстеження.

Довготривале оптичне відстеження є важливою проблемою для багатьох програм комп’ютерного зору в реальному сценарії. Розробка в цій області дуже фрагментарна, і цей *API* – це унікальний інтерфейс, корисний для підключення декількох алгоритмів та їх порівняння.

Ці алгоритми починаються з вікна обмежування цілі і за допомогою їх внутрішнього подання вони уникають зміщення під час відстеження.

Є три основні компоненти: *TrackerSampler*, *TrackerFeatureSet* і *TrackerModel*. Перший компонент – це об’єкт, який обчислює виправлення над кадром на основі останнього цільового місця розташування. *TrackerFeatureSet* – клас, який керує функціями, можливо підключити багато видів (*HAAR, HOG, LBP, Feature2D* тощо). Останній компонент – це внутрішнє подання цілі, це модель зовнішності. Він обчислює траєкторію (найімовірніші цільові стани). Клас *TrackerTargetState* представляє можливий стан цілі. *TrackerSampler* і *TrackerFeatureSet* – це візуальне зображення цілі, натомість *TrackerModel* є статистичною моделю.

*Videostab.* Модуль стабілізації відео містить набір функцій та класів, які можна використовувати для вирішення проблеми стабілізації відео.

*Viz*. У цьому модулі описано вікно візуалізації 3*D*, а також класи та методи, які використовуються для взаємодії з ним.

Вікно візуалізації 3*D* використовується для відображення віджетів (*Widget*), і воно забезпечує декілька методів взаємодії зі сценою та віджетами.

3.1.5. Бібліотека *Tesseract* та *Leptonica*

*Tesseract* – вільне програмне забезпечення для розпізнавання текстів, що розроблювалась *Hewlett-Packard* з 1985 до 1994 року, а згодом розвиток було замороженно до тих пір як проект викупив *Google*.

Постачається з ліцензією *Apache 2.0* з 2006 року.

*Tesseract* має ряд переваг, саме з ними пов’язаний вибір цієї бібліотеки для розпізнавання символів в рамках даної роботи:

* має відкритий вихідний код;
* виконує розпізнавання при роботі з чорно-білим текстом;
* велика кількість документації;
* сумісний з бібліотекою комп’ютерного зору (*OpenCV*).

Наразі програма розпізнає багато мов серед яких є українська (працює з   
*UTF-8*).

*Tesseract OCR* для роботи з зображеннями використовує бібліотеку *Leptonica*, яка в свою чергу використовує бібліотеки *libjpeg, libpng, libtiff* і *zlib.*

*Leptonica* – бібліотека для роботи з зображеннями, їх обробки. Має реалізацію алгоритмів вирівнювання сторінок, використання фонового зображення та інших. Написана на мові *C*. Використовується у *Java*, тому є популярною серед *Android* розробників.

*Libtiff* – це програмне забезпечення забезпечує підтримку формату файлів зображень тегів (*TIFF*), широко використовуваного формату для зберігання даних зображень.

*Zlib –* бібліотека стиснення даних без втрат для використання практично на будь-якому комп’ютерному апаратному забезпеченні та операційній системі. Формат даних zlib сам по собі портативний на різних платформах.

*Libjpeg* – це безкоштовна бібліотека з функціями для обробки формату даних зображення *JPEG*. Він реалізує кодек *JPEG* (кодування та декодування) поряд з різними утилітами для обробки *JPEG*-даних. Він написаний на *C* та розповсюджується як безкоштовне програмне забезпечення разом із його вихідним кодом за умовами ліцензії на користувальницьку дозвільну програму (схожу на *BSD*), яка вимагає атрибуції.

*Libpng* – це офіційна довідкова бібліотека портативної мережевої графіки (*PNG*) (спочатку називається *pnglib*). Це незалежна від платформи бібліотека, яка містить функції *C* для обробки зображень *PNG.* Він підтримує майже всі можливості *PNG*, є розширюваним і широко застосовується та перевіряється. *Libpng* залежить від *zlib.*

## **3.2. Програмні та апаратні вимоги до програмного модулю**

Використання розроблюваного програмного модулю вимагає наявність комп’ютера з наступними мінімальними характеристиками:

* операційна система: *Microsoft Windows* 7/8/8.1/10
* процесор: *Intel Celeron* та вище;
* оперативна пам’ять: 2 *Гб*;
* наявність 500 *Мб* вільного дискового простору;
* наявність клавіатури та мишки;
* можливість підключення для мережі для під’єднання до *IP-*камери;
* наявність бібліотек *Qt*.

Також для використання ідентифікації номерного знаку у реальному часі необхідно мати *IP*-камеру.

## **3.3. Навчання каскадів Харра для програмного модуля**

Для роботи програмного модуля ідентифікації автомобільного номера необхідно натренувати два каскади Харра. Перший для ідентифікації номерного знаку автомобіля на зображенні, другий – для ідентифікації символів на номерному знаці.

Основна перевага детектора Хаара це швидкість. Завдяки швидкій обробці, зображення можна з легкістю обробляти у реальному часі. Детектор Хаара використовується для розпізнавання більшості класів об’єктів. До них належать особи та інші частини тіла людей, номери автомобілів, пішоходи, дорожні знаки, тварини і.т.д. Детектор Хаара реалізований в бібліотеці *OpenCV*. Це дже зручно адже готові реалізації *OpenCV* є під більшість існуючих операційних систем (*Android, Windows, Linux, Ios*).

Отже будемо використовувати утиліту *PictureCropper* для навчання каскадів Харра що вбудована у пакет постачання *OpenCV*.

Для навчання нам необхідно зібрати реальні фотографії об’єкта, «хорошу» вибірку (рис. 3.3).

