НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК _____

«До захисту допущено»
В. о. завідувача кафедри
_____ О. І. Ролік
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 151 Автоматизація та комп'ютерно-інтегровані технології

на тему: «Алгоритми підрахунку людей методами комп'ютерного бачення»

Виконав:
студент ІІ курсу, групи ІА-61м
Антосєв Андрій Сергійович _____

Керівник:
Доцент, к.т.н., доцент
Дорогий Я. Ю. _____

Рецензент:
Доц. каф. кібербезпеки та ЗІСТ, к.т.н., доц.
Цуркан В. В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах
Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою
Спеціальність – 151 Автоматизація та комп'ютерно-інтегровані технології

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С. Ф. Теленик
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Антосєву Андрію Сергійовичу

1. Тема дисертації «Алгоритми підрахунку людей методами комп'ютерного бачення», науковий керівник дисертації Дорогий Ярослав Юрійович, доцент, к.т.н., доцент, затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Термін подання студентом дисертації: _____.

3. Об'єкт дослідження - алгоритми підрахунку людей

4. Предмет дослідження - алгоритм підрахунку людей методом обробки послідовності зображень.

5. Перелік завдань, які потрібно розробити: Аналіз існуючих рішень; розроблення алгоритму виявлення активності; розроблення алгоритму слідкування; дослідження колірних моделей; вирішення задачі взаємодії об'єктів; дослідження параметрів алгоритму; порівняння з існуючими рішеннями

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: зображення поетапного перетворення кадру для виявлення аткивного об'єкту; зображення проблемних ситуацій взаємодії об'єктів.

7. Перелік публікацій: 1) «Алгоритм підрахунку людей на основі методів комп'ютерного бачення» - Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка - №66; 2) «Дослідження впливу колірних моделей при визначенні кольорових характеристик об'єктів розпізнавання на зображенні» - VI

Міжнародна науково-практична конференція «Summer InfoCom Advanced Solutions 2018».

8. Дата видачі завдання: _____. _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Проведення аналізу існуючих рішень	15.03.18-23.03.18	
2	Розроблення алгоритму виявлення активності	24.03.18-29.03.18	
3	Розроблення алгоритму слідкування	30.03.18-09.04.18	
4	Дослідження колірних моделей	10.04.18-17.04.18	
5	Вирішення задачі взаємодії об’єктів	18.04.18-28.04.18	
6	Дослідження параметрів алгоритму	29.04.18-02.05.18	
7	Порівняння з існуючими рішеннями	03.05.18-14.05.18	

Студент А. С. Антосєв

Науковий керівник дисертації Я. Ю. Дорогий

РЕФЕРАТ

Магістерська дисертація на тему “Алгоритми підрахунку людей на основі методів комп’ютерного бачення”: 105 ст., 41 рис., 31 табл., 2 додатки, 21 джерело.

Об'єкт дослідження - алгоритми підрахунку людей.

Мета роботи – аналіз існуючих алгоритмів підрахунку людей на основі обробки зображень методами комп’ютерного бачення і розробка нового оптимального алгоритму.

В більшості алгоритмів підрахунку людей, що використовують методи комп’ютерного бачення недостатня точність особливо під впливом світла на зображення. Також на якість роботи таких алгоритмів впливають особливості приміщення де проходить відеоспростереження.

За допомогою популярних методів комп’ютерного бачення, а також після аналізу основних колірних моделей та вибору найбільш підходящої з них, було розроблено власний алгоритм, що надає якомога кращий результат незважаючи на вищевказані фактори впливу.

Для розробки та реалізації алгоритму у вигляді програмного забезпечення використовувалася мова програмування Python і його бібліотека комп’ютерного бачення OpenCV.

Результати дослідження використовуються в програмному забезпеченні компанії Quadrox.

Прогнозні припущення про розвиток дослідження – комбінація з іншими методами аналізу даних.

ЛІЧИЛЬНИК, ОБРОБКА ЗОБРАЖЕНЬ, КОМП’ЮТЕРНЕ БАЧЕННЯ, ВИЯВЛЕННЯ АКТИВНОСТІ, OPENCV, КОЛІРНА МОДЕЛЬ, RGB, HSV, LAB, РОЗПІЗНАВАННЯ.

ABSTRACT

Master's thesis " People counting algorithms on the basis of computer vision" consists of 105 pages, 41 figures, 31 tables, 2 appendices, 18 sources.

The object of the study is people counting algorithms.

The purpose of the work is to analyze existing people counting algorithms based on image processing using computer vision and to develop a new optimal algorithm.

Most algorithms for counting people using computer vision have insufficient accuracy, especially under the influence of light on the image. Also, the quality of the work of such algorithms is influenced by the room features where the video is recorded.

With popular methods of computer vision, and after analyzing the basic color models and selecting the most suitable of them was developed algorithm that provides the best possible result despite the impact of the above factors.

To develop and implement the algorithm in the form of software programming language Python and its library of computer vision OpenCV were used.

The results of the study are used in the software of Quadrox company.

Foreseeable assumptions about the development of the study - a combination with other methods of data analysis.

COUNTER, IMAGE PROCESSING, COMPUTER VISION, ACTIVITY DETECTION, OPENCV, COLOR MODEL, RGB, HSV, LAB, RECOGNITION.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ, ОДИНИЦЬ І ТЕРМІНІВ

ЛІЗ - лінійне інваріантне зміщення

ОРС - оптичне розпізнавання символів

АРНЗ - автоматичне розпізнавання номерного знака

ЗКЗ - злиття комп'ютерних зображень

ЄЗР - єдине значення розкладу

AAS - Automated-Attendance-System

OCPC - Overhead Camera People Counter

ВСТУП

Актуальність. Тема, що розкривається в дослідженні, є дуже актуальною у зв’язку з швидким розповсюдженням прикладного використання методів комп’ютерного бачення. В системах для розпізнавання та підрахунку людей все частіше використовується комбінація декількох методів обробки даних, що разом забезпечують надзвичайно точних результат. Одним з найважливіших компонентів таких систем є метод аналізу послідовності зображень. Проте, незважаючи на розповсюдженість його застосування, реалізація цього методу в більшості застосунках не є ідеальною і страждає від різноманітних шкідливих факторів, тож потребує вдосконалення.

Об’єктом дослідження є алгоритми підрахунку людей, що потребують вдосконалення.

Предметом дослідження є алгоритм підрахунку людей методом обробки послідовності зображень.

Зв’язок роботи з науковими програмами, планами, темами. Тематика роботи включена в науково-технічні плани кафедри автоматики та управління в технічних системах Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». Результати дослідження також використовуються в програмному забезпеченні компанії Quadrox для збору інформації про відвідуваність магазинів та торгових центрів.

Метою дослідження є розробка вдосконаленого методу обробки послідовності зображень з камери відеоспостереження, адже покращення окремих складових системи підвищує її загальну якість.

Задачі дослідження. Першим етапом дослідження є аналіз існуючих рішень, недоліки яких враховуються у власному алгоритмі. Далі необхідно дослідити та обрати ті технології, що найбільше підходять для вирішення поставлених задач. Наступним етапом є саме розроблення комплексу алгоритмів, що складається з алгоритму виявлення активності, слідкування за рухомими об’єктами, та перетинання лінії. Фінальним завданням є дослідження результатів роботи.

Методами дослідження, що використовуються для досягнення мети, є універсальні методи комп’ютерного бачення, в більшості - морфологічні перетворення зображень.

Наукова цінність. Новизна магістерського дослідження полягає у розробці алгоритму, точність якого є більшою ніж у аналогічних алгоритмів за рахунок більш детального аналізу даних зображення, а також аналізу взаємодії об’єктів розпізнавання.

Практична цінність полягає в тому, що розроблений алгоритм можна використовувати або як самостійний механізм, або як частину складної системи для підрахунку людей.

Апробація результатів. Стислий опис дослідження та розробки алгоритму був опублікований у вигляді статті «Алгоритм підрахунку людей на основі методів комп’ютерного бачення» у науковому виданні “Вісник НТУУ "КПІ". Інформатика, управління та обчислювальна техніка №66” (Додаток А). Результати порівняння колірних моделей були опубліковані у вигляді тез «Дослідження впливу колірних моделей при визначенні кольорових характеристик об’єктів розпізнавання на зображенні» на VI Міжнародній науково-практичній конференції «Summer InfoCom Advanced Solutions 2018» (Додаток Б).

1 АНАЛІЗ ТЕХНОЛОГІЙ ТА ІСНУЮЧИЙ РІШЕНЬ

1.1 Загальні положення про комп’ютерне бачення

Як люди, ми сприймаємо тривимірну структуру оточуючого нас світу з очевидною легкістю. Подумайте про яскравість тривимірного сприйняття, коли ви дивитесь на вазу квітів, що стоять на столі поруч із вами. Ви можете визначити форму та прозорість кожного пелюстка через тонкі візерунки світла та тіні, які грають на її поверхні та легко відстежують кожну квітку з фону сцени (рис. 1.1). У людській зоровій системі немає проблем, що інтерпретують витончені варіації прозорості та затінення на цій фотографії і правильну сегментацію об'єкта з його фону. Дивлячись на рукописний портрет колективу людей, ви можете легко підрахувати (і назвати) всіх людей у зображенні та навіть здогадатися про емоції з їхнього вигляду обличчя. Перцептивні психологи протягом десятиліть намагаються зрозуміти, як функціонує візуальна система, і, навіть якщо вони можуть вигадувати оптичні ілюзії, щоб розірвати деякі його принципи (рис. 1.3), повне вирішення цієї загадки залишається невловимим.

Рисунок 1.1 - Приклад кольорового зображення [1]

Дослідники комп'ютерного бачення паралельно розвивали математичні методи для відновлення тривимірної форми та появи об'єктів у зображенні. Тепер у нас є надійні методи для точного обчислення часткової 3D-моделі середовища з тисяч точок, що збігаються на фотографії (рис 1.2а). Отримавши достатньо великий набір ракурсів на певний об'єкт, ми можемо створювати точні, щільні моделі 3D-поверхні, використовуючи стереозвук (рис. 1.2б). Ми можемо відслідковувати людину, що рухається у складному зображенні (рис 1.2в). Завдяки помірному успіху ми можемо спробувати знайти і назвати всіх людей на фотографії, використовуючи комбінацію виявлення та розпізнавання обличчя, одягу та волосся (рис 1.2г). Однак, незважаючи на всі ці досягнення, залишається мрією, щоб комп'ютери інтерпретували зображення на тому ж рівні, що й дворічна дитина (наприклад, підрахування всіх тварин на зображенні), і поки що це залишається невловимим. Чому комп’ютерне бачення таке складне? Частково це пов'язано з тим, що бачення є зворотною проблемою, в якій ми прагнемо відновити деякі невідомі дані з недостатньою інформацією, щоб повністю вказати рішення. Тому ми повинні вдаватися до фізичних і імовірнісних моделей, щоб неоднозначно розглядати потенційні рішення. Проте моделювання візуального світу у всій його багатій складності набагато складніше, ніж, скажімо, моделювання голосового такту, який виробляє розмовні звуки.

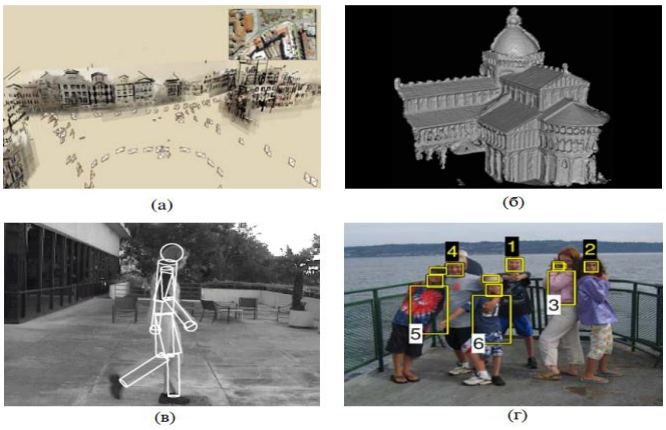

Рисунок 1.2 - Деякі приклади алгоритмів та програм комп'ютерного бачення[1]

Напрямні моделі, які ми використовуємо при комп'ютерному зорі, зазвичай розробляються в фізиці (радіометрія, оптика, сенсорний дизайн) та в комп'ютерній графіці. Обидві ці галузі моделюють, як об'єкти рухаються, і відтворюють в анімації, як відбивається світло від їх поверхонь, розсіюється атмосферою, переломлюється через лінзи камери (або людськими очима), і нарешті проеціюється на плоске (або вигнуте) зображення. Незважаючи на те, що комп'ютерна графіка ще не є ідеальною (повний комп'ютерний анімаційний фільм з людськими персонажами ще не вдалося створити в тій мірі, коли можна відрізнити справжніх людей від роботів-андроїдів та комп'ютерних анімаційних людей), в обмежених областях, таких як відтворення сцени, що складається з повсякденних об'єктів або анімацій вимерлих істот, таких як динозаври, ілюзія реальності є досконалою.

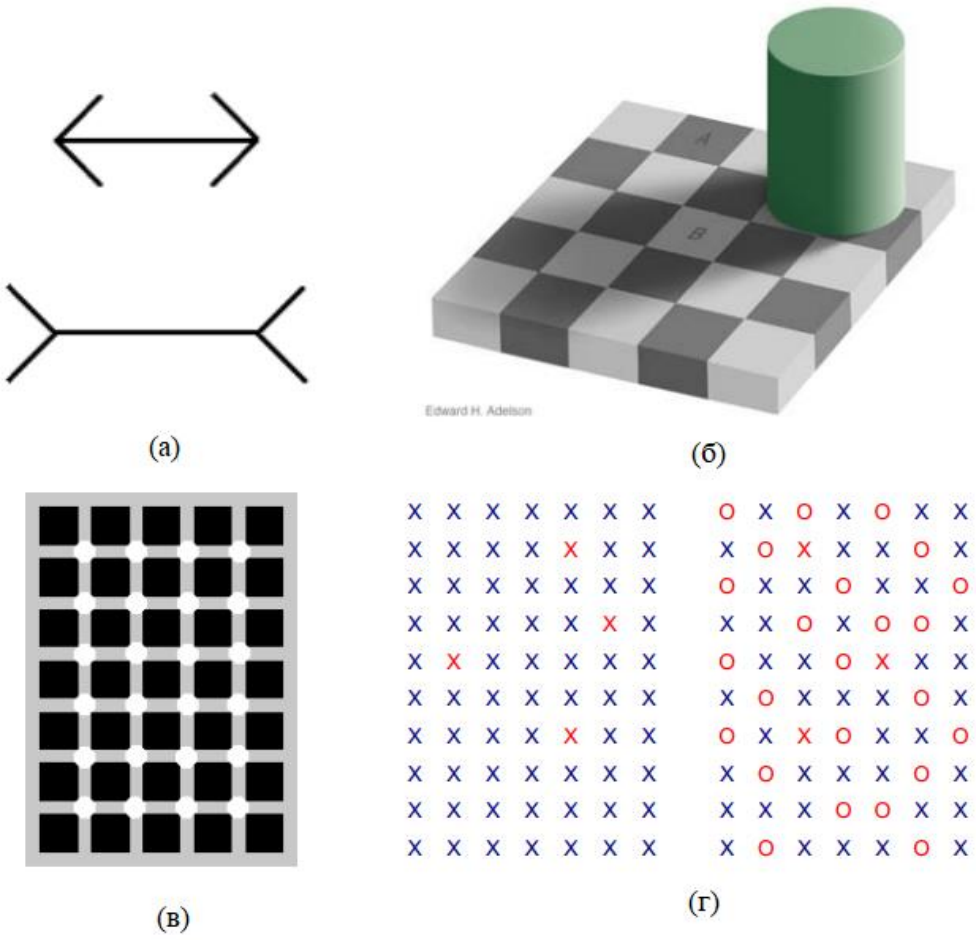

Рисунок 1.3 - Приклади оптичних ілюзій [1]

Деякі загальні оптичні ілюзії та те, що вони можуть сказати нам про візуальну систему (рис 1.3):

a) Класична ілюзія Мюллера-Лієра, де довжина двох горизонтальних ліній виглядає різною, ймовірно, пов'язана з уявними ефектами перспективи.

b) "білий" квадрат B в тіні та "чорний" квадрат A на світлі фактично мають однакову абсолютну інтенсивність. Перцепція обумовлена постійністю яскравості, спробою візуальної системи знижувати освітлення при інтерпретації кольорів.

c) Варіація ілюзії германської сітки Коли ви дивитесь на фігуру, на перехрестях з'являються сірі плями.

d) Розрахуйте червоні X в лівій половині фігури. Тепер підрахуйте їх у правій половині. Це значно важче? Пояснення пов'язано з ефектом сплеску (Treisman 1985), який говорить нам про операції паралельного сприйняття та інтеграції шляхів у мозку.

У комп'ютерному баченні ми намагаємося зробити зворотне, тобто описати світ, який ми бачимо в одному або декількох зображеннях, і відтворити його властивості, такі як форма, освітлення та розподіл кольорів. Дивно, що люди та тварини роблять це без зусиль, а алгоритми комп'ютерного зору настільки схильні до помилок. Люди, які не працювали у цій галузі, часто недооцінюють труднощі проблеми. (Колеги на роботі часто просили мене знайти програмне забезпечення та називати всіх людей на фотографіях, щоб вони могли отримати більш точний результат). Це хибне розуміння того, що бачення повинно бути легким, з’явилось з перших днів штучного інтелекту, коли спочатку вважалося, що пізнавальні (логічні докази та планування) частини інтелекту були по суті найскладнішими, ніж перцептивними складовими.

Хороша новина полягає в тому, що комп'ютерне бачення сьогодні використовується у різноманітних реальних застосунках, серед яких:

- Оптичне розпізнавання символів (ОРС): читання рукописних поштових кодів на листах (рис. 1.4a) та автоматичне розпізнавання номерного знака (АРНЗ);

- Інспекція машини: швидка перевірка деталей для забезпечення якості, використовуючи стерео бачення з спеціалізованим підсвічуванням для вимірювання похибок на крилах літальних апаратів або частинах кузова (рис 1.4б) або пошук дефектів сталевих виливків за допомогою рентгенівського зору;
- Роздрібна торгівля: розпізнавання об'єктів для автоматичних смуг відправлення (рис. 1.4в);
- 3D-моделювання (фотограмметрія): повністю автоматизована побудова 3D-моделей з аерофотознімків, що використовуються в таких системах, як Bing Maps;
- Медична візуалізація: реєстрація доопераційного та внутрішньоопераційного зображень (рис. 1.4г) або виконання довготермінових досліджень морфології людського мозку, коли вони старіють;
- Автомобільна безпека: виявлення несподіваних перешкод, таких як пішоходи на вулиці, в умовах, коли активні методи зору, такі як радіолокація або лідар, не працюють належним чином (рис. 1.4д);
- Транслювання матчів: злиття комп'ютерних зображень (ЗКЗ) з кадрами в прямому ефірі шляхом відстеження точок функцій у вихідному відео, щоб оцінити рух 3D-камери та форму навколишнього середовища. Такі методи широко використовуються в Голлівуді (наприклад, у фільмах, таких як Jurassic Park); вони також вимагають використання точного матування для вставки нових елементів між переднім та фоновим елементами .

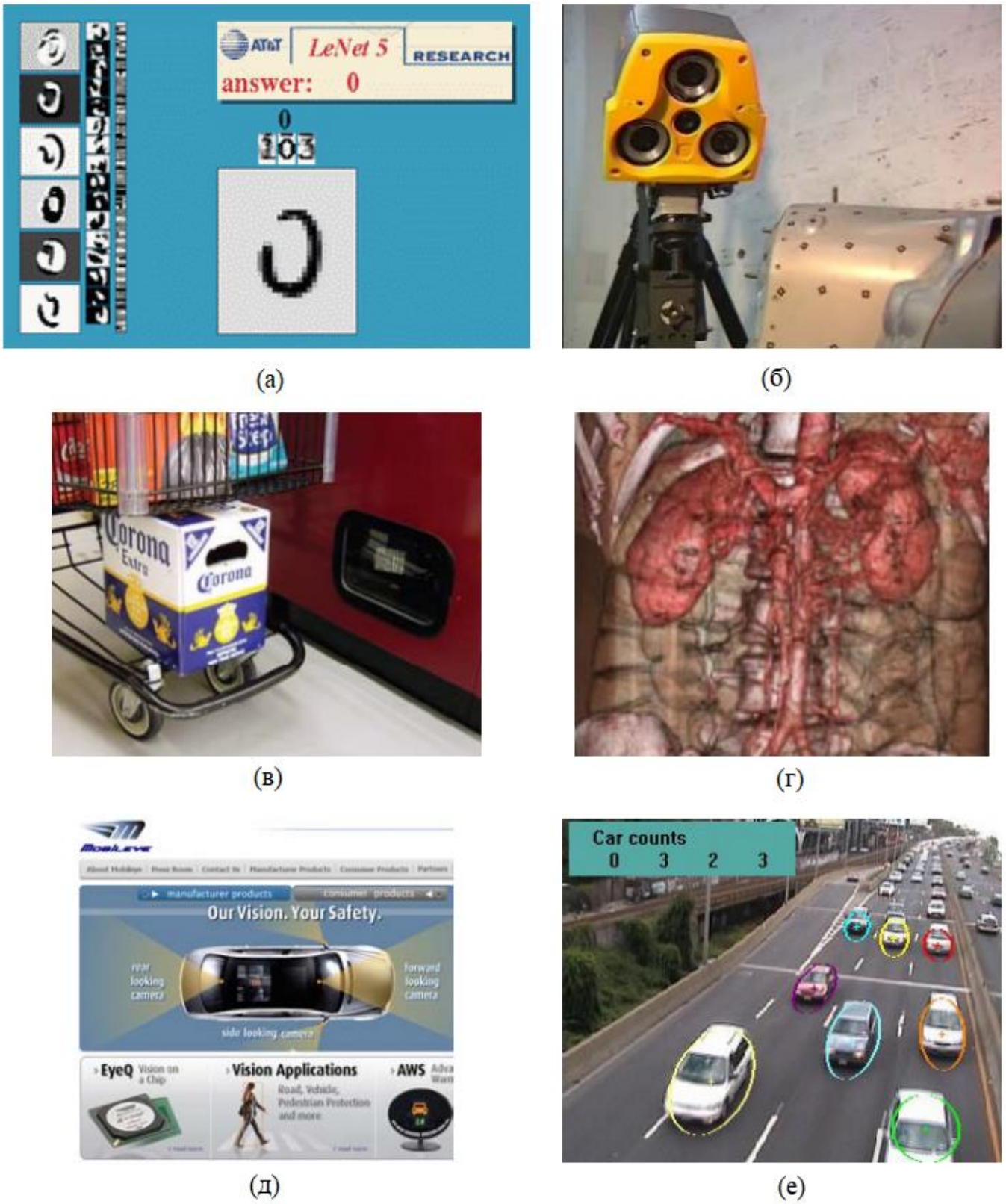

Рисунок 1.4 - Деякі промислові програми комп'ютерного бачення [1]

- Захоплення руху (mocap): використання рефлекторних маркерів, що переглядаються з декількох камер або інших методів зору, для захоплення акторів для комп'ютерної анімації;
- Нагляд: моніторинг для вторгнень, аналіз трафіку на трасі (рис. 1.4е) та пули моніторингу для слідкування за жертвами катастроф;
- Відстеження відбитків пальців та біометрія: для автентифікації автоматичного доступу, а також криміналістичних застосунків.

Веб-сайт Девіда Лоу [2] про застосунки для промислового зору перелічує багато інших цікавих промислових застосувань комп'ютерного бачення. Хоча вищезазначені програми є надзвичайно важливими, вони в основному стосуються досить спеціалізованих видів зображень та вузьких областей.

Далі зосереджується увага на більш широких додатках на рівні споживачів, таких як речі, які ви можете зробити з власними фотографіями та відео. До них відносяться:

- Зшивання: перетворення фотографій, що перекриваються, у єдину зшиту панораму (рис. 1.5а);
- Брекетинг експозиції: злиття декількох експозицій, зроблених у складних умовах освітлення (сильні сонячні промені та тіні), в одне повністю відтворене зображення (рис. 1.5б);
- Морфінг: перетворення зображення одного з ваших друзів в щось інше, використовуючи безперервний морфо перехід (рис. 1.5в);
- 3D-моделювання: перетворення одного або кількох знімків у 3D-модель об'єкта або особи, яку ви фотографуєте (рис. 1.5г);

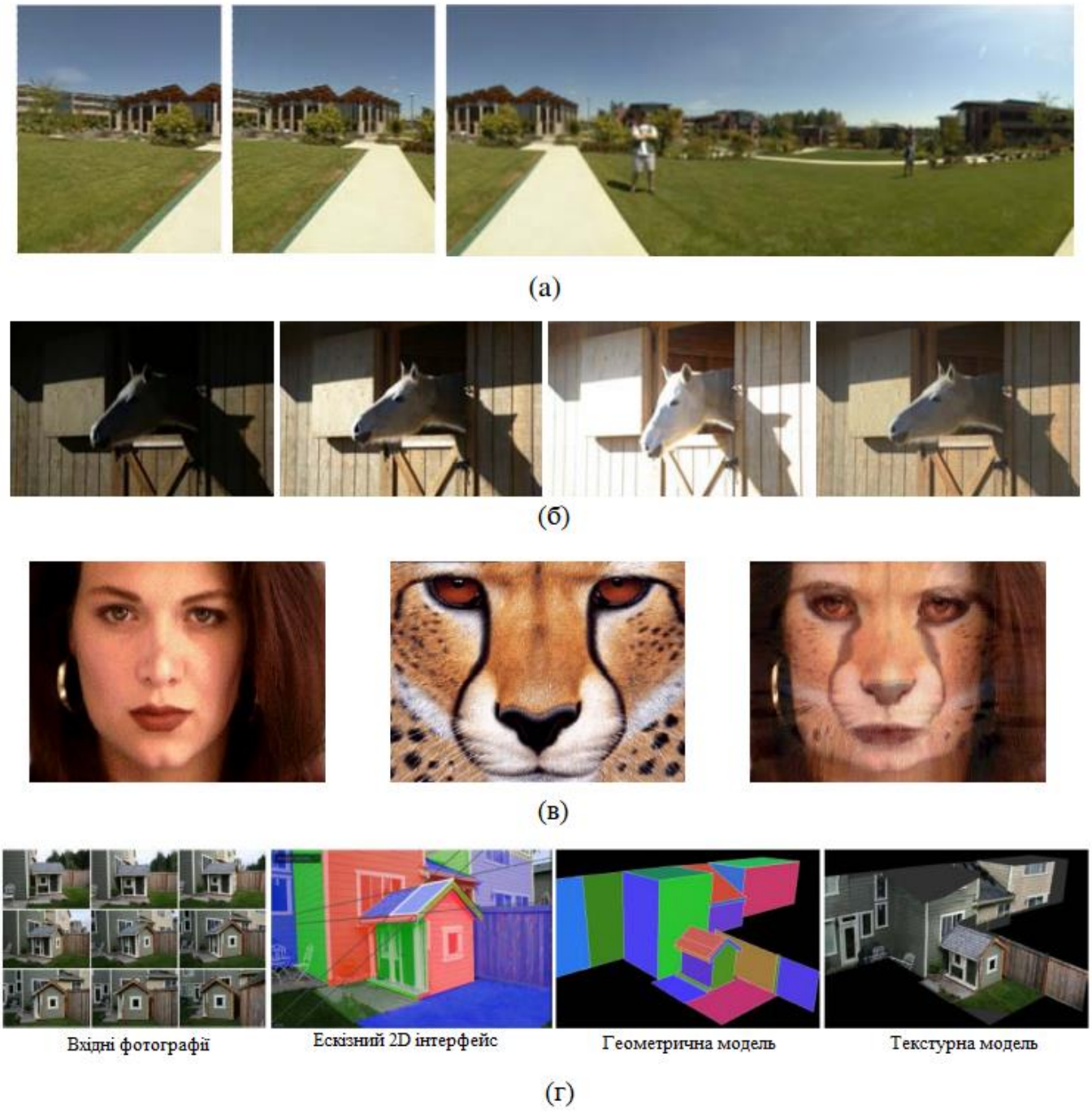

Рисунок 1.5 - Деякі користувацькі програми комп'ютерного бачення [1]

- Переміщення відеосигналу: додавання 2D-зображень або 3D-моделей у відео, автоматично відстежуючи орієнтири поблизу з використанням оцінок руху для видалення тряски з ваших відео ;
- Фото-покрокові керівництва: переміщення великої колекції фотографій, таких як інтер'єр вашого будинку, шляхом польоту між різними фотографіями в 3D;

- Виявлення обличчя: для покращення фокусування камери, а також для більш релевантного пошуку зображень;
- Візуальна автентифікація: автоматична реєстрація членів родини на вашому домашньому комп'ютері, коли вони сідають перед веб-камерою.

