РІВНЕНСЬКИЙ ДЕРЖАВНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет математики та інформатики
Кафедра інформатики та прикладної математики

«До захисту допущено»
Завідувач кафедри
_____ Батишкіна Ю. В.
«___»_____20___ р.

Дипломний проект (робота)

ступеня «Магістр»

з напряму підготовки (спеціальності) 122 – Комп'ютерні науки

на тему: Розробка програмного комплексу для автоматизації ідентифікації графічних об'єктів

Виконав: студент II курсу, групи М-КН-21

Хільчук Сергій Володимирович _____

Керівник: к.ф.-м.н., доц. Мороз І. П. _____

Консультант: _____ _____

Рецензент: д.т.н., проф. Бомба А. Я. _____

Засвідчую, що у цьому дипломному проекті немає запозичених матеріалів з праць інших авторів без відповідних посилань.
Студент _____

Рівне – 2020 року

ЗМІСТ

,

ВСТУП

Актуальність. Останні роки характеризуються значними успіхами у сфері розробки та застосування інтелектуальних систем різноманітного призначення. Поширення набувають інтелектуальні інформаційні системи, експертні системи, розрахунково-логічні та ін. у військовій справі, медицині, економіці, промисловості, транспорті, що працюють без втручання людини.

Сучасні комп'ютерні програми здатні відтворювати інтелектуальні функції людини. Наприклад, на основі штучного інтелекту працює багато додатків для розпізнавання та ідентифікації об'єктів, які використовуються в охоронних системах, де потрібно на автоматичному рівні розпізнати об'єкти та вжити необхідних заходів. Кожна з таких програм містить наступні основні блоки: базу знань, алгоритми прийняття рішень та алгоритми інтелектуального введення-виведення інформації. Останні, у комплексі із відповідними технічними системами, забезпечують комп'ютеру здатність "відчувати" навколишній світ.

Відповідно актуальною є проблема розпізнавання графічних об'єктів, зокрема, для експертно-криміналістичних підрозділів. Саме тому було прийняте рішення створити програмний продукт для автоматизації ідентифікації слідів взуття людей з метою підвищення ефективності роботи експертів.

Мета роботи: вивчення методів ідентифікації графічних об'єктів та розробка спеціалізованої програми для автоматизації процесу ідентифікації слідів взуття людини.

Об'єктом дослідження є методи обробки графічних зображень та методи ідентифікації графічних об'єктів.

Предмет дослідження: застосування методу контурних функцій для ідентифікації графічних об'єктів.

Завдання. Для досягнення поставленої мети потрібно:

- дослідити предметну область (базові проблеми криміналістики,

сучасний стан вирішення проблем розпізнавання та ідентифікації, сучасні інформаційні технології збереження та обробки графічних зображень, базові алгоритми прийняття рішень при розпізнаванні графічних об’єктів, сучасні технології розробки прикладних програм);

- розробити вимоги до програмного виробу;
- спроектувати та розробити структури даних;
- реалізувати алгоритми обробки графічних файлів та прийняття рішень на основі методу контурних функцій;
- навчитися проектувати і розробляти інтерфейс користувача з використанням об’єктно-орієнтованих технологій (зокрема пакету Visual Studio .NET).

У процесі виконання дослідження спроектовано та реалізовано програмний додаток - систему автоматизованого управління графічними об'єктами, що призначена для підтримки функціонально діяльність експертів НДЕКЦ (обробки, збереження зображень у графічних файлах та автоматизованого розпізнавання слідів взуття людини на основі використання методу контурних функцій). Ряд рішень закладених в проект надає НДЕКЦ можливість:

- формувати базу даних слідів взуття;
- реалізувати автоматичний пошук відповідних графічних об’єктів у сформованій базі даних.

Обсяг роботи: Робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел, додатків. Перший розділ присвячений дослідженню проблем розпізнавання та ідентифікації образів. У другому розділі описується використання бібліотеки EMGUCV, алгоритми обробки зображень та ідентифікації. Третій розділ містить опис використаних технологій. У четвертому розділі розглядаються базові алгоритми ідентифікації. У п’ятому розділі - реалізація програмного додатку, інструкція користувача. Список літератури містить 24 джерела.

РОЗДІЛ 1
ПРОБЛЕМАТИКА РОЗПІЗНАВАННЯ ТА ІДЕНТИФІКАЦІЇ ОБРАЗІВ

1.1. Базові положення теорії розпізнавання та ідентифікації

Предмет розпізнавання образів – об'єднує ряд наукових дисциплін. Їх пов'язує пошук рішення загальної задачі − виділити елементи, які належать конкретному класу, серед множини розмитих елементів, що відносяться до кількох класів.

Образ − це опис будь-якого елемента як представника відповідного класу образів.

У випадку коли множина образів розділяється на класи, які не перетинаються, бажано використовувати для віднесення цих образів до відповідних класів якийсь автоматичний пристрій.

Між образами та класами образів існує деяке ієрархічне підпорядкування. Букви алфавітів і цифри є образами, якщо буквено-цифрові символи розглядаються як клас образів.

Друковані і рукописні зображення, наприклад букви А є образами букви англійського алфавіту А, яка представляє в цьому випадку клас образів [1].

Як наука, теорія розпізнавання образів почала формуватися наприкінці 50-их років. Спочатку це була змістовна постановка задачі (але не формальна!) Вона полягала в тому, що треба було побудувати машину, яка б могла класифікувати різні ситуації так, як це робили живі істоти. Така загальна постановка задачі привела до того, що виникли різні напрямки досліджень. Деякі вчені будували моделі процесу сприйняття. Інші вважали, що головне − це створення алгоритмів навчання розпізнаванню образів, щоб розв’язувати конкретні задачі. Дехто шукав постановки нових математичних проблем. Якщо спочатку досить успішно вдалося просунутись по всіх напрямках, то з часом успіхи значно зменшились. Виникли два протилежні погляди на проблему

розпізнавання. Перший полягав у тому, що потрібно знайти такий опис об’єктів, використовуючи апріорні дані про них, що знаходження принципу класифікації вже не буде таким складним. Другий полягав у тому, що головну проблему вважали в пошуку правила класифікації серед заданої множини вирішальних правил [1].

В цілому проблема розпізнавання образів складається з двох частин: навчання та розпізнавання. Навчання здійснюється шляхом показу окремих об'єктів із зазначенням їх приналежності того чи іншого способу. В результаті навчання розпізнає система повинна здобути здатність реагувати однаковими реакціями на всі об'єкти одного образу і різними - на всі об'єкти різних образів. Дуже важливо, що процес навчання повинен завершитися тільки шляхом показів кінцевого числа об'єктів без будь-яких інших підказок. Об'єктом навчання можуть бути або картинки, або інші візуальні зображення (літери), або різні явища зовнішнього світу, наприклад звуки, стану організму при медичному діагнозі, стан технічного об'єкта в системах управління і ін. Важливо, що в процесі навчання вказуються тільки самі об'єкти і їх приналежність образу. За навчанням слідує процес розпізнавання нових об'єктів, який характеризує дії вже навченої системи. Автоматизація цих процедур і становить проблему навчання розпізнаванню образів. У тому випадку, коли людина сама розгадує або придумує, а потім нав'язує машині правило класифікації, проблема розпізнавання вирішується частково, так як основну і головну частину проблеми (навчання) людина бере на себе.

Проблема адаптації розпізнавання образів цікава як з прикладної, так і з принципової точки зору. З прикладної точки зору рішення цієї проблеми важливо перш за все тому, що воно відкриває можливість автоматизувати багато процесів, які до сих пір пов'язували лише з діяльністю живого мозку.

Коло завдань, які можуть вирішуватися за допомогою систем, що розпізнають, надзвичайно широкий. Сюди відносяться не тільки задачі розпізнавання зорових і слухових образів, а й завдання розпізнавання складних

процесів і явищ, що виникають, наприклад, при виборі доцільних дій керівником підприємства або виборі оптимального управління технологічними, економічними, транспортними або військовими операціями. Перш ніж почати аналіз будь-якого об'єкта, потрібно отримати про нього певну, будь-яким способом впорядковану інформацію. Така інформація є характеристикою об'єктів, на множинні сприймаючих органів розпізнавальної системи [2].

Ідентифікація − це процес визначення схожості об’єкта та зразку, це не порівняння властивостей об’єкта, а їх конкретне співставлення для виявлення схожості. Ідентифікація використовується зокрема в криміналістиці для виявлення слідів з місця злочину певній особі. Проблемою ідентифікації є автоматизація процесу, зазвичай ідентифікацію здійснює людина власними силами, це відповідно займає деякий проміжок часу. Щоб навчити машину здійснювати ідентифікацію необхідно підібрати правильний інструментарій, визначити відповідні алгоритми порівняння властивостей об’єкта і провести навчання щоб домогтися від системи отримання максимально точних результатів [3].

1.2. Класи та їх властивості

Одне з найвдаліших формулювань ключової парадигми теорії розпізнавання таке: будь-який об’єкт у природі є унікальним, всі об’єкти є типізованими. Зміст цієї парадигми такий. Кожний об’єкт характеризується тими чи іншими властивостями. Наявність чи відсутність таких властивостей, а також якісні та кількісні характеристики цих властивостей розглядаються як ознаки об’єкта.

Клас образів − деяка категорія, що визначається рядом властивостей спільних для всіх її елементів.

Класифікація - процес призначення міток класу об'єктам, відповідно до деякого опису властивостей цих об'єктів. Класифікатор - пристрій, який в якості

вхідних даних отримує набір ознак об'єкта, а в якості результату видає мітку класу.

Верифікація - процес зіставлення примірника об'єкта з однією моделлю об'єкта або описом класу.

Прикладами завдань класифікації є:

- розпізнавання символів;
- розпізнавання мови;
- встановлення медичного діагнозу;
- прогноз погоди;
- розпізнавання осіб;
- класифікація документів та ін.

Унікальність будь-якого об’єкта означає те, що в природі не існує двох різних об’єктів, для яких збігаються абсолютно всі ознаки, а це дозволяє, принаймні теоретично, відрізнити один об’єкт від іншого. Але деякі ознаки різних об’єктів можуть збігатися, і це дає підстави говорити про те, що ці об’єкти належать до одного типу або класу.

Фундаментальні поняття «клас» та «об’єкт» неможливо повністю формалізувати. Спробуємо навести їх неформальні визначення. Об’єктом у теорії розпізнавання прийнято називати будь-яку сутність, що існує або могла б існувати в реальному світі, а також будь-яке явище або процес. Це дуже широке визначення, подальші уточнення можуть бути пов’язані з тим чи іншим звуженням нашого розуміння про те, що саме треба вважати об’єктом.

Так, реально існуюча Ейфелева вежа буде вважатися об’єктом практично у будь-якій інтерпретації, а «розвиток наукових досліджень про етнокультурний стан Ефіопії та Антарктиди за період з липня 1898 року по січень 1927 року» –можливо, ні. Тут усе залежить від специфіки конкретної задачі і від мети, з якою вона вирішується. Класом теорії розпізнавання образів прийнято називати сукупність об’єктів, які мають ті чи інші спільні ознаки. Клас може об’єднувати фізично існуючі сутності (наприклад, людина, яблуко)

або бути абстрактним поняттям (горе, економічний крах і т.п.). Ознаки, які дають можливість відрізнити представників одного класу від іншого, прийнято називати інформативними ознаками. Ознаки, спільні для всіх представників класу, називатимемо інваріантами класу. Іноді вживають формальніше визначення класу: класом називається сукупність об'єктів, пов'язаних між собою деяким відношенням еквівалентності, або, в крайньому разі, толерантності [4].