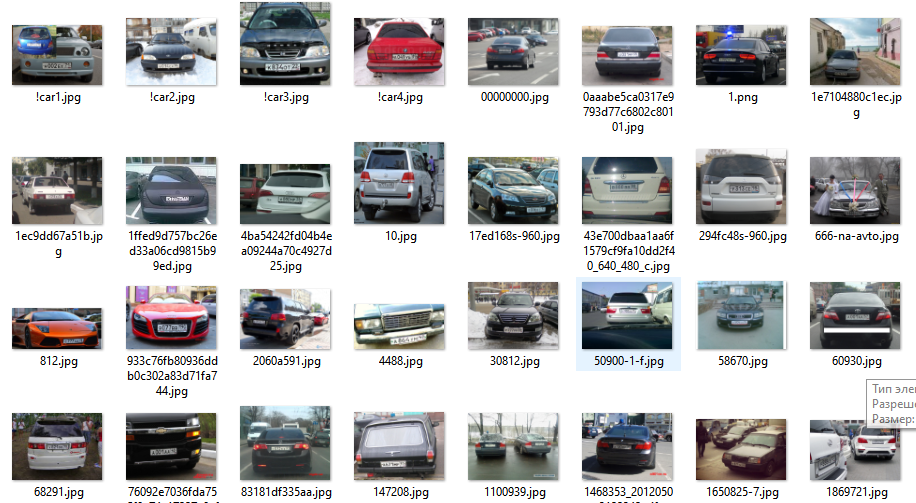


Рис. 3.3. Приклад фотографій для «хорошої» вибірки

Для кращого результату подальшого розпізнавання краще обирати фотографії зроблені при різних умовах, освітлені, різному куті нахилу номерних знаків, різній степені забрудненості. Чим більше вибірка буде схожа на те що будемо розпізнавати, тим краще модуль буде розпізнавати номери.

Також нам необхідно створити вибірку з невдалими фотографіями, так звану «погану» вибірку, на фотографіях якої буде відсутній об’єкт розпізнавання. Фотографії мають бути зроблені в тому ж середовищі що і фотографії «хорошої» вибірки, інакше подальше робота модулю буде некоректна.

Для тренування взято 3000 зображень «хорошої» вибірки та 4000 зображень «поганої» вибірки (рис. 3.4). Відповідно, чим різноманітніша вибірка, тим точніше розпізнаються об’єкти та довше триває навчання.

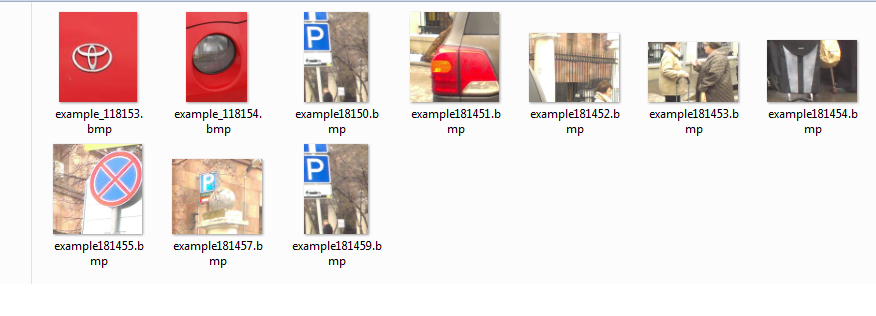


Рис. 3.4. Приклад фотографій для «поганої» вибірки

Навчання починається з підготовки правильної ієрархії папок та створення *.dat* файлів.Файли виборок різняться за структурою.

Структура «поганої» вибірки складається лише з відносних шляхів до файлів (рис. 3.5).



Рис. 3.5. Приклад структури для «поганої» вибірки

Структура «хорошої» вибірки складається з шляхів, координат розміщення об’єкту та його розмірів (рис. 3.6).

Для кожного зображення зазначається лише один об’єкт.

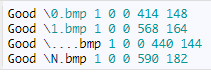


Рис. 3.6. Приклад структури для «хорошої» вибірки

За допомогою програми *PictureCropper* необхідно вручну виділити область що має бути збережена (область розміщення об’єкта). Після проходження по всім навчальним зображенням підготовка для навчання буде завершена, після чого натреновано каскад за допомогою команд консолі: *opencv\_traincascade,* та *opencv\_traincascade.*

Теж саме виконано для тренування каскаду Харра для ідентифікації символів номерного знаку.

Як результат отримано два .*xml* файли що містять треновані каскади Харра   
(рис. 3.7).

  
Рис. 3.7. Результат тренування каскадів Харра

# **3.4. Висновки до розділу**

В третьому розділі розглянуто інструмент розробки програмного забезпечення *Qt*, мову програмування *С++* , стандартну бібліотеку шаблонів *STL*, та бібліотеки *OpenCV, Teseract OCR, Leptonica*.

Також визначено основні системні вимоги до програмного модуля ідентифікації автономерів, та підготовлено каскади Харра для виявлення області номера та розміщення символів.

До підготовки входить створення «хорошої» та «поганої» тренувальної вибірки, та подальше тренування, в результаті якого створено два вихідних .*XML* файли. Перший файл буде використовуватись для знаходження області з автомобільним номером на вхідному зображенні, другий – для знаходження символів на зображенні автомобільного номера.

# **РОЗДІЛ 4 ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ІДЕНТИФІКАЦІЇ РУХОМОГО ОБ’ЄКТА У ПРОСТОРІ**

# **4.1. Реалізація алгоритму ідентифікації рухомого об’єкта у просторі**

4.1.1. Підключення бібліотек та попереднє оголошення змінних і функцій

Спочатку нам необхідно підключити бібліотеки що будуть використовуватись та оголосити глобальні змінні.