Найкраще в цих застосунках це те, що вони вже знайомі більшості студентів; вони, або принаймні їхні технології, які студенти можуть негайно оцінять та використовують з власними засобами масової інформації. Оскільки комп'ютерне бачення є складною темою, з урахуванням широкого спектру математики, що охоплюється, і сутнісно складний характер вирішенних проблем, веселощі та відповідні проблеми для роботи можуть бути дуже мотиваційними та надихаючими.

Інша основна причина, чому ця тема зосереджує увагу на додатках, полягає в тому, що вони можуть бути використані для формулювання та обмеження потенційно відкритих, ендемічних проблем зору. Наприклад, якщо хтось приходить до мене і запитує якісний і хороший детектор країв зображення, то моє перше питання, як правило, “Навіщо?” Які проблеми вони намагаються вирішити і чому вони вважають, що виявлення країв є важливим компонентом? Якщо вони намагаються знайти обличчя, я зазвичай вказую, що найуспішніші детектори обличчя використовують комбінацію виявлення кольору шкіри та простих функцій блокування. Якщо вони намагаються узгодити складові дверей і вікон у будівлі з метою реконструкції тривимірного зображення, я кажу їм, що ребра є прекрасною ідеєю, але краще налаштувати крайовий детектор для довгих країв і зв'язати їх разом у прямі лінії з загальними точками зникнення перед збігом .

Таким чином, краще відразу замислитись про підходящу проблему та виділити відповідні методи, а не обрати першу техніку, про яку ви, можливо, чули. Цей тип роботи від проблем до рішень характерний для інженерного підходу до вивчення зору і відображає власний досвід у цій галузі. По-перше, необхідно придумати детальне визначення проблеми та прийняти рішення щодо обмежень та специфікацій проблеми. Потім намагатись з'ясувати, які методизазвичай використовують, реалізувати декілька з них, оцінити їх продуктивність і, нарешті,

робити вибір. Для того, щоб цей процес працював, важливо мати реалістичні тестові дані, як синтетичні, які можуть бути використані для перевірки правильності та аналізу чутливості до шуму, так і реальних даних, характерних для способу остаточного використання системи.

В цьому розділі розділі частково розглядається науковий підхід до основних проблем зору. Тут наявний намір виробити найкращі фізичні моделі системи: як створюється сцена, як світло взаємодіє зі сценою та атмосферними ефектами, і як працюють датчики, включаючи джерела шуму та невизначеності. Завдання полягає в тому, щоб спробувати перевернути процес придбання, щоб запропонувати найкращий опис сцени.

Часто використовується статистичний підхід до формулювання та вирішення проблем комп'ютерного зору. Де це доцільно, розподіли ймовірностей використовуються для моделювання сцени та процесу отримання шумного зображення. Асоціацію попередніх розподілів з невідомими часто називають байєсовським моделюванням. Можна поєднати функцію ризику або втрату з неправильною оцінкою відповіді та налаштувати свій алгоритм виведення, щоб мінімізувати очікуваний ризик. (Розглянемо робота, який намагається оцінити відстань до перешкоди: зазвичай безпечніше недооцінювати, ніж переоцінювати). За статистичними методами це часто допомагає зібрати багато навчальних даних, з яких можна вивчати імовірнісні моделі. Нарешті, статистичні підходи дозволє вам використовувати перевірені методи висновків для оцінки найкращої відповіді (або розподілу відповідей) та кількісної оцінки невизначеності в отриманих оцінках.

Оскільки більша частина комп'ютерного бачення полягає у вирішенні зворотних задач або оцінці невідомих величин, поставлений великий акцент на алгоритмах, особливо тих, які добре працюють на практиці. Для багатьох проблем, пов'язаних із комп’ютерним баченням, дуже легко придумати математичний опис проблеми, яка не відповідає реальним умовам реального світу або не піддається стабільній оцінці невідомих. Необхідно розглянути алгоритми, які є надійними для

шуму та відхиленнями від наших моделей і досить ефективними з точки зору ресурсів проміжку часу та простору. [1]

Зображення формуються за допомогою взаємодії елементів 3D-сцен, освітлення та оптики камери та датчиків, тож далі давайте розглянемо перший етап в більшості комп’ютерних застосунків, а саме: використання обробки зображень для попередньої обробки зображення та перетворення їх у форму, придатну для подальшого аналізу. Приклади таких операцій включають корекцію експозиції та балансування кольорів, зменшення шумів зображення, збільшення різкості або вирівнювання зображення, обертаючи його (рис. 1.6). Хоча деякі можуть розглядати процеси обробки зображень поза компетенцією комп'ютерного бачення, більшість додатків для комп'ютерного зору, таких як обчислювальна фотографія та навіть розпізнавання, потребують уваги при розробці етапів обробки зображень для досягнення прийнятних результатів.

У цьому розділі ми переглядаємо стандартні оператори обробки зображень, які відображають значення пікселів від одного зображення до іншого.

Рисунок 1.6 - Деякі основні операції обробки зображень [3]

Почати обговорення необхідно найпростіших видів перетворень зображень, а саме тих, які маніпулюють кожним пікселем незалежно від сусідніх (розділ 1.2.4). Такі перетворення часто називають точковими операторами або точковими процесами. Далі ми розглядаємо оператори околиць (області), де кожен новий

піксель залежить від невеликої кількості сусідніх вхідних значень (розділи 1.2.1, 1.2.2, 1.2.3 та 1.2.5). Зручним інструментом для аналізу (а часом і прискорення) таких операцій сусідства є перетворення Фур'є. Оператори сусідства можуть бути скомпоновані для створення пірамід і пікселів зображень, які корисні для аналізу зображень за різними роздільною здатністю (масштабами) та для прискорення деяких операцій. Іншим важливим класом глобальних операторів є геометричні перетворення, такі як повороти, ножиці та перспективні деформації. Нарешті, вводяться глобальні підходи до оптимізації обробки зображень, які включають мінімізацію енергетичної функціональної або, рівнозначно, оптимальної оцінки з використанням випадкових моделей Байєса Маркова [3].

1.1.1 Лінійна фільтрація

Локально-адаптивне вирівнювання гістограми є прикладом оператора сусідства або локального оператора, який використовує колекцію значень пікселів поблизу даного пікселя для визначення його кінцевого вихідного значення. Для прикладу на рисунку 1.7 зображення зліва згортається з фільтром посередині, для отримання зображення справа. Світло-блакитні пікселі позначають сусідство джерела для світло-зеленого пікселя призначення. Окрім виконання локальної тонового налаштування, сусідні оператори можуть бути використані для фільтрації зображень, щоб додати м'яке розмиття, різкість деталей, підкреслення країв або видалення шуму (рис. 1.8 б-г). У цьому розділі ми розглянемо лінійні оператори фільтрування, які включають в себе масивні комбінації пікселів у невеликих кварталах. У розділі 1.2.2 ми розглядаємо нелінійні оператори, такі яку розділі 1.2.3 - морфологічні фільтри та дистанційні перетворення у розділі 1.2.5.

Найбільш часто використовуваним типом оператора сусідства є лінійний фільтр, в якому значення вихідного пікселя визначається як зважена сума значень вхідних пікселів (формула 1.1),

$$□(□,□) = \sum_{□,□} □(□ + □, □ + □)□(□, □) \quad (1.1)$$

Записи ядра ваги або маски *h(k,l)* часто називаються коефіцієнтами фільтра. Вищезгаданий кореляційний оператор може бути більш компактно позначений як

$$□ = □ \otimes □ \quad (1.2)$$

Загальним варіантом за цією формулою є

$$□(□,□) = \sum_{□,□} □(□ - □, □ - □)□(□, □) = \sum_{□,□} □(□, □)□(□ - □, □ - □) \quad (1.3)$$

де знаком зсувів у *f* було знехтувано. Це називається оператором згортки,

$$□ = □ * □ \quad (1.4)$$

а *h* являється функцією імпульсної відповіді. Причиною цієї назви є те, що функція ядра, *h*, згортована з імпульсним сигналом, *δ(i, j)* (зображення, де 0 є скрізь, крім оригіналу) відтворює себе як, *h*δ=h*, тоді як кореляція виробляє відбитий сигнал.

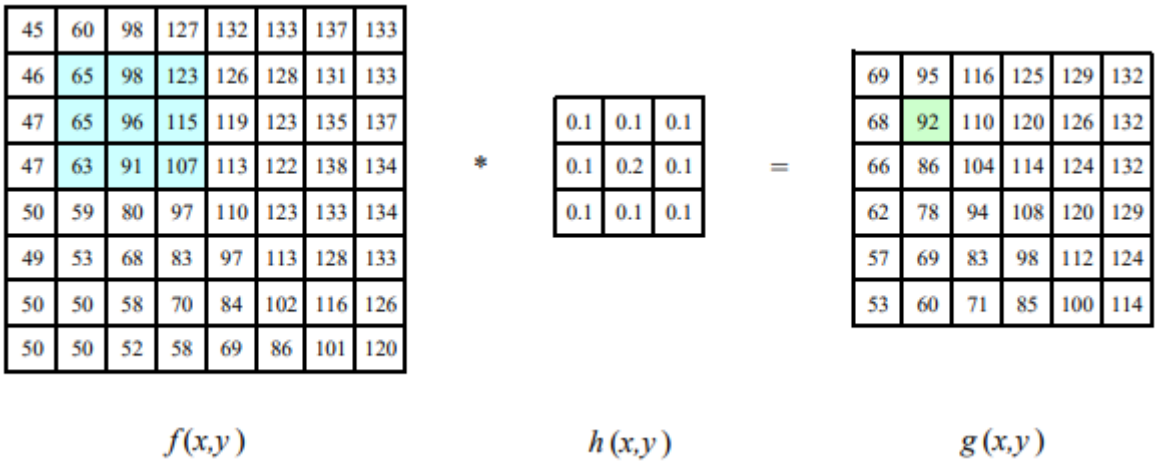

Рисунок 1.7 - Фільтрація сусідства (згортка) [4]

Насправді, рівняння (1.3) можна інтерпретувати як суперпозицію (додавання) зміщених функцій імпульсної відповіді *h(i-k, j -l)*, помножену на значення вхідних пікселів *f(k, l)*. Конволюція має додаткові приємні властивості, наприклад, вона є як комутативною, так і асоціативною. Також перетворення Фур'є двох згортених зображень є продуктом їх окремих перетворень Фур'є.

Обидві кореляції та згортки є операторами лінійного інваріантного зміщення (ЛІЗ), які відповідають принципу суперпозиції,

$$\mathcal{O} \circ (\mathcal{I}_0 + \mathcal{I}_1) = \mathcal{O} \circ \mathcal{I}_0 + \mathcal{O} \circ \mathcal{I}_1 \tag{1.5}$$

і принцип інваріантності зрушень

$$\mathcal{I}(\mathcal{x}, \mathcal{y}) = \mathcal{I}(\mathcal{x} + \mathcal{a}, \mathcal{y} + \mathcal{b}) \Leftrightarrow (\mathcal{O} \circ \mathcal{I})(\mathcal{x}, \mathcal{y}) = (\mathcal{O} \circ \mathcal{I})(\mathcal{x} + \mathcal{a}, \mathcal{y} + \mathcal{b}) \tag{1.6}$$

що означає, що перемикання сигналу переміщується із застосуванням оператора ($\circ$ - це оператор ЛІЗ). Інший спосіб говорити про інваріантність зрушення полягає в тому, що оператор "поводиться скрізь однаково".

$$\begin{array}{|c|c|c|c|c|} \hline 72 & 88 & 62 & 52 & 37 \\ \hline \end{array} * \begin{array}{|c|c|c|} \hline 1/4 & 1/2 & 1/4 \\ \hline \end{array} \Leftrightarrow \frac{1}{4} \begin{bmatrix} 2 & 1 & . & . & . \\ 1 & 2 & 1 & . & . \\ . & 1 & 2 & 1 & . \\ . & . & 1 & 2 & 1 \\ . & . & . & 1 & 2 \end{bmatrix} \begin{bmatrix} 72 \\ 88 \\ 62 \\ 52 \\ 37 \end{bmatrix}$$

Рисунок 1.8 - Одномірна сигнальна згортка як розріджена множина матриці-вектора, $g = Hf$ [4]

Рисунок 1.9 - Деякі операції: (а) оригінальне зображення; (б) розмита; (в) загострений; (г) згладжений фільтром із збереженням краю; (д)подвійне зображення; (е) розширена; (є) дистанційне перетворення; (ж) пов'язані компоненти[4]

Інколи може бути використана версія з кореляцією або згорткою змінного варіанту, наприклад,

$$\square(\square,\square) = \sum_{\square,\square} \quad \square(\square - \square, \square - \square)\square(\square,\square;\square,\square) \tag{1.7}$$

де *h(k, l; i, j)* - ядро згортка в пікселі *(i, j)*. Наприклад, таке просторово змінюване ядро може бути використане для моделювання розмивання в зображенні через різноманітний глибинний дефокус.

Кореляція та згортка можуть бути записані як множинність матричного вектора, якщо спочатку ми перетворимо двовимірні зображення *f (i, j)* і *g (i, j)* в растрово-упорядковані вектори *f* та *g*,

$$\square = \square\square \tag{1.8}$$

де (розріджена) матриця *H* містить ядра згортки. Рисунок 1.8 показує, як одновимірну згортку можна представити в матрично-векторній формі.

Матриця множення, показана на рисунку 1.8, страждає від граничних ефектів, тобто результати фільтрації зображення в цій формі призведуть до затемнення кутових пікселів. Це відбувається тому, що оригінальне зображення фактично замінюється 0 значеннями там, де ядро згортки виходить за межі вихідних зображень.

Для того, щоб компенсувати це, було розроблено ряд альтернативних режимів підключення або розширення (рис. 1.10):

- нуль: встановити всі пікселі за межами вихідного зображення до 0 (хороший вибір для альфа-матових різаних зображень);
- константа (колір кордону): встановлювати всі пікселі за межами вихідного зображення до вказаного значення кордону;
- затиск (реплікація або затискання до краю): повторити пікселі краю на невизначений час;
- (циклічне) обгортання (повторення або плитка): цикл "навколо" зображення в "тороїдальній" конфігурації;
- дзеркало: відображає пікселі по краю зображення;
- продовжити: подовжити сигнал, витягнувши дзеркальну версію сигналу з значення пікселя краю.

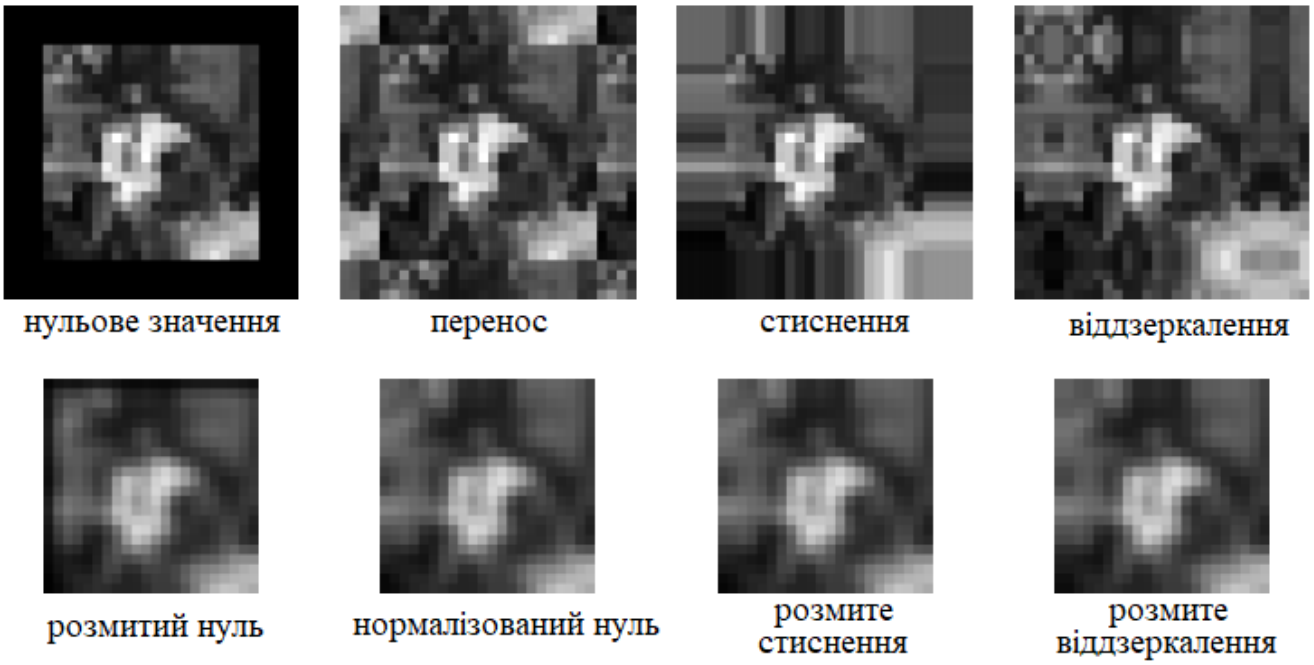

Рисунок 1.10 - Поліпшення кордону (верхній рядок) та результати розмивання полірованного зображення (нижній рядок) [4]

Нормальне нульове зображення є результатом розділення (нормалізації) розмитого зображеного зображення RGBA за допомогою відповідного м'якого альфа-значення.

У комп'ютерній графічній літературі [5] ці механізми відомі як режим обгортки (OpenGL) або режим адресації текстур (Direct3D). Формули для кожного з цих режимів залишаються читачеві.

Рисунок 1.10 показує ефекти підкладки зображення за допомогою кожного з вищезгаданих механізмів, а потім згладжує отримане полірованне зображення. Як ви бачите, нульова підкладка затемняє краї, затискання (реплікація) поширює значення кордонів всередині, дзеркальне (відбиття) заповнення зберігає кольори поруч із границями. Подовжувальна підкладка (не показана) фіксує пікселі кордону (під час розмиття).

Альтернативою заповнюванню є розмивання зображень RGBA з нульовою підкладкою, а потім розділення отриманого зображення на його альфа-значення,

щоб видалити ефект затемнення. Результати можуть бути досить хорошими, як це видно з нормованого нульового зображення на рисунку 1.10.

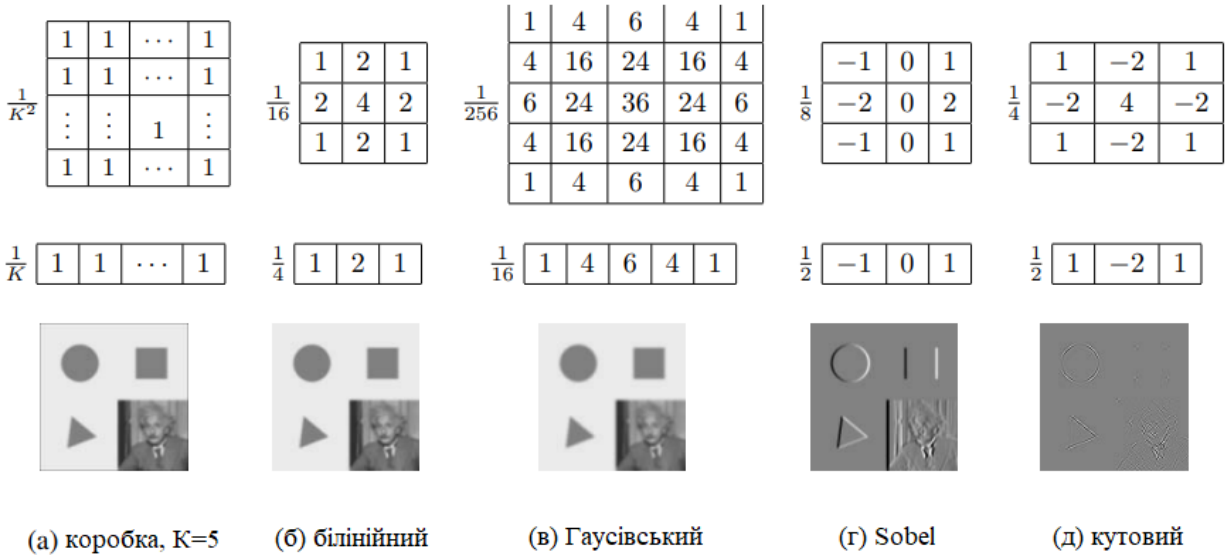

Рисунок 1.11 - Лінійні фільтри [4]

Процес виконання згортки вимагає K^2 операцій (багаторазового додавання) на піксель, де K - розмір (ширина або висота) ядра згортки, наприклад фільтр коробки на малюнку 1.11а. У багатьох випадках ця операція може бути значно збільшена, спочатку виконуючи одномірну горизонтальну згортку, а потім одновимірну вертикальну згортку (що вимагає всього 2K операцій на піксель). Ядро згортка, для якого це можливо, вважається роздільним.

Для кожного з зображень (а - д) рисунка 1.11 показується ядро 2D фільтра (зверху), відповідне горизонтальне ядро 1D (середнє) та фільтруване зображення (внизу). Фільтроване зображення Sobel та кутові зображення підписано, збільшено на 2 × та 4 ×, відповідно, і додано до сірого зсуву перед відображенням.

Легко показати, що двовимірне ядро K, яке відповідає послідовній згортці з горизонтальним ядром h і вертикальним ядром v, є зовнішнім продуктом двох ядер,

$$K = vh^T \quad (1.9)$$

(для деяких прикладів див. малюнок 1.11). Завдяки підвищеній ефективності, конструкція згортання ядер для комп'ютерних зображень часто впливає на їх відокремленість.

Як ми можемо визначити, чи справді ядро *K* роздільне? Це часто можна зробити шляхом перевірки аналітичної форми ядра. Більш прямим методом є обробка 2D ядра як 2D-матриці *K* і прийняття його єдиного значення розкладу (ЄЗР),

$$\square = \sum_{\square} \quad \square_{\square}\square_{\square}\square^{\square}_{\square} \tag{1.10}$$

Якщо перше одиничне значення $\square_0$ **відрізняється від нуля, ядро роздільне і** $\sqrt{\square_0}\square_0$ **і** $\sqrt{\square_0}\square_0^{\square}$ забезпечують вертикальні та горизонтальні ядра. Наприклад, лапласиан Гаусівського ядра може бути реалізований як сума двох роздільних фільтрів.

Що робити, якщо ваше ядро не є відокремленим, але ви все ще хочете швидше реалізувати його? Перона, який вперше встановив зв'язок між сепарабельністю ядра та ЄЗР, пропонує використовувати більше значень у серії, тобто підсумовувати ряд роздільних згорток. Чи варто це робити чи ні, залежить від відносних розмірів *K* та кількості значущих сингулярних значень, а також інших міркувань, таких як когерентність кеш-пам'яті та локація пам'яті.

Тепер, коли ми описали процес виконання лінійної фільтрації, давайте розглянемо ряд часто використовуваних фільтрів.

Найпростіший фільтр для реалізації - це фільтр переміщення середнього фільтру або фільтру коробки, який просто розраховує середні значення пікселів у вікні *K*×*K*. Це еквівалентно злиттю зображення з ядром всіх одиниць, а потім масштабування (рис. 1.11a). Для великих ядер, більш ефективне виконання полягає в тому, щоб пересунути вікно, що рухається, через кожний сканлайн (у відокремлюваному фільтрі), додавши останній піксель і віднімаючи найстаріший

піксель від поточної суми. Це пов'язано з концепцією підсумкових таблиць площ, які ми коротко опишемо.

Більш плавне зображення можна отримати шляхом роздільного згортання зображення частково-лінійною "покриваючої" функції (також відомої як фільтр Бартлетта). На рисунку 1.11б показана версія 3×3 цього фільтра, яка називається білінійним ядром, оскільки це зовнішній продукт двох лінійних (першого порядку) сплайнів.

Звернення лінійної покривної функції з собою дає кубічний аппроксимаційний сплайн, який називається ядром «Гауса» (рис. 1.11в) представлений у піраміді Лапласа). Зверніть увагу, що приблизні гаусові ядра також можуть бути отримані ітераційною згорткою з фільтрами коробки . У програмах, де фільтри дійсно повинні бути обернено симетричними, слід використовувати ретельно налаштовані версії відібраних гаусів.

Ядра, про які ми щойно говорили, - це всі приклади розмивання (згладжування) або низькочастотні ядра (оскільки вони проходять через нижні частоти при зменшенні частот). Наскільки добре вони роблять це? Зазвичай використовується аналіз частоти-простір Фур'є для перевірки точної частотної характеристики цих фільтрів. Також представляється фільтр *sinc ((sin x) / x)*, який виконує ідеальну фільтрацію низьких частот.

На практиці згладжування ядра часто використовуються для зменшення високочастотного шуму.

Як не дивно, згладжування ядра також може використовуватися для загострення зображень за допомогою процесу, що називається негострим маскування. Оскільки розмивання зображення зменшує високі частоти, додавання деякої різниці між оригіналом та розмитим зображенням робить його чіткішим,

$$\square_{\square\square\square\square} = \square + \square(\square - \square_{\square\square\square\square} * \square) \qquad (1.11)$$

Фактично, до появи цифрової фотографії це був стандартний спосіб загострити зображення в приміщенні темної кімнати: створювати розмитий ("позитивний") негатив від вихідного негативного шляхом неправильного фокусування, а потім накладати два негативи перед друком остаточного зображення, який відповідає

$$□_{□□□□□□□□} = □(1 - □□_{□□□□} * □) \quad (1.12)$$

Це вже не лінійний фільтр, але він працює.

Лінійна фільтрація також може використовуватися як етап попередньої обробки до екстракції краю та алгоритмів виявлення відсоткових точок. Рисунок 1.11г показує простий *3×3* крайовий екстрактор, який називається оператором Sobel, що є відокремленою комбінацією горизонтальної центральної різниці (має таку назву, оскільки горизонтальна похідна розташована в центрі пікселя) та вертикальний фільтр покриття (для згладжування результатів) . Як видно на зображенні нижче ядра, цей фільтр ефективно підкреслює горизонтальні краю.