Нагадаємо, що відношенням еквівалентності називається відношення, яке є симетричним, рефлексивним і транзитивним (наприклад, відношення «дорівнювати»). Для відношення ж толерантності властивість транзитивності в цілому не виконується (наприклад, відношення «бути схожим»). Набори інформативних та інваріантних ознак можуть збігатися, але це зовсім не обов'язково. Інколи, щоб уникнути непорозумінь, ми називатимемо об'єкти і класи Реального світу відповідно Р-об'єктами і Р-класами. Зрозуміло, що Р-об'єкт може належати будь-якому числу Р-класів. Р-об'єкти часто називаються реалізаціями, або зразками Р-класів. Можна виокремити такі основні властивості класів:

1. Усі представники класу мають певний набір спільних ознак (випливає з визначення).

2. Змінюваність реалізацій класів. По-перше, різні об'єкти, що належать одному класу, можуть бути не схожими між собою. Вони повинні мати спільні інваріантні ознаки, але всі інші ознаки можуть як завгодно варіювати. По-друге, один і той самий об'єкт може змінюватися з часом і навіть поступово переходити від одного класу до іншого (наприклад, перетворення пуголовка на жабу). Усе це свідчить про те, що розпізнати чіткі межі класу часто неможливо.

3. Ознайомлення з деякою скінченим числом представників одного класу дає можливість впізнавати інших представників цього класу. Взагалі кажучи, ця властивість може не виконуватися. Але якщо ця властивість виконується (принаймні, теоретично), це є дуже сильним і важливим твердженням. Воно по

суті означає можливість навчатися на прикладах, тобто на основі спостереження певної кількості прикладів (можливо, разом з контрприкладами), сформулювати правило розпізнавання, яке дає змогу відрізняти представників даного класу від представників іншого (можливо, з певною достовірністю тобто з певним процентом помилок).

У деяких випадках правилом розпізнавання може бути предикат, який залежить від інформативних або інваріантних ознак, інколи правило розпізнавання реалізується у вигляді деякої складної процедури. Якщо навчання на прикладах неможливе або неефективне, правило розпізнавання інколи можна задати явно. Якщо це зробити не вдається, єдиною можливістю для надійного розпізнавання залишається запам'ятовування всіх можливих представників даного класу.

Цей випадок не цікавий з теоретичного погляду, і його можна реалізувати лише, якщо число можливих представників не є надто великим. Створення пристроїв, які виконують функції розпізнавання різних об'єктів, у багатьох випадках відкриває можливість заміни людини як елемента складної системи спеціалізованим автоматом. Така заміна дозволяє значно розширити можливості різних систем, що виконують складні інформаційно-логічні задачі [5].

Якість робіт, які виконує людина на будь-якому робочому місці залежить від кваліфікації, досвіду, сумлінності, стану. У той же час автомат, що її замінює, діє одноманітно і забезпечує завжди однакову якість, якщо він справний. Але не тільки зазначена заміна і звільнення людини від виконання рутинних операцій є причиною створення і пошуку шляхів створення ряду систем розпізнавання. У деяких випадках людина узагалі не в змозі вирішувати цю задачу зі швидкістю, яка задається обставинами, не залежно від якостей і психологічного стану приймаючого рішення (наприклад: протиракетний маневр літака в складних метеоумовах). А автомат з такими задачами може легко

справлятися. Отже, основні цілі заміни людини в задачах розпізнавання зводяться до наступного:

- Звільнення людини від одноманітних рутинних операцій для вирішення інших важливіших задач.
- Підвищення якості виконуваних робіт.
- Підвищення швидкості вирішення задач.

Практичні реалізації методів розпізнавання носять назву систем розпізнавання (СР). Центральним завданням розпізнавання образів є побудова на основі систематичних теоретичних і експериментальних досліджень ефективних обчислювальних засобів (об'єднуваних в понятті «Системи розпізнавання») для віднесення описів з об'єктів, явищ, процесів до відповідних класів. Таким чином, СР – складна динамічна система, яка складається у загальному випадку з колективу підготовлених фахівців і сукупності технічних засобів здобуття і переробки інформації, що забезпечують на основі спеціально сконструйованих алгоритмів рішення задачі класифікації відповідних об'єктів, явищ або процесів. Постають такі питання:

- які функції виконують рецептори в частині первинної обробки результатів виявлення об'єктів, явищ;
- які характеристики ліній передачі даних від рецепторів до мозку як ЦВС;
- які ознаки виділяє система обробки;
- які алгоритми використовує мозок для вирішення завдання класифікації, оптимального управління процесом розпізнавання;
- як людині вдається позбавитися від специфічності, властивої технічним СР і тому подібні. Отже, розпізнавання образів в техніці – необхідний елемент процесу механізації та автоматизації машин, пристроїв і систем для заміни людини там, де використовується важка фізична праця;
- реалізації швидких реакцій в управлінні там, де немає часу на роздуми;
- заміни людини в так званих рутинних операціях, тобто, у повторюваних діях, і не вимагають розумових зусиль. У результаті зіставлення

конкретних рішень і розробок виявилось, що не дивлячись на різноманіття та особливості додатків, завдання створення систем розпізнавання мали багато спільного, не залежного від вказаної специфіки.

Ось чому для вироблення методичних підходів теорії розпізнавання є сенс виділяти загальні повторювані прийоми, а їх число природно має бути обмеженим і легко об’єднуваним в завдання. Самі ж ці завдання повинні були бути ключовими для створення будь-якої системи розпізнавання. У результаті виявилось, що знайдений методичний підхід до побудови систем розпізнавання образів інваріантний до предметної області.

На прикладах видно, що підходи до побудови систем розпізнавання практично нічим не відрізняються, не дивлячись на специфіку самих створюваних систем. У результаті ми отримали загальні уявлення про послідовність вирішення і складові завдання створення системи розпізнавання. Не дивлячись на відмінність предметних областей підходи до побудови СР – однакові.

Система розпізнавань захворювань серця будувалася так само, як і система розпізнавання літаків, але замінити її вона не може. Аналогічно СР літаків не може застосовуватися для вирішення завдань розпізнавання захворювань серця. Системи розпізнавання об’єктів (явищ), які створюються людиною завжди вузько спеціалізовані на відміну від її власних природних можливостей. Що ж до загального підходу до побудови будь-якої системи, то тепер, якщо у нас є деяка сукупність об’єктів або явищ, які потрібно розпізнавати (класифікувати), на основі узагальнення дій при створенні СР в двох розглянутих прикладах ми знаємо, що послідовність вирішення відповідних завдань наступна:

- відповідно до вибраного принципу сукупність об’єктів або явищ підрозділяється на ряд класів (кажуть: призначається алфавіт класів);
- розроблюється сукупність ознак (кажуть: словник);

- на мові словника ознак описується кожен клас;
- вибираються і (або) створюються засоби визначення ознак;
- на обчислювальних засобах реалізується алгоритм зіставлення апостеріорних та апріорних даних і приймається рішення про результати розпізнавання.

Хоча послідовність дій визначена, залишаються запитання:

- як краще здійснювати розбиття об’єктів (літаки, захворювання та ін.) на класи;
- як накопичувати та обробляти апріорну інформацію;
- з яких міркувань вибирати ознаки;
- як описувати класи на мові ознак;
- на основі яких методів порівнювати апріорну та апостеріорну інформацію;
- коли і як з’являється вся система розпізнавання.

Отже, головні висновки:

1. Завдання, які вирішуються в процесі створення систем розпізнавання, інваріантні відносно предметної області, мають багато спільного, базуються на єдиному методологічному підході.

2. Кожна система розпізнавання індивідуальна і призначається лише для одного цілком конкретного виду об’єктів або явищ. Якщо знайдена сфера застосування розпізнавання, то відповідна система повинна розроблятися заново з урахуванням нових специфічних властивостей об’єктів (явищ), що визначають як систему вимірів характеристик, так і словник ознак, алфавіт класів та алгоритм ухвалення рішень.

3. СР повинна створюватися методом послідовних наближень внутрішньої структури на її математичній моделі у міру накопичення потрібної інформації [5].

1.3. Базові методи розпізнавання образів

Виділяють 3 групи методів розпізнавання:

- Порівняння із зразком. До цієї групи відносяться структурні методи і методи що використовують приближення і відстань (класифікація по найближчому середньому і по відстані до найближчого сусіда) [6].

Класифікація по найближчому середньому значенні.

У класичному підході до систем розпізнавання вектор ознак, що характеризує кожен клас, виходить в наслідок навчання системи і відомий заздалегідь або на основі будь-яких моделей передбачається в режимі реального часу. Один з найпростіших алгоритмів класифікації використовує вектор математичного очікування класу (середнє значення).

Отже, невідомий об'єкт буде відноситись до класу і, якщо він істотно ближче до вектору математичного очікування класу і, ніж до векторів математичних очікувань інших класів. Такий метод можна застосовувати, коли точки ознак розташовані дуже тісно і далеко від точок інших класів.

Класифікація за відстанню до найближчого сусіда. Цей метод відносить невідомий вектор ознак до класу, окремі зразки якого знаходяться ближче всіх. Такі зразки називаються найближчими сусідами. При класифікації за найближчим сусідом не потрібно знати моделей розподілу класів в просторі, необхідна тільки інформація про еталонні зразки. Принцип роботи алгоритму побудований на визначенні мінімальної відстані до зразка ознаки з бази даних. Також рішення можна поліпшити, якщо шукати серед сусідів. Для класифікації кожного з об'єктів тестової вибірки необхідно послідовно виконати наступні операції:

- Обчислити відстань до кожного з об'єктів навчальної вибірки.
- Відібрати k об'єктів навчальної вибірки, відстань до яких мінімально.

- Клас класифікуємого об'єкта - це клас, який найчастіше трапляється серед k найближчих сусідів.

Переваги такого підходу очевидні:

- в будь-який момент можна додати нові зразки в базу даних;
- деревовидні і сіткові структури даних дозволяють скоротити кількість обчислюваних відстаней.

Даний алгоритм - один з найпростіших алгоритмів класифікації, тому на реальних завданнях він часто виявляється неефективним. Крім точності класифікації, проблемою цього класифікатора є швидкість класифікації: якщо в навчальній вибірці N об'єктів, в тестовому виборі M об'єктів, а розмірність простору - K, то кількість операцій для класифікації тестової вибірки може бути оцінений як O (K * M * N) [7].

Структурний метод. Такий метод розпізнавання застосовується для об'єктів, які можна структурно розділити на складові. Також важливо, щоб при розпізнаванні система знайшла такі ознаки, які точно дозволять сказати, що об'єкт належить до цього класу і ні до якого іншого. Як приклад використання, можна привести задачу розпізнавання писаних символів або фігур. Друга назва цього методу - синтаксичний, так як він має на увазі використання мови опису образів, який структурно описує кожен елемент і піделементи, структурно розділяючи образ на підобрази. Метод буде корисний для розпізнавання складних образів, що складаються з багатьох образів нижчого, простого рівня.

- Статичні методи. Прикладом цієї групи служить Баєсовий метод прийняття рішення. Статистичні методи засновані на обчисленні ймовірності.

Баєсовий метод прийняття рішень. Даний метод заснований на теоремі Баєса і визначенні апріорних ймовірностей, тобто ймовірність результатів або належність об'єкта певного класу змінюється після отримання нових експертних оцінок (підтвердження наявності нових ознак). Поява того чи іншого способу є випадковою подією і ймовірність цієї події можна описати за допомогою закону розподілу ймовірностей багатовимірної випадкової

величини ξ в тій чи іншій формі. Знаючи елементи навчальної вибірки можна відновити імовірнісні характеристики цього середовища. Баєсовий класифікатор на основі спостережуваних ознак відносить об'єкт до класу, до якого цей об'єкт належить з найбільшою ймовірністю. Як уже зазначалося, в основі методу лежить теорема Баєса. Припустимо, що ми розглядаємо деяку випадкову величину X, яка має щільність ймовірності p (x, w) з параметром w. Але нам потрібно отримати дані про інший випадкової величини w, що має деякий розподіл ймовірності τ (w). Нехай в результаті спостережень отримані статистичні дані x. Із визначення умовної ймовірності слідує:

Також із цього визначення слідує, що:

Підставивши другий вираз в перший отримаємо формулу Баєса

Розподіл Pr (w) називають апріорним розподілом ймовірностей можливих значень w. Даний розподіл відомо раніше, ніж будуть отримані статистичні дані, тобто нові ознаки (експертні оцінки) [8].