Головні бібліотеки що будуть використовуватись:

* *QWidget* (відповідає за створення вікна програми);
* *baseapi.h (API* бібліотеки *tesseract*);
* *allheaders.h* (необхідні для роботи *tesseract* залежності з бібліотеки *leptonica*);
* *core.hpp* (ядро бібліотеки *opencv*);
* *highgui.hpp* (робота з відео);
* *imgproc.hpp* (функції обробки зображень);
* *objdetect.hpp* (робота з каскадами Харра);
* *algorithm (*функція сортування та інші алгоритми);
* *iostream* (ввід/вивід інформації);
* *string* (функції роботи з рядками);
* *vector* (контейнер для зберігання даних);
* *math.h* (основні математичні операції).

Далі оголошуємо константні змінні. Змінна яка містить числа та змінна яка містить літери що можуть зустрічатися у автомобільному номері *listDigit,* та *listSymbol* відповідно.

*string listDigit = "0123456789";  
 string listSymbol = "ABСЕТРОКМНХІ";*

Встановлюємо значення мінімальний куту нахилу *minAgle,* максимального куту *maxAgle* та кроку перебору кута next*Step.* Ці константні змінні необхідні для функції пошуку кута нахилу.

*const double minAgle = -10;*

*const double maxAgle = 10;*

*const double* next*Step = 0.1;*

Виконуємо попереднє оголошення функцій що будуть розглянуті далі.

*bool recCarNumber(const cv::Mat& frame);*

*std::vector<std::string> getCarNumber() const;*

*std::vector<cv::Mat> getZoneNumber() const;*

*bool segmentation(cv::Mat& src);*

*double findCut(cv::Mat& zona);*

*void rotImg(cv::Mat& img, const double cut);*

*int buttomLimit(cv::Mat& zona);*

*int topLimit(cv::Mat& zona);*

*int histogramLimit(cv::Mat& zona);*

*int rightLimit(cv::Mat zona, bool color);*

*int rightLimit(cv::Mat zona, bool color);*

*bool segmentation();*

4.1.2. Конструктор та деструктор

У конструкторі перевіряється наявність файлів та підключаються каскади Харра для пошуку автомобільного номера та пошуку символів. Файл *haar\_detect\_car\_number.xml* являє собою тренований каскадний класифікатор для розпізнавання області номерного знаку, *haar\_detect\_symbol.xml* відповідає за знаходження розміщення кожного окремого символу для подальшого їх розпізнавання. Деструктор відповідає за звільнення пам’яті.

*ANR::ANR()*

*{*

*if(!firstRec.load("haar\_detect\_car\_number.xml "))*

*throw std::logic\_error("file not found.");*

*if(!secondRec.load("haar\_detect\_symbol.xml "))*

*throw std::logic\_error("file not found.");*

*OCR.Init(NULL, "numbRec");*

*OCR.SetPageSegMode(tesseract::PSM\_SINGLE\_CHAR);*

*}*

*ANR::~ANR()*

*{*

*}*

4.1.1. Реалізація алгоритму нормалізації кута нахилу

Згідно алгоритму нормалізації кута нахилу встановлюємо першочергове значення шуканого кута у нуль. Потім перебором шукаємо кут при якому значення нижньої границі буде мінімальним, цей кут і буде шуканим кутом.

Функція пошуку кута нахилу:

*double Anr::findCut(cv::Mat& zona)*

*{*

*int min = zona.size().height;*

*double cut = 0;*

*cv::Mat tmp;*

*for(double i = minAgle; i < maxAgle; i += nextStep)*

*{*

*tmp = zona.clone();*

*rotImg(tmp, i);*

*int bot= bottomLimit(tmp);*

*if(bot < min)*

*{*

*cut = i;*

*min = bot;*

*}*

*}*

*return cut;*

*}*

Використовується у попередній функції. Відповідає за поворот зображення на заданий кут.

Функція повороту кута нахилу:

*void Anr::rotImg(cv::Mat& zona, const double cut)*

*{*

*cv::Mat rot\_mat(2, 3, CV\_32FC1);*

*cv::Point c = cv::Point(zona.cols/2, zona.rows/2);//c-centre*

*double scale = 1;*

*matrix = cv::getRotationMatrix2D(c, cut, scale);*

*cv::warpAffine(zona, zona, matrix, zona.size());*

*}*

4.1.2. Реалізація алгоритмів пошуку границь автомобільного номера

Починаючи від середини номерного знаку до проходження усього зображення шукаємо поточне значення гістограми яскравості та порівнюємо з попереднім, таким чином буде знайдено область з максимальним гістограмним значенням, це і буде нижня границя.

Функція відповідає за пошук нижньої границі автомобільного номера:

*int Anr::bottomLimit(cv::Mat& zona)*

*{*

*size\_t height = zona.size().height;*

*int count = 0;*

*cv::Mat data;*

*for(int i = height/2; i < height; ++i)*

*{*

*data = zona.row(i);*

*int count = cv::countNonZero(data);*

*if(count < count/2)*

*return i;*

*count = count;*

*}*

*return height;*

*}*

Процес пошуку відповідає пошуку нижньої границі. Функція відповідає за пошук верхньої границі автомобільного номера гістограмним методом:  
  
*int Anr:: histogramLimit (cv::Mat& zona)*

*{*

*size\_t height = zona.size().height;*

*cv::Mat data;*

*for(int h = 0; h < height / 2; h++)*

*{*

*hist = zona.row(h);*

*int count = cv::countNonZero(hist);*

*if(count > height\*0.5){*

*return h;}*

*}*

*return 0;*

*}*

Використовуючи каскади Харра шукаємо верхні границі кожної літери після чого визначаємо середнє значення висоти літер, воно буде являти собою верхню границю автомобільного номера.