Простий детектор кутів (рис. 1.11д) шукає одночасно вертикальні та вертикальні похідні. Однак, як ви бачите, він реагує не тільки на кути площі, але й уздовж діагональних країв [4].

1.1.2 Нелінійна фільтрація

Фільтри, які ми розглядали до цього, були лінійними, тобто їх відповідь на суму двох сигналів така ж, як і сума окремих відповідей. Це еквівалентно тому, що кожний вихідний піксель - це зважене підсумовування деякої кількості вхідних пікселів. Лінійні фільтри легше складаються і піддаються аналізу частотного відгуку.

У багатьох випадках краща продуктивність може бути отримана за допомогою нелінійної комбінації сусідніх пікселів. Розглянемо, наприклад, зображення на рисунку 1.12д, де шум, не Гаусівський, а ударний шум, тобто іноді має дуже великі

значення. У цьому випадку регулярне розмивання з гаусовим фільтром не дозволяє видалити шумові пікселі, а замість цього перетворює їх на м'які (але все-таки видимі) місця (рис 1.12е).

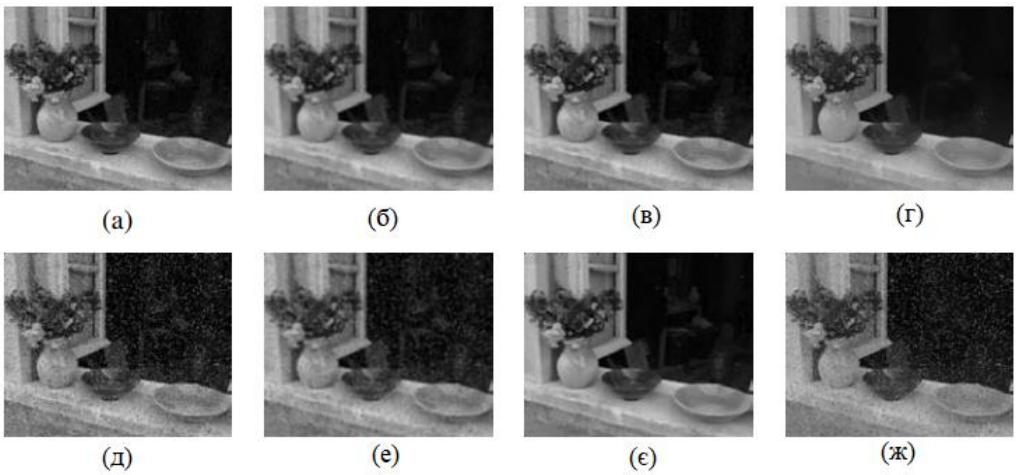

Рисунок 1.12 - Медіана та двостороння фільтрація [6]

Трохи детальніше розглянемо рисунок 1.12: (а) оригінальне зображення з Гаусовським шумом; (б) фільтр Гауса; (в) медіанний фільтр; (г) двосторонній фільтр; (д) оригінальне зображення з ударним шумом; (е) фільтр Гауса; (є) медіанний фільтується; (ж) двосторонній фільтр. Зауважте, що двосторонній фільтр не може видалити ударний шум, оскільки шумові пікселі занадто відрізняються від сусідів[6].

1.1.3 Морфологічні перетворення

Хоча нелінійні фільтри часто використовуються для поліпшення відтінків сірого та кольорових зображень, вони також широко використовуються для обробки бінарних зображень. Такі зображення часто трапляються після операції, що складається з порога, наприклад, перетворення відсканованого відтінку сірого у бінарне зображення для подальшої обробки,

$$\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases} \tag{1.13}$$

наприклад, оптичного розпізнавання символів.

Найпоширеніші операції двійкового зображення називаються морфологічними операціями, оскільки вони змінюють форму основних бінарних об'єктів. Для виконання такої операції ми спочатку обробляємо бінарне зображення за допомогою елемента двійкового структурування, а потім вибираємо значення бінарного виводу залежно від порогового результату згортки. Це не звичайний спосіб опису цих операцій, але він вважається простим способом уніфікації процесів. Елементом структурування може бути будь-яка форма, від простого *3×3* коробкового фільтра до складніших структур диска . Він навіть може відповідати певній формі, яку шукають у зображенні.

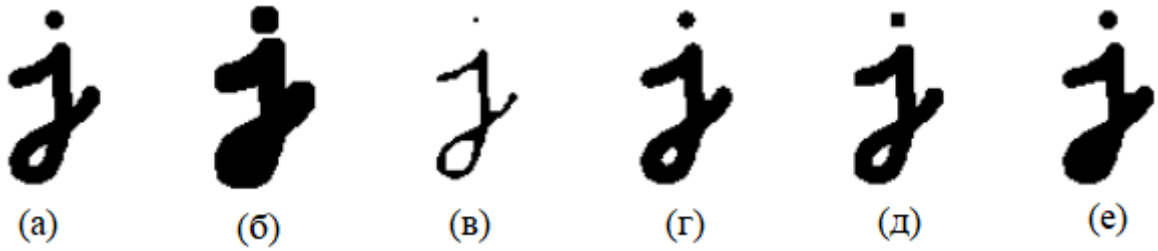

Рисунок 1.13 - Двійкові морфологічні перетворення [7]

На рисунку 1.13 показано крупне зображення згортки бінарного зображення *f* з *3×3* структуруючим елементом *s* і отриманими зображеннями для операцій, описаних нижче. Детальніше про рисунок 1.13: (а) оригінальне зображення; (б) dilation; (в) erosion; (г) majority; (д) opening; (е) closing. Елемент структурування для всіх прикладів - 5×5 кв. Majority - це тонке округлення різких кутів. Opening не дозволяє усунути точку, оскільки вона не досить широка.

Нехай

$$\square = \square \otimes \square \tag{1.14}$$

буде цілочисельним значенням кількості одиниць всередині кожного елемента, отримано від сканування зображенням, а *S* - розмір елемента структурування (кількість пікселів). Стандартні операції, що використовуються в двійковій морфології, включають:

- dilation: dilate(f, s) = θ(c, 1);
- erosion: erode(f, s) = θ(c, S);
- majority: maj(f, s) = θ(c, S/2);
- opening: open(f, s) = dilate(erode(f, s), s);
- closing: close(f, s) = erode(dilate(f, s), s).

Як ми бачимо з рисунку 1.13, розширення збільшує (потовщує) об'єкти, що складаються з одиниць, тоді як ерозія стискає їх (зменшує). Операції з відкриття та закриття, як правило, залишають великі регіони і не впливають на гладкі межі, одночасно видаляючи невеликі об'єкти або отвори та згладжуючи межі.

Поки ми не будемо використовувати багато математичної морфології, а просто помітимо, що це зручний інструмент для того щоб, коли потрібно, очистити деякі порожні зображення. Ви можете знайти додаткові подробиці щодо морфології в інших підручниках з комп'ютерного зору та обробки зображень, а також статті та книги, що стосуються саме цього тема [7].

1.1.4 Трансформації кольору

Хоча кольорові зображення можна розглядати як довільні векторні функції або збірки незалежних ліній, зазвичай їх представляють як високо корельовані сигнали з сильними поєднанням процесу формування зображення, дизайном сенсора, і сприйняттям людини.

Фактично, додавання одного і того ж значення до кожного колірного каналу не тільки збільшує очевидну інтенсивність кожного пікселя, але також може вплинути

на тональність та насиченість пікселя. Як ми можемо визначити та маніпулювати такими величинами, щоб досягти потрібних перцептивних ефектів?

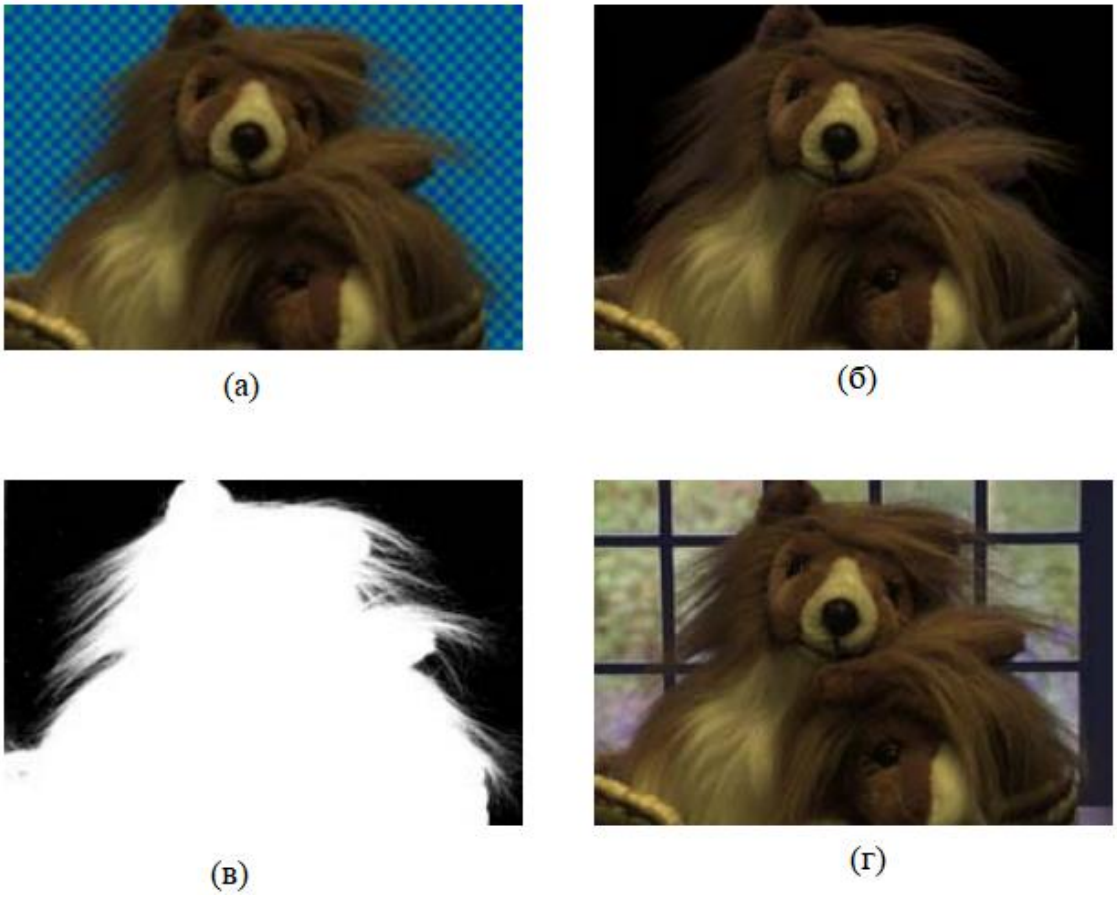

Рисунок 1.14 - Матування та композитування зображень [8]

На рисунку 1.14 представлено матування та композитування зображень: (а) вихідне зображення; (б) витягнутий об'єкт переднього плану F; (с) альфа-матовий α, представлений у градаціях сірого; (г) новий композитний C.

Повертаючись до досягання потрібних перцептивних ефектів, початку можна обчислити координати кольоровості (або навіть більш прості кольорові коефіцієнти), та після цього виконувати маніпулювання світлістю Y для повторні оброчки дійсного RGB зображення з таким же відтінком і насиченістю.

Аналогічним чином, балансування кольорів (наприклад, для компенсації освітлення) може виконуватися або шляхом множення кожного каналу на інший масштабний коефіцієнт або більш складним процесом відображення в колірний простір XYZ, зміною номінальної білої точки та відображенням назад до RGB , який

можна записати за допомогою лінійної матриці перетворення 3×3 кольорового повороту [8].

1.1.5 Дистанційні перетворення

Дистанційне перетворення корисно для швидкого обчислення відстані до кривої або набору точок з використанням двопрохідного растрового алгоритму . Воно має безліч застосунків, включаючи набори рівнів, швидке підстроювання шаф (двостороння вирівнювання зображень) , накладання зображень для зшивання чи змішування, а також найближчу точка вирівнювання .

Відстань перетворення *D(i, j)* двійкового зображення *b(i, j)* визначається наступним чином. Нехай *d(k, l)* - деяка метрика відстані між змінами пікселів. Два загальноприйнятих показника включають манхеттенську відстань

$$□_1(□,□) = |□| + |□| \tag{1.15}$$

і евклідову відстані

$$□_2(□,□) = \sqrt{□^2 + □^2} \tag{1.16}$$

Віддалене перетворення після цього визначається як

$$D(i,j) = \min_{k,l:b(k,l)=0} d(i-k, j-l), \tag{1.17}$$

Тобто, це відстань до найближчого фонового пікселя, значення якого дорівнює 0.

Перетворення відстані міського блоку *D1* можна ефективно обчислити, використовуючи прямий та зворотний пропуски простого алгоритму растрового сканування, як показано на рисунку 1.15. Під час прямого переходу кожний

ненульовий піксель в *b* замінюється мінімумом від 1 плюс відстань його північного або західного сусіда. Під час зворотного проходу відбувається те ж саме, за винятком того, що мінімум обидва перевищує поточне значення *D* і 1 плюс відстань південних та східних сусідів (рис. 1.15).

Детальний опис рисунку 1.15: (а) оригінальне двійкове зображення; (б) зверху вниз (вперед) растровий розмах: зелені значення використовуються для обчислення помаранчевих значень; (в) знизу вгору (назад) растровий розмах: зелені значення зливаються зі старими помаранчевими значеннями; (г) остаточне дистанційне перетворення.

Ефективне обчислення евклідової відстані є більш складним. Тут недостатньо просто зберегти мінімальну скалярну відстань до кордону протягом двох проходів. Замість цього, векторна відстань, що складається як з координат *x* та *y* відстані до кордону, повинна зберігатись та порівнюватися за допомогою правил квадрата відстані (гіпотенузи). Крім того, для отримання достовірних результатів потрібно використовувати більші регіони пошуку.

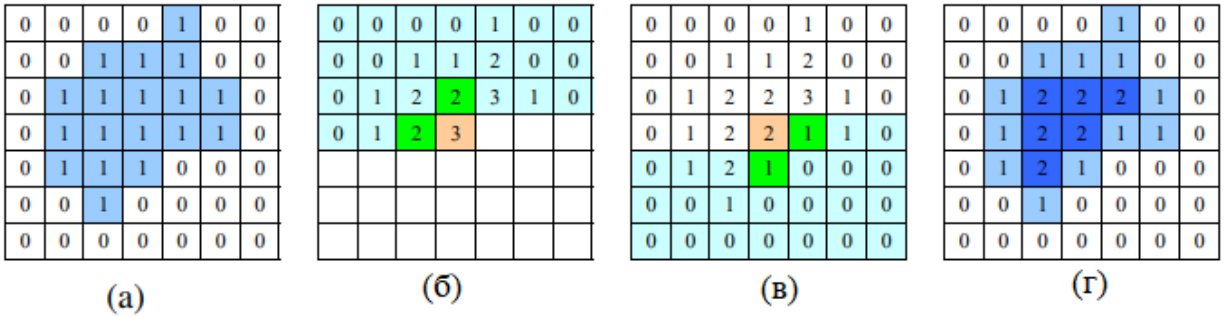

Рисунок 1.15 - Відстань трансформації міського блоку [9]

Рисунок 1.9є показує перетворення відстані, обчислену з бінарного зображення. Зверніть увагу, як значення зростають далі від чорних регіонів і утворюють хребти в білій області оригінального зображення. Через це лінійне зростання від початкових граничних пікселів, віддалене перетворення також іноді називають перетворення згоранням дерева, оскільки воно описує час, коли вогонь, що починається всередині чорного регіону, споживає будь-який заданий піксель,

оскільки він нагадує подібні фігури, що використовуються в деревообробному та промисловому дизайні. Хребти на відстані перетворюються в скелет (або медіальне перетворення осей (МАО)) регіону, де обчислюється перетворення, і складається з пікселів, які мають однакову відстань до двох (або більше) меж.

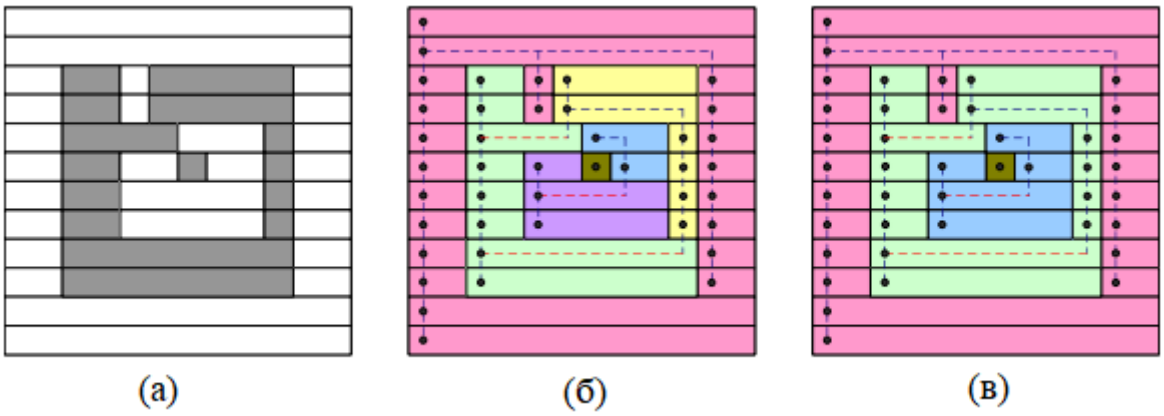

Рисунок 1.16 - Обчислення з'єднаних компонентів [9]

На рисунку 1.16 показано: (а) оригінальне зображення у градаціях сірого; (б) горизонтальні смуги (вузли), з'єднані вертикальними (графічними) краями (пунктирними синіми), - прутки - псевдоцвітні з унікальними кольорами, успадкованими від батьківських вузлів; (с) повторне фарбування після об'єднання суміжних сегментів.

Корисним продовженням базового перетворення дистанції є підписане віддалене перетворення, яке обчислює відстані до граничних пікселів для всіх пікселів. Найпростіший спосіб створити це - обчислити перетворення дистанції як для оригінального бінарного зображення, так і для його доповнення, а також для того, щоб відмовитися від одного з них перед об'єднанням. Оскільки такі поля відстані мають тенденцію бути гладкими, їх можна зберігати більш компактно (з мінімальними втратами в відносній точності) за допомогою сплайну, визначеного за структурою даних з квадратним або восьмизначним даними. Такі попередньо підписані віддалені перетворення можуть бути надзвичайно корисними для ефективного вирівнювання та об'єднання 2D-кривих та 3D-поверхонь, особливо

якщо векторна версія відстані перетворюється, тобто покажчик від кожного пікселя або вокселю до найближчого межі або елемента поверхні зберігається та інтерполюється. Підписані поля відстані також є важливим компонентом еволюції набору рівня, де вони називаються характерними функціями[9].

1.2 Аналіз існуючих рішень

Існує досить велика кількість програм, завдання яких – підрахунок людей, що входять, або виходять з приміщення. Для виконання цієї задачі використовуються різні методи, які в свою чергу мають свої переваги та недоліки. Зазвичай кінцевий продукт представляє собою поєднання різних методів аналізу даних, таких як: обробка стерео- або моно-зображення, використання інфрачервоного датчику, сканування Wi-Fi або Bluetooth сигналів.

Гарним прикладом такого рішення є V-Count 3D+ Alfa [10]. У ньому використовують синхронну обробку зображення з двох камер, а також аналіз Wi-Fi сигналів.

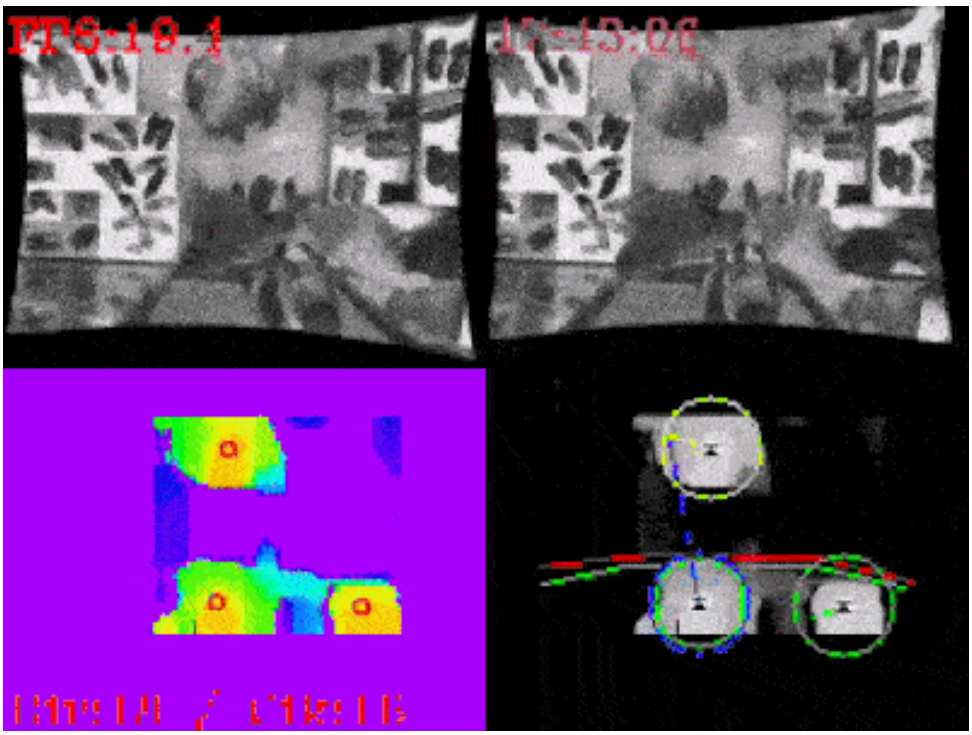

Рисунок 1.17 - V-Count 3D+ Alfa [10]

Як зазначають розробники продукту, V-Count 3D Alpha + забезпечує мінімальний рівень точності підрахунку 98% зі своєю технологією зі стереозвизненням, найкращою технологією, яка на даний момент може використовуватись для підрахунку рухомих обєктів. Також компанія-розробник повідомляє, що це найточніший і надійний спосіб вимірювання активності людей у фізичному просторі.

Двосторонній компонент пристрою може подраховувати людей одночасно на вхід і вихід.

V-Count 3D Alpha + має технологію Wi-Fi, вбудовану в пристрій. Окрім обчислення кількості користувачів, він надає дані відстеження Wi-Fi, включаючи швидкість з'єднання.

V-Count 3D Alpha + призначений для роздрібних торговців, розважальних центрів, операторів торгових центрів, медичних закладів та всіх тих користувачів, які хочуть зрозуміти, як часто люди входять, переміщуються та виходять з певних просторів. Пристрій може забезпечити контроль широкого діапазону трафіку, черг та інших складних ситуацій. Це дає додаткову інформацію про клієнтів та характер їх покупок. Компанія представляє декілька версій фізичних пристроїв із певними специфікаціями. Існують такі пристрої, як V-Count 3D+ Mini, V-Count 3D+ Alpha, V-Count Heat Map, V-Count Wi-Fi Tracker та V-Count Queue. Всі вони мають своє певне призначення та досить точно виконують поставлені перед ними задачі. Проте, як вже було раніше сказано, дана система є складною та складається з декількох окремих підсистем для підрахунку людей, що в комбінації дає дуже точний результат. Окрім цього така складність та різноманітність в свою чергу відображається на ціні продукту.

Тож окрім вищезгаданих складних систем слід виділити такі, що використовують лише один метод для підрахунку відвідуваності. Наприклад Automated-Attendance-System (AAS)[11]. В основі логіки для аналізу відвідуваності лежить метод розпізнавання облич. Для виконання функцій обробки зображення в

даному продукті використовується бібліотека Emgu CV, яка є .Net обгорткою навколо бібліотеки Open CV [12]. Розпізнавання облич людей виконується за допомогою каскадів haarcscade_eye та haarcascade_frontalFace_default.

Іншим прикладом програми, що використовує обробку моно-зображення для аналізу є OCPC (Overhead Camera People Counter) [13]. Для обробки зображень в даному рішенні також використовується функціонал бібліотеки OpenCV. Для правильного функціонування алгоритму передбачається розміщення камери на стелі приміщення та задання координат лінії, по пересіченні якої відбувається реакція.

Проте жодне з існуючих рішень не надає стовідсоткової точності. Покращення результату може бути досягнуте шляхом вдосконалення окремих методів. До того ж при наявності високоточних методів, буде спрощена їх комбінація, що призведе до зменшення собівартості таких рішень.

Слід зазначити, що жодне з рішень, в основі яких лежить аналіз послідовності кадрів з відео, не є стійким до впливу світла на приміщення де проходить відеоспростереження. Також великою проблемою, що впливає на якість роботи алгоритму є взаємодія об’єктів розпізнавання. В момент часу, коли декілька людей взаємодіють один з одним на відео, алгоритми, що про які йде мова, некоректно оброблюють такі ситуації, що призводить до неточностей у їх роботі.

1.3 Огляд бібліотек комп'ютерного бачення

Для вирішення деяких підзадач алгоритму логічним було б використання вже існуючих інструментів у галузі комп’ютерного бачення. Можна виділити наступні популярні бібліотеки, функціонал яких призначений для розв’язання таких задач.

1.3.1 Visual something libraries

VXL (Visual something Libraries) [14] – це колекція бібліотек, написаних мовою програмування C++, та призначена для виконання задач в області комп’ютерного

бачення. VXL була розроблена на ANSI/ISO C++ як портативна та мультиплатформенна система. Сюди входять такі бібліотеки як:

- VNL (numerics): Числові контейнери та алгоритми. Наприклад, матриці, вектори, розкладання, оптимізатори.
- VIL (imaging): завантаження, збереження та маніпулювання зображеннями у багатьох звичайних форматах файлів, включаючи дуже великі зображення.
- VGL (geometry): Геометрія для точок, кривих та інших елементарних об'єктів в 1, 2 або 3 вимірах.
- VSL (streaming I/O), VBL (basic templates), VUL(utilites): Різні незалежні від платформи функції.

Окрім основних бібліотек існують бібліотеки, що охоплюють чисельні алгоритми, обробку зображень, координаційні системи, геометрію камери, стерео, відео маніпуляції, відновлення структури від руху, моделювання імовірності, графічний дизайн, класифікація, оцінка надійності, відстеження функцій, топологія, робота над структурами, 3D-зображення та багато іншого.

Кожна базова бібліотека є нескладною, і її можна використовувати без посилання на інші основні бібліотеки. Подібним чином, другорядні бібліотеки не залежать від основних, тому ви можете скласти і пов'язати лише ті бібліотеки, які вам дійсно потрібні.

VXL розробляється та використовується міжнародною командою дослідників, включаючи деяких провідних експертів у сфері комп'ютерного бачення.

1.3.2 LTI

LTI-lib – об'єктно-орієнтована бібліотека, що пропонує достатньо великих набір інструментів та алгоритмів для обробки зображень [15]. Була розроблена на кафедрі технічних комп'ютерних наук (Lehrstuhl fuer Technische Informatik) LTI в Аахенському технологічному університеті в рамках багатьох дослідницьких

проектів з комп'ютерного бачення, що стосуються робототехніки, розпізнавання об'єктів та розпізнавання мов та жестів.

Основною метою LTI-Lib є створення об'єктно-орієнтованої бібліотеки на C++, що спрощує спільне використання та підтримку коду, але забезпечує при цьому швидкі алгоритми, які можуть бути використані в реальних програмах.