Лінгвістичний (синтаксичний) метод.

Якщо опис образів здійснюється за допомогою підобразів та їх співвідношень, то для конструювання системи розпізнавання використовують лінгвістичний або синтаксичний підхід з використанням принципу загальності властивостей.

Основне припущення, яке робиться в цьому методі, ґрунтується на тому, що образи, які належать одному і тому ж класу, володіють рядом загальних властивостей або ознак, які відображають подібність таких образів. Ці загальні властивості можна частково ввести в пам’ять системи розпізнавання. Коли системі представити некласифікований образ, то вибирається набір визначуваних ознак, причому останні інколи кодуються, а потім вони порівнюються з ознаками, закладеними в пам’яті системи розпізнавання. При використанні цього методу основна задача полягає у виділенні загальних властивостей за вибором образів, належність яких шуканому класу відома.

Вибір методу синтезу системи не розв’язує до кінця проблеми складання конкретної програми і проблеми реалізації. У більшості випадків є образи, що представляють кожний з розглядуваних класів. У таких випадках можна скористатися методами розпізнавання, які називаються «навчання з учителем». У схемі «навчання з учителем» система навчається розпізнавати образи за допомогою різного роду адаптивних схем. У деяких прикладних задачах для елементів, які належать до визначеного класу, навчаючі множини невідомі. У таких випадках, властиво, і можна звернутися до методів «розпізнавання з учителем». Розпізнавання за схемою навчання з учителем характеризується тим, що відома правильна класифікація кожного навчаючого образу.

У випадку навчання без учителя потрібно конкретно вивчити класи образів. Типовим прикладом лінійного представлення є блок-схеми, технічні та архітектурні креслення і т.д. Лінійні структури можуть бути описані аналітично, а з використанням ЕОМ можна порівняно легко формувати складні структури. Ці структури використовують також у тих випадках, коли для вводу зображення застосовують методи слідкування за контуром чи іншими траєкторними точками. Яка б структура не використовувалась, в машинному представленні найкраще використовувати єдиний вид структури даних – ланцюговий код. Це послідовність з восьми цифр .

Перевага ланцюгового коду – його компактність. Код дозволяє представити складні дані про зображення, які складається з великого числа областей, у стиснутому вигляді. Оскільки ланцюгові коди представляють дані про границю, їх зручно застосовувати в ЕОМ до задач, які потребують зберігання форми областей. Границі для цілої області можуть зберігатися у формі лінійних списків [9].

Математичний метод.

В основу математичного підходу покладені правила класифікації, які формулюються і виводяться в рамках визначеного математичного формалізму з

допомогою принципів загальності властивостей і кластеризації. Коли образи деякого класу являють собою вектори, компонентами яких є дійсні числа, то цей клас можна розглядати як кластер. Побудова системи розпізнавання, яка базується на реалізації цього принципу, визначається просторовим розміщенням окремих кластерів. Якщо кластери, що відповідають різним класам, рознесені далеко один від одного, то можна користуватися простими схемами розпізнавання, наприклад, класифікацією за принципом мінімальної відстані.

Інший підхід відомий як метод потенціальних функцій, відрізняється тим, що об'єднуються деякі розмиті множини, які описуються так званою потенціальною функцією. При цьому здійснюється апроксимація не розв'язуючою, а дискримінантною функцією. Її значення вираховується шляхом складання значень «потенціалу», який зменшується в міру зникання від деяких центрів, що вибираються в процесі навчання. Цей підхід є еквівалентним апроксимації дискримінантної функції за допомогою функціонального ряду.

Суть полягає в тому, що на просторі вхідних векторів X задається функція, яка називається «потенціалом». Потенціал визначається наближенням двох точок і задається як функція відстані між точками. Потенціальна функція така, що вона монотонно зменшується із збільшенням відстані.

Отже, результатом побудови є потенціальне поле, яке розбиває весь простір на дві частини: де значення додатні і від'ємні. Поверхня, на якій потенціал дорівнює нулю, називається розділяючою [9].

Евристичний метод

За основу евристичного підходу взяті інтуїція і досвід людини: в ньому використовуються принципи перерахування членів класу і загальні властивості. Зазвичай системи, побудовані такими методами, включають набір специфічних процедур, розроблених для конкретних задач розпізнавання. Хоча евристичний підхід відіграє велику роль в розпізнаванні, небагато може бути сказано про

загальні принципи синтезу, бо розв’язання кожної конкретної задачі потребує використання специфічних прийомів розробки. Це означає, що структура і якість евристичної системи в значній мірі визначається талантом та досвідом роботи тих, хто її розробляє.

У більшості випадків зображення одного класу зберігають більш-менш постійні розміри і форми. У цих випадках розпізнавання можна здійснити шляхом порівняння зображень зі зразками. Такі зразки називають масками, трафаретами, еталонами.

Найпростіший підхід до розпізнавання образів базується на порівнянні їх з еталонами. У цьому випадку деяка множина образів, по одному з кожного класу образів, знаходиться в пам’яті машини. Вхідний образ, який потрібно розпізнати (невідомого класу) порівнюється з еталоном кожного класу.

Класифікація базується на раніше вибраному критерії співставлення подібності. Іншими словами, якщо вхідний образ краще відповідає еталону і-го класу образів, ніж у будь-якому іншому еталоні, то вхідний образ класифікується як належність до і-го класу образів. Такий підхід використаний в більшості випадків для читання друкованих літер і банківських чеків. Недоліком цього підходу, тобто співставлення з еталоном, є те, що важко вибрати еталон, який найбільше підходить з кожного класу образів і встановити необхідний критерій відповідності. Ці труднощі особливо суттєві, коли образи, які належать одному класу, можуть мати значні спотворення.

Типовим прикладом цього явища є розпізнавання рукописних літер. Більш вдосконалений підхід ґрунтується на тому, що замість порівняння вхідного образу з еталоном, класифікація базується на деякій множині відібраних вимірів. Ці відібрані виміри називаються ознаками, і є малочутливими щодо звичайних змін і спотворень. У цьому випадку розпізнавання образу можна розглядати в двох задачах.

Перша задача ґрунтується у визначенні, які вимірювання повинні були б бути зроблені на вхідному образі. Звичайний розв’язок задачі про те, що міряти,

є певною мірою суб’єктивний, а також залежать від практичних обставин. На сьогодні дуже мало зроблено в конструюванні загальної теорії вибору вимірювальних ознак.

Друга задача розпізнавання образів базується на класифікації (тобто основу на вимірюваннях відібраних ознак [9].

1.4. Елементарні методи ідентифікації

Ідентифікація за кількістю точок об’єкта. Метод заснований на порівнянні кількості точок в периметрах об’єкта, що ідентифікується і еталону.

Використовується перший рівень подання ознак об’єкта .

тут η_S и η_I є кількістю точок контуру еталону і відповідно об’єкта що тестується, а ε_N — класифікаційний допуск, який визначає точність порівняння об’єктів.

Ідентифікація за контуром не інваріантна до масштабу і не може розділяти об’єкти за формою, однак її можна використовувати на попередньому етапі, коли потрібно класифікувати об’єкти по кількості точок їх периметру. Наприклад, відділити великі об’єкти з кількістю точок 100 і більше від малих, створених перешкодами, які складаються із 5-10 точок.

Ще одним напрямком, де метод може бути використаним, є необхідність ідентифікації об’єктів, описання форми яких не може бути отриманим. Прикладом таких образів можуть бути дефектом фарбування тканини, коли на однорідному фоні текстури тканини знаходяться об’єкти неправильної форми, які підлягають розпізнаванню. Опису форми таких дефектів немає і не може бути, однак відомо, що дефектом є об’єкт, який має деякі лінійні розміри, що перевищують деякий поріг. Їх значення легко перераховуються в кількість точок периметру. Припускається, що периметр дефекту значно більший, ніж контури елемента текстури дослідної тканини [10].

Класифікація за габаритами. Поняття габаритів добре визначено для геометричних фігур з чіткими прямолінійними межами. Для будь–якого відомого об’єкта легко уявити його розташування в просторі і місце, яке він займає. Так легко визначити габарити об’єкта, розташованого на (рис. 1.1). Для цього об’єкта габаритами будуть висота і ширина прямокутника, в який він може бути вписаним. А от для об’єкта розташованого на (рис. 1.2) визначити габарити буде набагато важче.

Рис. 1.1. Геометричний об'єкт з прямокутними формами

Рис. 1.2. Геометричний об’єкт з непрямокутними формами

Метод LH. Нехай r визначена на інтервалі $[0, \quad 360^0]$ з кроком 1^0, тоді назвемо габаритами графічного об’єкта наступну трійку параметрів

таку, що

Визначимо функцію розпізнавання за габаритами λ_{LH} (LH-метод) у вигляді

Тут i_j^{δ} і i_j^{0} — габарити еталону і поточного об’єкта, а $\varepsilon_\ell, \varepsilon_h$, ε_φ — класифікаційні допуски, визначаючі точність розпізнавання за довжино, висотою і кутом повороту об’єкта.

Класифікація за компактністю. Доволі часто форму фігури описують через визначення компактності, яке показує відхилення від її форми від найбільш згрупованого об’єкта на площині в евклідовому просторі. Компактність інваріантна до зміщення, повороту і масштабу, а також до дзеркального відображення, але залежить від щільності (роздільна здатність) зображення і записується як

(1.7)

де C— компактність; L—довжина контуру фігури (кількість точок), а S—площа фігури (кількість точок, обмежена контуром). Теоретично найбільш компактною фігурою є округлість, для якої значення компактності дорівнює .
Але на практиці зручніше брати за значення компактності округлості 1, тобто поділеним на коефіцієнт Тоді межі зміни компактності будуть знаходитися на інтервалі $[1, \infty]$, що є достатньо доброю характеристикою для попередніх етапів ідентифікації в методі почергового зважування.

Функцію розпізнавання за компактністю записують у вигляді

де C_S і C_O — компактність еталону і об'єкта, а ε_C — класифікаційний допуск за компактністю.

Класифікація за площею. Розгляд терміну «площа фігури» призводить до необхідності вирахування подвійного інтегралу виду(0-го моменту об'єкту)

Зазвичай такий інтеграл обраховується шляхом заміни змінних і зведення його до інтегралу за контуром. В дискретному випадку його обрахування проводиться шляхом заміни подвійних інтегралів на подвійну суму. Але для реального об'єкта такий спосіб обрахування не завжди зручний, особливо для об'єктів з невипуклим периметром, також потребує значних обчислювальних затрат, і цей метод уже не можна відносити до елементарних. Використовуючи поняття контурної функції, можна ввести сурогатне розуміння площі об'єкта у вигляді площі, яка лежить під контурною функцією і вираховується як сума площ трикутників, які мають основу на контурній функції, і сторони радіуси-вектори відносно центру тяжіння [10].

1.5. Практичне застосування

Методи розпізнавання образів та відповідні технічні засоби вирішують цілий ряд проблем.

Технічна діагностика. На виробництві часто виникає потреба автоматизувати контроль якості деталей. Задача полягає в тому, щоб виявити,

чи є деталь дефектною, чи ні. Якщо ж з'ясовується, що деталь має дефект, часто потрібно визначити тип цього дефекту.

Медична діагностика. Системи розпізнавання часто використовуються і в медичній практиці. Найтиповіша ситуація полягає в тому, що ті чи інші захворювання діагностуються на основі аналізу кардіограм, рентгенівських знімків і т. п.