Функція відповідає за пошук верхньої границі автомобільного номера що використовує каскад Харра:

*int Anr::topLimit(cv::Mat& zona)*

*{*

*std::vector<cv::Rect> liters;*

*secondRect.detectMultiScale(zona, liters);*

*std::vector<cv::Rect>::iterator result;*

*std::sort (liters.begin(), liters.end(), [ ](const Rect& r1, const Rect & r2) {return r1.y > r2.y;});*

*int aveHeight = 0;*

*for(auto& s : liters)*

*{*

*aveHeight += s.y;*

*}*

*if(!liters.empty())*

*aveHeight /= liters.size();*

*return liters.empty() ? histogramLimit(zona) : averHeight;*

*}*

Підготовлюємо зображання виконуючи розмиття та розтягнення, після чого виконуємо обрахування гістограм яскравості для всіх стовпчиків зображення починаючи з середини. Рішення про розташування границі приймається з врахуванням того, біла машина чи чорна.

Функціявідповідає за пошук лівої границі автомобільного номера:

*int Anr::leftLimit(cv::Mat zona, bool color)*

*{*

*cv::Mat elem = getStructuringElement(cv::MORPH\_RECT,*

*cv::Size( (2 \* 1 )+ 1,( 2 \*1)+1 ),*

*cv::Point( 1, 1) );*

*cv::erode(zona, zona, elem);*

*cv::dilate(zona, zona, elem);*

*size\_t width = zona.size().width;*

*double height= zona.size().height;*

*cv::Mat data;*

*for(int i = 2; i < width/2; ++i)*

*{*

*data = zona.col(i);*

*int count = cv::countNonZero(data);*

*//color* істина якщо білий

*if((!color && count > height\*0.5) || (color && count < height\*0.60))*

*{*

*return i;*

*}*

*}*

*return 0;*

*}*

Функціявідповідає за пошук правої границі автомобільного номера:

*int Anr::rightLimit(cv::Mat zona, bool color)*

*{*

*cv::Mat elem = getStructuringElement(cv::MORPH\_RECT,*

*cv::Size( (2 \* 1 )+ 1,( 2 \*1)+1 ),*

*cv::Point( 1, 1) );*

*cv::erode(zona, zona, elem);*

*cv::dilate(zona, zona, elem);*

*size\_t width = zona.size().width;*

*double height= zona.size().height;*

*cv::Mat data;*

*for(int i = width-2; i > width/2; --i)*

*{*

*data = zona.col(i);*

*int count = cv::countNonZero(data);*

*//color* істина якщо білий

*if((!color && count > height\*0.5) || (color && count < height\*0.60))*

*return i+1;*

*}*

*return width;*

*}*

4.1.2. Реалізація алгоритму сегментації автомобільного номера

Над зображенням виконується перетворення в ч/б та бінаризація, після чого обрізаємо зображення по його границям. Застосуємо фільтр середніх частот та розмиття по гаусу для видалення шуму. Після цього застосовуємо детектор границь Кенні та знаходимо контур символів. Знайдені контури що відповідають шаблонам поміщаємо в контейнер для подальшого розпізнавання.

Функція що розбиває номер на сегменти та підготовлює символи до розпізнавання:

*bool Anr::segmentation(cv::Mat& src)*

*{*

*std::vector<std::vector<cv::Point> > conturs;*

*std::vector<mZon> cOut;*

*cv::Mat cOutput, srcB\_w, tresh;*

*cvtColor(src, srcB\_w, cv::COLOR\_BGR2GRAY);*

*treshdata(srcB\_w, tresh, 0, 255, CV\_THRESH\_BINARY | CV\_THRESH\_OTSU);*

*medianBlur(tresh, tresh, 5);*

*double cut = findCut(tresh);*

*rotImg(tresh, cut);*

*int bottom = bottomLimit(tresh);*

*int top = topLimit(tresh);*

*tresh = tresh(cv::Rect(0, top, tresh.size().width, bottom-top));*

*int leftBound = std::max(leftLimit(tresh, true),leftLimit(tresh, false));*

*int rightBound = std::min(rightLimit(tresh, true), rightLimit(tresh, false));*

*rotImg(src, cut);*

*src = src(cv::Rect(left, top, right-left, bottom-top));*

*if((src.size().height < 60) && (src.size().width < 240))*

*resize(src, src, Size(240, 60));*

*cvtColor(src, srcB\_w, cv::COLOR\_BGR2GRAY);*

*// Крок 1: Перетворення в Ч/Б*

*treshdata(srcGray, srcB\_w, 0, 255, CV\_THRESH\_BINARY | CV\_THRESH\_OTSU);*

*// Крок 2: Бінаризація зображення*

*medianBlur(tresh, tresh, 3);*

*//Крок 3: Фільтр середніх частот*

*cv::blur(srcB\_w, srcB\_w, cv::Size(3,3));*

*// Крок 4: Розмиття по Гаусу*

*cv::Canny(srcB\_w, cannyOutput, 100, 300, 3);*

*// Крок 5: Використання детектора границь*

*// Знайти контур*

*cv::findContours(cOut, conturs, hiera, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0));*