Вона розроблена з використанням GCC під Linux та Visual C ++ під Windows NT.

Багато класів інкапсулюють функціональність Windows / Linux, щоб спростити роботу з конкретним системним або апаратним кодом (наприклад, класи для багатопотокової синхронізації, вимірювання часу та доступу до послідовного порту).

Решта класів, кількість яких є більш ніж 500, займаються головним чином наступними функціями:

- Лінійна алгебра. Матриці, вектори, розрахунок рішень лінійних рівнянь, статистики, векторів тощо.
- Класифікація та кластеризація. Радіально-базисні функції класифікації, методи опорних векторів, метод k-середніх, нечіткі C-засоби, класифікаційна статистика - лише деякі приклади того, що ви можете зробити з LTI-Lib.
- Обробка зображення. Більшість класів займаються проблемами обробки зображень. Доступні різні підходи сегментації, лінійні фільтри, вейвлет-перетворення, керовані фільтри та багато іншого.
- Візуалізація та інструменти малювання. Найважча частина при розробці алгоритмів обробки зображень у C ++ це показувати тимчасові зображення під час налагодження. Завдяки об'єктно-орієнтованій архітектурі LTI-Lib, вам просто потрібно створити об'єкт переглядача та надати йому зображення, яке потрібно показати. Якщо вам потрібно надати додаткову інформацію про це зображення (текст, еліпси, прямокутники, лінії або точки), ви можете використовувати один із об'єктів малювання.

1.3.3 OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення з відкритим кодом для комп'ютера та машинного навчання. OpenCV була розроблена для забезпечення загальної інфраструктури застосунків комп'ютерного бачення, а також для прискорення їх використання у комерційних продуктах. Будучи ліцензованим продуктом BSD, OpenCV дозволяє підприємствам легко використовувати та змінювати код.

Бібліотека має понад 2500 оптимізованих алгоритмів, що включає в себе повний комплект як класичного, так і сучасного комп'ютерного бачення та алгоритмів машинного навчання. Ці алгоритми можуть бути використані для виявлення та розпізнавання облич, визначення об'єктів, класифікації людських дій у відео, відстеження рухів камери, відстеження 3D-об'єктів, створення тривимірних точкових хмар із стереокамер, зшивання зображень разом для отримання зображення високої роздільної здатності цілісної сцени , знаходження схожих зображень з бази даних зображень, виправлення кольору очей на фотографіях, створених із використанням спалаху, стеження за рухами очей, налаштовування маркерів, накладання їх на додану реальність тощо. OpenCV має більше 47 000 користувачів та приблизну кількість завантажень більше 14 мільйонів. Бібліотека широко використовується в компаніях, дослідницьких групах та органах влади.

Окрім добре зарекомендованих компаній, таких як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато нових компаній, таких як Applied Minds, VideoSurf та Zeitera, які широко використовують OpenCV. Розгорнуті програми OpenCV охоплюють діапазон вирішення проблем від зшиття об'ємних зображень вулиці, обладнання спростереження в Китаї, допомога роботам в навігації, забиранні об'єктів в Willow Garage, виявлення надзвичайних ситуацій, перевірка злітно-посадкових смуг в Туреччині, перевірка етикеток на продуктах на заводах у всьому світі та багато іншого.

OpenCV має інтерфейси для C ++, Python, Java та MATLAB і підтримується на таких відомих платформах як Windows, Linux, Android та Mac OS. OpenCV спрямовується в основному на додатки в режимі реального часу та використовує інструкції MMX та SSE, коли це можливо. В даний час активно розвиваються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10 разів більше функцій, які складають або підтримують ці алгоритми. OpenCV було написано на C++ тож має шаблонний інтерфейс, завдяки якому без проблем працює з контейнерами STL.

Бібліотека OpenCV має функцію ручного налаштування на використання високооптимізованого коду IPP. На рисунку 1.18 показано порівняння OpenCV, OpenCV з IPP, та бібліотеками, розглянутими раніше - LTI та VXL. Продуктивність було основною задачею OpenCV, у зв'язку з тим, що потребувалось використання комп'ютерного бачення в реальному часі. Той факт, що OpenCV була написана з допомогою високопродуктивного коду на мовах програмування C та C++ ніяк не зв'язаний із IPP, який автоматично підключається до OpenCV для покращення продуктивності.

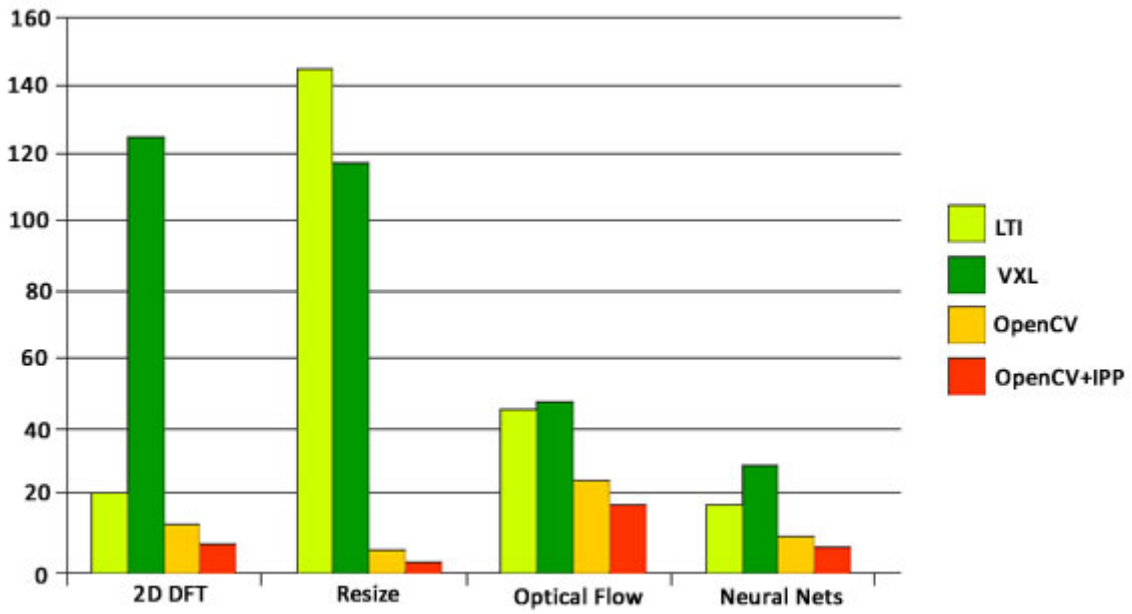

Рисунок 1.18 - Порівняння OpenCV (з та без IPP) та бібліотек комп'ютерного бачення LTI та VXL

Як видно з діаграми, представленої на рисунку 1.18, по всім чотирьом критеріям ефективності бібліотека OpenCv показує кращі результати, OpenCV + IPP переважає OpenCV без IPP (діаграма показує результати, пропорційні часу виконання для кожної з бібліотек по чотирьом критеріям).

Ще однією перевагою використання бібліотеки OpenCV в проекті є наявнічть HTML-документації, що поставляється разом з кодом бібліотеки .

В папці .../opencv/docs можна знайти документ index.html, завантаживши який можна побачити список посилань, кожне з яких відповідає за певні групи алгоритмів, що містяться в бібліотеці. Наприклад, посилання CV містить обробку зображень, аналіз структури зображень, Machine learning (ML) - функції кластеризації, класифікації та аналізу дані, CVCAM - работа с камерой и т.д. Документація, представлена в такому вигляді, допомагає при розробці проекту. Крім того, існує більш новий тип документації по бібліотеці OpenCV, а саме, документація у форматі Wiki. Документація містить вказівки по збірці OpenCV, що використовують середовище розробки Eclipse IDE, розпізнавання осіб з OpenCV, бібліотеку відеоспостереження, список літератури, сумісні камери та посилання на спільноти. Дану документацію можна знайти за адресою [16].

Існує також документація у форматі Wiki з більш унікальною інформацією за різними допоміжними функціями. Дана документація містить опис наступних понять: стерео відповідність, точка зору морфінгу камер, 3d стеження в режимі стерео, об'єкт відповідності (PCA), функції для розпізнавання об'єктів та впровадження схованої Марковської моделі (CMM).

Основна документація по бібліотеці OpenCV представлена англійською мовою, однак на сьогоднішній день існують і російськомовні аналоги даної документації, що багато в чому сприяє тому факту, що більшість російськомовних розробників використовують саме OpenCV при створенні своїх проектів.

Бібліотека OpenCV є структурованою бібліотекою. Вона розділена на п'ять основних компонентів: алгоритми обробки зображень, високорівнісні алгоритми

комп'ютерної зору, MLL - бібліотека машинного навчання, HighGUI - процедури та функції введення та виведення для зберігання та завантаження відео та зображень, CXCore - основні структури даних.

Структура бібліотеки представлена на рисунку 1.19. На цьому рисунку представлений ще один - п'ятий компонент бібліотеки, а саме CvAux (експериментальні алгоритми). CvAux містить: 1D та 2D скриті Марківські моделі, техніку статистичного розпізнавання з використанням методу динамічного програмування, вбудовану CMM, розпізнавання сигнатур завдяки стерео зору, стерео зів, текстовий дескриптор, пошук скелета (центральних ліній) об'єктів сцени, відеоспостереження тощо.

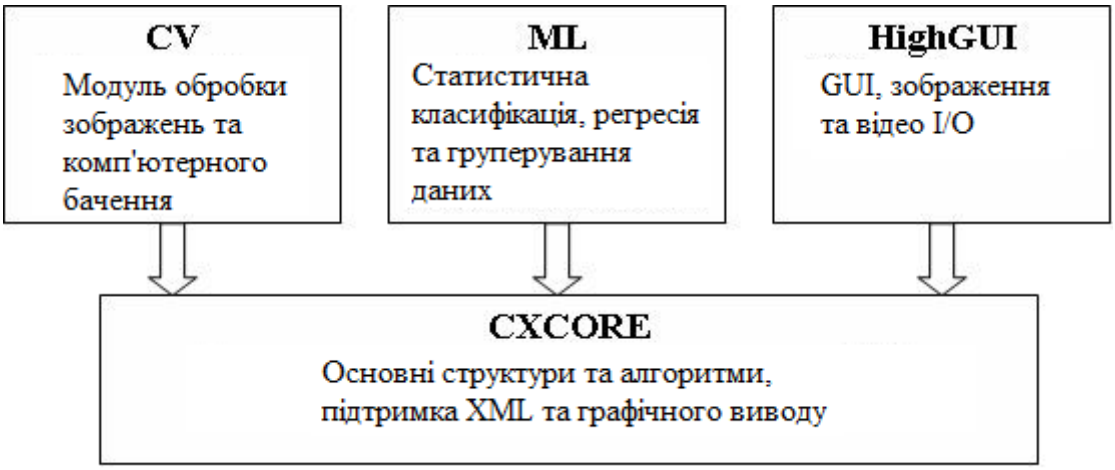

Рисунок 1.19 - Базова структура OpenCV [17]

Ще однією перевагою використання OpenCV є її переносимість. OpenCV розробляється як портативна бібліотека. В самому початку розробка велася на Borland C ++,

MSVC ++, і компілятори Intel. Все це означало той факт, що код в мовах C і C ++ повинен був бути стандартним, щоб створити легку підтримку кроссплатформенності. В початок розробки підтримувана архітектура була 32-бітна Intel з операційною системою Windows, трохи пізніше з підтримкою ОС Linux. Після того як компанія Apple почала використовувати процесори Intel, з'явилася також підтримка MAC OS X. Трохи пізніше з'явилася підтримка і 64-бітної

архітектури Intel. На сьогоднішній день бібліотека OpenCV адаптована майже для всіх комерційних систем, крім того, вона працює і на процесорах AMD, в яких також доступний IPP для підвищення продуктивності (при використанні MMX - мультимедійного розширення) [17].

Отже було вирішено використовувати бібліотеку OpenCV з деяких причин. По-перше вона показує найкращий результат по швидкодії в порівнянні з іншими. По-друге, ця бібліотека має надзвичайно просте та різноманітне API, що дозволяє вирішувати безліч задач у різний спосіб, з використанням тих чи інший популярних алгоритмів.

1.4 Вибір колірних моделей для алгоритму

Під час виявлення окремих об’єктів на зображенні недостатньо інформації лише про його форму та розташування, адже такі параметри в більшості випадків не несуть в собі унікальних знань про такі об’єкти. Найбільш важливими даними про об’єкт є його колір. Саме визначення кольорової характеристики зображення та подальший її аналіз суттєво спрощує задачу розпізнавання та слідкування за об’єктами.

Після виділення окремого об’єкта на зображенні, він представляється у вигляді набору пікселей із певними числовими значеннями кольорів. Кольори в свою чергу задаються у відповідності до колірної моделі, якою представлене зображення.

Колірна модель є абстрактною математичною моделлю, що описує спосіб, яким кольори можуть бути представлені як кортежі чисел. Як правило, як три або чотири значення або компоненти кольору. Коли ця модель асоціюється з точнішим описом того, як компоненти повинні інтерпретуватися, результуючий набір кольорів називається кольоровим простором. Також під колірною моделлю необхідно розуміти спосіб відображення колірної гами в дискретному вигляді, для представлення її в обчислювальних, цифрових системах.

Декілька слів про колірні простори. Можна виділити трикомпонентний колірний простір стимулів та кольоровий простір CIE XYZ.

Можна визначити колірний простір стимулів як лінійний простір, якщо задати координати x, y, z як значення стимулів, відповідних відгуку колб довгохвильового (L), середньохвильового (M) і короткохвильового (S) діапазону оптичного спектру. Початок координат (S, M, L) = (0, 0, 0) представлятиме чорний колір. Білий колір не матиме чіткої позиції в даному визначенні діаграми всіляких кольорів, а буде визначатися, наприклад, через кольорову температуру, певний баланс білого або яким-небудь іншим способом. Повний колірний простір людини має вигляд конуса у формі підкови. Принципово дане подання дозволяє моделювати колір будь-якої інтенсивності — починаючи з нуля (чорного кольору) до нескінченності. Однак, на практиці, людські рецептори перенасичуються або навіть можуть бути пошкоджені випромінюванням екстремальної інтенсивності, тому дана модель не застосовна для опису кольору в умовах надзвичайно високих інтенсивностей випромінювань і також не розглядає опис кольору в умовах дуже низьких інтенсивностей (оскільки у людини використовується інший механізм сприйняття через палички).

Будучи лінійним простором, простір кольорових стимулів має властивість адитивного змішування — сума двох колірних векторів буде відповідати кольору, рівному отриманому змішанням цих двох кольорів (див. також: Закон Грассмана). Таким чином, можна описувати будь-які кольори (вектора колірного простору) через лінійну комбінацію кольорів, обраних як базис. Такі кольори називають основними (англ. primary colors). Найчастіше основними кольорами вибирають червоний, зелений і синій (модель RGB), проте можливі інші варіанти базису основних кольорів. Вибір червоного, зеленого і синього оптимальний з кількох причин, наприклад, бо при цьому мінімізується кількість точок колірного простору, для представлення яких використовуються негативні координати, що має практичне значення для передання кольору (не можна відтворювати колір випромінюванням з негативною інтенсивністю). Цей факт випливає з того, що піки чутливостей L, M і S колб припадають на червону, зелену і синю частини видимого спектру.

Деякі колірні моделі використовуються для відображення кольору, наприклад відтворення кольору на екранах телевізорів і комп'ютерів, або кольорового друку на принтерах. Використовуючи явище метамерії, пристрої передачі кольору не відтворюють оригінальний спектр зображення, а лише імітують стимульну складову цього спектру, що в ідеалі дозволяє отримати картину, яка не відрізняється людиною від оригінальної сцени. На практиці таке, як правило, неможливо, оскільки пристрої відтворення працюють не в повній гамі і мають неідеальні випромінювачі.

Одним з перших математично визначених колірних просторів є колірний простір CIE XYZ (також відомий як колірний простір CIE 1931), створений Міжнародною комісією з освітлення в 1931 році. Ці дані були виміряні для спостереження за людьми та поля зору 2 градусів. У 1964 році були опубліковані додаткові дані для поля зору 10 градусів.

Зауважте, що табличні криві чутливості мають певну кількість сваволі в них. Форми окремих кривих чутливості X, Y і Z можуть бути виміряні з розумною точністю. Проте загальна функція світності (яка фактично є вагою сумою цих трьох кривих) є суб'єктивною, оскільки вона включає в себе запит випробовувача, чи є два джерела світла однаковою яскравістю, навіть якщо вони мають абсолютно різні кольори. Подібним чином, відносні величини кривих X, Y та Z довільно вибираються для отримання рівних площ під кривими. Можна також визначити коректний колірний простір з кривою чутливості X, що має вдвічі більшу амплітуду. Цей новий колірний простір буде мати іншу форму. Криві чутливості в колірному просторі CIE 1931 та 1964 рр. Масштабуються з рівними площинами під кривими.

Іноді колір XYZ представлений координатами яскравості, Y, а також хроматичністю x та y, які визначаються як:

$$x = X / (X + Y + Z), \quad (1.18)$$

$$y = Y / (X + Y + Z). \quad (1.19)$$

Математично x та y - проекційні координати, а кольори діаграми кольоровості займають область дійсної проективної площини. Оскільки криві чутливості CIE мають однакові площі під кривими, світло з плоским енергетичним спектром відповідає точці (x, y) = (0.333,0.333).

Значення для X, Y та Z отримуються шляхом інтегрування продукту спектру світлового пучка та опублікованих функцій корекції кольорів.

Повертаючись до колірний моделей слід виділити основні колірні моделі такі, як RGB, CMYK, HSV та Lab.

1.4.1 RGB

RGB – найбільш розповсюджена адитивна колірна модель, що представляє собою спосіб синтезу кольору за допомогою накладання червоного, синього та зеленого світла відповідно [18].

Кольорову модель RGB використовують при відображенні зображень в електронних системах та пристроях, таких як телевізори, комп'ютери, фотоапарати тощо. До початку електронного віку кольорова модель RGB вже мала тверду теоретичну базу, яка базується на людському сприйнятті кольорів.

RGB - залежна від пристрою колірна модель: різні пристрої визначають або відтворюють задане значення RGB по-різному, оскільки кольорові елементи (наприклад, люмінофори або барвники) та їх відображення окремих складових R, G і B відрізняються в залежності від виробника та технології створення. Окрім цього якість відтворення кольорів на одному й тому самому пристрої може відрізнятися з часом. Таким чином, значення RGB не визначає один і той самий колір на різних пристроях без певного управління кольором.

Типовими пристроями вводу, що сприймають зображення у RGB є кольорові телевізійні та відеокамери, сканери зображень та цифрові камери. Типовими пристроями виводу у RGB є телевізори різних технологій (CRT, LCD, плазма, OLED тощо), дисплеї для комп'ютерів і мобільних телефонів, відеопроектори,

багатоколірні світлодіодні дисплеї та великі екрани. З іншого боку, кольорові принтери - це не RGB-пристрої, а субтрактивні кольорові пристрої (що, як правило, використовують колірну модель CMYK).

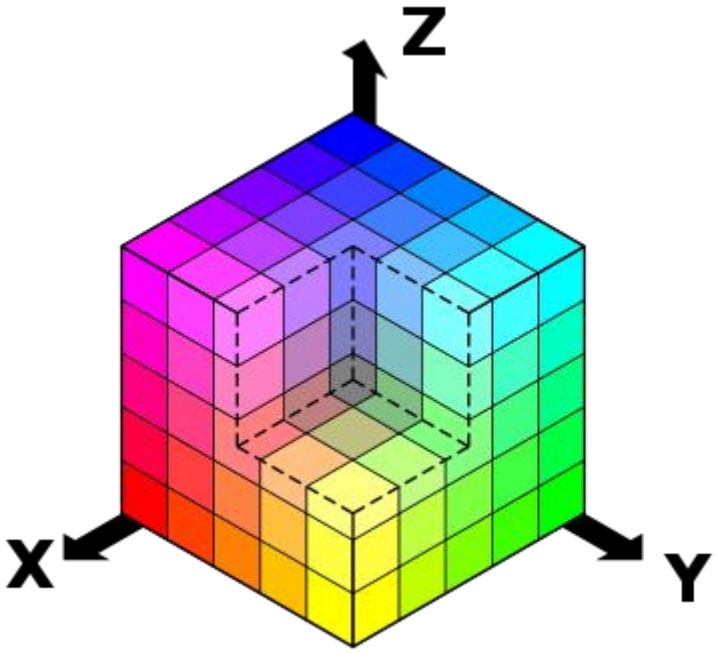

Рисунок 1.20 - Тривимірне представлення RGB моделі

Щоб сформувати RGB колір, необхідно накласти три світлових пучка (червоний, зелений та синій). Кожен з трьох пучків називається компонентом цього кольору, і кожен з них може мати довільну інтенсивність.

Кольорова модель RGB є адитивною в тому сенсі, що три світлових пучка об'єднуються, додаючи довжини хвиль, щоб отримати остаточний колірний спектр. Це, по суті, є протилежним до субтрактивної кольорової моделі, яка застосовується до фарб, чорнил, барвників та інших речовин, колір яких залежить від відображення світла, під яким ми бачимо їх.

При інтенсивності 0 для кожного компонента отримується темний колір (без світла, вважається чорним), а максимальна інтенсивність кожного дає білий колір; якість цього білого залежить від природи первинних джерел світла, але якщо вони правильно збалансовані, результат є нейтральним білим, що відповідає білій точці системи. Коли інтенсивність для всіх компонентів однакова, результат залежить від

інтенсивності у вигляді тіні сірого, темного або світлого. Коли інтенсивність відрізняється, результат - кольоровий відтінок, більш-менш насичений залежно від різниці більших або менших інтенсивностей основних кольорів.

Коли один з компонентів має найсильнішу інтенсивність, колір є відтінком цього основного компонента (червоного, зеленуватого або синюватого кольору), а коли два компоненти мають таку ж сильну інтенсивність, то колір є відтінком вторинного кольору (тіні з блакитного, пурпурового або жовтого кольорів). Другий колір утворюється сумою двох основних кольорів однакової інтенсивності: блакитний - це зелений та синій, пурпуровий - червоний та синій, жовтий - червоний та зелений. Кожен вторинний колір є доповненням одного основного кольору. Коли основний та його додатковий вторинний колір об'єднуються, результатом є білий колір: блакитний доповнює червоний, малиновий доповнює зелений, а жовтий доповнює синій.

Найпростішим методом для створення кольорової характеристики зображення є підрахунок кількості пікселів однакового кольору з використанням RGB моделі. Ідея полягає в тому, щоб порівнювати такі масиви даних з різних кадрів та слідкувати за об’єктами. Такий підхід є найлегшим як у розуміння та і з боку використання обчислювальних ресурсів. Проте існує ряд суттєвих недоліків. Так як кожна складова кольору має значення у межах від 0 до 255, то кількість можливих кольорів складає 16 777 216. При чому при найменшому впливі на об’єкт світла або зміни ракурсу до камери велика частина точок будуть змінювати своє кольорове значення. Для спрощення сприйняття кольору виконується нормалізація таких значень. Попередньо обирається в якому діапазоні цілих чисел повинні відображатись кожен з каналів. Такий діапазон зазвичай є унікальним для різних ситуацій чи потреб. Часто обирають максимальне значення - 7. Тобто кожен кольоровий канал, що попередньо був у діапазоні від 0 до 255, після нормалізації лежатиме в межах цілих чисел від 0 до 7 відповідно. Як нескладно зрозуміти, що для нормалізації необхідно поділити кожне значення кольорового каналу на 32. Маючи спрощені значення кольорів, з ними на порядок легше працювати адже невеликі

зміни у їх відтінках у частині випадків не відображаються на їх цифровому значенні. Проте даних підхід є досить ненадійним. Нормалізовані кольори мають різку зміну тону, а також накладання тіні на частину об’єкта певного кольору може змінити його сприйняття.

1.4.2 CMYK

Кольорова модель CMYK є субтрактивною кольоровою моделлю, що використовується при кольоровому друку, а також для опису самого процесу друку. CMYK складається з чотирьох значень, що використовуються в кольоровому друці: блакитний, пурпурний, жовтий та ключ (чорний).

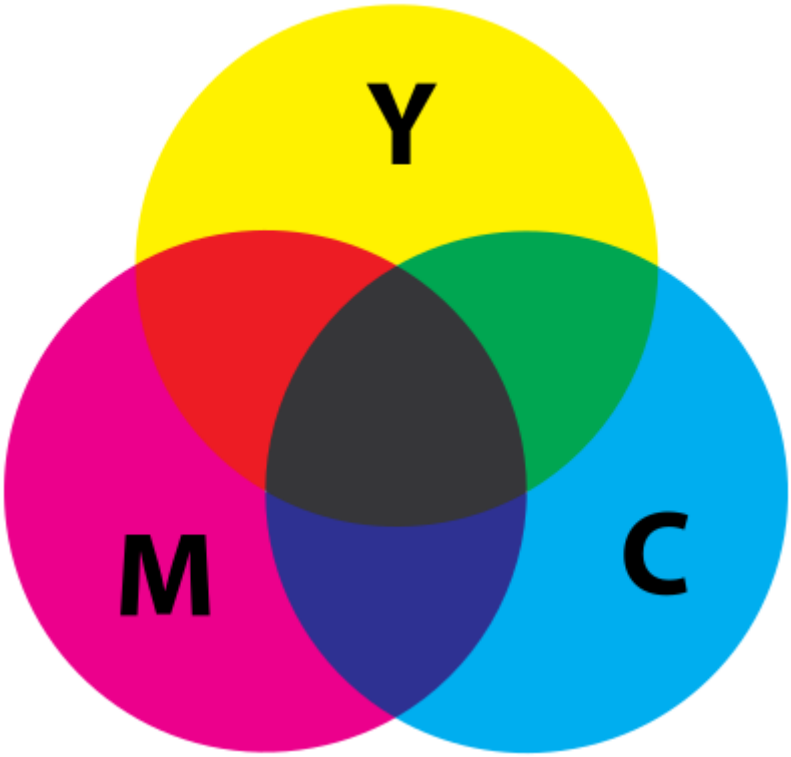

Рисунок 1.21 - образне представлення CMYK моделі

Причиною того, що чорні чорнила називаються ключовими, є те, що в чотирикольоровому друці, блакитний, пурпуровий та жовтий кольори вирівнюються за допомогою чорного, що додає контрасту результуючому кольору. Деякі джерела

вказують на те, що "K" в CMYK походить від останньої букви слова “black” і обрано тому, що B вже означає синій. Однак деякі люди не згодні з цим, тому що в основних кольорах CMYK немає синього кольору (який створюється змішуванням блакитного і пурпурного). Деякі джерела стверджують, що K походить з "Key", тому що чорний часто використовується як контур і друкується спочатку. В сучасному друці чорний друкують останнім, щоб отримати більш глибокі, чисті тіні на відміну від того, як це робиться за допомогою CMY, коли основні кольори надруковані над чорним кольором. Модель CMYK частково або повністю наносить кольори на більш світлий, зазвичай білий, фон. Така модель називається субтрактивною, оскільки чорнила "віднімають" тони червоного, зеленого та синього від білого світла. Біле світло мінус червоне дає блакитне, біле світло мінус зелене дає пурпурне, і біле світло мінус синє дає жовтий колір.