Розпізнавання літер. Окрім усього іншого, ця проблема має велике значення для власне комп'ютерних технологій. Системи розпізнавання літер працюють разом зі сканерами – пристроями, які використовуються для введення до комп'ютера друкованих зображень і текстів. При введенні друкованого тексту сканер формує лише графічне зображення; для того щоб створити текстовий документ, з яким може працювати текстовий редактор, необхідно впізнати на цьому зображенні окремі літери. Аналогічно розпізнавання літер є необхідним для підтримки пристроїв рукописного введення. Цими пристроями, зовні схожими на звичайну авторучку, часто комплектуються надпортативні комп'ютери (персональні помічники). Основна мета цих пристроїв – замінити введення з клавіатури, що є незручним для багатьох користувачів.

Розпізнавання мови. Сьогодні інтенсивно розвиваються технології, пов'язані, по перше, з голосовим керуванням комп'ютером, а по друге – з введенням текстів з голосу.

Робототехніка. Застосування методів розпізнавання в робототехніці є абсолютно природним і необхідним, оскільки роботи повинні безпосередньо сприймати зовнішній світ і, відповідно, мати пристрої машинного зору.

Охоронні системи. Застосування методів розпізнавання в охоронних системах пов'язано в першу чергу з проблемою ідентифікації. Наприклад, потрібно ідентифікувати певну особу, щоб визначити, чи має вона право входити на територію, що охороняється. Розвиваються також системи, які вирішують проблему ідентифікації відбитків пальців і т. п. [4].

РОЗДІЛ 2
ОБРОБКА ГРАФІЧНИХ ЗОБРАЖЕНЬ ЗАСОБАМИ EMGU.CV

2.1. Формати збереження графічної інформації

Комп’ютерна графіка— область діяльності, в якій комп'ютери поряд зі спеціальним програмним забезпеченням використовуються в якості інструменту як для створення (синтезу) і редагування зображень, так і для оцифрування візуальної інформації, отриманої з реального світу, з метою подальшої її обробки і зберігання [11]. Існує чотири види комп’ютерної графіки, які відрізняються принципом зберігання і створення зображення:

Растрова графіка - це зображення, складені з пікселів - маленьких кольорових квадратиків, розміщених в прямокутній сітці. Піксель - це найменша одиниця цифрового зображення. Якість растрового зображення безпосередньо залежить від кількості пікселів, з яких воно складається - чим більше пікселів тим більше деталей можна відобразити [12].Основними форматами зберігання растрових зображень є:

TIF. При збереженні ілюстрації в цьому форматі не використовується жоден з видів компресії (стиснення). У цьому форматі отримують максимально можливу ступінь якості та відповідності, збереженої у файлі копії зображення. Це єдиний формат, який використовується в професійному дизайні для зберігання зображень високої якості. Якісні TIF-зображення можуть займати кілька сотень мегабайт. TIF-формат є найкращим вибором при передачі зображень і растрової графіки в векторні програми і видавничі системи.

JPG. Цей формат використовується для стиснення зображення в десятки разів. Формат дозволяє використовувати різні ступені стиснення, роблячи тим самим вибір або в бік збільшення якості, або в сторону зменшення файлу. У професійній поліграфії цей формат не використовується через істотні втрати якості зображення. Для перегляду зображення на екрані монітора або для

роздруківки на принтері якості JPG-формату досить. У форматі JPG використовується метод стиснення jpeg. Цим методом краще стискаються растрові зображення фотографічної якості і погано стискаються логотипи або схеми. У цьому форматі добре і з меншими втратами стискаються великі зображення з високою роздільною здатністю 200-300 ppi і погано стискаються з низьким розширенням 72-150 ppi. Небажано зберігати зображення в JPG-форматі, де важливі всі тонкощі передачі кольору, так як під час стиснення відбувається відкидання деякої колірної інформації.

GIF. Це формат растрової графіки, створений спеціально для КС. Цей формат має метод стиснення, який позначається LZW. Цей формат має обмежену палітру кольорів. Основне обмеження GIF полягає в тому, що кольорове зображення може мати трохи більше 256 кольорів, тому кольори в цьому форматі стають грубими, а саме зображення зернистим. Не використовується в поліграфії і не рекомендується для зображень, призначених для монітора або принтера. В GIF-форматі пікселі зображення записуються через рядок. За цією технологією, отримавши тільки частина файлу вже можна побачити зображення цілком, але з низькою якістю. У випадку з контрастністю зображення з чіткими кордонами між квітами або у випадку з однотонним зображенням при використанні цього формату велика ступінь стиснення, ніж JPG, причому якість не змінюється.

PNG. Це формат, розроблений відносно недавно, призначений для того, щоб замінити GIF-формат. У ньому використовується метод стиснення без втрат якості, який позначається deflate. Стислі індексовані файли (з невеликою кількістю кольорів) мають менший розмір порівняно з аналогічними GIF-файлами. Глибина кольору в файлах може бути будь-якою до 48 біт. На відміну від GIF-формату PNG підтримує не тільки прозорість, але і напівпрозорість. У файловому форматі PNG записана інформація про гамах корекції.

PDF. Це незалежний від графічних програм формат для створення електронної документації, презентацій, а також для передачі графіки через

мережі. PDF-файли створюються шляхом конвертації з PostScript-файлу або функцією експорту. Програми Photoshop, Illustrator можуть створювати тільки лише односторінковий PDF-файл. Всі дані в форматі PDF можуть стискатися. Причому до різного типу інформації застосовуються різні типи стиснення. Файл PDF може бути оптимізований - з нього видаляються повторювані елементи, встановлюється посторінковий порядок завантаження сторінок з пріоритетом спочатку для тексту, потім для графіки. Формат PDF використовується для передачі по мережах в компактному вигляді графіки і тексту.

PSD. Це внутрішній формат програми Photoshop. Став підтримуватися ще більшою кількістю графічних програм. Цей формат дозволяє записувати зображення з багатьма шарами і додатковими альфа-каналами, а також з каналами простих кольорів і контурами і іншою специфічною інформацією.

BMP. Растровий формат, який є рідним графічним форматом Windows. Підтримується всіма редакторами. У цьому форматі зберігаються невеликі растрові зображення, призначені для використання в системі Windows. Це формат невисокої якості і з низьким ступенем стиснення. Його не рекомендується використовувати не для web-дизайну, не для передачі [12].

Векторна графіка. Різноманітність форматів векторної графіки значно менша, і практично кожний векторний графічний редактор використовує свій власний формат зберігання зображень. Наприклад:

WMF. (англ. Windows MetaFile – метафайл Windows) – універсальний формат для програм, що працюють в ОС Windows. Використовується для зберігання колекції графічних зображень Microsoft Clip Gallery. Можливі розширення імен файлів – WMF, EMF, WMZ, EMZ.

CGM. (англ. Computer Graphic Metafile – метафайл комп'ютерної графіки) – широко використовується як стандартний формат векторних графічних даних в мережі Інтернет. Стандартне розширення імен файлів CGM.

SVG.(Scalable Vector Graphics – векторна графіка, що масштабується) – універсальний формат для двовимірної графіки, який дає змогу з високою якістю зберігати у файлі текст, графічне зображення і анімацію. Файли можуть додатково стискатися програмами-архіваторами. Опрацьовується практично всіма векторними графічними редакторами. Широке застосування отримав у інженерній графіці і при розробці веб-сайтів. Стандартне розширення імен файлів SVG.

CDR. (англ. CorelDRaw files – файли CorelDraw) – стандартний формат файлів векторного графічного редактора CorelDraw. Зображення у файлі може мати кілька сторінок, дає змогу зберігати не тільки векторну графіку, а й текст і растрові зображення. Максимальний розмір малюнка 45 х 45 м. Файли даного формату можуть мати розширення імені CDR або CDT.

AI. (англ. Adobe Illustrator files – файли Adobe Illustrator) – стандартний формат файлів редактора векторної графіки Adobe Illustrator. Зберігає у файлі тільки одну сторінку. Файли мають розширення імені AI.

Сумісність форматів векторної графіки дуже низька. Складність перетворення даних з одного векторного формату в інший полягає у використанні різними програмами різних алгоритмів побудови графічних примітивів [13].

Фрактальна графіка. Фрактальна графіка - одна із перспективних і таких, що найшвидше розвивається, видів комп'ютерної графіки.

Фрактал - структура, що складається з частин, подібних цілому. Одним з основних властивостей є самоподоба. (Фрактус - складається з фрагментів).

Фрактальна графіка, як і векторна обчислюється, але відрізняється тим, що ніякі об'єкти в пам'яті не зберігаються. Зображення будується за рівнянням, або системі рівнянь, тому нічого крім формули зберігати не треба. Змінивши коефіцієнти можна отримати зовсім іншу картину.

Таким чином, дрібні об'єкти повторюють властивості всього об'єкта. Процес спадкування можна продовжувати до нескінченності.

Отриманий об'єкт носить назву - фрактальної фігури. Абстрактні композиції можна порівняти зі сніжинкою, з кристалом.

Фрактальна графіка заснована на математичних обчисленнях. Базовим елементом фрактальної графіки є сама математична формула, тобто ніяких об'єктів в пам'яті комп'ютера не зберігається і зображення будується виключно по рівняннях.

Фрактальними властивостями володіють багато об'єктів живої і неживої природи (сніжинка, гілка папороті).

Здатність фрактальної графіки моделювати образи обчислювальним шляхом часто використовують для автоматичної генерації незвичайних ілюстрацій [14].

Тривимірна графіка. 3D графіка або тривимірна графіка - це один з розділів комп'ютерної графіки, комплекс прийомів і інструментів, які дозволяють створити об'ємні об'єкти за допомогою форми і кольору. Від двомірних зображень вона відрізняється тим, що має на увазі побудову геометричної проекції тривимірної моделі сцени (віртуального простору) на площину, робиться це за допомогою спеціалізованих програм. Отримана модель може відповідати об'єктам реального світу (наприклад, будівля, людина, автомобіль, астероїд) або бути цілком абстрактною (проекція чотиривимірного фрактала).

Сьогодні 3D графіка міцно увійшла в багато сфер нашого життя - це: будівництво (візуалізація об'ємних архітектурних зображень будівель, об'єктів, інтер'єру, екстер'єру), виробництво (об'єктне моделювання), телебачення (модельовані фото в глянцевих журналах, відеоролики, спецефекти в кіно),ігрова індустрія (3D-анімація і віртуальні світи, розробка комп'ютерних ігор), поліграфія (створення поліграфічної продукції),реклама (електронні презентації і каталоги, рекламні щити та ін.) і т.д.

3D-графіка - один з найбільш ефективних інструментів в рекламі, що дозволяє розширити вплив на потенційного клієнта і підвищити якість пропонованої як реальному, так і у віртуальному світі [14].

2.2. Основні методи обробки зображень

Ще в середині XX століття обробка зображень була здебільшого аналоговою і виконувалась оптичними пристроями. Подібні оптичні методи досі важливі, в таких областях як, наприклад, голографія. Тим не менш, з різким зростанням продуктивності комп'ютерів, ці методи все в більшій мірі витіснялися методами цифрової обробки зображень. Методи цифрової обробки зображень зазвичай є більш точними, надійними, гнучкими і простими в реалізації, ніж аналогові методи. У цифровій обробці зображень широко застосовується спеціалізоване обладнання, таке як процесори з конвеєрною обробкою інструкцій та багатопроцесорні системи. Особливою мірою це стосується систем обробки відео. Обробка зображень виконується також за допомогою програмних засобів комп'ютерної математики, наприклад, MATLAB, Mathcad,Maple,Mathematica таін. Для цього в них використовуються як базові засоби, так і пакети розширення Image Processing [15].