*for(auto& c : conturs)*

*{*

*mZon zon(c);*

*if(((int)(zon.height \* zon.width)) == 0)*

*continue;*

*if(zon.height < 0.3 \* srcB\_w.size().height ||*

*zon.height > 0.9 \* srcB\_w.size().height ||*

*zon.width < srcB\_w.size().width \* 0.03 ||*

*zon.width > srcB\_w.size().width \* 0.15 ||*

*zon.width > zon.height ||*

*zon.min.x < zon.width \* 0.03 ||*

*(zon.min.x + zon.width) > srcB\_w.size().width \* 0.97 ||*

*isDuplicat(zon, cOut))*

*continue;*

*contursOut.push\_back(zon);*

*}*

*if(contursOut.size() >= 8)*

*{*

*std::sort(contursOut.begin(), cOut.end()); carSymb.push\_back(CarSymbZon(src, contursOut));*

*}*

*return !carSymb.empty();*

*}*

4.1.4. Реалізація алгоритму розпізнавання символів автомобільного номера

Для розпізнавання символів оброблюємо фрагмент зображення з символом видаляючи з нього зайве та приводячи до потрібного розміру. Далі застосовуємо бібліотеку *tesseract* та зберігаємо символи.

Функція відповідає за розпізнавання символів автомобільного номера:

*bool Anr::recCarNumberLetters()*

*{*

*for(auto& l : carSymb)*

*{*

*std::string seq;*

*for(size\_t j = 0; j < l.zonaZonSymbols.size(); j++)*

*{*

*cv::Mat rimg = cvCreateImage(cv::Size(100, 100), 8, 1);*

*for(int j= 0; j < rimg.size().width; j++)*

*for(int i=0;i < rimg.size().height; i++)*

*recimage.at<int char>(i, j) = 255;*

*int minx = l.zonaZonSymbols.at(j).min.x;*

*int miny = l.zonaZonSymbols.at(j).min.y;*

*int height = l.zonaZonSymbols.at(j).height+l.zonaZonSymbols.at(j).height\*0.05;*

*int width = l.zonaZonSymbols.at(j).width+l.zonaZonSymbols.at(j).width\*0.05;*

*cv::Mat subImg = (l.zona)(cv::Rect(minx, miny, width, height));*

*cv::Mat b\_w = cvCreateImage(subImg.size(), 8, 1);*

*cv::cvtColor(subImg, b\_w, CV\_RGB2GRAY);*

*cv::treshdata(b\_w, b\_w, 0, 255, cv::THRESH\_BINARY\_INV |cv::THRESH\_OTSU);*

*cv::resize(b\_w, b\_w, cv::Size(20, 30));*

*for(int j = 0; j < b\_w.size().width; j++)*

*for(int i=0; i < b\_w.size().height; i++)*

*rimg.at<int char>(40+i, 40+j) = (b\_w.at<int char>(i, j) == 255 ? 0 : 255);*

*equalizeHist(rimg, rimg);*

*int lvl = 5;*

*cv::GaussianBlur(rimg, rimg, cv::Size(lvl, lvl), 0);*

*OCR.SetVariable("tessedit\_char\_whitelist", ((j == 0 || j == 4 || j == 5) ? symbolChar.c\_str() : symbolDigit.c\_str()));*

*OCR.TesseractRect(rimg.data, 1, rimg.step1(), 0, 0, rimg.cols, rimg.rows);*

*char\* symbol = OCR.GetUTF8Text();*

*while(symbol[0] == ' ')*

*{*

*while((lvl \* lvl) % 2 == 0)*

*lvl++;*

*cv::GaussianBlur(rimg, rimg, cv::Size(lvl, lvl), 0);*

*OCR.TesseractRect(rimg.data, 1, rimg.step1(), 0, 0, rimg.cols, rimg.rows);*

*symbol = OCR.GetUTF8Text();*

*lvl++; }*

*text.push\_back(symbol[0]);}*

*carNumberText.push\_back(seq);*

*}*

4.1.5. Реалізація загального алгоритму ідентифікації рухомого об’єкта у   
 просторі

Для уникнення невірного розпізнавання очищаємо буфери в яких зберігаються координати номерного знаку та символів. Шукаємо за допомогою каскадів Харра номерний знак та виконуємо пошук літер та їх розпізнавання.

Функція розпізнавання автомобільного номера на поточному кадрі

*bool Anr::recCarNumber(const cv::Mat& frame)*

*{*

*this->sourseFrame = frame;*

*licenseZonas.clear();*

*carSymb.clear();*

*carNumberText.clear();*

*std::vector<cv::Rect> zonas;*

*cv::Mat b\_w;*

*cv::cvtColor(sourseImage, b\_w, CV\_BGR2GRAY);*

*bool resize = sourseFrame.size().width/scale > 480 || sourseFrame.size().height/scale > 320;*

*if(resize)*

*cv::resize(b\_w, b\_w, cv::Size(sourseFrame.size().width/scale, sourseFrame.size().height / scale), 0, 0, cv::INTER\_LINEAR);*

*firstRec.detectMultiScale(b\_w, zonas, 1.1, 10, 0, Size(70, 21), Size(500, 150));*

*for(auto& p : zonas)*

*{*

*cv::Point zonaBegin= cv::Point((p.x)\*(resize ? scale : 1), (p.y)\*(resize ? scale : 1));*

*cv::Point zonaEnd = cv::Point((p.width)\*(resize ? scale : 1), (p.height)\*(resize ? scale : 1));*

*licenseZonas.push\_back(sourseFrame(cv::Rect(zonaBegin.x,zonaBegin.y, zonaEnd.x, zonaEnd.y)));*

*}*

*for(auto& p : licenseZonas)*

*segmentation(p);*

*if(!carSymb.empty())*

*recCarNumberLetters();*

*return true;*

*}*

В випадку коли користувач вирішив обробити відеоматеріал що зберігається в системі, розбиває обраний користувачем відеофайл на кадри та викликає функцію розпізнавання для кожного отриманого зображення.