В адитивних кольорових моделях, таких як RGB, білий колір - це "аддитивна" комбінація всіх основних кольорових складових, у той час як чорний - це відсутність світла. У моделі CMYK навпаки: білий - це натуральний колір паперу або інший фон, а чорний - результат повної комбінації кольорових складових. Щоб заощадити витрати на чорнил, а також виробляти більш глибокі чорні тони, ненасичені та темні кольори створюються за допомогою чорних фарб замість комбінації з блакитного, пурпурового та жовтого кольорів [19].

1.4.3 HSV

Отже хотілось би звернути увагу на наступний метод, що вирішує недоліки свого попередника. Метод складання кольорової характеристики на основі HSV моделі враховує ситуації зі зміною тону кольору під час затінення об’єкта. Для початку декілька слів про дану кольорову модель. HSV відображає кольори у вигляді трьох наступних складових: колірний тон (Hue), насиченість (Saturation) та значення кольору (Value) [20]. Колірний тон - це значення “чистого” кольору, тобто

без накладання чорного або білого тонів. Його значення зазвичай представляють в межах $0 - 360^o$, або просто від 0 до 100 (рис. 1.22).

Рисунок 1.22 - Шкала колірного тону (hue)

Насиченість відповідає за відтінок білого у кольорі і лежить у межах від 0 до 100, або від 0 до 1. При показнику 1 колір “чистий”, при 0 - білий. Цей параметр може змінюватись при засвічені об’єкта білим світлом. Тобто при попаданні на об’єкт променів білого відтінку значення Hue, в ідеальному випадку, не змінюється. Змінює своє значення параметр Saturation. Отже можна зазначити, що обираючи колірну модель HSV, покращується сприйняття алгоритмом впливу світла на об’єкт розпізнавання.

Останнім параметром є - значення кольору. Дана складова відповідає за яскравість кольору та задається в аналогічних межах від 0 до 1, або від 0 до 100. При наближенні показника до 0 колір приймає чорний відтінок. Таку особливість колірної моделі дуже зручно використовувати для аналіз падаючої на об’єкт розпізнавання тіні. Візуалізація HSV простору приведена на рисунку 1.23 у вигляді циліндричної системи координат.

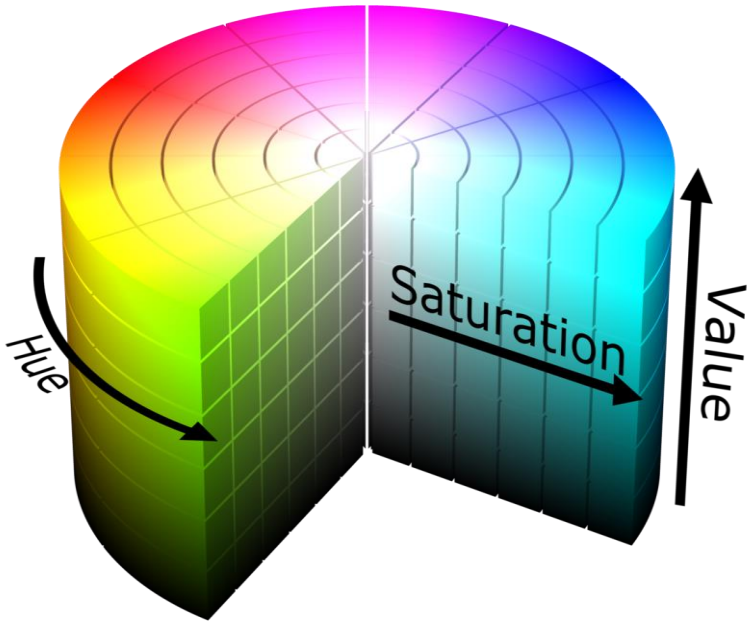

Рисунок 1.23 - Візуалізація HSV моделі

Колірні суміші в RGB просторі можуть відтворювати широкий спектр кольорів, однак співвідношення між складовими червоного, зеленого та синього світла та отриманого кольору не є інтуїтивним, особливо для недосвідчених користувачів, і для користувачів, які знайомі з субтрактивним змішуванням кольорів фарб або моделей традиційних художників на основі відтінків та відтінків. Крім того, ні адитивні, ні субтрактивні колірні моделі не визначають співвідношення кольорів так само, як робить людське око.

Наприклад, уявіть, що ми маємо RGB-дисплей, колір якого контролюється трьома повзунками від 0 до 255, один з яких контролює інтенсивність червоного, інший - зеленого та третій - синього кольорів. Якщо ми почнемо з відносно барвистого помаранчевого кольору з значеннями sRGB R = 217, G = 118, B = 33 і хочемо зменшити його кольоровість наполовину до менш насиченого помаранчевого, нам потрібно буде перетягнути повзунки, щоб зменшити R до 31 , збільшити G на 24, збільшити B на 59. Проте це не було досить інтуїтивним і в спробі використати більш традиційні та зрозумілі моделі змішування кольорів

дослідники комп'ютерної графіки в PARC та NYIT в середині 1970-х років розробили модель HSV.

Така колірна модель в більшості випадків дає кращі результати ніж RGB, проте вони не є ідеальними. Проблеми полягають у сприйнятті кольорів та світла. Іноді при засвічені або затіненні об’єкта він приймає відтінок іншого кольору чим спотворює дані алгоритму обробки.

1.4.4 Lab

Наступною необхідно виділити колірну модель Lab. Дана модель розроблялась з метою представлення кольорів у вигляді якомога ближчому до того, як їх сприймає людина. В просторі Lab відокремили хроматичну складову кольору від світлості. Тим самим Lab однозначно визначає колір, значення якого задається трьома параметрами. За світлість відповідає L, що лежить в діапазоні від 0 до 100, тобто від темного до світлого. Колір же визначається двома координатами - a та b. перша координата визначає відношення зеленого до червоного, а друга відношення синього до жовтого. На рисунках 1.24 та 1.25 показані палітри кольорів при значенні світлості у 25 та 75 відсотків відповідно. Так як в моделі Lab світлість відокремлюється від кольору дія світла на зображення майже не відображає змін у сприйнятті кольорів алгоритмом. А отже створення кольорової характеристики об’єкта виконується дуже подібно на різних кадрах під різним впливом на цих об’єктів світла, що є надзвичайно важливим при обробці послідовності кадрів із метою виділення окремих об’єктів [21].

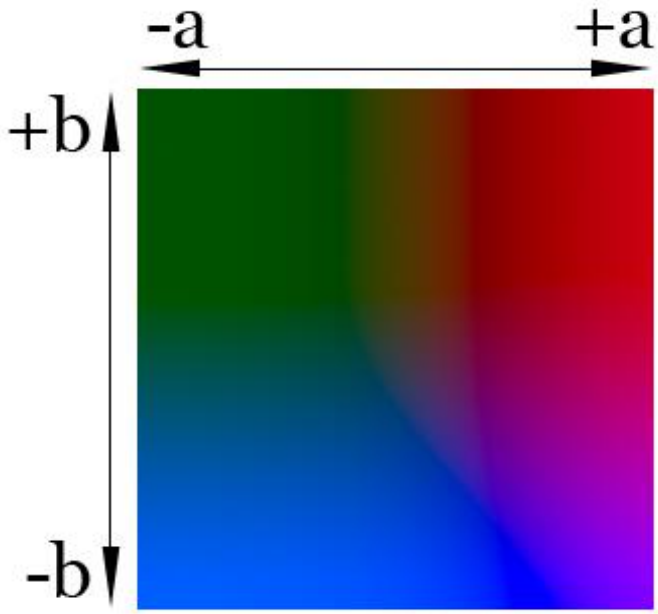

Рисунок 1.24 - Палітра кольорів у колірній моделі Lab при значенні світлості у 25 відсотків

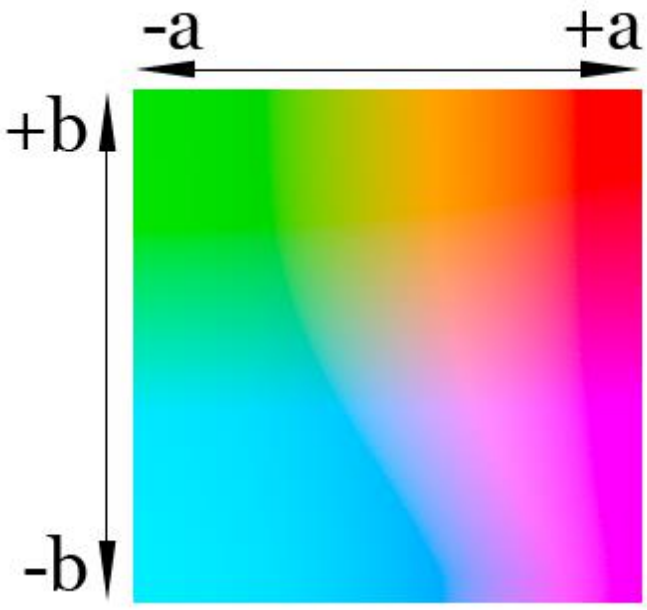

Рисунок 1.25 - Палітра кольорів у колірній моделі Lab при значенні світлості у 75 відсотків

Важливо відмітити, що перетворення RGB зображення в Lab - ресурсозатратний процес. Це може стати на заваді при реалізації алгоритму з

використанням моделі Lab на машинах із невисокими обчислювальними здібностями, наприклад Raspberry Pi.

1.5 Висновки за розділом

Завдяки комп’ютерному баченню сьогодні вирішується безліч проблем, пов’язаних із обробкою зображення. Однією з таких проблем є і підрахунок людей за допомогою камери відеоспостереження. Існує багато підходів для вирішення тих або інших задач, наприклад морфологічні перетворення зображень, лінійна або нелінійна фільтрації. Найбільш популярні підходи розглянуті в даному розділі. Далі проведено аналіз існуючих рішень: як складних систем, так і застосунків, в основі яких лежить метод обробки послідовності зображень. В кінці розділу розглянуто основні колірні моделі, які можна використовувати при обробці зображень.

2 РОЗРОБЛЕННЯ КОМПЛЕКСУ АЛГОРИТМІВ

Для досягнення мети алгоритму необхідно виділити окремі основні його задачі:

- Виявлення активних об’єктів на відео з камери,
- Слідкування за об’єктами
- Аналіз ситуацій із пересічення об’єктами контрольної лінії.

2.1 Алгоритм виявлення активності

Для виявлення активних об’єктів на відео необхідно виконати ряд операцій. В алгоритмі використовується порівняння поточного та попереднього зображення, та подальша обробка результату порівняння (рис. 2.1). По-перше необхідно завжди пам’ятати попередній кадр із послідовності кадрів відео потоку. Також для збільшення швидкості роботи алгоритму рекомендовано зменшити зображення. Мною було використаний розмір 320 на 240 пікселів. Далі необхідно конвертувати кадри стандартної колірної моделі RGB у ту, з якою алгоритм показує якомога кращі результати. В розділі 3.1 проведено дослідження колірних моделей, та обрано найкращу з них, а саме *Lab*. За допомогою функції OpenCV *cv2.cvtColor* конвертуємо кадр у потрібну нам колірну модель. Складову світлості *L* кожної точки зменшуємо в 2 рази. Це необхідно для зменшення впливу світла та тіні на знаходження активних об’єктів. Повністю ігнорувати світлість не потрібно тому, що різниця між сусідніми кадрами невелика і можна не врахувати деяку частину рухомого об’єкта. Маючи таким чином два оброблених послідовних кадри необхідно виконати їх віднімання для знаходження пікселів, що змінились (рис 2.2). Віднімання кадрів виконується наступним чином: від кожної складової колірної моделі кожного пікселя із послідовності по модулю віднімаються значення відповідних складових такого ж порядкового пікселя на другому кадрі. В результаті ми отримуємо новий кадр, ті пікселі, на яких нічого не змінилось мають значення (0, 0, 0), на інших зазначено на скільки по модулю змінився кожен з каналів. Далі, для

простоти операцій та розрахунків, нам необхідно працювати з чорно-білим зображенням. Для цього ми повинні залишити лише одне максимальне значення для кожної точки на результуючому кадрі. Так, як OpenCV представляє складові кольору в межах від 0 до 255, то і результат віднімання їх по модулю також буде лежати в таких межах, а значить і вихідне чорно-біле зображення буде складатися із пікселів із числовим значенням від 0 до 255. Саме значення в такому діапазоні необхідні для подальшої роботи.

Рисунок 2.1 - Приклад послідовних кадрів для обробки

Рисунок 2.2 - Різниця сусідніх кадрів

З отриманих даних ми повинні забрати ту активність на зображенні, якою можуть бути сторонні зміни, такі як шуми на відео. При цьому відповідні пікселі змінюються на невелике значення (рис. 2.3). Тож їх необхідно ігнорувати. Для цього ми встановлюємо поріг, який є унікальним для різних приміщень, де проходить

зйомка відео, та змінюємо значення елементів масиву пікселів наступним чином: якщо значення елемента масиву менше заданого порогу (тобто колір пікселя наближених до чорного) – задаємо його значення 0 (прибираємо піксель), в іншому випадку – задаємо значення 255. Таким чином ми позбуваємось зайвих пікселів і виділяємо активність, що залишилась (рис. 2.4).

Рисунок 2.3 - Приклад виділення всіх змінених пікселів

Рисунок 2.4 - Виділення цікавої зміни зображення

Далі необхідно отримати окремі суцільні об’єкти. Для цього в OpenCV є функція *cv2.findContours*. Вона повертає масив контурів окремих груп пікселів. Та активність, яку ми отримали, недостатньо точно повторює контури реальних об’єктів, і якщо використати функцію *cv2.findContours* зараз, то результатом буде велика кількість менших контурів, як можна бачити на прикладі (рис. 2.4). Для того,

щоб отримати один суцільний об’єкт із декількох сусідніх контурів використовуємо функцію *cv2.dilate* із значенням параметру *kernel = 11x11*. Кожен піксель на зображенні замінюється набором пікселів *11x11*, в результаті чого зображення активності збільшується і заповнюються деякі проміжки між сусідніми контурами (рис. 2.5). Виконання операції розширення перекриває майже всі проміжки між частинами об’єкта, проте деякі неточності залишаються.

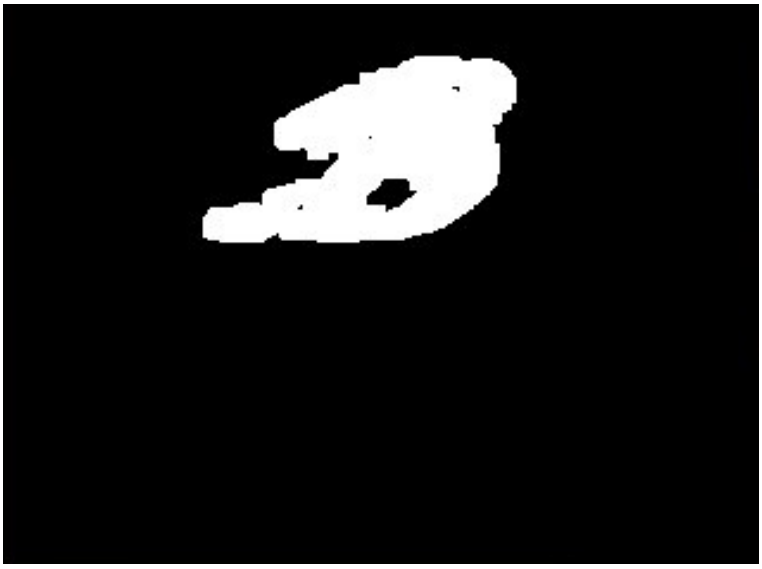

Рисунок 2.5 - Контур після операції *dilate*

Для усунення остаточних недоліків необхідно виконати операцію *cv2.morphologyEx*. Дана функція створена для морфологічних трансформацій над зображеннями. У параметр функції можна передати одну з таких операцій, як Openin, Closing, Morphological Gradient, Top Hat, Black Hat. Кожна з них виконує свої важливі функції, нас цікавить саме операція *Closing*. Дана функція призначена для закривання невеликих чорних проміжків в білих об’єктах. Це остаточно дозволить об’єднати всі частини розбитого об’єкта. Результат виконання функції зображено на рисунку 2.6.

Рисунок 2.6 - Контур після операції *morphology CLOSE*

Отриманий контур хибно вважати остаточним. Справа в тому, що при виконанні операції *dilate*, контур суттєво збільнився в розмірах і не відповідає розмірам реальної людини. Необхідно виконати операцію, протилежну до *dilate*. В OpenCV для таких випадків розроблена функція *erode*. Застосувавши таке саме ядро *11x11*, дана функція зменшує розміри об’єкта по його контурам. В результаті чого ми отримуємо суцільну групу пікселів, що дуже точно повторює контури об’єкта розпізнавання. Остаточний контур представлено на рисунку 2.7.

Рисунок 2.7 - Остаточний контур рухомого об’єкта

Виконання приведених вище операцій, в багатьох випадках, дозволяє отримати суцільні контури людей, що рухаються на відео. По отриманню контурів рухомих об’єктів необхідно їх попередньо проаналізувати та за деякими критеріями відсіяти непотрібні. По-перше відкидаються контури із площею менше за заданий поріг (в даному випадку 2000 $пікселів^2$) . Також, як показала практика, деякі камери можуть інколи віддавати зіпсовані кадри з полосами на всю ширину, або висоту кадру. В таких випадках отримані артефакти можуть сприйматись, як активні об’єкти. Тому необхідно також ігнорувати такі об’єкти. Решта об’єктів підлягає подальшому аналізу та обробці.

2.2 Алгоритм слідкування за об’єктами

Для того, щоб коректно оцінювати положення кожного об’єкта відносно контрольної лінії, необхідно розуміти яким об’єктам на одному кадрі відповідають об’єкти на наступному кадрі.

Кожен об’єкт на зображенні запам’ятовується. Контур кожного нового об’єкта перевіряється з контуром вже існуючих об’єктів наступним чином: якщо контури об’єкта в нового кадру пересікаються із деяким контуром вже існуючого об’єкта з минулих кадрів на площу більшу 50% від площі одного з цих двох контурів – вважаємо, що новий об’єкт є оновленим станом старого, і перезаписуємо його нове положення на зображенні.

Якщо збігів нового контуру із старими не виявлено – новий об’єкт додається до списку активних об’єктів. Об’єкти, що не оновилися під час обробки кадру мають час життя - 1.5 секунди. Якщо за цей час вони не оновилися – видаляємо такі об’єкти з пам’яті.

2.3 Алгоритм перетинання контрольної лінії

Під визначенням «контрольна лінія» мається на увазі ламана, що складається із секторів (відрізків), по перетинанню будь-якого з них в обох напрямках необхідно підраховувати кількість об’єктів, які його пересікають. «Контрольна лінія» задається у вигляді координат декількох точок. Для фіксування перетинання лінії перевіряється чи не перетнув об’єкт один із секторів лінії. Поняття “сектор” вирішено було ввести через особливості приміщень, в яких необхідно рахувати людей. На початку створення алгоритму контрольна лінія представлялася у вигляді відрізка, проте іноді було дуже незручно задавати її на вертикальному зображенні деяких приміщень. Тож було вирішено зображувати лінію, як сукупність відрізків. Далі будемо розглядати процес перетинання саме сектора (або простої лінії), адже саме він гарантує перетинання контрольної лінії як такої. Для перетинання необхідно виконання двох умов.

Перша умова: необхідна зміна положення об’єкта відносно сектора. Для цього на кожному кадрі повинно бути визначено поточне положення об’єкта по відношенню до контрольної лінії.

Для простоти розрахунків та пришвидшення роботи алгоритму перевіряється положення центра об’єкта. Проте в даному випадку потрібно враховувати таку ситуацію, коли людина знаходиться на лінії та стоїть на місці. Тоді центр об’єкта може перетинати лінію безліч разів, що негативно вплине на результат алгоритму. Для цього було вирішено ввести таке поняття, як «контрольний простір» (рис. 2.8), тобто невеликий простір по обидва боки сектора, входження в який необхідне об’єкту для зарахування пересічення лінії. Ширина такого простору задається для кожного зображення різна.

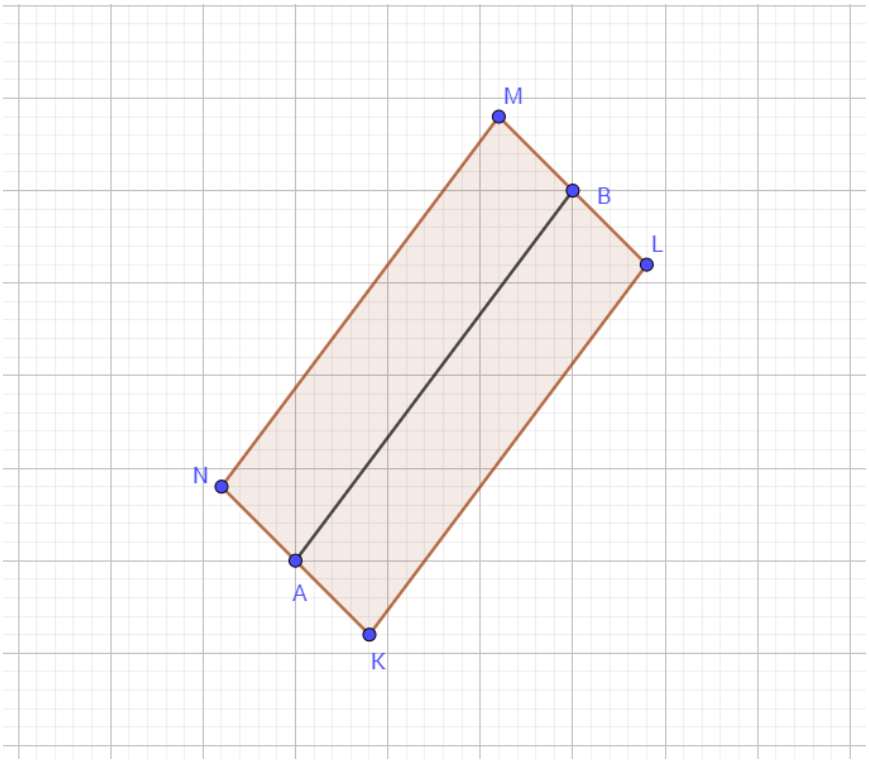

Рисунок 2.8 - Контрольний простір *KLMN* лінії *AB*

Координати точок *K, L, M, N* розраховуються за відповідними формулами, вважаючи, що *KLMN* – прямокутник, точка *A* – середина *KN*, точка *B* – середина *LM*, а довжина сторін *KN = LM* задається попередньо на етапі калібрування контрольної лінії.

Положення точки відносно лінії визначається наступним чином: Нехай існує контрольна лінія *AB*, контрольний простір, заданий точками *K, L, M, N* та центр об’єкта *C*. А також позначимо два простори, зліва та справа від лінії, як *P1* та *P2* (рис. 2.9).

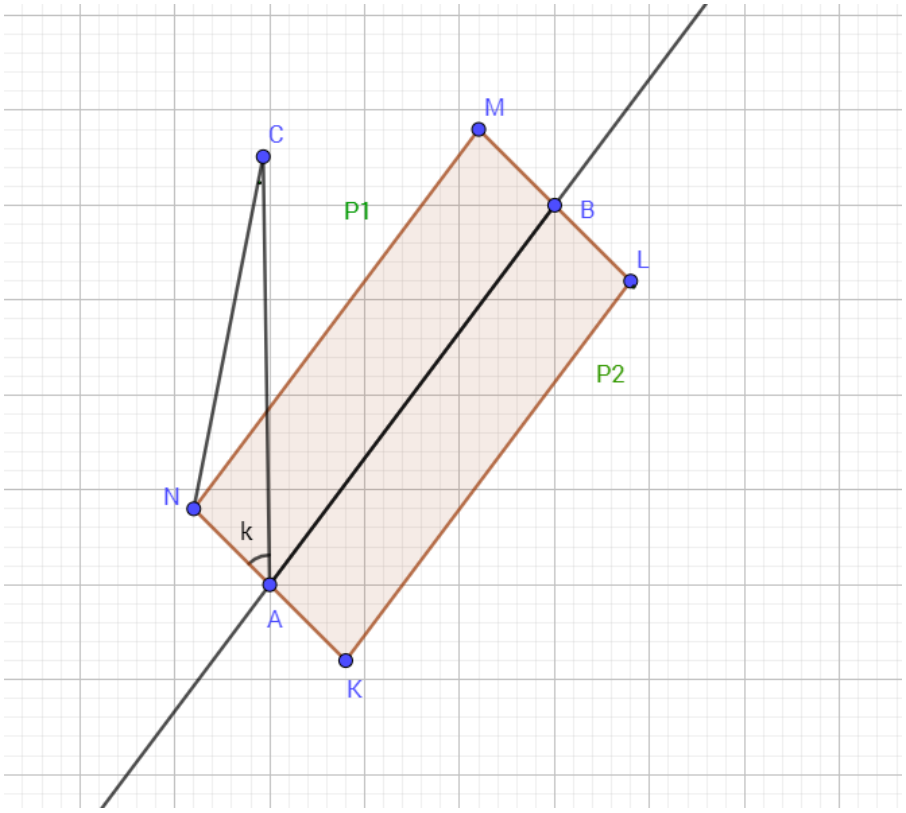

Рисунок 2.9 - Приклад положення точки *C* відносно лінії *AB*

Тоді центр об’єкта *C* знаходиться у просторі *P1*, якщо:

$$\square\square\square\ \square < \frac{\square}{2} \tag{2.1}$$

Або, центр об’єкта *C* знаходиться у просторі *P2*, якщо:

$$\square\square\square\ \square > \frac{\square}{2} \tag{2.2}$$

Друга умова: об’єкт повинен потрапити до «контрольного простору» та вийти з нього. Ця умова важлива в тих випадках, коли контрольна лінія задана не на всю ширину зображення, і нам необхідно ігнорувати зміну положення об’єкта за межами відрізку.

Зустрічаються випадки коли декілька людей, проходячи близько один до одного пересікають лінію. Такі групи людей можуть сприйматися як один великий

об’єкт. Отже якщо зафіксовано перетинання лінії потрібно проаналізувати ширину та довжину об’єкта при перетинанні.

Для зручності розрахунків габаритів об’єкта контур необхідно представити у вигляді геометричної фігури, що якомога краще повторювала б силуети контуру і займала найменший зайвий простір навколо об’єкта. Спочатку були спроби працювати із прямокутником, проте, як показали практика, він займає багато зайвого місця, особливо біля його кутів. Тоді було прийнято рішення обводити контури за допомогою еліпса. Бібліотека OpenCV надає зручну функцію для отримання еліпса по контуру - *cv2.fitEllipse*, що повертає координати центру еліпса, його велику на малі вісь та кут нахилу в системі координат.

Для знаходження ширини об’єкта необхідно знайти його діаметр, що проходить під таким самим кутом, під яким нахилена до еліпса «контрольна лінія». Спочатку знаходиться кут нахилу лінії відносно еліпса. Нехай кут нахилу лінії до осі координат α, кут нахилу еліпса до осі координат β. Тоді кут нахилу лінії до еліпса визначається як:

$$ = 180 + - \quad (2.3)$$

Ширина об’єкта, або довжина діаметра під кутом , розраховується за формулою:

$$ = ((\frac{ }{ })^2 + (\frac{ }{ })^2)^{-\frac{1}{2}}, \quad (2.4)$$

де *a, b* – малий та великий діаметри еліпса відповідно.