Основні методи обробки зображень. Більшість методів обробки одновимірних сигналів (наприклад, медіанний фільтр) застосовні і до двомірних сигналів, якими є зображення. Деякі з цих одновимірних методів значно ускладняються з переходом до двовимірних сигналів. Обробка зображень вносить сюди кілька нових понять, таких як зв'язність і ротаційна інваріантність, які мають сенс тільки для двомірних сигналів. В обробці сигналів широко використовуються перетворення Фур'є, а також вейвлет-перетворення і фільтр Габора. Обробку зображень поділяють на обробку в просторовій області (перетворення яскравості, гама-корекція і т. д.) і частотній (перетворення Фур'є, і т. д.). Перетворення Фур'є дискретної функції

(зображення) просторових координат є періодичним по просторових частотах з періодом 2pi [15].

2.3. Бібліотека функцій для роботи з графічними файлами EMGU.CV

EMGUCV (бібліотека комп'ютерного зору з відкритим кодом) — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки та аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відслідковування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення спільних елементів на різних зображеннях [16].

Бібліотека розроблена Intel і нині підтримується WillowGarage та Itseez. Сирцевий код бібліотеки написаний мовою C# і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях [17].

Офіційно проект EMGUCV був запущений у 1999 році за ініціативою IntelResearch з ціллю розвивати CPU-ресурсомісткі додатки. Основними вкладниками у проект була Intel'sPerformanceLibraryTeam та певна кількість експертів з чисельної оптимізації у Inter Russia. На перших етапах розвитку OpenCV основними задачами бібліотеки були:

1. Розвивати дослідження у напрямку комп'ютерного зору, забезпечуючи добре оптимізований та відкритий код бібліотеки.
2. Поширювати знання у сфері комп'ютерного зору, забезпечуючи загальну інфраструктуру, яку б могли розвивати розробники, таким чином код ставатиме більш легким для сприйняття та обміну.
3. Розвивати засновані на роботі з комп'ютерним зором комерційні додатки, створюючи незалежну від платформи, оптимізовану та безкоштовну

бібліотеку. Для цього використовувалася ліцензія, яка не вимагала від таких комерційних додатків бути відкритими.

Перша альфа-версія EMGUCV була оприлюднена на IEEE конференції з комп'ютерного зору й розпізнавання образів у 2000 році, і п'ять бета-версій було випущено у період між 2001 і 2005 роками. Перша версія 1.0 була випущена у 2006 році. У середині 2008 року, EMGUCV отримала корпоративну підтримку від Willow Garage і знову перейшла у стадію активної розробки. «Пре-релізна» версія 1.1 була випущена у жовтні 2008 року.

Другий великий випуск EMGUCV відбувся у жовтні 2009 року. EMGUCV 2 включала в себе серйозні зміни у інтерфейсі C++. Ці зміни спрямовані на більш прості, тип-безпечні моделі, додавання нових функцій, і кращу реалізацію існуючих моделей в плані швидкодії (особливо на багатоядерних системам). Офіційні релізи надалі відбуваються кожні 6 місяців[3]і розробкою займається незалежна команда з Росії, яка підтримується комерційними корпораціями.

У серпні 2012 року, підтримку EMGUCV було передано некомерційній організації, EMGUCV.org.

Бібліотека містить понад 2500 оптимізованих алгоритмів, серед яких повний набір як класичних так і практичних алгоритмів машинного навчання і комп'ютерного зору. Алгоритми EMGUCV застосовують у таких сферах:

1. Аналіз та обробка зображень.
2. Системи з розпізнавання обличчя.
3. Ідентифікації об'єктів.
4. Розпізнавання жестів на відео.
5. Відслідковування переміщення камери.
6. Побудова 3D моделей об'єктів.
7. Створення 3D хмар точок зі стерео камер.
8. Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю.

9. Система взаємодії людини з комп'ютером.
10. Пошуку схожих зображень із бази даних.
11. Усування ефекту червоних очей при фотозйомці зі спалахом.
12. Стеження за рухом очей.
13. Аналіз руху.
14. Ідентифікація об'єктів.
15. Сегментація зображення
16. Трекінґ відео.
17. Розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності та інші [17].

РОЗДІЛ 3
СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ ДОДАТКІВ

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології WindowsForms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються MicrosoftWindows, WindowsMobile, WindowsPhone, WindowsCE, .NETFramework, .NETCompactFramework та MicrosoftSilverlight [18].

3.1. Технологія WPF

Технологія WPF (Windows Presentation Foundation) є частиною екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів [19].

Якщо при створенні традиційних додатків на основі WinForms за створення елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF засновані на DirectX. У цьому полягає ключова особливість рендерингу графіки в WPF: використовуючи WPF, значна частина роботи по відображенні графіки, як найпростіших кнопок, так і складних 3D-моделей, лягає на графічний процесор відеокарти, що також дозволяє скористатися апаратним прискоренням графіки.

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

Переваги WPF

- Використання традиційних мов .NET-платформи - C # і VB.NET для створення логіки додатка.
- Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованої на xml і представляє альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C # / VB.NET
- Незалежність від розширення екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різним розширенням.
- Нові можливості, яких складно було досягти в WinForms, наприклад, створення трьохвимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін.
- Хорошу взаємодію з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms.
- Багаті можливості по створенню різних додатків: це і мультимедіа, і двомірна і триьохвимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого.
- Апаратне прискорення графіки - незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, всі компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіку більш плавною.
- Створення додатків під безліч ОС сімейства Windows - від Windows XP до Windows 10.

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення додатків з великою кількістю

тривимірних зображень, перш за все для ігор, краще використовувати інші засоби - DirectX або спеціальні фреймворки, такі як Monogame або Unity.

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

Архітектура WPF

Схематично архітектуру WPF можна представити таким чином: архітектура WPF. WPF розбивається на два рівня: managed API і unmanaged API (рівень інтеграції з DirectX). Managed API (керований API-інтерфейс) містить код, що виконується під управлінням загальномовного середовища виконання .NET - Common Language Runtime. Цей API описує основний функціонал платформи WPF і складається з наступних компонентів:

- **PresentationFramework.dll**: містить всі основні реалізації компонентів і елементів управління, які можна використовувати при побудові графічного інтерфейсу.
- **PresentationCore.dll**: містить всі базові типи для більшості класів з PresentationFramework.dll.
- **WindowsBase.dll**: містить ряд допоміжних класів, які застосовуються в WPF, але можуть також використовуватися і поза даною платформою.
- Unmanaged API використовується для інтеграції вищого рівня з DirectX:
- milcore.dll: власне забезпечує інтеграцію компонентів WPF з DirectX. Даний компонент написаний на некерованому коді (C / C ++) для взаємодії з DirectX.
- **WindowsCodecs.dll**: бібліотека, яка надає низькорівневу підтримку для зображень в WPF.00

Ще нижче власне знаходяться компоненти операційної системи і DirectX, які проводять візуалізацію компонентів програми, або виконують іншу низько

рівневу обробку. Зокрема, за допомогою низько рівневому інтерфейсу Direct3D, який входить до складу DirectX, відбувається трансляція.

Тут також на одному рівні знаходиться бібліотека user32.dll. І хоча вище говорилося, що WPF не використовує цю бібліотеку для рендерингу і візуалізації, проте для ряду обчислювальних задач (що не включають візуалізацію) дана бібліотека продовжує використовуватися [19].

3.2. Технологія JSON

Для зберігання оброблених фотографій у нашому застосунку ми використовували технологію JSON.

JSON (JavaScript Object Notation, об'єктна нотація JavaScript) - формат даних, призначений спеціально для використання JavaScript-кодом, виконуваних на веб-сторінках всередині браузера. Це формат даних за замовчуванням, що використовуються службами ASP.NET AJAX, створені в Windows Communication Foundation (WCF).

Його також можна використовувати при створенні служб AJAX без інтеграції з ASP.NET; в даному випадку форматом за умовчанням є XML, проте можна вибрати і JSON [20].

Нарешті, якщо потрібна підтримка JSON, однак створювана служба не є службою AJAX, серіалізатор DataContractJsonSerializer дозволяє безпосередньо серіалізувати об'єкти .NET в дані JSON і десеріалізувати такі дані назад в екземпляри типів .NET.

При роботі з JSON підтримуються ті ж (за деякими винятками) типи .NET, що підтримуються серіалізатором DataContractSerializer. Список типів, див. Розділ Types Supported by the Data Contract Serializer. До них відноситься більшість примітивних типів, більшість типів масивів і колекцій, а також складні типи, в яких використовуються атрибути DataContractAttribute і DataMemberAttribute.

Порядок членів даних при використанні JSON не має значення. Зокрема, навіть якщо заданий атрибут Order, JSON-дані все одно можна серіалізувати в будь-якому порядку.

Тип JSON при десеріалізації не обов'язково повинен відповідати вказаним у таблиці JSON. Наприклад, тип Int зазвичай зіставляється числу JSON, однак може бути успішно десеріалізований з рядка JSON, за умови, що рядок містить допустиму кількість. Тобто, і { "q": 42}, і { "q": "42"} допустимі, якщо є член даних типу Int з ім'ям "q".

Поліморфна серіалізація полягає в можливості серіалізувати похідний тип там, де очікується його базовий тип. Це підтримується для серіалізації JSON з WCF, аналогічно тому, як підтримується серіалізація XML. Наприклад, можна серіалізувати MyDerivedType де MyBaseType очікується, або серіалізувати Int де очікується Object.

При десеріалізації похідного типу там, де очікується базовий тип, інформація типу може бути втрачена, за винятком випадків десеріалізації складних типів. Наприклад, при серіалізації типу Uri там, де очікується тип Object, буде отриманий рядок JSON. Якщо цей рядок потім десеріалізувати назад в тип Object, буде повернуто .NET-тип String. Десеріалізатор не знає, що рядок спочатку мав тип Uri. Як правило, коли очікується тип Object, всі рядки JSON десеріалізуются як рядки .NET, а всі масиви JSON, використовувані для серіалізації колекцій, словників і масивів .NET, десеріалізуются як .NET-об'єкти Array типу Object, незалежно від того, якими були вихідні типи. Логічний тип JSON зіставляється .NET-типом Boolean. Однак, коли очікується тип Object, числа JSON десеріалізуются в .NET-типи Int32, Decimal або Double (найбільш підходящий тип вибирається автоматично) [21].

При десеріалізації в тип інтерфейсу DataContractJsonSerializer виконує десеріалізацію так, як ніби оголошений тип - об'єкт.

При роботі зі своїми власними базовими і похідними типами звичайно потрібно використовувати атрибути KnownTypeAttribute,

ServiceKnownTypeAttribute або еквівалентний механізм. Наприклад, якщо у вас є клас Animal в яком повертається значення і він фактично повертає екземпляр Cat (похідний від Animal), необхідно або застосувати KnownTypeAttribute, щоб Animal тип або ServiceKnownTypeAttribute для операції і вкажіть Cat тип в цих атрибутах [22].

Приклад створення власного класу для серіалізації і десеріалізації зображень (рис. 3.1).

Рис. 3.1. Клас для JSON.

РОЗДІЛ 4
ПРОБЛЕМА АВТОМАТИЗОВАНОЇ ІДЕНТИФІКАЦІЇ ВІДБИТКІВ СЛІДІВ

4.1. Постановка проблеми

Суттєвий вплив освітленості може істотно понизити якість роботи систем автоматизованого розпізнавання слідів. Зокрема, при низькому рівні освітлення заднього або переднього плану, розпізнавання і виявлення слідів виконується значно гірше, це пов’язано із тінями на фото, вони можуть зіпсувати певні шаблони сліду. З іншого боку, занадто яскраве освітлення теж впливає на шаблон. Щоб уникнути проблем з освітленням, використовують додаткові підходи обробки зображення, які нормалізують освітлення через створення гістограми, або методів машинного навчання, які оброблюють зображення з поточною загальною інтенсивністю зображення.