Функція отримання поточного кадру з відео та його розпізнавання:

*void MainWindow::on\_pushButton\_clicked()*

*{*

*QString filename=QFileDialog::getOpenFileName(this,tr("Open File"),"C:\\","All files(\*.\*)") ;*

*if (filename!="")*

*{*

*cv::VideoCapture capVideo;*

*capVideo.open(filename);*

*}*

*if (!capVideo.isOpened())*

*{*

*ui->label1->setText("error: reading video file");*

*return 1;*

*else ui->label1->setText("File is open");*

*if (capVideo.get(CV\_CAP\_PROP\_FRAME\_COUNT) < 2)*

*{*

*ui->label1->setText("error: video file must have at least two frames");*

*return 0;*

*}*

*cv::Mat imgFrame1;*

*cv::Mat imgFrame2;*

*ui->widget->addWidget(capVideo);*

*capVideo.read(imgFrame1);*

*capVideo.read(imgFrame2);*

*QString detectedNumber ="";*

*while (capVideo.isOpened())*

*{*

*cv::Mat img = cv::imread(imgFrame1, CV\_LOAD\_IMAGE\_COLOR);*

*ANR\* image; і*

*image = new ANR;*

*image->recCarNumber(img);*

*image->getZoneNumber();*

*for(auto& x : image->getCarNumber())*

*detectedNumber+=x;*

*detectedNumber+="\n";*

*ui->textEdit->setText(detectedNumber);*

*imgFrame1 = imgFrame2;*

*if ((capVideo.get(CV\_CAP\_PROP\_POS\_FRAMES) + 1) < capVideo.get(CV\_CAP\_PROP\_FRAME\_COUNT))*

*{*

*capVideo.read(imgFrame2);*

*}*

*else {*

*ui->label1->setText("end of video");*

*break;*

*} }}*

Забезпечує підключення до камери відеоспостереження та передає кожен кадр вхідного відеопотоку на розпізнавання.

Функція отримання поточного кадру з *IP*-камери та його розпізнавання:

*void MainWindow::on\_pushButton\_2\_clicked(){*

*CvCapture \*c = cvCaptureFromFile("http://"+ui->lineEdit->text(););*

*if (c != NULL) {*

*ui->label->setText("Connect successful");*

*vidcap = new Vidcap(c);*

*FrameSize sizeWH;*

*sizeWH.width = 640;*

*sizeWH.height = 480;*

*vidcap->setSize(sizeWH);*

*// сигнал на початок трансляції відео*

*emit vidcap->start();*

*while(true)*

*{*

*cv::Mat img = cv::imread(cvQueryFrame(capture), CV\_LOAD\_IMAGE\_COLOR);*

*ANR\* image;*

*image = new ANR;*

*image->recCarNumber(img);*

*image->getZoneNumber();*

*for(auto& x : image->getCarNumber())*

*{*

*detectedNumber+=x;*

*}*

*detectedNumber+="\n";*

*ui->textEdit->setText(detectedNumber);*

*}*

*}*

*else {*

*ui->label->setText("Video device not found.");*

*}*

*}*

Функція зупинення трансляції з камери:

*void MainWindow::on\_pushButton\_3\_clicked()*

*{*

*ui->label->setText("Camera disconnect.");*

*emit vidcap->stop();*

*}*

4.1.6. Функції отримання і відображення даних

Функція видаляє дублікати номерів та передає на вивід розпізнаний автомобільний знак. Код функції:

*std::vector<std::string> Anr::getCarNumber() const*

*{*

*isDupl(carNumberText);*

*return this->carNumberText;*

*}*

Функція відображує область з автомобільним номером малюючи на зображенні прямокутних зі знайденою областю зеленим кольором. Код функції:

*std::vector<cv::Mat> Anr::getZoneNumber() const*

*{*

*size\_t nwnd = 0;*

*cv::Mat img = p.clone();*

*for(auto& l : license Zonas)*

*{*

*if(!std::equal(p.begin<int>(), p.end<int>(), l.zona.begin<int>())) continue;*

*rectcut(img, f.min, f.max, cv::Scalar(0,255,0), 2);*

*}*

*return this->licenseZonas;*

*}*

## **4.2. Тестування роботи програмного модуля**

Перевіримо роботу програмного модуля при підключенні до IP-камери.

Результат роботи програмного модуля (рис. 4.1.)

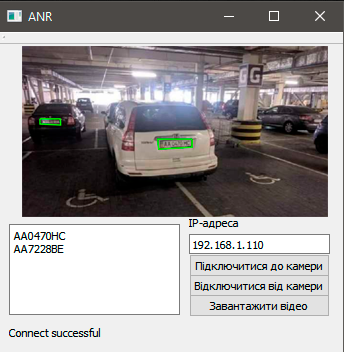


Рис. 4.1. Результат роботи програмного модулю при підключенні до камери

Перевіримо роботу програмного модуля при завантаженні відео з системи.

Результат роботи програмного модуля (рис. 4.2.)

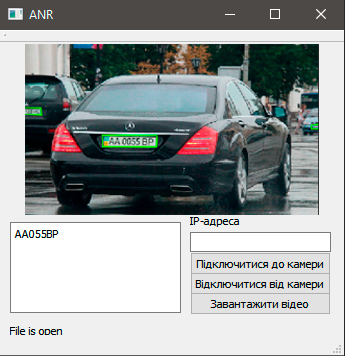


Рис. 4.2. Результат роботи програмного модулю при завантаженні відео

Як видно з результату виконання програми, номера автомобілів що мають менше восьми символів (неповне зображення номера) програмний модуль також знаходить але не може розпізнати. Номера зображення яких є повним розпізнаються успішно в 97 відсотків випадків. Також розпізнаються частково забруднені номера.