Довжина об’єкта розраховується як довжина діаметра еліпса, що є перпендикулярним до діаметра, розрахованого вище:

$$ = ((\frac{ (90-)}{ })^2 + (\frac{ (90-)}{ })^2)^{-\frac{1}{2}} \quad (2.5)$$

Перед роботою алгоритму необхідно вручну відкалібрувати також і приблизні ширину та довжину одиночного об’єкта (тобто об’єкта, що рівний розмірам однієї людини).

Знаючи габарити об’єкта, що пересік лінію, та приблизні габарити одиничного об’єкта, можна розрахувати скільки людей перетнули лінію.

Нехай ширина та довжина одиничного об’єкта задані значеннями *singleObjectWidth (sow)* та *singleObjectHeight (soh)*, тоді кількість людей *n* в об’єкті, що пересік лінію, розраховується наступним чином:

$$n = \frac{\text{sow}}{\text{soh}} * \frac{\text{soh}}{\text{soh}} \tag{2.6}$$

Після чого *n* округлюється до цілого числа і приймає остаточний результат.

2.4 Висновки за розділом

Для виконання поставленої задачі розроблено окремі складові алгоритму, а саме: виявлення активних об’єктів на відео з камери; слідкування за об’єктами; аналіз ситуацій із перетинання об’єктами контрольної лінії. Проблему виявлення активності вирішено завдяки застосуванню морфологічних перетворень над зображеннями та реалізовано за допомогою відповідних функцій бібліотеки OpenCV. Задачу слідкування за об’єктами, а також логіку перетинання об’єктами контрольної лінії реалізовано за допомогою простих геометричних функцій та алгоритмів. Також виявлено деякі особливості взаємодії об’єктів, що детально описані в наступному розділі.

3 ДОСЛІДЖЕННЯ РОБОТИ АЛГОРИТМУ

3.1 Дослідження впливу вибору колірної моделі на роботу алгоритму

Для порівняння впливу вибору колірної моделі на результат дії алгоритму розпізнавання було проведено наступне дослідження. Було обрано послідовність з десяти кадрів, попередньо записаних на відео з камери. На уривку відео зображена людина, що проходить по нерівномірно освітленому приміщенню, тобто на об’єкт спостереження на деяких кадрах падає світло або тінь. На кожному кадрі виділяється контур людини та знаходяться всі точки всередині контуру, кожна з яких має свій колір у відповідній колірній моделі. Далі розраховується скільки пікселів якого кольору має цей об’єкт на кожному кадрі. Такий набір даних далі вважаємо за колірну характеристику об’єкта. Після чого порівнюється колірна характеристика на одному кадрі з попередньою на скільки відсотків вони подібні. Слід зазначити, що відхилення до 10 відсотків можна вважати допустимим, адже розміри контуру можуть змінюватись в процесі знаходження об’єкта на зображенні, а також під час руху деякі частини об’єкта закриваються або навпаки з’являються

Таблиця 3.1 - Порівняння колірних моделей

№ кадру	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RGB	74%	68%	38%	59%	67%	71%	27%	53%	62%
CMYK	74%	68%	38%	59%	67%	71%	27%	53%	62%
HSV	93%	87%	73%	92%	90%	91%	69%	85%	89%
Lab	98%	96%	93%	96%	97%	92%	89%	93%	94%

Також необхідно зауважити, що перед проведенням досліду значення каналів були нереалізовані. Адже без нормалізації результати порівняння сусідніх кадрів складали близько 8% при нормальних умовах використовуючи колірну модель *RGB*. Після нормалізації діапазони кожного параметру було зменшено у 8 разів, що не сильно спотворює дані.

На кадрах 1-3 на об’єкт падає світло; кадр 4 - вихід об’єкта у рівномірно освітлений простір; 5-7 об’єкт знаходиться у рівномірному освітленні. На кадрі 8 об’єкт заходить в тінь і далі він рухається в затіненій частині приміщення.

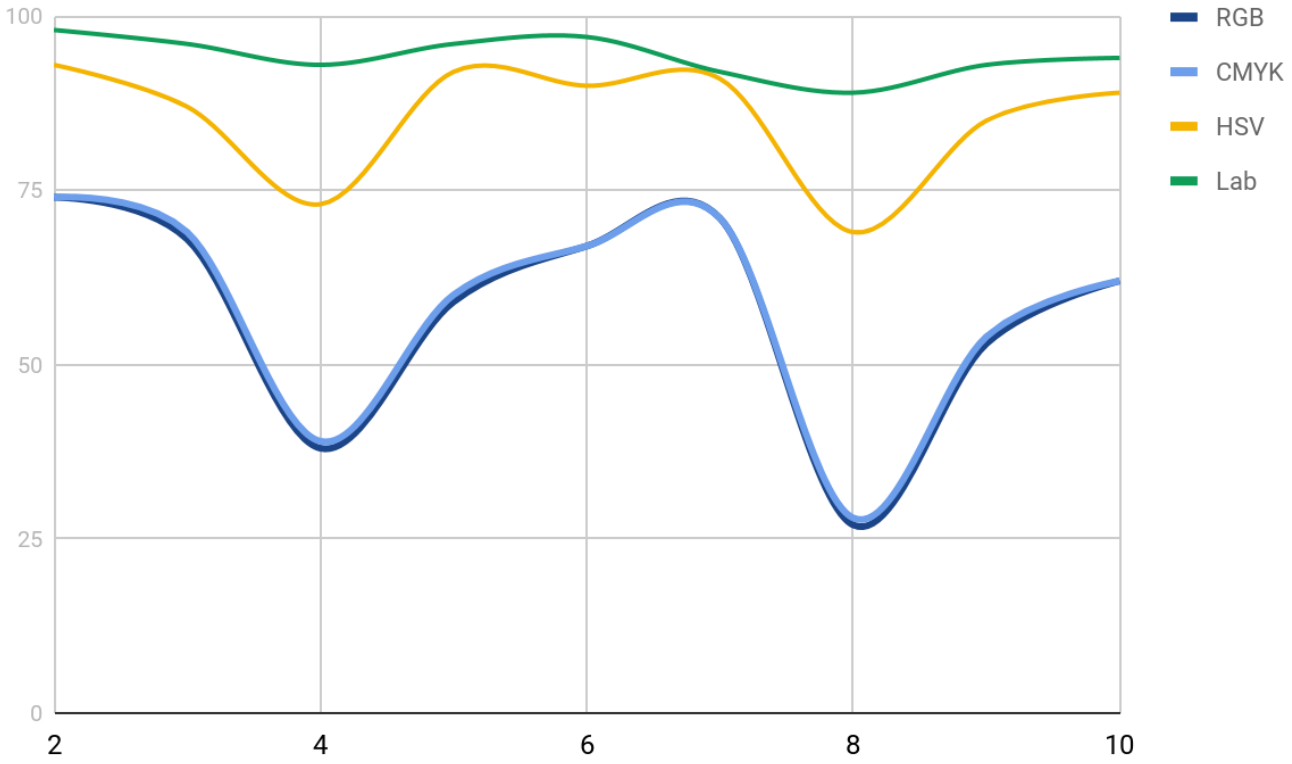

Рисунок 3.1 - Графічне зображення порівняння колірних моделей

З таблиці 3.1 можна побачити, що використовуючи модель *RGB* схожість колірних характеристик об’єкта на сусідніх кадрах 60-70%. Проте при зміні освітлення результати різко погіршуються. Таку закономірність легко помітити при зображенні порівняння за допомогою графіка на рисунку 3.1. Цю проблему вирішують моделі *HSV* та *Lab*. З їх використанням вплив світла та тіні впливає на такий значний. Найкраще себе показую саме *Lab* модель, через відокремлення значення світлості від кольору. Також слід зауважити, що використання двох останніх моделей при визначенні активності на відео значно покращує знаходження контурів рухомого об’єкта, адже поява тіні на зображенні ігнорується, що не можна

сказати про модель *RGB*, де невелика зміна відтінку буде сприйматись, як рухомий об’єкт.

Таким чином при розробці алгоритму обробки зображень з урахуванням кольору рекомендовано обирати колірну модель *Lab*, зважаючи на її переваги над іншими.

3.2 Дослідження взаємодії декількох об’єктів

Як було зазначено раніше, для аналізу інформації про пересікання об’єктом лінії використовуються дані про положення об’єкта відносно лінії, а точніше положення центра еліпса. За нормальних обставин центр зміщується більш-менш рівномірно, в наслідок чого слідкування відбувається без помилок. Але існує ряд проблем, які слід було враховувати в роботі алгоритму. Так як об’єктом є замкнена область “рухомих” пікселів, то очевидна ситуація, коли декілька осіб можуть підходити близько один до одного, утворюючи з двох груп пікселів на одному кадрі одну велику групу. Під час такого процесу розроблений алгоритм, по-перше, оновить дані про положення обох об’єктів, в результаті чого вони матимуть один і той же контур на наступних кадрах (рис. 3.2). А по-друге, центри обох об’єктів різко змінять своє положення, що може негативно сказатись на результаті. Аналогічно існує і така проблемна ситуація, коли декілька людей навпаки розходяться утворюючи декілька об’єктів. В цьому випадку центр цих новоутворених об’єктів також змінюється на недопустимо велику відстань.

Рисунок 3.2 - Об’єднання двох людей в один об’єкт

Для вирішення цих проблем була поставлена задача розробки логіки взаємодії об’єктів. Для початку необхідно було слідкувати чи не перетнув кожен із об’єктів контрольну лінії під час від'єднання або злиття з іншим об’єктом. Це необхідно, адже розроблений алгоритм, для прикладу під час розділення великого об’єкта на декілька маленьких, обере один із нових об’єктів в якості свого оновленого стану, коли решта буде вважатись як абсолютно нові об’єкти. Тож якщо дана ситуація відбулася посеред кадру, і після від'єднання новий будь-який об’єкт опинився з іншого боку контрольної лінії - втрачається цікава нам інформація, що є недопустимим для користувача.

Так само під час об’єднання декількох об’єктів в один алгоритм оновлює інформацію про контур першого, в той час як інші будуть проігноровані, хоча насправді в цей момент деякі з них могли пересікти лінію.

Іншим питанням при взаємодії є впізнавання об’єктів. Проблема полягає в наступному: два об’єкти рухаються один на зустріч іншому по різні сторони контрольної лінії. Для наочності позначимо кожну людину окремо певним кольором (рис. 3.3). Безпосередньо на лінії контури обох пересікаються і вони утворюють один великий об’єкт, як було зображено раніше на рисунку 3.2.

Рисунок 3.3 - Приклад кадру коли дві людини
рухаються назустріч один одному

Далі вони продовжують рух, кожен в своєму напрямку. Відійшовши на достатню відстань велика група пікселів розділяється на дві, і утворюються два об’єкти знову по обидві сторони від контрольної лінії (рис. 3.4).

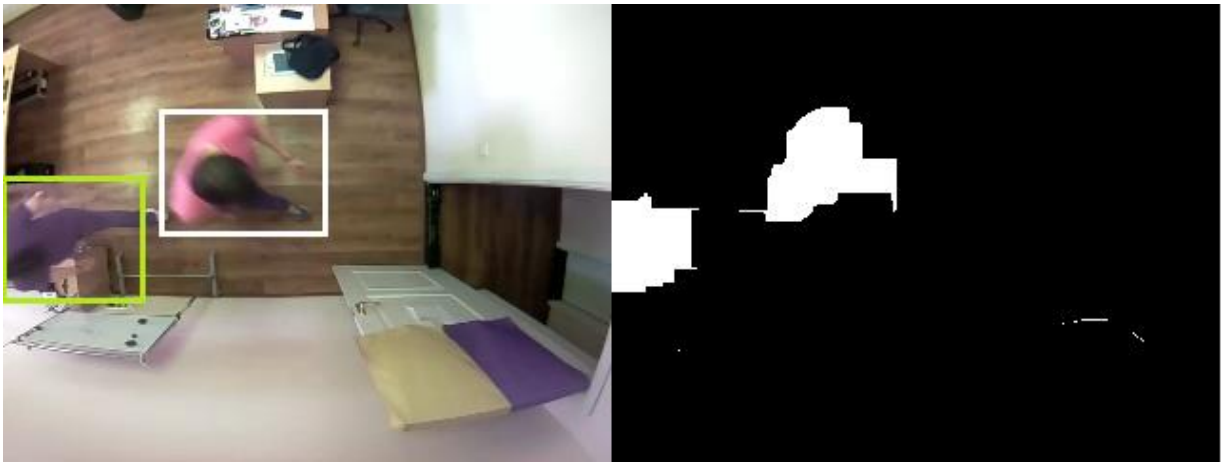

Рисунок 3.4 - Приклад кадру коли дві людини
розійшлись в різних напрямках (варіант 1)

Проте алгоритм не знає якій людини до об’єднання у великий контур відповідає яка людина на поточному кадрі, адже алгоритм бачить лише групи пікселів без додаткової інформації. Для прикладу в іншій ситуації (рис. 3.5) алгоритм може неправильно зрозуміти де саме яка людина.

Рисунок 3.5 - Приклад кадру коли дві людини
розійшлись в різних напрямках (варіант 2)

Саме через це необхідно аналізувати колірні характеристики кожного об’єкта, та брати їх до уваги під час взаємодії між ними.

3.2.1 Аналіз кольору об’єкта

В деякі моменти роботи алгоритму необхідно аналізувати колір об’єктів. У кожної людини на зображенні є певні особливості кольору на одязі. Було б дуже зручно, якщо алгоритм міг би помітити певні відмінності у кольорі об’єктів, та врахувати ці дані під час взаємодії. Тож необхідно проаналізувати всі пікселі що входять в контур об’єкта на кольоровому зображенні.

Кожен об’єкт має координати свого контуру. Для знаходження всіх точок всередині нього необхідно виконати наступний порядок дій:

1. Отримати зменшену копію поточного кольорового кадру;
2. Виконати маштабування контуру об’єкта відповідно до розмірів зменшеного зображення;
3. Створити чорне зображення аналогічного розміру із робочим кадром;
4. Намалювати на чорному зображенні контур об’єкта та залити його будь-яким, відмінним від чорного кольором;

5. Накласти ці два зображення одне на одне (рис 3.6).

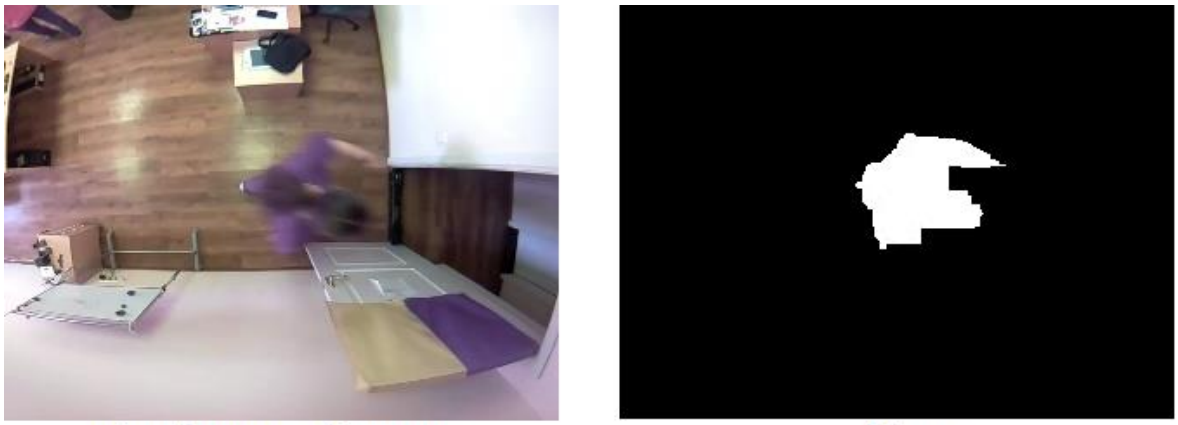

Рисунок 3.6 - Кадр із рухомим обєктом та маска цього кадру

Під час накладання необхідно порівнювати відповідні пікселі на обох зображеннях. Якщо піксель на кадрі-масці є чорним - це означає, що даний піксель не входить у контур. В іншому випадку вважаємо, що піксель нам цікавий і зберігаємо інформацію про його колір у масив.

Після знаходження всіх точок всередині контуру необхідно нормалізувати значення кольорів, так як діапазон від 0 до 255 (в такому діапазоні представляє складові кольорів OpenCV) є завеликим і кількість різноманітних комбінацій, а отже і схожих за кольором точок буде дуже велика.

Для нормалізації попередньо задаються нові межі діапазону значень. Шляхом проведених експериментів було обрано межі від 0 до 7 (можна обирати межі на свій розсуд в залежності від якості зображення). Після чого розраховується коефіцієнт нормалізації $□_□$ (формула 3.1), і визначається спрощене значення кольору точки. Кожна із колірних складових $□_{□□□□}$ визначається як ціла частина від ділення оригінального значення $□_{□□□□}$ на коефіцієнт нормалізації (формула 3.2).

$$□_□ = \frac{256}{8} = 32 \quad (3.1)$$

$$□_{□□□□} = [\frac{□_{□□□□}}{□_□}] \quad (3.2)$$

Отримавши масив спрощених значень кольорів необхідно підрахувати кількість однакових за кольором точок, що далі називаємо колірною характеристикою об’єкта. Для зручної роботи з колірною характеристикою в подальшому її зручно представити у вигляді набору даних типу ключ-значення, де ключ - певний колір із нормалізованими складовими, а значення - кількість точок певного кольору всередині контуру об’єкта. При обраному діапазоні від 0 до 7 нормалізованих значень, враховуючи, що значення кольору складається з трьох параметрів, кількість комбінацій різноманітних кольорів становить 512. Приклад колірної характеристики приведено в таблиці 3.2.

Таблиця 3.2 - Приклад колірної характеристики об’єкта

Значення кольору (нормалізоване)	Кількість пікселів
[0, 0, 0]	12
[0, 0, 1]	5
[0, 0, 2]	0
...	
[7, 7, 5]	14
[7, 7, 6]	0
[7, 7, 7]	22

3.2.2 Об’єднання об’єктів

Важливо зупинитись на одній з проблем взаємодії об’єктів, а саме об’єднання декількох об’єктів розпізнавання в один. Під час того, як дві групи пікселів зійшлись в одну створюється новий об’єкт, який повинен зберігати в собі інформації про ті об’єкти що об’єднались. При цьому необхідно додатково визначити декілька важливих параметрів цих об’єктів. По-перше необхідно оприділити положення

кожного з об’єктів відносно контрольної лінії. Це важливо в тих випадках, коли після злиття центр результуючого великий об’єкта опинився з іншого боку лінії (рис. 3.7).

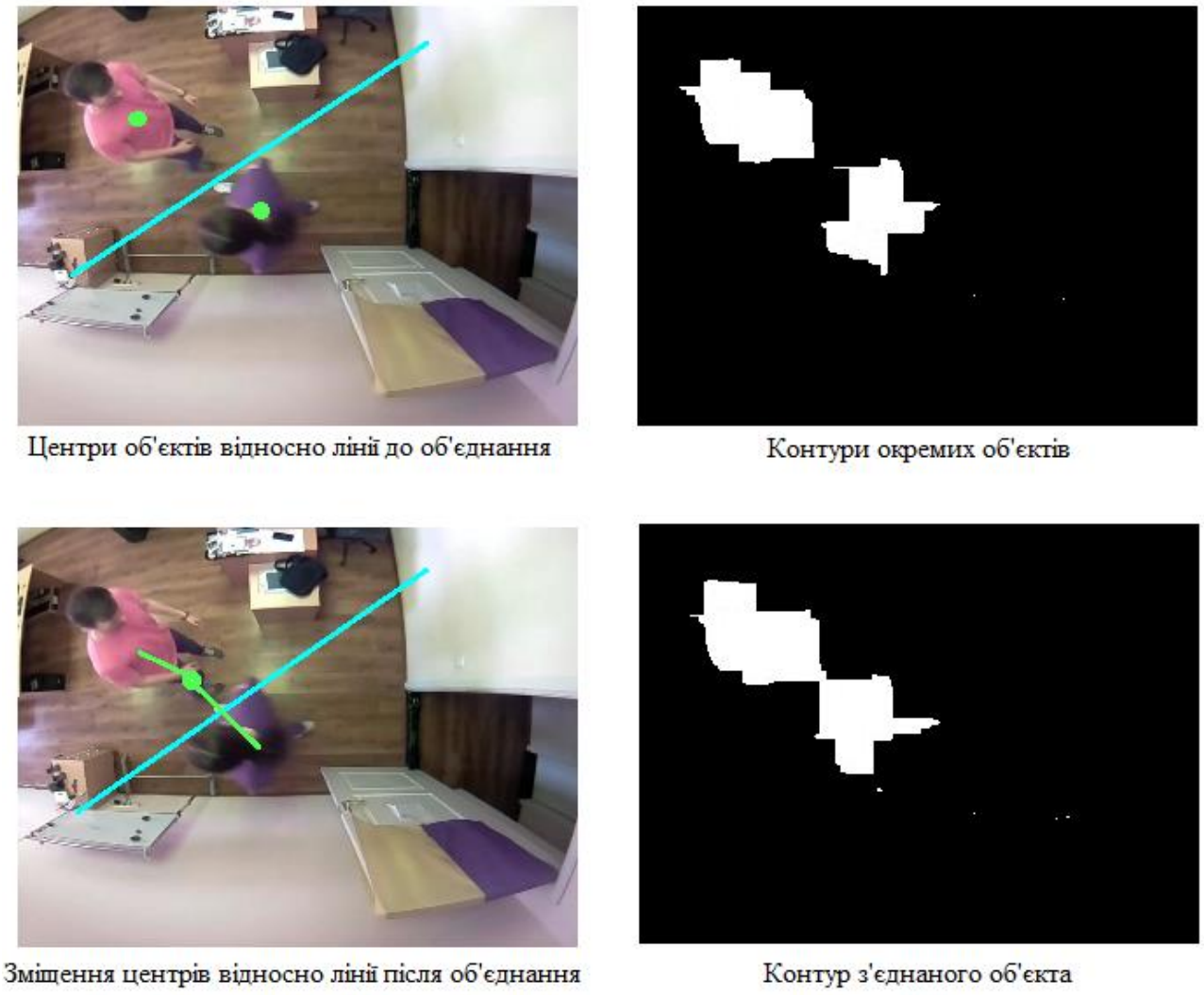

Рисунок 3.7 - Обєднання двох обєктів із пересіканням лінії

Визначення положення відносно лінії відбувається аналогічним чином як було описано в розділі 2.3. Обирається перший сектор складної лінії (або єдиний сектор, якщо ця лінія проста), та знаходиться положення об’єкта відносно нього. Після об’єднання об’єктів перевіряється їх поточне положення, і якщо воно відмінне від попереднього, тобто об’єкт пересік лінію, посилається сигнал про перетинання.

3.2.3 Розділення об’єктів

Утворення мульти-об’єкта, тобто об’єкта, що зберігає інформацію про декілька попередньо поєднаних, допомагає правильно співставити людей, що розділились. Процес визначення того, що об’єкт розділився на декілька менших виглядає наступним чином. Алгоритм аналізує всі контури на новому кадрі, співставляє їх з останніми контурами вже існуючих об’єктів розпізнавання, та намагається оновити останні. В результаті якщо деякому об’єкту підходить декілька контурів, і не можна визначити яких з них є його новим положенням необхідно виконати операції розділення мульти-об’єкта. Якщо ж розділення відбувається зі звичайним об’єктом, то його оновленим станом обирається будь-який з цих нових контурів, а інші створюються як нові об’єкти. Такий виняток відбувається коли у межі слідкування камери увійшло декілька людей близько один до одного, і неможливо дізнатись їх окремі колірні характеристики.

Розділення мульти-об’єкта відбувається за наступним алгоритмом:

1. В якості зразкової колірної характеристики обирається знайдений на новому кадрі контур;
2. Зразкова колірна характеристика порівнюється із кожною характеристикою збережених у мульти-об’єкті, та визначається коефіцієнт подібності двох характеристик за алгоритмом, описаним далі;
3. Той об’єкт, колірна характеристика якого найбільш подібна до зразкової, тобто коефіцієнт подібності є найбільшим, видаляється із мульти-об’єкта, і оновлює своє положення на положення зразкового контуру із усіма його параметрами;
4. Далі обирається наступний контур і аналогічним чином визначається найбільш підходящий йому об’єкт.

Коефіцієнт подібності двох колірних характеристик визначається наступним чином. Як зазначалось раніше, колірна характеристика виглядає як послідовність наборів ключ-значення. Більшість значень зазвичай дорівнюють нулю, адже об’єкт

на зображенні складається з невеликого набору різноманітних кольорів. Спочатку для кожної з двох характеристик необхідно обрати лише ті пари, значення яких відрізняється від 0, тобто ті кольори, які присутні в зображенні об’єкта. Та визначити середнє значення *averageColorValue* таких кольорів. Ініціалізуємо коефіцієнт подібності *compareKoef* значенням 0. Далі порівнюємо значення колірних характеристик однакового кольору, ті відповідно до таблиці 3.3 змінюємо *compareKoef* з кожним таким порівнянням (*i*-те значення першої колірної характеристики позначимо як *v1*, а *i*-те значення другої як *v2*).

Таблиця 3.3 - порівняння значень елементів колірних характеристик

Умова порівняння	Зміна *compareKoef*
v1 = 0 and *v2* = 0	*compareKoef* += 0.5
(v1 !=0 and *v2* != 0) and (*v1* = *v2)*	*compareKoef* += 1
(*v1* or *v2* = 0) and (*v1* != *v2*)	*compareKoef* -= 0.5
v1 !=0 and *v2* != 0	*compareKoef* += *delta*

delta розраховується за формулою 3.5:

$$□ = \frac{□1^2}{□□□□□□□□□□□□□□□□□□_1} \quad (3.3)$$

$$□ = \frac{□2^2}{□□□□□□□□□□□□□□□□□□_2} \quad (3.4)$$

$$□□□□□ = \left(\frac{□□□(□,□)}{□□□(□,□)}\right)^2 \quad (3.5)$$

Необхідно пояснити чому обрані саме такі зміни коефіцієнта порівняння. При порівнянні двох колірних характеристик важливо помічати унікальність кольорів двох об’єктів. Тобто ситуації, коли певний колір відсутній або присутній в обох об’єктах одночасно, необхідно вважати корисними. І навпаки в тих випадках, коли колір є унікальним для одного з об’єктів, важливо зазначити, що така розбіжність є неодмінно суттєвою. Для прикладу можна уявити, що на одному зображенні людина мала червону стрічку на одязі, а на іншому - ні.

3.3 Дослідження параметрів алгоритму

На якість роботи алгоритму можуть впливати наступні фактори:

- Освітлення: рівномірне; наявність об’єктів-джерел освітлення (вікна, ліхтарі) з деякого боку від поля зору камери. Впливає на виділення активності на кадрі.
- Ширина проходу до приміщення. Впливає на об’єднання людей в об’єкти.
- Висота розташування камери. Впливає на розміри одиничного об’єкта.

Було проведено дослідження роботи алгоритму в залежності від значень параметрів на прикладі трьох різних приміщень із різними характеристиками.

Нижче приведені важливі характеристики приміщень:

- Приміщення А: рівномірне освітлення, широкий прохід.
- Приміщення Б: наявність декількох вікон із сторони входу, широкий прохід.
- Приміщення В: нерівномірне освітлення, вузький прохід.