Також проблемою при розпізнаванні може бути пошкодження сліду, зверху впав предмет який деформував його, або ж нецілісність сліду, при внесенні в базу відбиток внесеться з усіма дефектами. Але під час ідентифікації очікується що й і шуканий зразок матиме дефект.

В ідеалі знімок потрібно робити перпендикулярно до поверхні на якій він знаходиться, але ж бувають випадки коли фотографія зроблена під кутом, тоді виникають певні складності при ідентифікації слідів. Вирішується дана проблема достатньою кількістю еталонних зображень.

Роздільна здатність зображення, є важливим фактором при здійсненні розпізнавання, чим чіткіше зображення отримується на вході до системи, тим достовірніші результати ми отримаємо на виході. Відстань з якої фотографують сліди, відіграє важливу роль під час розпізнавання сліду, якщо фото зроблене з більшої відстані, то слід здаватиметься меншим порівняно з аналогічним фото з ближчої дистанції. Розмір відбитку взуття, є важливим фактором під час

розпізнавання, наприклад взуття розміру 40 та розміру 34 може мати однаковий протектор, і система може ідентифікувати ці два сліди як одну людину, щоб вирішити дану проблему потрібно при фотографуванні сліду поряд класти лінійку щоб експерт який вноситиме інформацію в базу даних, добавив також в сантиметрах висоту сліду, що дозволить автоматично визначити розмір взуття людини.

Проблемою є автоматичне виявлення правильного порогу визначення бінаризації фото, етап на якому зображення транслюється до чорно білого вигляду, для максимально точного виявлення цю роботу повинен виконувати експерт вручну, регулюючи рівень контрастності повзунком в додатку, автоматизація даного етапу потребує набагато більшого поглиблення в дану проблематику та підключення машинного навчання.

4.2. Використання методу контурних функцій для прийняття рішення в процесі автоматизованої ідентифікації відбитків слідів

У першому розділі вже згадувалися алгоритми отримання границь. Отримані границі досить просто перетворюються в контури. Для алгоритму Кенні це відбувається автоматично, для інших алгоритмів потрібна додаткова бінаризація. Отримати контур для бінарного алгоритму можна наприклад алгоритмом жука.

Контур є унікальною характеристикою об'єкта. Часто це дозволяє ідентифікувати об'єкт по контуру. Існує потужний математичний апарат, що дозволяє це зробити. Апарат називається контурним аналізом (рис. 4.1) .

Рис. 4.1. Схема роботи контурних функцій

Алгоритм контурних функцій чудово підходить для ідеальних умов отриманого зображення, він дуже швидко працює і має зрозумілу логіку.

Виходячи з викладеного вище матеріалу, можна зробити висновок, що контурний аналіз є універсальним способом класифікації об'єктів, заснованим на аналізі форми.

Контурний аналіз є перспективним методом для вирішення задачі класифікації графічних об'єктів.

Проте, необхідно відзначити, що аналізовані контури можуть бути істотно перекручені в результаті впливу зовнішніх факторів. Найбільший вплив надає наявність тіні від яскравого сонячного світла. З цієї причини подальші зусилля будуть спрямовані на розвиток методу корекції вихідного контуру з метою збільшення властивостей інваріантності до зовнішніх умов фотозображення [23].

Розглянемо поняття інформаційної складової деякого вектора властивостей. Нехай, вона визначена як множина властивостей *I*, яка складається із підмножин $\mathcal{I}^i \subset \mathcal{I}, i = \overline{0,n}$. Індекс i означає рівень підмножини. Так, наприклад, $\mathcal{I}^0$ буде підмножиною первинних ознак, $\mathcal{I}^1$ — вторинних ознак і т. д. Таким чином, вся інформаційна частина вектора властивостей буде складатися із

$$\mathcal{I} = \bigcup_{i=0}^{n} \mathcal{I}^{(i)}; \quad \mathcal{I}^{(i)} \bigcap \mathcal{I}^{(j)} = \emptyset; \quad i \neq j. \tag{4.1}$$

Крім того, можлива ситуація, коли

$$\mathcal{I}^{\dagger}((i)) = F(\mathcal{I}^{\dagger}((i-1))), \mathcal{I}^{\dagger}((i-2)), \ldots). \tag{4.2}$$

Причому в реальних випадках деякі підмножини $\mathcal{I}^{(i)}$ можуть бути пустими або не використовуватися, а процес розпізнавання може бути побудований з використанням як ознак окремих рівнів, так і їх комбінацій.

Нехай множина координат $(x_i, y_i), i = \overline{1,k}$, створює контур деякого графічного об’єкта на площині (рис. 4.2).

Рис. 4.2. Об’єкт із 16 точок

Множина $\mathcal{J}^{(0)}$ первинних векторів властивостей визначена як множина точок на площині:

$$\mathcal{J}^{(0)} = \left\{ i_l^{(0)} = (x_l, y_l), \qquad l = \overline{1,k} \right\}, \tag{4.3}$$

Які створюють контур графічного об'єкта, а k кількість точок периметру цього об'єкта.

Нехай визначено множину $\mathcal{J}^{(0)}$ деякого об'єкта. Елемент множини властивостей $\xi_0^{(1)} = v_0(\mathcal{J}^{(0)}) \in \mathcal{J}^{(1)}$ називають центром тяжіння об'єкта, якщо

$$\xi_0^{(1)} = \left(x_c = \frac{1}{k}\sum_{l=1}^{k} x_l, \qquad y_c = \frac{1}{k}\sum_{l=1}^{k} y_l \right). \tag{4.4}$$

Вводиться формальний опис сигнатурної функції наступним чином. Нехай визначені елементи множин $\mathcal{J}^{(0)}$ і $\mathcal{J}^{(1)}$ для деякого об'єкта. Функціоналом перетворення сигнатури контуру $r = v_3(v_2(v_1(\mathcal{J}^{(0)}, \mathcal{J}^{(1)})))$ називають наступну послідовність функцій v , застосованих до $\mathcal{J}^{(0)}$ і$\mathcal{J}^{(1)}$:

1. $\mathcal{J}^{(0')} = v_1(\mathcal{J}^{(0)}, \mathcal{J}^{(1)})$, де v_1 — функція перетворення декартових координат в полярні у вигляді $(R, \varphi) = \left\{ \left((x - x_c)^2 + (y - y)^2\right)^{\frac{1}{2}}, arctg\frac{x}{y} \right\}$,

тобто отримання набору $\mathcal{J}^{(0)}$ в полярних координатах $\mathcal{J}^{(0')} = (R_l, \varphi_l), l = \overline{1,k}.$

2. $\mathcal{J}^{(0'')} = v_2\left(\mathcal{J}^{(0')}\right)$, де v_2 — функція сортування множини $\mathcal{J}^{(0')}$ за кутом φ в порядку його зростання. Далі припускається, що якщо зустрічаються дві і більше точок з однаковим кутом, то вибирається точка з максимальним значенням вектора R.

3. $r = v_3\left(\mathcal{J}^{(0'')}\right)$,де v_3 —функція інтерполяції точок об'єкта по всьому периметру з заданим фіксованим кроком, визначеним як $\Delta\varphi = 0.5, 1, 2 \ldots.$ кутових градусів, що дає 720, 360 і 180… точок розгортання дослідного контуру в залежності від потрібної точності представлення об'єкта. Очевидно, що перетворення v_1 і v_2 будуть однозначними лише для випуклих контурів графічних об'єктів. Однак необхідно зауважити, що ці перетворення здійснюються відносно центру тяжіння об'єкта. Тобто поняття випуклості тут визначається через число ліній контуру об'єкта, який буде перетинати промінь,

що іде від його центра тяжіння. Для випуклого в цьому сенсі об'єкта це число завжди повинно бути рівне 1. В іншому випадку виникають неоднозначності в виконанні перетворення v_1 і v_2, що вирішуються за визначенням (3). Приклади розгортки контурних функцій (див. рис. 4.3).

Рис. 4.3. Приклади розгорток контурних функцій [24]

Нехай на [0, $360°$] визначена функція r. Нормованою функцією r_N будемо називати $r-$ функцію, нормовану відносно її максимуму на [0, $360°$]. Коефіцієнтом нормування назвемо число $\eta = \frac{1}{r_{\max}}$, де $r_{\max}$ — максимальне значення, що отримує функція r на інтервалі [0, $360°$]. Дальше в цілях спрощення запису будемо прибирати індекс N, беручи до уваги, що функція r завжди нормована [24].

Введемо поняття узагальненої обробки зображень I_m як деякої послідовності функцій:

$$I_{m_l} = f_l\left(I_{m_{l-1}}, \theta_{f_l}, t_{f_l}\right), l = \overline{1, L}, \tag{4.6}$$

де f_l —функція перетворення зображень, які створюють множину F;

$\theta_{f_l} = (\beta_1, \beta_2, \ldots, \beta_m)$—вектор параметрів для функції f_l з множини Θ;

а $\tau_f = \left(t_{f_1}, t_{f_2}, \ldots, t_{f_L}\right) \in T$ — вектор номерів t_{f_l} функції f_l із F, що визначає послідовність їх виклику. Множина векторів властивостей об'єктів $\omega_{I_m} = (\overline{x_i}, i = \overline{1, k})$, отриманих із зображень I_m, будемо записувати

$$\omega_{I_m} = \{I_\eta, \theta_f, \tau_f\}, \eta = \overline{1, N}, \tag{4.7}$$

Де індекс η позначає об'єкт множини ω_{I_m}, I_η — інформаційну частину η -го вектора властивостей, а $\theta_f = \{\theta_{f_l}, l = \overline{1, L}\}$.

Запишемо тепер функцію розпізнавання λ, засновану на обчисленні деякої метрики в просторі ознак, таку що

$$\lambda = \begin{cases} 1, & \rho(x_{I_m}, x_s) < \varepsilon, \\ 0, & \rho(x_{I_m}, x_s) < \varepsilon. \end{cases} \tag{4.8}$$

Тут x_{I_m} і $x_s I_\eta$ — вектори властивостей поточного і еталонного об'єктів; ρ —

Деяка метрика, а ε — класифікаційний допуск розпізнавання. Якщо$\rho < \varepsilon$, то рахуватимемо, що вектор x_{I_m} належить до класу ω_s, якщо $\rho \geq \varepsilon$, то не належить. Також вважатимемо, що процедурна частина вектору властивостей об'єкта x_{I_m} в цій функції не використовується.

Нехай множина ω_{I_m} створена векторами ознак всіх графічних об'єктів, виділених із зображення I_m. Представимо процес їх ідентифікації у вигляді послідовності призначення різних функцій розпізнавання λ до елементів множини ω_{I_m}:

$$\omega_{I_m}^{q} = \lambda_q\left(\omega_{I_m}^{q-1}, \overline{x_s}, \overline{\theta_{\lambda_q}}, \varepsilon_{\lambda_q}\right), q = \overline{1, Q}, \tag{4.9}$$

$$\omega_{I_m}^{q} \subset \omega_{I_m}^{q-1} \subset \cdots \omega_{I_m}^{0}, \tag{4.10}$$

Де $\omega_{I_m}^{q-1}$ і $\omega_{I_m}^{q}$— підмножини із ω_{I_m} на $q-1$ і $q-m$ етапах розпізнавання; λ_q — функції, які формують множину методів розпізнавання $\bigwedge$; $\theta_{\lambda_q} = (\gamma_1, \gamma_2, \ldots, \gamma_m)$— вектор параметрів функції λ_q; ε_{λ_q}— класифікаційний допуск для λ_q; Q—число використовуваних методів розпізнавання. Формула (4.10) означає, що нерозпізнані на кроці qоб'єкти відкидаються в подальшому розгляді.