## **4.3. Висновки до розділу**

В ході виконання даного розділу програмно реалізовано програмний модуль ідентифікації рухомого об’єкта у просторі та перевірено його роботу.

Перевагами розроблюваного модулю ідентифікації автомобільного номера (об’єкта) є:

* низькі системні вимоги для використання;
* кросплатформеність;
* легка інтегрованість у інші програмні засоби;
* можливість розпізнавання номера як з відеофайлу так і з *IP* камери;
* швидкість оброблюваних результатів (Середня швидкість розпізнавання одного номерного знаку становить 1.3 с);
* можливість розпізнавання частково забруднених номерів;
* процент розпізнавання 97 відсотків;
* можливість додання розпізнавання номерних знаків інших країн з мінімальними змінами коду.

Недоліками програмного забезпечення є:

* необхідність належного освітлення;
* необхідність якісного відеоматеріалу;
* малий діапазон зміщення відеокамери від осі руху автомобіля.

# **ВИСНОВКИ**

В ході виконання дипломної роботи розглянуто що ідентифікація рухомого об’єкта у просторі є досить розповсюдженою задачею. Ця задача існує у безлічі варіацій у різних сферах життя, починаючи від розваг і закінчуючи безпекою та медициною.

Як об’єкт що розпізнається обрано автомобільний номер.

Програмний модуль може забезпечувати контроль знаходження автомобільного транспорту в навантажувальної зоні підприємства, може дозволити проконтролювати заповнюваність території в реальному часі, що дає можливість вжити заходів щодо підвищення ефективності розпізнавання номерів автомобілів, а з системою оплат повністю виключить можливість зловживання або розкрадання з боку найманих працівників.

Перевагами, які надає програмний модуль є:

* підвищення рівня безпеки та контролю автомобільного транспорту на об’єкті;
* виключається можливість для третіх осіб проникнути на територію, що охороняється;
* автоматичне ведення звітності про транспортні засоби з можливістю формування численних звітів;
* систему розпізнавання автомобільних номерів можна легко інтегрувати в загальну базу даних організації.

У першому розділі розглянуто сучасні засоби для ідентифікації автомобільних номерів. Розглянуто програмно-апаратний комплекс *NumberOK* та його варіації(має точність розпізнавання 95 відсотків), автоматизовану систему *CarGo Enterprise* (точність розпізнавання 94 відсотки), систему *CVS*-Авто (точність розпізнавання 96 відсотків), модуль розпізнавання автомобільних номерів *iPera EX-LPR* (точність також 96 відсотків проте немає можливості розпізнавання Українських номерів), та програмне забезпечення *IPVideoRecord* і *Hunter LPR.*

З огляду вказаного програмного забезпечення можна зробити висновок що основними недоліками існуючих засобів є:

– висока вартість;

– високі апаратні вимоги;

– залежність від операційної системи *Microsoft Windows*.

Рівень розпізнавання автомобільних номерів у розглянутого програмного забезпечення коливається в межах від 94 до 96 відсотків, що є досить високим показником.

Також розглянуто методи для ідентифікації об’єкта у просторі: метод кваріації, метод порівняння шаблонів, метод Віоли-Джонса, та штучні нейронні мережі на прикладі нейромережі Гопфліда та нейромережі Кохонена.

Серед методів. Обрано метод Віоли-Джонса створений спеціально для вирішення задач розпізнавання об’єктів у реальному часі.

В другому розділі більш детально розглянуто метод Віоли-Джонса та розглянуто його складові.

Основними принципами, на яких базується даний метод є: робота з об’єктами в інтегральному відображенні (для збільшення швидкості обчислення об’єктів), робота з ознаками Хаара, за допомогою яких відбувається пошук об’єкта і його характеристик, використання бустінгу з метою вибору найбільш суттєвих ознак для шуканого об’єкта на вказаній частині зображення. Всі ознаки надходять на вхід класифікатора, який дає результат «вірно» або «брехня», використовуються каскади ознак для швидкого відкидання вікон, де не знайдено об’єкт.

Також побудовано схеми наступних алгоритмів для побудови програмного модулю ідентифікації автомобільного номера:

* алгоритм нормалізації кута нахилу;
* алгоритм пошуку нижньої границі автомобільного номера;
* алгоритм пошуку верхньої границі автомобільного номера;
* алгоритм пошуку бічних границь автомобільного номера;
* алгоритм сегментації автомобільного номера на символи;
* алгоритм розпізнавання символів;
* алгоритм роботи модуля розпізнавання автомобільного номера;
* загальний алгоритм програмного модулю.

Після чого побудовано структурну схему програмного модуля розпізнавання автомобільних номерів.

В третьому розділі розглянуто інструмент розробки програмного забезпечення *Qt*, мову програмування *С++* , стандартну бібліотеку шаблонів *STL*, та бібліотеки *OpenCV* (бібліотека оптичного зору)*, Teseract OCR* (бібліотека розпізнавання символів)*, Leptonica* (бібліотека роботи з зображеннями).

Вибір саме цих інструментів розробки обумовлено кросплатформеность усіх обраних компонентів, що надало можливість зробити модуль незалежним від системи *Windows.* Програмній модуль має сумісність як з *Linux* системами так і системами *Microsoft Windows.*

Також визначено основні системні вимоги до програмного модуля ідентифікації автономерів, та підготовлено каскади Харра для виявлення області номера та розміщення символів.