Параметри, що підлягають калібруванню наступні:

- Поріг чутливості: *threshold*
- Ширина контрольного простору: *dist*
- Ширина одиничного об’єкта: *sow*

Було записано відео тривалістю 24 години з кожного приміщення, отримані наступні результати роботи алгоритму в залежності від каліброваних параметрів.

Таблиця 3.4 - Аналіз роботи алгоритму на прикладі приміщення А

№	threshold	dist	sow	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	15	10	50	581/453		73/92
2	25	10	50	454/434		94/96
3	40	10	50	380/351		89/84
4	30	15	50	438/427		97/98
5	30	15	60	428/422		99/99

Таблиця 3.5 - Аналіз роботи алгоритму на прикладі приміщення Б

№	threshold	dist	sow	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	30	15	60	603/598		91/92
2	35	15	60	588/585		94/94
3	40	15	60	573/569		96/96
4	40	10	60	571/569		96/97
5	40	10	55	564/561		98/98

Таблиця 3.6 - Аналіз роботи алгоритму на прикладі приміщення В

№	threshold	dist	sow	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	40	10	55	441/427		88/90
2	38	10	55	434/422		89/91
3	38	15	60	434/421		89/91
4	38	15	70	354/356		91/92
5	38	15	65	411/404		94/95

Також на якість роботи алгоритму впливають деякі фактори, які на даному етапі неможливо передбачити. До таких ситуацій можна віднести віддзеркалення об’єктів на відповідних поверхнях, реєстрування інших рухомих об’єктів, що не є людьми, таких як візки, двері, тощо. В таких випадках похибка може перевищувати 10%. Для запобігання таких ситуацій необхідна комбінація із іншим методом, наприклад додавання до алгоритму даних датчика температури.

3.4 Порівняння з іншими рішеннями

Далі приведено порівняння якості роботи розробленого алгоритму з деякими з існуючих рішень. Було обрано такі рішення, в основі яких лежить лише один з методів аналізу зображень, так само як представлений вище алгоритм.

Такими рішеннями є Automated-Attendance-System та Overhead Camera People Counter.

Вхідні дані для порівняння роботи рішень обрані ті самі, що і в розділі «Дослідження роботи алгоритму», а саме 3 відео тривалістю 24 години у різних за характеристиками приміщеннях. Проте для Automated-Attendance-System відео було знято на окрему камеру, розташовану під кутом до входу, щоб можливо було зафіксувати обличчя людей, що входять у приміщення.

Всі алгоритми буди попередньо налаштовані у відповідності до приміщення, задавши найбільш підходящі параметри (де можливо).

Таблиця 3.7- Порівняння роботи рішень на прикладі приміщення А

№	**Рішення**	**Результат (In/Out)**	**Реальні дані (In/Out)**	**Точність, %**
1	Власний алгоритм	428/422		99/99
2	OCPC	419/415		98/99
3	AAS	388/388		91/93

Таблиця 3.8 -Порівняння роботи рішень на прикладі приміщення Б

№	Рішення	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	Власний алгоритм	564/561		98/98
2	OCPC	532/529		97/96
3	AAS	501/501		91/91

Таблиця 3.9 - Порівняння роботи рішень на прикладі приміщення В

№	Рішення	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	Власний алгоритм	411/404		94/95
2	OCPC	350/356		90/92
3	AAS	247/247		64/64

Отримані результати показують, що аналіз відвідуваності методом розпізнавання облич є найменш точним. Це пов’язано із освітленням та якістю зображення. Як бачимо у третьому випадку, при поганому освітленні точність алгоритму різко зменшилась.

Якщо порівнювати розроблений алгоритм із схожим аналогом (в даному випадку Overhead Camera People Counter). То результати приблизно однакові, проте слід врахувати, що завдяки більш детальній логіці пересікання лінії, врахування розмірів об’єкта, а також механізму слідкування власний алгоритм показує дедалі кращі результати при погіршенні умов, а саме освітлення та розмірів приміщення.

3.5 Висновки за розділом

Важливим фактором впливу на результати роботи алгоритмів для підрахунку людей з використанням методу обробки відео з камери відеоспостереження є вплив світла, а точніше постійна зміна освітлення в приміщенні де відбувається запис відео. Отже необхідно було проаналізувати вплив колірної моделі на вирішення вищезгаданої проблеми. В першому підрозділі розглянуто такі моделі як RGB, CMYK, HSV та Lab. Враховуючи їх переваги та недоліки обрано модель Lab.

Так як камера відеоспостереження може знаходитись в приміщеннях різної форми, з різними висотами стелі та іншими відмінностями, було вирішено виділити окремі параметри, що можуть бути налаштовані окремо для різних випадків застосування. В даному розділі було проведено експериментальне дослідження і виявлено найкращі значення кожного з параметрів.

В завершенні проведено порівняльний аналіз результатів роботи розробленого алгоритму з роботою існуючих рішень. Проведено дослідження роботи кожного в різних умовах освітлення та в приміщеннях з різними особливостями та показано переваги власного алгоритму над іншими.

4 РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

4.1 Опис ідеї проекту

Метою розділу є розроблення програми для підрахунку людей, використовуючи розроблений в роботі алгоритм. Розглянемо зміст ідеї, можливі напрямки застосування, основні переваги, які зможе отримати користувач представлено у таблиці 4.1

Таблиця 4.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	1. Виявлення активності	Інформація про рух в приміщенні
	2.Підрахунок рухомих об’єктів.	Статистика про рух об’єктів в певних напрямках

Алгоритм, що використовується в програмі має велику точність підрахунків, особливо в умовах складного освітлення та форми приміщення.

Далі проведено аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників). Визначено перелік техніко-економічних властивостей та характеристик ідеї.

Серед конкурентів було вирішено обрати продукти різних за можливостями виробників, а саме V-Count 3D+ Alfa та Automated-Attendance-System (AAS).

Проведено порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні).

Таблиця 4.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	V-Count	AAS			
1.	Технічна	-швидкодія -точність аналізу даних -простота у використанні	-Висока точність -великі затрати ресурсів	-Недостатня точність		-точність аналізу даних	-швидкодія -простота використання -малі витарти ресурсів
2.	Економічна	Відносно невеликі витрати на обладнання	Необхідність витрат на обладнання	Невеликі витраті на обладнання			Невелика ціна на ринку
3.	Надійність	Не обмежений термін дії	Не обмежений термін дії	Не обмежений термін дії			Довговічність не менша ніж у конкурентів

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару).

Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (табл. 4.3):

Таблиця 4.3 - Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
	Алгоритм підрахунку людей	Вже розроблено	Технологія доступна
		Мова програмування python, бібліотека OpenCV	Технологія доступна
		Мова програмування python, бібліотека pygame	Технологія доступна
		Окремий стаціонарний комп’ютер, опреційна система Windows	Технологія доступна
		Raspberry Pi, опреційна система Raspbian	Технологія доступна
Обрані технології реалізації ідеї проекту: В якості платформи для встановлення та запуску програми було обрано Raspberry Pi через її компактність та коштовність. Враховуючи оптимізованість алгоритму, потужності Raspberry Pi вистачає для запуску програми. Для реалізації користувацького інтерфейсу було обрано бібліотеку pygame, адже при використанні програм віддаленого керування (TeamViewer, Remote Desktop) бібіолтека OpenCV має деякі проблеми з відображенням графічного інтерфейсу.			

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану

ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

1) Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 4.4).

Таблиця 4.4 -Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	7
2	Загальний обсяг продаж, грн/ум.од	15000 за рік
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Сертифікація
6	Середня норма рентабельності в галузі (або по ринку), %	50

Ринок є привабливий для входу та потребує новітніх алгоритмів для вирішення задач.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиці 4.5).

Таблиця 4.5-Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Підрахунок людей,	Потенційна	-потреба в продуті	-отримання результатів

що входять або виходять з приміщення	група клієнтів є підприємці, бізнесмени	оптимізації -застосування в приміщеннях різної форми -доступна ціна ціна -простота встановлення	в реальному часі -просте встановлення продукту -можливість задання складної контрольної лінії.

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 4.6 та 4.7). Фактори в таблиці подаються в порядку зменшення значущості.

Таблиця 4.6 - Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	З’являються конкуренти	На ринку утворюються конкуренти з якіснішими продуктами	-Реклама -Удосконалення продукту
2	Збільшення ціни на розроблення	Збільшення ціни на нові моделі Raspberry Pi	Зміна цінової політики

Таблиця 4.7 - Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Попит на товар	Попит на програми для підрахунку людей	-Пришвидшення розробки -збільшення обсягів виробнитва
2	Зменшення ціни на розроблення	Збільшення ціни на нові моделі Raspberry Pi	-Пришвидшення розробки -Вихід на нові ринки збуту

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиця 4.8).

Таблиця 4.8 -Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.Вказати тип конкуренції - олігополія	Домінує мала кількість фірм	-Пришвидшення розроблення унікального продукту -Впровадження реклами
2. За рівнем конкурентної боротьби національний	Боротьба ведеться на національному ринку	Вихід на всесвітній ринок
3. За галузевою ознакою - внутрішньогалузева	Боротьба між товаритсвами галузі обробки зображення	Вдосконалення алгоритму
4. Конкуренція за видами товарів: - товарно-видова	Різновиди однієї категорії товару, які здатні задовольнити конкретне бажання покупця	Відсутня
5. За характером конкурентних переваг - не цінова	Ключовим фактором конкурентності є точність аналізу даних	Вдосконалення точності агоритму
6. За інтенсивністю - не марочна	Не прив’язаний до певної марки, може використовувати будь-де	Співпраця з різими марками

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 4.9).

Таблиця 4.9-Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Наведені в таблиці 4.2	потенційні конкуренти можуть з’явитися	Raspberry Pi Foundation	Клієнти потребують більшу точність за	Будь-які прямі конкуренти

		досягнувши високої точності розробленого алгоритму		меншу ціну	
Висновки:	Невелика кількість продуктів з великою точність аналізу даних	Є можливість виходу конкурентів на ринок	Постачальник забезпечую лише платфому для виконання алгритму.	Не диктують умови роботи на ринку	необхідно постійно вивчати наукову сферу даної галузі та покращувати результати алгоритму

Робота на ринку можлива, попри наявність конкурентів, через досягнуту високу точність алгоритму при невеликій собівартості реалізації продукту. Сильними сторонами алгоритму є його швидкість роботи, собівартість та умови виконання.

На основі аналізу конкуренції, проведеного в таблиця 4.9, а також із урахуванням характеристик ідеї проекту (табл. 4.2), вимог споживачів до товару (таблиці 4.5) та факторів маркетингового середовища (таблиці № 4.6-4.7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 4.10.

Таблиця 4.10 -Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Точність	Алгоритм показує кращі результати точності, на відміну від аналогів
2	Собівартість	Використання невеликої кіськості обчислювальних ресурсів дозволяє використовувати дешевші технології
3	Оптимізованість	Алгоритм поєднує в собі високу точність отримання результату в реальному часі при використанні невисоких обчислювальних потужностей

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11). (С.П. – стартап проект, К.1 – Конкурент 1, К.2 – Конкурент 2)

Таблиця 8.11-Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з … (Конкурент 1,2,3)						
			–3	–2	–1	0	+1	+2	+3
1	Точність	18					К.2	С.П	К.1
2	Собівартість	20		К.1					К.2 С.П.
3	Оптимізованість	20				К.2		К.1	С.П.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.10).

Таблиця 4.12 -SWOT- аналіз стартап-проекту

Сильні сторони: собівартість, точність	Слабкі сторони: певні виняткові ситуації в яких зменшується точність аналізу.
Можливості: попит на товар, зменшення коштів на розроблення	Загрози: збільшення ціни на розроблення, з’являються конкуренти

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок (див. табл. 4.9, аналіз потенційних конкурентів).

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 4.13).

Таблиця 4.13 -Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Потреба в оптимізації	Висока	До одного року
2	Запуск реклами	висока	До одного місяця
3	Комбінація з іншими методами аналізу зображення	середня	Необмежено

Після аналізу зазначити обрану альтернативу.

Першою альтернативою є запуск реклами через простоту та швидкість даного комплексу заходів.

4.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 4.14).

Таблиця 4.14 -Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Організація, що потребує збір статистики про відвідування певних закладів	Споживачі готові сприйняти продукт.	Попит є в цільовому сегменті	Конкуренція існує, все залежить від витрат клієнта на продукт	Продукт простий у впровадження його на ринок

Які цільові групи обрано: торгівля, безпека, розважальні комплекси

За результатами аналізу потенційних груп споживачів (сегментів) обираються цільові групи, для яких пропонується товар, та визначається стратегія охоплення ринку:

- якщо компанія зосереджується на одному сегменті – вона обирає стратегію концентрованого маркетингу;
- якщо працює із кількома сегментами, розробляючи для них окремо програми ринкового впливу – вона використовує стратегію диференційованого маркетингу;
- якщо компанія працює із всім ринком, пропонуючи стандартизовану програму (включно із характеристиками товару/послуги) – вона використовує масовий маркетинг.

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (табл. 4.15).

Таблиця 4.15 -Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Стратегія виклику лідера	Стратегія диференційного маркетингу	Індивідуальний підхід до клієнта; краща якість відповдно до місця реалізації	Стратегія диференціації передбачає надання товару важливих з точки зору споживача відмітних властивостей, які роблять товар відмінним від товарів можливих конкурентів.

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 4.16).

Таблиця 4.16 -Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
Ні	Шукати нових та забирати споживачів у конкурентів	Необхідна реалізація важливих для клієнта функцій, отже копіювання деяких властивостей існуючих продуктів обов’язкова	Стратегія заняття конкурентної ніші

На основі вимог споживачів з обраних сегментів до постачальника (стартап-компанії) та до продукту (див. табл. 4.14), а також в залежності від обраної базової стратегії розвитку (табл. 4.15) та стратегії конкурентної поведінки (табл. 4.16) розробляється стратегія позиціонування (табл. 4.17). що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17 -Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)

	умовах			

Результатом виконання підрозділу є узгоджена система рішень щодо ринкової поведінки стартап-компанії, яка визначає напрями роботи стартап-компанії на ринку.

4.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у таблиці 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18 -Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Точність аналізу	Точний підрахунок людей	У конкурентів є такий функціонал, але він не є настільки якісним
2	Швидкість	Витрача часу на обчислення мінімальна	Робота в реальному часі
3	Ціна	Мінімальні грошові витрати	Конкуренти програють у ціні продукту

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 4.19).

Таблиця 4.19 -Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
І. Товар за задумом	Алгоритм працює а режимі реального часу, має високу точність та можливість калібрування за допомогою користувацького інтерфейсу

II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Калібрування параметрів алгоритму 2. Налаштування контрольної лінії 3. Запуск алгоритму в режимі налаштування або в режимі фонового процесу		
	Якість: Документація із вказаними рекомендованими налаштуваннями для різних ситауцій		
	Raspberyy Pi зі встановленим програмним забезпеченням. Кабелі для підключення		
	Марка: Quadrox Inc. “QCounter”		
	Програмне забезпечення		
	Raspberry Pi		
Закритий код програми, шифрування			

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20 -Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
20 тис. грн	25 тис. грн	450 тис. грн	5 тис. грн – 8 тис. грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.21):

- проводити збут власними силами або залучати сторонніх посередників (власна або залучена система збуту);
- вибір та обґрунтування оптимальної глибини каналу збуту;
- вибір та обґрунтування виду посередників.

Таблиця 4.21 -Формування системи збуту

Специфіка	Функції збуту, які має	Глибина	Оптимальна

закупівельної поведінки цільових клієнтів	виконувати постачальник товару	каналу збуту	система збуту
Знаходження на сайті продукту, оплата продукту, доставка продукту на почтове відділення або адресу замовника	Зберігання, доставка на товару через компанії доставки	Виробни - споживач	Web-сайт

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.22).

Таблиця 4.22- Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Ретельно обирають товар, багато порівнюють	Веб-сайт, телефон	Підтримка, індивідуальний підхід, постійне оновлення	Донесення переваг до клієнтів	Точність статистики про відвідування

Результатом пункту 5 є ринкова (маркетингова) програма, що включає в себе концепції товару, збуту, просування та попередній аналіз можливостей ціноутворення, спирається на цінності та потреби потенційних клієнтів, конкурентні переваги ідеї, стан та динаміку ринкового середовища, в межах якого буде впроваджено проект, та відповідну обрану альтернативу ринкової поведінки.

Висновки: Є можливість ринкової комерціалізації проекту. Попит на товар присутній, попри наявність продуктів-аналогів. Розроблений проект має свої переваги над конкурентами у вигляді співвідношення ціна - якість. З огляду на потенційні групи клієнтів є перспективи для входження на ринок. Отже конкурентоспроможність продукту висока. В якості базової стратегії розвитку обрана стратегія диференціювання. В якості альтернативи впровадження доцільно обрати комбінацію з іншими методами аналізу зображення та розміщення реклами

продукту. Подальше вдосконалення алгоритму необхідне через постійний розвиток технологій в даній галузі.

ВИСНОВКИ

Під час написання магістерської дисертації розглянуто питання розробки алгоритму для підрахування людей методом обробки послідовності зображень з камери відеоспостереження. Проведено огляд популярних рішень та виділено особливості кожного з них. Окрім цього розглянуто та проаналізовано найбільш популярні бібліотеки для вирішення проблем комп'ютерного бачення. Для використання у розробці алгоритму обрано найбільш потужну, а саме бібліотеку OpenCV. Так як зображення представляються за допомогою певної колірної моделі, було проведено аналіз популярних колірних моделей таких, як RGB, CMYK, HSV та Lab. Проведено порівняння моделей у відповідності до поставленої задачі, в результаті якого обрано модель Lab для подальшого використання її в процесі розробки алгоритму. Також проведено дослідження роботи алгоритму в залежності від налаштувань алгоритму та впливу зовнішніх факторів, таких як освітлення, розміри та форми приміщення. Після визначення найбільш підходящих параметрів виконано порівняння результатів роботи алгоритму із іншими аналогічними рішеннями. Результатом дослідження можна вважати розроблення вдосконаленого алгоритму для підрахунку людей, який в подальшому слід використовувати в комбінації з іншими методами аналізу даних для досягнення ідеального результату системи слідкування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Szeliski R. What is computer vision? / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 3–10.
2. Lowe D. The Computer Vision Industry [Електронний ресурс] / David Lowe – Режим доступу до ресурсу: http://www.cs.ubc.ca/~lowe/vision.html.
3. Szeliski R. Image processing / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 101–102.
4. Szeliski R. Linear filtering / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 111–118.
5. T. Moller, E. Haines // Real-Time Rendering / T. Moller, E. Haines.. – С. 124.
6. Szeliski R. Non-linear filtering / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 122–124.
7. Szeliski R. Morphology / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 127–129.
8. Szeliski R. Color transforms / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 104–105.
9. Szeliski R. Distance transforms / Richard Szeliski // Computer Vision: Algorithms and Applications / Richard Szeliski., 2010. – С. 129–131.
10. V-Count 3D+ Alfa [Електронний ресурс] – Режим доступу до ресурсу: https://v-count.com/solutions/people-counting/.
11. Automated Attendance System Using Face Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Hybrid-SyntaX/ Automated-Attendance-System-Using-Face-Recognition.
12. OpenCV [Електронний ресурс] – Режим доступу до ресурсу: https://opencv.org/.
13. People counting algorithm using an overhead video camera [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/agurz/overhead-camera-people-counter.
14. C++ Libraries for Computer Vision Research and Implementation [Електронний ресурс] – Режим доступу до ресурсу: http://vxl.sourceforge.net/#docs.

15. The LTI-Lib Introduction [Електронний ресурс] – Режим доступу до ресурсу: http://ltilib.sourceforge.net/doc/homepage/index.shtml.
16. Open Source Computer Vision Library [Електронний ресурс] – Режим доступу до ресурсу: http://opencvlibrary.SourceForge.net.
17. Булатников Е. В. СРАВНЕНИЕ БИБЛИОТЕК КОМПЬЮТЕРНОГО ЗРЕНИЯ ДЛЯ ПРИМЕНЕНИЯ В ПРИЛОЖЕНИИ, ИСПОЛЬЗУЮЩЕМ ТЕХНОЛОГИЮ РАСПОЗНАВАНИЯ ПЛОСКИХ ИЗОБРАЖЕНИЙ / Е. В. Булатников, А. А. Гоева. // Вестник Московского государственного университета печати. – 2015. – С. 88–90.
18. Лисиця В. Т. КОЛІРНІ МОДЕЛІ ТА ЗАКОНИ ПОШИРЕННЯ СВІТЛА / В. Т. Лисиця.–2012.– С. 19.
19. CMYK color model [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/CMYK_color_model.
20. Комп’ютерна графіка "Навчальний посібник". // Тернопільський національний технічний університет імені Івана Пулюя. – 2017. – С. 44.
21. ПРОЦЕДУРА КОРРЕКЦИИ ЦВЕТОВОГО БАЛАНСА ЦИФРОВЫХ ИЗОБРАЖЕНИЙ В ЦВЕТОВОМ ПРОСТРАНСТВЕ LAB. // Збірник наукових праць Харківського університету Повітряних Сил. – 2014. – №1. – С. 122–124

Додаток А

Наукова стаття

УДК 004.855.5

ДОРОГИЙ Я.Ю.,
АНТОСЄВ А. С.

АЛГОРИТМ ПІДРАХУНКУ ЛЮДЕЙ НА ОСНОВІ МЕТОДІВ КОМП'ЮТЕРНОГО БАЧЕННЯ PEOPLE COUNTING ALGORITHM ON THE BASIS OF COMPUTER VISION

В статті розглянуте питання реалізації алгоритму для підрахунку людей з використанням методів обробки моно-зображень, та дослідження результатів його роботи

In the article the question of implementation of people counting algorithm using monocular video analytics methods, and analysis of its results.

Ключові слова: лічильник, обробка зображень, комп'ютерне бачення, виявлення активності, обробка моно-зображень

Вступ

Апарат комп'ютерного бачення широко використовують для розв'язання різноманітних задач. Важливим завданням в багатьох сферах є – підрахунок людей, наприклад, в цілях безпеки або для збору статистичних даних. Тож враховуючи можливості комп'ютерного бачення, було вирішено створити алгоритм для вирішення даної задачі із використанням популярних підходів та методів обробки зображення. Для досягнення максимальної точності використовується комбінація різних методів аналізу. Тож необхідно вдосконалювати окремі методи, що в свою чергу вплине на результат їх поєднання. Мною було вирішено розробити алгоритм з використанням обробки моно-зображень (або відео з камери з однією лінзою).

Аналіз існуючих рішень

Існує досить велика кількість програм, завдання яких – підрахунок людей, що входять, або виходять з приміщення. Для виконання цієї задачі використовуються різні методи, які в свою чергу мають свої переваги та недоліки. Зазвичай кінцевий продукт представляє собою поєднання різних методів аналізу даних, таких як: обробка стерео- або моно-зображення, використання інфрачервоного датчику, сканування Wi-Fi або Bluetooth сигналів. Гарним прикладом такого рішення є V-Count 3D+ Alfa [1]. У ньому використовують синхронну обробку зображення з двох камер, а також аналіз Wi-Fi сигналів.

Окрім вищезгаданих складних систем слід виділити такі, що використовують лише один метод для підрахунку відвідуваності. Наприклад Automated-Attendance-System (AAS)[2]. В основі логіки для аналізу відвідуваності лежить метод розпізнавання облич. Для виконання функцій обробки зображення в даному продукті використовується бібліотека Emgu CV [3], яка є .Net обгорткою навколо бібліотеки Open CV [4]. Розпізнавання облич людей виконується за допомогою каскадів haarcscade_eye та haarcascade_frontalFace_default.

Іншим прикладом програми, що використовує обробку моно-зображення для аналізу є OCPC (Overhead Camera People Counter) [5]. Для обробки зображень в даному рішенні також використовується функціонал бібліотеки Open CV. Для правильного функціонування алгоритму передбачається розміщення камери на стелі приміщення та задання координат лінії, по пересіченні якої відбувається реакція.

Проте жодне з існуючих рішень не надає стовідсоткової точності. Покращення результату може бути досягнуте шляхом вдосконалення окремих методів. До того ж при наявності високоточних методів, буде спрощена їх комбінація, що призведе до зменшення собівартості таких рішень.

Мета роботи

Метою роботи є розробка власного алгоритму для підрахунку людей, використовуючи методи обробки моно-зображень. Вдосконалений метод обробки моно-зображення в подальшому можна буде використовувати в аналізі стерео-зображень.

Виділення основних задач алгоритму

Для досягнення мети алгоритму необхідно виділити основні його задачі:

- Виявлення активних об'єктів на відео з камери,
- Слідкування за об'єктами
- Аналіз ситуацій із пересічення об'єктами контрольної лінії.

Для вирішення деяких підзадач алгоритму логічним було б використання вже існуючих інструментів у галузі комп'ютерного бачення. Можна виділити наступні популярні бібліотеки, функціонал яких призначений для розв'язання таких задач:

- VXL (Visual-something-Libraries) [6] – це колекція бібліотек, написаних мовою програмування C++, та призначена для виконання задач о області комп'ютерного бачення. Сюди входять бібліотеки для обробки зображень, математичних алгоритмів, обробки відео, класифікації, моделювання ймовірностей та багато іншого.
- LTI – бібліотека, також написана мовою програмування C++, що пропонує достатньо

великих набір інструментів та алгоритмів для обробки зображень [7]. Вона є кросплатформеною та моє досить зручне у використанні API.

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

Було вирішено використовувати бібліотеку OpenCV з деяких причин. По-перше вона показує найкращий результат по швидкодії в порівнянні з іншими. По-друге, ця бібліотека має надзвичайно просте та різноманітне API, що дозволяє вирішувати безліч задач у різний спосіб, з використанням тих чи інший популярних алгоритмів.

Алгоритм виявлення активності

Для виявлення активних об’єктів на відео необхідно виконати ряд операцій. В алгоритмі використовується порівняння поточного та попереднього зображення, та подальша обробка результату порівняння (рис. 1). По-перше необхідно завжди пам’ятати попередній кадр із послідовності кадрів відео потоку. Кожне зображення необхідно конвертувати із колірної моделі RGB у чорно-біле зображення (Grayscale). Також для збільшення швидкості роботи алгоритму рекомендовано зменшити зображення. Мною було використаний розмір 320 на 240 пікселів.

Далі до зображення у вигляді вектору розмірності 76800 було використано накладання шуму для згладжування контурів об’єктів за допомогою функції OpenCV cv2.GaussianBlur [8]. Після чого необхідно виконати операцію віднімання векторів зображень для отримання різниці між кадрами – тобто масиву пікселів, що змінилися (рис 2.).

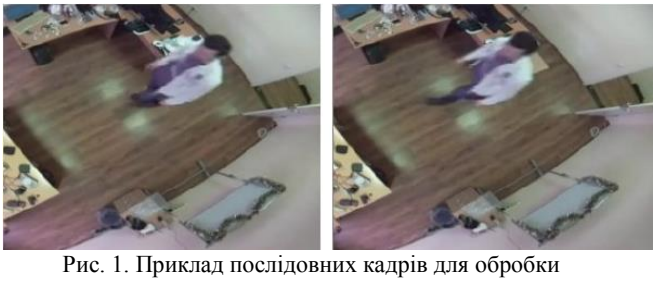

Рис. 1. Приклад послідовних кадрів для обробки

Рис. 2. Різниця сусідніх кадрів

З отриманих даних ми повинні забрати ту активність на зображенні, якою можуть бути сторонні зміни, такі як незначна зміна освітлення. Так як при зміні яскравості картинки відповідні пікселі змінюються на невелике значення (рис. 3) – необхідно ігнорувати такі зміни. Для цього ми встановлюємо поріг, який є унікальним для різних приміщень, де проходить зйомка відео, та змінюємо значення елементів масиву пікселів наступним чином: якщо значення елемента масиву менше заданого порогу (тобто колір пікселя наближених до чорного) – задаємо його значення 0 (прибираємо піксель), в іншому випадку – задаємо значення 255. Таким чином ми позбуваємось зайвих пікселів і виділяємо активність, що залишилась (рис. 4).