Порядок застосування функцій λ_q в (4.9) будемо називати методом послідовного зважування і представляти як

$$x_{\Sigma} = \{\lambda_1, \lambda_2, \ldots, \lambda_Q\}. \tag{4.11}$$

В (4.9)-(4.11) початкова множина векторів ознак ω_{I_m} приймається за $\omega_{I_m}^{0}$, а результатом виконання (4.9) буде шукана множина розпізнаних об'єктів $\omega_{I_m}^{R}$. Тепер вектор властивостей еталону можна записати у вигляді

$$x_s = \{I_s, \theta_\lambda, \tau_\lambda\}.$$

Тут $\tau_\lambda = (t_{\lambda_1}, t_{\lambda_2}, \ldots, t_{\lambda_Q})$—вектор номерів в послідовності методів λ_Σ із (4.11);

I_s—інформаційна частина вектору еталона; $\theta_\lambda = \{\theta_{\lambda_1}, \theta_{\lambda_2}, \ldots, \theta_{\lambda_Q}, \varepsilon_\Sigma\}$—розширений вектор параметрів функції λ_q, індекс Мзалежить від τ_{λ_q} і визначає кількість параметрів в функціях λ_q.

Тепер процес розпізнавання, який полягав в перенесені об’єктів зображення $x_{I_{m,\eta}}$, $\eta = \overline{1,N}$, $\forall x_{I_{m,\eta}} \in \omega_{I_m}$ до деякого класу ω_s, можна записати

$$\beta(\omega_{I_m}, \lambda_q, x_s), \qquad q = \overline{1,Q}, \tag{4.12}$$

де β —функціонал ідентифікації, в якому визначена послідовність методів розпізнавання λ, задана τ_{λ_Σ} для множини ω_{I_m} [24].

РОЗДІЛ 5
РЕАЛІЗАЦІЯ ДОДАТКУ

5.1. Налаштування середовища розробки

Для інсталяції VisualStudio необхідно завантажити інсталяційний файл з офіційного сайту Unity (https://visualstudio.microsoft.com/vs/) . Перед завантаженням розробнику необхідно вирішити: завантажувати безкоштовну , комерційну чи професійну версію. Меню вибору можна побачити на рис.4.1.

Після того, як розробник вирішив, він має обрати необхідне поле на сайті. Найкращим варіантом для студентів і початківців, а також для всіх, хто вже має досвід роботи з середовищами розробки, але тільки починає працювати зVisualStudio, пропонується завантажити безкоштовну версію.

При першому завантаженні розробнику надається місяць безкоштовного використання усіх функцій, за який розробнику надається змога зрозуміти, потрібні йому платні функції, чи ні. Отже, в цій роботі буде використовуватися безкоштовна версія. І для її завантаження необхідно натиснути на поле, виділене на (див. рис. 5.1), яке знаходиться на сайті, вказаному вище. Поки файл завантажується необхідно зареєструвати аккаунт на сайті.

Рис. 5.1.Вибір потрібної версії

Після завантаження, розробник запускає файл і проходить усі кроки інсталяції, по завершенню якої, буде відкрито вікно. У нього необхідно ввести

дані з акаунту, який зареєстрували на сайті (рис. 5.2). Підсумувавши все, написане вище, інсталяцію можна розбити на такі кроки:

1. Завантаження інсталяційного файлу.

2. Реєстрація на сайті Microsoft.

3. Інсталяція завантаженого файлу.

4. Активація програми.

Після виконання усіх цих кроків, інсталяцію програми можна вважати завершеною.

Рис. 5.2. Сайт реєстрації акаунта

Після завершення інсталяції можна приступати до роботи з ігровим рушієм. Одразу після запуску з’являється вікно, у якому можна побачити проекти, з якими працювали, а також кнопки для відкриття або створення нового проекту (рис. 5.3).

Рис. 5.3. Початкове вікно VisualStudio

Якщо користувач обирає проект, з яким він працював, він одразув ідкривається. Якщо було обрано новий проект, то з’являється нове вікно, у якому треба дати проекту назву, обрати місце його збереження і обрати тип застосування. Після цього кроку з’являється вікно з головним робочим інтерфейсом (рис. 5.4).

Рис. 5.4. Головний інтерфейсVisualStudio

Головний інтерфейс поділено на 6 зон:

- 1-Зона. Показується зовнішній вигляд вікна додатку.
- 2-Зона. Це файлова структура додатку.
- 3-Зона. Властивості застосування.
- 4-Зона. Код XAMLдля відображення всього в зоні №1
- 5-Зона. Вивід помилок та попереджень.

- 6-Зона. Головне навігаційне меню.

Підключення бібліотеки EMGUCV

Рис. 5.5. Меню підключення бібліотеки EMGUCV

Для підключення бібліотеки необхідно зайти в менеджер управління пакетами та знайти необхідну нам бібліотеку та підключити її (див. рис. 5.5).

Вона дозволить нам використовувати всі її інструменти для роботи з графічними об’єктами(функції, які дозволять нам значно простіше маніпулювати з графічними об’єктами).

5.2. Опис алгоритму

Для перевірки роботи застосунку необхідно спочатку створити базу з фотографіями відбитків взуття. І вже тоді можна розпочати подальшу обробку цих фотографій. Алгоритм обробки зображень (рис. 5.6).

Рис. 5.6. Алгоритм обробки зображення

Кроки алгоритму

1. Нам необхідно обрати ображення, яке ми плануємо обробляти.
2. Проводиться бінаризація. Використовуючи функції бібліотеки EMGU.CVми перетворюємо зображення до бінарного вигляду (чорні і білі пікселі), що дозволить нам в подальшому відкинути непотрібні нам області на фотографії і виділити ту область яка відповідає власне за відбиток взуття людини. Для ще кращого виділення області налаштовується контрастність зображення, яка коливається в межах від 0 до 100% (див. додаток А.).
3. Вважається, що всі сліди вносяться в базу у вертикальному положенні, саме для цього здійснюється поворот уже бінаризованого зображення.
4. Знаходження геометричного центру мас він необхідний нам для того щоб створити точку відліку для зображення, шукається він за формулою:

$$\left(X_c = \frac{1}{k}\sum_{i=0}^{k} X_i \, , \, Y_c = \frac{1}{k}\sum_{i=0}^{k} Y_i \right)$$

k- кількість чорних пікселів на зображенні, $\sum_{i=0}^{k} X_i$ - сума координат чорних пікселів по X, $\sum_{i=0}^{k} Y_i$ - сума координат чорних пікселів по У.

Приклад роботи (див. рис. 5.7) наведено нижче, де було знайдено і виділено червоним кольором центр мас для геометричної фігури трикутник на основі матриці з 0(уявно білий колір) і 1(уявно чорний колір) (див. додаток В. рис. В.1.).

Рис. 5.7. Приклад пошуку центру мас

5.3. Порядок використання програми

Для початку роботи в додатку необхідно ознайомитися з його інтерфейсом який поділено на 4 зони (див. рис. 5.8):

- 1-зона. Здійснюється показ початкового та обробленого фото.
- 2-зона. Стек панель для виводу результатів після виконання функції пошуку.
- 3-зона. Область де здійснюється редагування та обробка зображення.
- 4-зона. Меню для здійснення пошуку.

Рис. 5.8. Інтерфейс додатку

Для того щоб завантажити фото в зону 1 для подальшої обробки, необхідно натиснути клавішу «Завантажити фотографію». У провіднику що відкриється обираємо необхідний нам графічний файл (див. рис. 5.9).

Рис. 5.9. Провідник файлів для вибору зображення

Обраний нами файл потребує попередньої обробки (див. рис. 5.10):

- Вибір рівня контрастності для максимально точного відображення відбитку сліду взуття у чорно-білому вигляді.
- Здійснення повороту фото, щоб встановити зображення у вертикальне положення.
- Ввести довжину сліду взуття у сантиметрах в спеціальне поле.

Рис. 5.10. Попередня обробка графічного зображення

Після здійснення обробки фото можна його зберегти в базу даних натиснувши кнопку «Внести фотографію в базу», якщо збереження пройшло успішно то висвітися відповідне повідомлення на екран (див. рис. 5.11)

Рис. 5.11. Повідомлення про успішне додавання зображення у базу даних

Щоб виконати пошук за схожістю зображень у базі даних, в зону 1 потрібно завантажити та попередньо обробити зображення, в зоні 4 обрати поріг схожості(показник який у відсотках показує схожість шуканого зображення з уже наявними у базі даних), і натиснути кнопку «Пошук», дочекатися поки програму виконає всі необхідні обрахунки та виведе результати в зону 2 (див. рис. 5.12) (див. додаток В.).

Рис. 5.12. Виведення результатів пошуку

ВИСНОВКИ

У процесі виконання дослідження зібрано і опрацьовано літературні джерела з базових проблем криміналістики, проблематики теорії розпізнавання та ідентифікації, сучасних інформаційних технологій збереження та обробки графічних зображень, базових алгоритмів прийняття рішень при розпізнаванні графічних об’єктів, сучасних технологій розробки прикладних програм

Спроектовано та реалізовано програмний додаток - систему автоматизованого управління графічними об'єктами, що призначена для підтримки функціонально діяльність експертів НДЕКЦ (обробки, збереження зображень у графічних файлах та автоматизованого розпізнавання слідів взуття людини на основі використання методу контурних функцій). Ряд рішень закладених в проект надає НДЕКЦ можливість:

- формувати базу даних слідів взуття;
- реалізувати автоматичну ідентифікацію відповідних графічних об’єктів.

Програмний продукт реалізовано в програмному середовищі Microsoft Visual Studio. Для створення інтерфейсу було використано технологію розмітки WPF, яка дозволяє з легкістю змінювати та налаштовувати інтерфейс під необхідні потреби. Для збереження даних використали технологію JSON, усі файли були серіалізовані для зручності їх зберігання (див. додаток Б.). Обробка фото здійснювалася за допомогою вбудованих бібліотек EMGUCV в середовище розробки, за допомогою даної бібліотеки було реалізовано метод бінаризації фото (див. додаток А.).

Дослідження проведено на замовлення Рівненського НДЕКЦ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Прошина М. Ю. Розпізнавання образів [Електронний ресурс] / М. Ю. Прошина. – 2012. – Режим доступу до ресурсу: https://wiki.tntu.edu.ua/Розпізнавання_образів.

2. Глава 3: Системы распознавания образов (идентификации) [Електронний ресурс] – Режим доступу до ресурсу: http://www.codenet.ru/progr/alg/ai/htm/gl3_1.php.

3. Проблема розпізнавання. Проблема розпізнавання осіб. огляд методів її рішення. Огляд існуючих методів розпізнавання образів [Електронний ресурс] - Режим доступу до ресурсу: https://beasthackerz.ru/uk/fleshka/problema-raspoznavaniya-problema-raspoznavaniya-lic-obzor-metodov-ee-resheniya.html.

4. Копча-Горячкіна Г. Е. ТЕОРІЯ РОЗПІЗНАВАННЯ ОБРАЗІВ / Г. Е. Копча-Горячкіна // Навчально-методичний посібник. Частина І для студентів факультету інформаційних технологій напряму „Комп'ютерні науки" та „Програмна інженерія" ДВНЗ «Ужгородський національний університет» / Галина Ернестівна Копча-Горячкіна. – Ужгород: Видавництво ДВНЗ «Ужгородського національного університету», 2016. – С. 2–57.

5. Теорія розпізнавання образів. [Електронний ресурс] – Режим доступу до ресурсу: https://sites.google.com/site/navcalnijsajtdlastudentivvnz/teoreticni-osnovi-informatiki/laboratorni-roboti/lab_6.

6. А. С. Довбиш. ОСНОВИ ТЕОРІЇ РОЗПІЗНАВАННЯ ОБРАЗІВ / А. С. Довбиш, І. В. Шелехов – Суми: Сумський державний університет, 2015. – С. 109.