До підготовки входить створення «хорошої» та «поганої» тренувальної вибірки, (взято 3000 зображень «хорошої» вибірки та 4000 зображень «поганої» вибірки) та подальше тренування, в результаті якого створено два вихідних .*XML* файли. Перший файл використовується для знаходження області з автомобільним номером на вхідному зображенні, другий – для знаходження символів на зображенні автомобільного номера.

У четвертому розділі розроблено програмний модуль розпізнавання автомобільного номера та перевірено його роботу.

Модуль протестовано при різних вхідних даних (відео з різним освітленням, з різним кутом запису, якістю зображення та у різний час). Найкраще модуль впорався з задачами розпізнавання автомобільного номера в денній час, та с добре освітленими номерами. Час розпізнавання автомобільного номера з забрудненнями чи при нечіткому, неякісному відео більший, але в результаті модуль чудово справляєтся і з такими задачами.

Встановлено що перевагами розроблюваного модулю ідентифікації автомобільного номера (об’єкта) є:

* низькі системні вимоги для використання;
* кросплатформеність;
* легка інтегрованість у інші програмні засоби;
* можливість розпізнавання номера як з відеофайлу що зберігається на машині користувача так і з *IP*-камери;
* швидкість оброблюваних результатів (середня швидкість розпізнавання одного номерного знаку становить 1.3 *с*.);
* можливість розпізнавання частково забруднених номерів;
* процент розпізнавання 97 відсотків, що досягнуто завдяки різноманітній навчальній виборці та двом методам для розпізнавання верхньої границі автомобільного номера ;
* можливість додання розпізнавання номерних знаків інших країн з мінімальними змінами коду.

Недоліками програмного забезпечення є:

* необхідність належного освітлення;
* необхідність якісного відеоматеріалу;
* малий діапазон зміщення відеокамери від осі руху автомобіля.

Наразі програмний модуль використовується у вінницькій компанії «Токін». Компанією зроблено замовлення на програмний модуль розпізнавання автомобільних номерів що використовуєтся з метою відстежити ввезення і вивезення інструментів та обладнання з території підприємства, та перевірки номера автомобіля для виключення можливість вивезення товару за підробленими або помилковим супровідним документам.

Згодом планується розширення модулю та доповнення його новими функціями, серед яких:

* збільшення числа каналів відстеження;
* підключення баз даних та автоматичне порівняння автомобільного номера на вході з списиками рівнів доступу;
* додання можливості розпізнавання автомобільних номерів інших країн;
* контроль фізичних пристроїв для автоматичного відкриття/закриття воріт та шлагбаумів.

Також планується подальша оптимізація використання системних ресурсів.

# **СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ**

# **ДЖЕРЕЛ**

1. Бойченко С. В., Іваненко О. В. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – Київ: НАУ, 2017. –   
   63 с.
2. Единая система конструкторской документации. Текстовые документы: ГОСТ 2.106-96. – [Дата введения 1997–07–01]. – М.: Стандартинформ, 1997. – 39 с. – (Межгосударственный стандарт).
3. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Нац. стандарт України. – Вид. офіц. – [Чинний від 1995–02–23]. – Київ: Держстандарт України, 1995. – 39 с.
4. Мазуров Д. Математические методы распознавания образов / Д. Мазуров. – Екатеринбург: Урал, 2010. – 101 с.
5. Лепский A. Є. Математические методы распознавания образов / A. Є. Лепский, А. Г. Броневич. – Таганрог: ТТИ ЮФУ, 2009. – 155 с.
6. Копча-Горячкіна Г. Е. Навчально-методичний посібник до курсу “ ТЕОРІЯ РОЗПІЗНАВАННЯ ОБРАЗІВ ” / Г. Е. Копча-Горячкіна. – Ужгород: ДВНЗ «Ужгородського національного університету»,, 2016. – 59 с.
7. Інформаційні технології та комп’ютерна інженерія: збірник тез доповідей науково-практичної конференції, – Кіровоград: КНТУ, 2014. – 246 с.
8. Патрик Э. А. Основы теории распознавания образов / Э. А. Патрик. – Москва, 2004. – 407 с.
9. Пістунов І. М. Нейромережеві технології економіці та фінансах з розрахунками на комп’ютері / І. М. Пістунов, О. П. Антонюк., 2014. – 105 с.
10. Шлее М. *Qt5.3*. Профессиональное программирование на *С++* / М. Шлее. – Санкт-Петербург: БХВ-Петербург, 2015. – 928 с.
11. Шилдт Г. Полный справочник по *C++* / Герберт Шилдт. – Москва: Издательский дом «Вильямс», 2006. – 800 с.
12. *Qingliang Z. Qt 5 and OpenCV 4 Computer Vision Projects / Zhuo Qingliang. – Birmingham: Packt Publishing Ltd*, 2019. – 348 с.
13. Бишоп К. Распознавание образов и обучение машин / К. Бишоп., 2007.
14. Хайкин С. Нейронные сети. Полный курс. Учебное пособие / Саймон Хайкин., 2016. – 1114 с.
15. Селянкин В. В. Компьютерное зрение. Анализ и обработка изображений. Учебное пособие / Владимир Васильевич Селянкин., 2016. – 152 с.
16. *Merkov AB Pattern Recognition. Introduction to methods of statistical training / AB Merkov*., 2011. - 311 p.
17. Келер А. Изучаем *OpenCV 3* / А. Келер, Г. Брэдски., 2017. – 826 с.
18. Мерков А. Б. Распознавание образов. Построение и обучение вероятностных моделей / А. Б. Мерков., 2020. – 240 с.
19. Программирование на языке *С++* в среде *Qt Creator* / Е. Р.Алексеев, Г. Г. Злобин, О. В. Чеснокова, А. С. Чмыхало. – Москва: *ALT Linux*, 2015. –   
    448 с.