Рис. 3. Приклад виділення всіх змінених пікселів

Рис. 4. Виділення цікавої зміни зображення

Далі необхідно отримати окремі суцільні об’єкти. Для цього в OpenCV є функція cv2.findContours [9]. Вона повертає масив контурів окремих груп пікселів. Та активність, яку ми отримали, недостатньо точно повторює контури реальних об’єктів, як можна бачити на прикладі (рис. 4). Для того, щоб отримати один суцільний об’єкт із декількох сусідніх контурів використовуємо функцію cv2.dilate [10] із значенням параметру kernel = 3x3. Кожен піксель на зображенні замінюється набором пікселів 3x3, в результаті чого зображення активності збільшується і заповнюються деякі проміжки між сусідніми контурами, що в багатьох випадках дозволяє отримати суцільні контури людей, що рухаються на відео (рис. 5).По отриманню контурів рухомих об’єктів необхідно їх попередньо проаналізувати та за деякими критеріями відсіяти непотрібні. По-перше відкидаються контури із площею менше за заданий поріг (в даному випадку 2000 *пікселів*^{2}) . Також, як показала практика, деякі камери можуть інколи віддавати зіпсовані кадри з полосами на всю ширину, або висоту кадру. В таких випадках отримані артефакти можуть сприйматись, як активні об’єкти. Тому необхідно також ігнорувати такі об’єкти. Решта об’єктів підлягає подальшому аналізу та обробці.

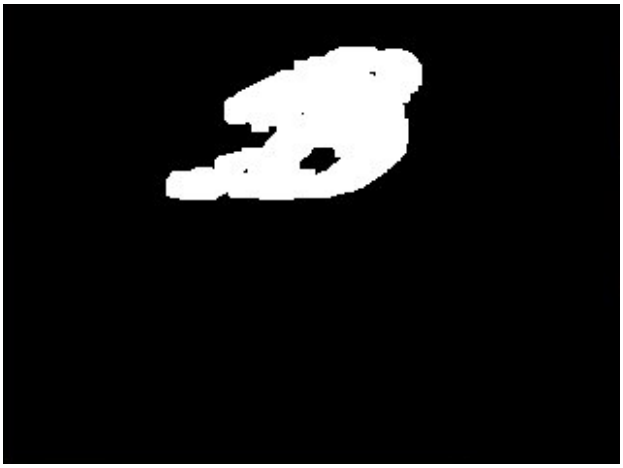

Рис. 5. Остаточний контур рухомого об’єкта

Алгоритм слідкування за об’єктами

Для того, щоб коректно оцінювати положення кожного об’єкта відносно контрольної лінії, необхідно розуміти яким об’єктам на одному кадрі відповідають об’єкти на наступному кадрі.

Кожен об’єкт на зображенні запам’ятовується. Контур кожного нового об’єкта перевіряється з контуром вже існуючих об’єктів наступним чином: якщо контури об’єкта в нового кадру пересікаються із деяким контуром вже існуючого об’єкта з минулих кадрів на площу більшу 30% від площі одного з цих двох контурів – вважаємо, що новий об’єкт є оновленим станом старого, і перезаписуємо його нове положення на зображенні.

Якщо збігів нового контуру із старими не виявлено – новий об’єкт додається до списку активних об’єктів. Об’єкти, що не оновилися під час обробки кадру мають час життя - 1.5 секунди. Якщо за цей час вони не оновилися – видаляємо такі об’єкти з пам’яті.

Алгоритм пересічення контрольної лінії

Під визначенням «контрольна лінія» мається на увазі відрізок, по пересіченню якого в обох напрямках необхідно підраховувати кількість об’єктів, які її пересікають. «Контрольна лінія» задається у вигляді координат двох точок. Для фіксування пересічення лінії необхідно виконання двох умов.

Перша умова: необхідна зміна положення об’єкта відносно лінії. Для цього на кожному кадрі повинно бути визначено поточне положення об’єкта по відношенню до контрольної лінії.

Для простоти розрахунків та пришвидшення роботи алгоритму перевіряється положення центра об’єкта. Проте в даному випадку потрібно враховувати таку ситуацію, коли людина знаходиться на лінії та стоїть на місці. Тоді центр об’єкта може пересікати лінію безліч разів, що негативно вплине на результат алгоритму. Для цього будо вирішено ввести таке поняття, як «контрольний простір» (рис. 6), тобто невеликий простір по обидва боки лінії, входження в який необхідне об’єкту для зарахування пересічення лінії. Ширина такого простору задається для кожного зображення різна.

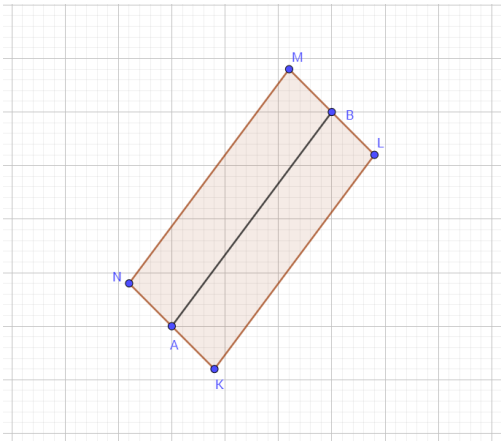

Рис. 6. Контрольний простір KLMN лінії AB

Координати точок K, L, M, N розраховуються за відповідними формулами, вважаючи, що KLMN – прямокутник, точка A – середина KN, точка B – середина LM, а довжина сторін KN = LM задається попередньо на етапі калібрування контрольної лінії.

Положення точки відносно лінії визначається наступним чином: Нехай існує контрольна лінія AB, контрольний простір, заданий точками K, L, M, N та центр об’єкта C. А також позначимо два простори зліва та справа від лінії як P1 та P2 (рис. 7).

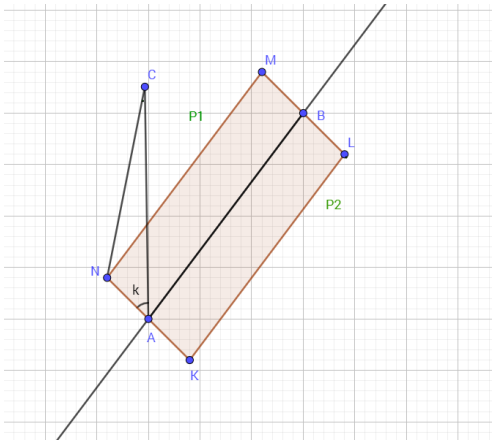

Рис. 7. Приклад положення точки C відносно лінії AB

Тоді центр об'єкта C знаходиться у просторі P1, якщо:

$$\text{[illegible]} < \frac{\text{[illegible]}}{2}$$

Або, центр об'єкта C знаходиться у просторі P2, якщо:

$$\text{[illegible]} > \frac{\text{[illegible]}}{2}$$

Друга умова: об'єкт повинен потрапити до «контрольного простору» та вийти з нього. Ця умова важлива в тих випадках, коли контрольна лінія задана не на всю ширину зображення, і нам необхідно ігнорувати зміну положення об'єкта за межами відрізку.

Зустрічаються випадки коли декілька людей, проходячи близько один до одного пересікають лінію. Такі групи людей можуть сприйматися як один великий об'єкт. Отже якщо зафіксовано пересічення лінії потрібно проаналізувати ширину та довжину об'єкта при пересіченні.

Для зручності розрахунків габаритів об'єкта контур необхідно представити у вигляді геометричної фігури, що якомога краще повторювала б силуети контуру і займала найменший зайвий простір навколо об'єкта. Спочатку були спроби працювати із прямокутником, проте, як показали практика, він займає багато зайвого місця, особливо біля його кутів. Тоді було прийнято рішення обводити контури за допомогою еліпса. Бібліотека OpenCV надає зручну функцію для отримання еліпса по контуру - cv2.fitEllipse, що повертає координати центру еліпса, його велику на малі вісь та кут нахилу в системі координат.

Для знаходження ширини об'єкта необхідно знайти його діаметр, що проходить під таким самим кутом, під яким нахилена до еліпса «контрольна лінія». Спочатку знаходиться кут нахилу лінії відносно еліпса. Нехай кут нахилу лінії до осі координат α, кут нахилу еліпса до осі координат β. Тоді кут нахилу лінії до еліпса визначається як:

$$\text{[illegible]} = 180 + \text{[illegible]} - \text{[illegible]}$$

Ширина об'єкта, або довжина діаметра під кутом, розраховується за формулою:

$$\text{[illegible]} = ((\frac{\text{[illegible]}}{\text{[illegible]}})^2 + (\frac{\text{[illegible]}}{\text{[illegible]}})^2)^{-\frac{1}{2}},$$

де *a, b* – малий та великий діаметри еліпса відповідно.

Довжина об'єкта розраховується як довжина діаметра еліпса, що є перпендикулярним до діаметра, розрахованого вище:

$$\text{[illegible]} = ((\frac{\text{[illegible]}\ (90 - \text{[illegible]})}{\text{[illegible]}})^2 + (\frac{\text{[illegible]}\ (90 - \text{[illegible]})}{\text{[illegible]}})^2)^{-\frac{1}{2}}$$

Перед роботою алгоритму необхідно вручну відкалібрувати також і приблизні ширину та довжину одиночного об'єкта (тобто об'єкта, що рівний розмірам однієї людини).

Знаючи габарити об'єкта, що пересік лінію, та приблизні габарити одиничного об'єкта, можна розрахувати скільки людей перетнули лінію.

Нехай ширина та довжина одиничного об'єкта задані значеннями *singleObjectWidth (sow)* та *singleObjectHeight (soh)*, тоді кількість людей *n* в об'єкті, що пересік лінію, розраховується наступним чином:

$$\text{[illegible]} = \frac{\text{[illegible]}}{\text{[illegible]}} * \frac{\text{[illegible]}}{\text{[illegible]}}$$

Після чого *n* округлюється до цілого числа і приймає остаточний результат.

Дослідження роботи алгоритму

На якість роботи алгоритму можуть впливати наступні фактори:

- Освітлення: рівномірне; наявність об'єктів-джерел освітлення (вікна, ліхтарі) з деякого боку від поля зору камери. Впливає на виділення активності на кадрі.
- Ширина проходу до приміщення. Впливає на скупчення людей у великі об'єкти.
- Висота розташування камери. Впливає на розміри одиничного об'єкта.

Було проведено дослідження роботи алгоритму в залежності від значень параметрів калібрування на прикладі трьох різних приміщень із різними характеристиками.

Нижче приведені важливі характеристики приміщень:

- Приміщення А: рівномірне освітлення, широкий прохід.
- Приміщення Б: наявність декількох вікон із сторони входу, широкий прохід.
- Приміщення В: нерівномірне освітлення, вузький прохід.

Параметри, що підлягають калібруванню наступні:

- Поріг чутливості: *threshold*
- Ширина контрольного простору: *dist*
- Ширина одиничного об'єкта: *sow*

Було записано відео тривалістю 24 години з кожного приміщення, отримані наступні результати роботи алгоритму в залежності від каліброваних параметрів.

Аналіз роботи алгоритму на прикладі приміщення А приведено в таблиці 1:

Таблиця 1

№	*threshold*	*dist*	*sow*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	15	10	50	581/453		73/92
2	25	10	50	454/434		94/96
3	40	10	50	380/351		89/84
4	30	15	50	438/427		97/98

5	30	15	60	428/422		99/99

Аналіз роботи алгоритму на прикладі приміщення Б приведено в таблиці 2:

Таблиця 2

№	*threshold*	*dist*	*sow*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	30	15	60	603/598		91/92
2	35	15	60	588/585		94/94
3	40	15	60	573/569		96/96
4	40	10	60	571/569		96/97
5	40	10	55	564/561		98/98

Аналіз роботи алгоритму на прикладі приміщення В приведено в таблиці 3:

Таблиця 3

№	*threshold*	*dist*	*sow*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	40	10	55	441/427		88/90
2	38	10	55	434/422		89/91
3	38	15	60	434/421		89/91
4	38	15	70	354/356		91/92
5	38	15	65	411/404		94/95

Також на якість роботи алгоритму впливають деякі фактори, які на даному етапі неможливо передбачити. До таких ситуацій можна віднести віддзеркалення об'єктів на відповідних поверхнях, реєстрування інших рухомих об'єктів, що не є людьми, таких як візки, двері, тощо. В таких випадках похибка може перевищувати 10%. Для запобігання таких ситуацій необхідна комбінація із іншим методом, наприклад додавання до алгоритму даних датчика температури.

Порівняння з іншими рішеннями

Далі приведено порівняння якості роботи розробленого алгоритму з деякими з існуючих рішень. Було обрано такі рішення, в основі яких лежить лише один з методів аналізу зображень, так само як представлений вище алгоритм.

Такими рішеннями є Automated-Attendance-System та Overhead Camera People Counter.

Вхідні дані для порівняння роботи рішень обрані ті самі, що і в розділі «Дослідження роботи алгоритму», а саме 3 відео тривалістю 24 години у різних за характеристиками приміщеннях. Проте для Automated-Attendance-System відео було знято на окрему камеру, розташовану під кутом до входу, щоб можливо було зафіксувати обличчя людей, що входять у приміщення.

Всі алгоритми буди попередньо налаштовані у відповідності до приміщення, задавши найбільш підходящі параметри (де можливо).

Порівняння роботи рішень на прикладі приміщення А приведено в таблиці 4:

Таблиця 4

№	*Рішення*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	Власний алгоритм	428/422		99/99
2	OCPC	419/415		98/99
3	AAS	388/388		91/93

Порівняння роботи рішень на прикладі приміщення Б приведено в таблиці 5:

Таблиця 5

№	*Рішення*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	Власний алгоритм	564/561		98/98
2	OCPC	532/529		97/96
3	AAS	501/501		91/91

Порівняння роботи рішень на прикладі приміщення В приведено в таблиці 6:

Таблиця 6

№	*Рішення*	Результат (In/Out)	Реальні дані (In/Out)	Точність, %
1	Власний алгоритм	411/404		94/95
2	OCPC	350/356		90/92
3	AAS	247/247		64/64

Отримані результати показують, що аналіз відвідуваності методом розпізнавання облич є найменш точним. Це пов'язано із освітленням та якістю зображення. Як бачимо у третьому випадку, при поганому освітленні точність алгоритму різко зменшилась.

Якщо порівнювати розроблений алгоритм із схожим аналогом (в даному випадку Overhead Camera People Counter). То результати приблизно однакові, проте слід врахувати, що завдяки більш детальній логіці пересікання лінії, врахування розмірів об'єкта, а також механізму слідкування власний алгоритм показує дедалі кращі результати при погіршенні умов, а саме освітлення та розмірів приміщення.

Висновки

В роботі розглянуто питання розробки алгоритму для підрахування людей на основі методів комп'ютерного бачення. Також було проведено дослідження роботи алгоритму в залежності від зовнішніх факторів. Після визначення найбільш підходящих параметрів виконано порівняння результатів роботи алгоритму із іншими рішеннями.

Список посилань

1. People counting [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://v-count.com/solutions/peoplecounting/ (дата звернення 03.11.2017). – Назва з екрана.

2. Hybrid-SyntaX [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://github.com/Hybrid-SyntaX/ Automated-Attendance-System-Using-Face-Recognition (дата звернення 12.11.2017). – Назва з екрана.

3. EmguOV [Електронний ресурс] : [Веб-сайт]. – Режим доступу: http://www.emgu.com/ wiki/index.php/Main_Page (дата звернення 01.12.2017). – Назва з екрана.

4. OpenCV [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://opencv.org/ (дата звернення 01.12.2017). – Назва з екрана.

5. People counting algorithm using an overhead video camera [Електронний ресурс] : [Веб-сайт]. – Режим досту- пу: https://github.com/agurz/overhead-camera-people-counter (дата звернення 01.12.2017). – Назва з екрана.

6. VXL - C++ Libraries for Computer Vision [Електронний ресурс] : [Веб-сайт]. – Режим доступу: http://vxl.sourceforge.net/#docs (дата звернення 02.12.2017). – Назва з екрана.

7. LTI-Lib [Електронний ресурс] : [Веб-сайт]. – Режим доступу: http://ltilib.sourceforge.net/ doc/homepage/index.shtml (дата звернення 02.12.2017). – Назва з екрана.

8. OpenCV: Image Filtering [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://docs.opencv.org/ 3.1.0/d4/d86/group__imgproc__filter.html#gaabe8c836e97159a9193fb0b11ac52cf1 (дата звернення 09.12.2017). – Назва з екрана.

9. OpenCV: Structural Analysis and Shape Descriptors [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https://docs.opencv.org/3.1.0/ d3/dc0/group__imgproc__shape.html#ga95f5b48d01abc7c2e0732db24689837b (да- та звернення 09.12.2017). – Назва з екрана.

10. Morphological Transformations; OpenCV 3.0.0-dev documentation [Електронний ресурс] : [Веб-сайт]. – Режим доступу: https:// docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc /py_morphological_ops/ py_morphological_ops. html#dilation (дата звернення 09.12.2017). – Назва з екрана.

Додаток Б

Теза

Дослідження впливу колірних моделей при визначенні кольорових характеристик об'єктів розпізнавання на зображенні

Антосєв Андрій Сергійович
магістрант
КПІ ім. Ігоря Сікорського
Київ, Україна
gibson12ant@gmail.com

Дорогий Ярослав Юрійович
К.т.н., доц.
КПІ ім. Ігоря Сікорського
Київ, Україна
Cisco.rna@gmail.com

В статті здійснено аналіз методів визначення кольорових особливостей зображень. Розглянуто популярні колірні моделі та окреслено їхні особливості.

Ключові слова: колірна модель, обробка зображень, комп'ютерне бачення, RGB, HSV, Lab, розпізнавання

Сьогодні вслід за швидким ростом обчислювальних можливостей комп'ютерів стають все більш популярними задачі комп'ютерного бачення. За допомогою програмного аналізу зображення вирішується безліч побутових та професійних проблем. Однією з яких є розпізнавання та слідкування за рухомими об'єктами.

Під час виявлення окремих об'єктів на зображенні недостатньо інформації лише про його форму та розташування, адже такі параметри в більшості випадків не несуть в собі унікальних знань про такі об'єкти. Найбільш важливими даними про об'єкт є його колір. Саме визначення кольорової характеристики зображення та подальший її аналіз суттєво спрощує задачу розпізнавання та слідкування за об'єктами.

Після виділення окремого об'єкта на зображенні, він представляється у вигляді набору пікселей із певними числовими значеннями кольорів. Слід виділити основні колірні моделі такі, як RGB, HSV та Lab.

RGB – найбільш розповсюджена адитвна колірна модель, що представляє собою спосіб синтезу кольору за допомогою накладання червоного, синього та зеленого світла відповідно [1].

Найпростішим методом для створення кольорової характеристики зображення є підрахунок кількості пікселів однакового кольору з використанням RGB моделі. Ідея полягає в тому, щоб порівнювати такі масиви даних з різних кадрів та слідкувати за об'єктами. Такий підхід є найлегшим як у розуміння та і з боку використання обчислювальних ресурсів. Проте існує ряд суттєвих недоліків. Так як кожна складова кольору має значення у межах від 0 до 255, то кількість можливих кольорів складає 16 777 216. При чому при найменшому впливі на об'єкт світла або зміни ракурсу до камери велика частина точок будуть змінювати своє кольорове значення. Для спрощення сприйняття кольору виконується нормалізація таких значень. Попередньо обирається в якому діапазоні цілих чисел повинні відображатись кожен з каналів. Такий діапазон зазвичай є унікальним для різних ситуацій чи потреб. Часто обирають максимальне значення - 7. Тобто кожен кольоровий канал, що попередньо був у діапазоні від 0 до 255, після нормалізації лежатиме в межах цілих чисел від 0 до 7 відповідно. Як нескладно зрозуміти, що для нормалізації необхідно поділити кожне значення кольорового каналу на 32. Маючи спрощені значення кольорів, з ними на порядок легше працювати адже невеликі зміни у їх відтінках у частині випадків не відображаються на їх цифровому значенні. Проте даних підхід є досить ненадійним. Нормалізовані кольори мають різку зміну тону, а також накладання тіні на частину об'єкта певного кольору може змінити його сприйняття.

Отже хотілось би звернути увагу на наступний метод, що вирішує недоліки свого попередника. Метод складання кольорової характеристики на основі HSV моделі враховує ситуації зі зміною тону кольору під час затінення об'єкта. Для початку декілька слів про дану кольорову модель. HSV відображає кольори у вигляді трьох наступних складових: колірний тон (Hue), насиченість (Saturation) та значення кольору (Value) [2]. Колірний тон - це значення "чистого" кольору, тобто без накладання чорного або білого тонів. Його значення зазвичай представляють в межах $0 - 360^o$, або просто від 0 до 100 (рис. 1).

Рис.1 - Шкала колірного тону (hue)

Насиченість відповідає за відтінок білого у кольорі і лежить у межах від 0 до 100, або від 0 до 1. При показнику 1 колір "чистий", при 0 - білий. Цей параметр може змінюватись при засвічені об'єкта білим світлом. Тобто при попаданні на об'єкт променів білого відтінку значення Hue, в ідеальному випадку, не змінюється. Змінює своє значення параметр Saturation. Отже можна зазначити, що обираючи колірну модель HSV, покращується сприйняття алгоритмом впливу світла на об'єкт розпізнавання.

Останнім параметром є - значення кольору. Дана складова відповідає за яскравість кольору та задається в аналогічних межах від 0 до 1, або від 0 до 100. При

наближенні показника до 0 колір приймає чорний відтінок. Таку особливість колірної моделі дуже зручно використовувати для аналіз падаючої на об'єкт розпізнавання тіні. Візуалізація HSV простору приведена на рис. 2 у вигляді циліндричної системи координат.

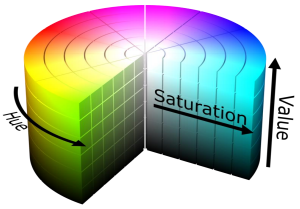

Рис. 2 - Візуалізація HSV моделі

Така колірна модель в більшості випадків дає кращі результати ніж RGB, проте вони не є ідеальними. Проблеми полягають у сприйнятті кольорів та світла. Іноді при засвічені або затіненні об'єкта він приймає відтінок іншого кольору чим спотворює дані алгоритму обробки.

Наступною необхідно виділити колірну модель Lab. Дана модель розроблялась з метою представлення кольорів у вигляді якомога ближчому до того, як їх сприймає людина. В просторі Lab відокремили хроматичну складову кольору від світлості. Тим самим Lab однозначно визначає колір, значення якого задається трьома параметрами. За світлість відповідає L, що лежить в діапазоні від 0 до 100, тобто від темного до світлого. Колір же визначається двома координатами - a та b. перша координата визначає відношення зеленого до червоного, а друга відношення синього до жовтого. Так як в моделі Lab світлість відокремлюється від кольору дія світла на зображення майже не відображає змін у сприйнятті кольорів алгоритмом. А отже створення кольорової характеристики об'єкта виконується дуже подібно на різних кадрах під різним впливом на цих об'єктів світла, що є надзвичайно важливим при обробці послідовності кадрів із метою виділення окремих об'єктів.

Важливо відмітити, що перетворення RGB зображення в Lab - ресурсозатратний процес. Це може стати на заваді при реалізації алгоритму з використанням моделі Lab на слабких машинах, наприклад Raspberry Pi.

Для порівняння впливу вибору колірної моделі на результат дії алгоритму розпізнавання було проведено наступне дослідження. Було обрано послідовність з десяти кадрів, попередньо записаних на відео з камери. На уривку відео зображена людина, що проходить по нерівномірно освітленому приміщенню, тобто на об'єкт спостереження на деяких кадрах падає світло або тінь. на кожному кадрі виділяється контур людини та знаходяться всі точки всередині контуру, кожна з яких має свій колір у відповідній колірній моделі. Далі розраховується скільки пікселів якого кольору має цей об'єкт на кожному кадрі. Такий набір даних далі вважаємо за колірну характеристику об'єкта. Після чого порівнюється колірна характеристика на одному кадрі з попередньою на скільки відсотків вони подібні. Слід зазначити, що відхилення до 10 відсотків можна вважати допустимим, адже розміри контуру можуть змінюватись в процесі знаходження об'єкта на зображенні, а також під час руху деякі частини об'єкта закриваються або навпаки з'являються

Таблиця 1 - Порівняння колірних моделей

№ кадру	2	3	4	5	6	7	8	9	10
RGB	74%	68%	38%	59%	67%	71%	27%	53%	62%
HSV	93%	87%	73%	92%	90%	91%	69%	85%	89%
Lab	98%	96%	93%	96%	97%	92%	89%	93%	94%

Також необхідно зауважити, що перед проведенням досліду значення каналів були нереалізовані. Адже без нормалізації результати порівняння сусідніх кадрів складали близько 8% при нормальних умовах використовуючи колірну модель RGB. Після нормалізації діапазони кожного параметру було зменшено у 8 разів, що не сильно спотворює дані.

На кадрах 1-3 на об'єкт падає світло; кадр 4 - вихід об'єкта у рівномірно освітлений простір; 5-7 об'єкт знаходиться у рівномірному освітленні. На кадрі 8 об'єкт заходить в тінь і рухається в темній частині приміщення.

З таблиці 1 можна побачити, що використовуючи модель RGB схожість колірних характеристик об'єкта на сусідніх кадрах 60-70%. Проте при зміні освітлення результати різко погіршуються. Цю проблему вирішують моделі HSV та Lab. З їх використанням вплив світла та тіні впливає на такий значний. Найкраще себе показую саме Lab модель, через відокремлення значення світлості від кольору. Також слід зауважити, що використання двох останніх моделей при визначенні активності на відео значно покращує знаходження контурів рухомого об'єкта, адже поява тіні на зображенні ігнорується, що не можна сказати про модель RGB, де невелика зміна відтінку буде сприйматись, як рухомий об'єкт.

Таким чином при розробці алгоритму обробки зображень з урахуванням кольору рекомендовано обирати колірну модель Lab, зважаючи на її переваги над іншими.

Література

1. Лисиця В. Т. КОЛІРНІ МОДЕЛІ ТА ЗАКОНИ ПОШИРЕННЯ СВІТЛА / В. Т. Лисиця.–2012.– С. 19.
2. Комп'ютерна графіка "Навчальний посібник". // Тернопільський національний технічний університет імені Івана Пулюя. – 2017. – С. 44.
3. ПРОЦЕДУРА КОРРЕКЦИИ ЦВЕТОВОГО БАЛАНСА ЦИФРОВЫХ ИЗОБРАЖЕНИЙ В ЦВЕТОВОМ ПРОСТРАНСТВЕ LAB. // Збірник наукових праць Харківського університету Повітряних Сил. – 2014. – №1. – С. 122–12