7. Зенин А. В. Информационные технологии [Електронний ресурс] / А. В. Зенин // Молодой ученый. – 2017. – Режим доступу до ресурсу: https://moluch.ru/archive/150/42393/.

8. Введение в Байесовские методы [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/170545/.

9. Царев А.Г. Принципы и методы автоматического распознавания образов [Електронний ресурс] / А. Г. Царев. – Режим доступу до ресурсу: https://cyberleninka.ru/article/n/printsipy-i-metody-avtomaticheskogo-raspoznavaniya-obrazov.

10. Гостев И. М. Методы идентификации графических обьектов на основе геометрической корреляции [Електронний ресурс] / И. М. Гостев. – 2010. – Режим доступу до ресурсу: http://www1.jinr.ru/Pepan/2010-v41/v-41-1/02_gost.pdf.

11. Що таке растрова графіка [Електронний ресурс] – Режим доступу до ресурсу: https://programming.in.ua/other-files/comp-graphics/42-what-it-raster-graphic.html.

12. Форматы графических файлов. Растровые и векторные форматы. [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetcom.ru/production/file-formats/.

13. Форматы векторной графики [Електронний ресурс] – Режим доступу до ресурсу: http://sayt-s-nulya.ru/formaty_vectornoy_grafiki.php.

14. Основные виды компьютерной графики Подробнее на esate.ru: http://esate.ru/article/cg/dizayn/osnovnye_vidy_kompyuternoy_grafiki/ [Електронний ресурс] – Режим доступу до ресурсу: http://esate.ru/article/cg/dizayn/osnovnye_vidy_kompyuternoy_grafiki/.

15. Обробка зображень [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Обробка_зображень.

16. Emgu CV is a cross platform. [Електронний ресурс] – Режим доступу до ресурсу: http://www.emgu.com/wiki/index.php/Main_Page.

17. OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/OpenCV.

18. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.

19. Введение в WPF [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://metanit.com/sharp/wpf/1.php.

20. JSON [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/JSON.

21. Что Такое JSON? [Електронний ресурс] – Режим доступу до ресурсу: https://www.hostinger.ru/rukovodstva/chto-takoe-json/.

22. Изолированная сериализация JSON с использованием DataContractJsonSerializer [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/framework/wcf/feature-details/stand-alone-json-serialization.

23. OpenCV шаг за шагом. Нахождение контуров и операции с ними [Електронний ресурс] – Режим доступу до ресурсу: http://robocraft.ru/blog/computervision/640.html.

24. Гостев И. М. Методи идентификации графических обьектов на основе геометрической корреляции [Електронний ресурс] / И. М. Гостев. – 2010. – Режим доступу до ресурсу: http://www1.jinr.ru/Pepan/2010-v41/v-41-1/02_gost.pdf.

ДОДАТКИ

Додаток А

Функції, які були використані для обробки графічних файлів.

```
3 references
static public BitmapImage Binarize(BitmapImage bitmapImage, double threshold)
{
    double region_size = 20;

    var result = new Image<Gray, byte>(ImageConverter.ToBitmap(bitmapImage)).ThresholdBinaryInv(new Gray(threshold), new Gray(255));

    int rows = Convert.ToInt32(Math.Ceiling(result.Size.Height / region_size));
    int cols = Convert.ToInt32(Math.Ceiling(result.Size.Width / region_size));

    var flags = new bool[rows, cols];
    var regions = new int[rows, cols];
    regions.Initialize();

    var bmp = CreateNonIndexedImage(result.Bitmap);

    for (int i = 0; i < bmp.Height; ++i)
    {
        for (int j = 0; j < bmp.Width; ++j)
        {
            if (bmp.GetPixel(j, i).R == 0)
            {
                System.Drawing.Point flag_pos = new System.Drawing.Point((int)(i / region_size), (int)(j / region_size));
                flags[flag_pos.X, flag_pos.Y] = ++regions[flag_pos.X, flag_pos.Y] > region_size * region_size * 0.5;
            }
        }
    }

    for (int i = 1; i < rows - 1; ++i)
    {
        for (int j = 1; j < cols - 1; ++j)
        {
            int count = Convert.ToInt32(flags[i + 1, j])
                + Convert.ToInt32(flags[i - 1, j])
                + Convert.ToInt32(flags[i, j + 1])
                + Convert.ToInt32(flags[i, j - 1]);

            if ((flags[i, j] && count <= 2) || (!flags[i, j] && count >= 2))
            {
                flags[i, j] = !flags[i, j];
            }
        }
    }
```

Рис. А. 1. Функція бібліотеки EMGUCV для бінаризації графічного зображення

```
    for (int i = 0; i < cols; ++i)
    {
        flags[0, i] = false;
        flags[rows - 1, i] = false;
    }

    for (int i = 0; i < rows; ++i)
    {
        flags[i, 0] = false;
        flags[i, cols - 1] = false;
    }

    for (int i = 0; i < bmp.Height; ++i)
    {
        for (int j = 0; j < bmp.Width; ++j)
        {
            System.Drawing.Point flag_pos = new System.Drawing.Point((int)(i / region_size), (int)(j / region_size));
            bmp.SetPixel(j, i, flags[flag_pos.X, flag_pos.Y] && bmp.GetPixel(j, i).R == 0 ? System.Drawing.Color.Black : System.Drawing.Color.White);
        }
    }

    result.Bitmap = bmp;
    return ImageConverter.ToBitmapImage(result.Bitmap);
}
```

Рис. А.2. Продовження функції (див. рис. А.1.)

```
private void Binarize(object sender, RoutedEventArgs e)
{
    ThresholdSlider.IsEnabled = IsBinarized;

    if (IsBinarized && MainImageSource != null)
    {
        MainImageSource = Binarizator.Binarize(OriginalImage, ThresholdSlider.Value);
    }
    else if (!IsBinarized && MainImageSource != null)
    {
        MainImageSource = OriginalImage;
    }
}
```

Рис. А.3. Виклик функції для здійснення бінаризації

Додаток Б

Функції для збереження інформації в файлові системі JSON

```
using System.Drawing;

namespace TrassogicalTool
{
    4 references
    public class ProcessedImage
    {
        2 references
        public Point Center { get; set; }
        2 references
        public int Long { get; set; }
        3 references
        public string Path { get; set; }
    }
}
```

Рис. Б.1. Стуктура файлу JSON

```
private void UploadImage(object sender, RoutedEventArgs e)
{
    if (MainImageSource == null)
        return;

    var path = $"{DatabasePath}{ProcessedImages.Count}.png";
    ImageConverter.ToBitmap(MainImageSource).Save(path);
    if ((sizeTextBox.Text.Length)< 1)
    {
        sizeTextBox.Text = "10";
    }
        ProcessedImages.Add(new ProcessedImage
    {
        Center = CenterOfImage(),
        Path = path,
        Long= Int32.Parse(sizeTextBox.Text)
    });

    File.WriteAllText(DatabasePath + "database.json", JsonConvert.SerializeObject(ProcessedImages));

    System.Windows.MessageBox.Show("Зображення успішно додано в базу.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);
}
```

Рис. Б.2. Функція збереження зображення в файл JSON

```
private void UploadMainImage(object sender, RoutedEventArgs e)
{
    var dialog = new OpenFileDialog
    {
        Filter = "Image files (*.jpg, *.jpeg, *.jpe, *.jfif, *.png, *.bmp) | *.jpg; *.jpeg; *.jpe; *.jfif; *.png; *bmp;",
        Title = "Оберіть зображення",
        Multiselect = false
    };

    try
    {
        if (dialog.ShowDialog() == System.Windows.Forms.DialogResult.OK)
        {
            var bmp_image = new BitmapImage();
            bmp_image.BeginInit();
            bmp_image.UriSource = new Uri(dialog.FileName);
            bmp_image.EndInit();
            OriginalImage = bmp_image;
            MainImageSource = IsBinarized ? Binarizator.Binarize(OriginalImage, ThresholdSlider.Value) : bmp_image;
            MainImagePath = dialog.FileName;
        }
    }
    catch (Exception)
    {
        System.Windows.MessageBox.Show("Щось пішло не так. Перевірте коректність обраного зображення та повторіть спробу.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Рис. Б. 3. Функція завантаження графічного зображення

Додаток В

Функції для пошуку та ідентифікації графічних об’єктів

```
public Point CenterOfImage()
{
    var center = new Point();
    var bitmap = ImageConverter.ToBitmap(MainImageSour
    int Height = bitmap.Height;
    int Width = bitmap.Width;
    int K = 0, SumX = 0, SumY = 0;

    for (int j = 0; j < Height; j++)
        for (int i = 0; i < Width; i++)
            if (bitmap.GetPixel(i, j).R == 0)
            { K++; SumY += j; }

    for (int j = 0; j < Height; j++)
        for (int i = 0; i < Width; i++)
            if (bitmap.GetPixel(i, j).R == 0)
                SumX += i;

    center.X = SumX / K; center.Y = SumY / K;
    return center;
}
```

Рис. В.1. Функція пошуку центру зображення

```
1 reference
double Identification(Bitmap firstImg,int firstSize,int secondSize, Point firstCenter, Bitmap secondImg, Point secondCenter)
{
if (firstSize >= (secondSize - 2) && firstSize <= (secondSize + 2))
{
    double CountOfChangeArea;//загальна кількість пікселів виділеної області;
    int k = 0;//кількість співпадінь
    double result;
    int minLeftLength, minRightLength, minTopLength, minDownLength; //мінімальні значення від центру координат для виділення необхідної області для порівняння
    minLeftLength = Math.Min(firstCenter.X, secondCenter.X);//мнімальний розмір зліва від центру
    minRightLength = Math.Min(firstImg.Height - firstCenter.X - 1, secondImg.Height - secondCenter.X - 1);//відступ справа.від загальної ширини віднімаємо точку центра
    minTopLength = Math.Min(firstCenter.Y, secondCenter.Y);//відстань зверху
    minDownLength = Math.Min(firstImg.Width - firstCenter.Y - 1, secondImg.Width - secondCenter.Y - 1);//відстань знизу

    for (int j = 0; j <= (minTopLength + minDownLength); j++)
        for (int q = 0; q <= (minLeftLength + minRightLength); q++)
        {
            if (firstImg.GetPixel((j + firstCenter.Y - minTopLength), (q + firstCenter.X - minLeftLength)) == secondImg.GetPixel((j + secondCenter.Y - minTopLength), (q + secondCenter.X - minLeftLength)))
                ++k;
        }
    CountOfChangeArea = (minTopLength + minDownLength + 1) * (minLeftLength + minRightLength + 1);//множимо висоту на ширину(виділена область яка порівнюється)
    result = 100 * (k / CountOfChangeArea);
    k = 0;

    return result;
}
else return 0.15;


}
```

Рис. В.2. Функція для ідентифікації графічних об’єктів

```
private void ShowImages(object sender, RoutedEventArgs e)
{
    if (MainImageSource == null)
        return;

    var images = new List<BitmapImage>();
    var paths = Directory.GetFiles(DatabasePath, "*.png");

    foreach (var path in paths)
    {
        var bmp_image = new BitmapImage();

        bmp_image.BeginInit();
        bmp_image.UriSource = new Uri(Path.GetFullPath(path));
        bmp_image.EndInit();


        Likeness.Add(bmp_image, Identification(
        ImageConverter.ToBitmap(MainImageSource),
      Int32.Parse(sizeTextBox.Text),
        ProcessedImages.First(x => x.Path == path).Long,
        CenterOfImage(),
            ImageConverter.ToBitmap(bmp_image),
            ProcessedImages.First(x => x.Path == path).Center));

        images.Add(bmp_image);
    }

    Images = images;
}
```

Рис. В.3. Функція відображення необхідних результатів після здійснення ідентифікації