РЕФЕРАТ

Дипломна робота: 49 ст. , 16 рис., 12 табл., 2 джерел та 2 додатки.

Метою даної роботи є реалізація методів по локалізації людської руки у відеопотоці. У роботі реалізовані такі методи як: віднімання фону, колірний фільтр на основі байесовскього класифікатора та обробка відеопотоку з камери глибини.

Результати роботи:

- реалізовані три підходи по локалізації людської руки на відео;
- проведений аналіз умов для надійної роботи алгоритмів;
- запропоновано і реалізовано ефективний метод навчання байесовскього класифікатора;
- проведено порівняння алгоритмів.

Результати даної роботи рекомендовано використовувати у системах взаємодії людини та комп’ютера. При подальших дослідженнях у цій області доцільно створити систему по взаємодії с комп’ютером на основі реалізованого методу детекції руки на відео.

РОЗПІЗНАВАННЯ ОБ’ЄКТІВ, ВЗАЄМОДІЯ КОМП’ЮТЕРА ТА ЛЮДИНИ, КАМЕРА ГЛИБИНИ, БАЙЕСОВСЬКИЙ КЛАСИФІКАТОР, КОЛІРНИЙ ПРОСТІР.

ABSTRACT

The thesis: 49 p. , 16 fig., 12 tabl., 2 sources and 2 appendices.

The purpose of this thesis is to realize and analyze 3 approaches of human hand localization in video stream. In the thesis were realized the following methods: background subtraction, color filter based on Bayes classifier and processing the video stream from depth camera.

Thesis results:

- were realized 3 methods of human hand localization in video stream;
- for each method were analyzed the optimal working conditions;
- proposed a convenient way for Bayes classifier training;
- made a comparison of methods.

Current thesis results are proposed for using in computer-human interaction systems. In further researches it is reasonable to create a computer-human interaction system based on realized approaches and gesture recognition with support vector machine or neural networks.

OBJECT RECOGNITION, COMPUTER-HUMAN INTERACTION, DEPTH CAMERA, BAYES CLASSIFIER, COLOR SPACE.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВС — Bayes classifier ПК — портативний комп’ютер

ВСТУП

В наші часи складно уявити своє життя без комп'ютера. Він стоїть удома, на роботі, в школі чи університеті. Основний способ комунікації з цим складним девайсом є усім відомі пристрої вводу такі як: клавіатура, мишка чи в більш продвинутому випадку графічний планшет, який є просто незамінний для малювання на ПК.

Права рука людини, що працює за комп'ютером, більше 90% часу знаходиться на мишці, оскільки основна частина користувачів ПК використовує операційні системи з графічними оболонками де більша частина простих керуючих операцій здійснюється за допомоги курсора, який контролюється саме мишою. Варто подумати про те, що робить у цей час ліва рука користувача. Вона дуже часто бездіє чи нажимає прості гарячі комбінації з двухтрьох кнопок на клавіатурі, хоча це малоймовірний випадок оскільки мізерний відсоток людей знає більше 5 гарячих комбінацій. Тобто перша очевидна проблема взаємодії з ПК – мала ефективність основних способів комунікації.

Взагалі перехід операційних систем на графічну оболонку можна вважати дуже великим кроком уперед. Більшість користувачів ПК сприйняли цей крок позитивно адже для запуску програми треба лише навести на її іконку курсор та натиснути один-два рази на ліву кнопку миші. Перша комп'ютерна миша, що була відносно доступною для простих людей, мала лише одну функціональну кнопку та коштувала приблизно 25$. Вона була випущена разом із операційною системою Apple Macintosh в якій якраз з'явилась підтримка віконного інтерфейсу. Це значно пришвидшило роботу з файлами та інші рутинні операції оскільки пару кліків мишкою заміняли доволі таки складні команди в терміналі. Проте розвиток технологій та потреб користувачів призводить до того, що миша вже не може повністю покривати множину найчастіших команд користувача лише трьома кнопками та сенсором руху. Заміна середньої кнопки на колесо якраз є прикладом додавання нової функціональності миші для покриття більшої кількості команд. Саме розвиток інтернету та браузерів призвів до того, що дуже часто потрібно листати достатньо довгі сторінки і колесо прокрутки для цього підходить набагато краще. Також дуже часто до мишки додають допоміжні функціональні

кнопки, які можна запрограмувати на якусь дію чи навіть послідовність дій. Це також вимушений крок розробників мишок, проте площина поверхні мишки обмежена і місце для кнопок з часом закінчиться.

В останні пару років комп'ютерна миша еволюціонувала на багатьох пристроях у сенсорну панель. Завдяки тому, що сенсорна панель може розпізнавати до десяти пальців, з'являється можливість обробки комплексних жестів для масштабування зображення, прокрутки текста або інтернет сторінки чи навіть створення власних жестів та програмування виконання певних дій при фіксуванні цього жеста. На данний момент це один із самих ефективних мишкоподібних засобів вводу оскільки покриває дуже велику множину команд з можливістю її розширення.

Проте останній рік був дуже насиченим в плані дослідження проблеми взаємодії людини та ПК. Не так давно була випущенна камера Microsoft Kinect для приставки Xbox One. Завдяки поєднанню потоків з двох серсорів RGB та depth з'явилась реальна можливість відслудковування об'єктів у просторі. Звісно це було можливо і раніше за допомоги декількох RGB сенсорів чи навіть спеціальних рухових сенсорів, проте великим недоліком цих методів можна вважати достатньо масштабні обчислення або велику кількістю проводів. Поєднавши у камері кольоровий сенсор та сенсор відстані виходить людина не повинна взагалі нічого на себе чіпляти, не потрібно одягати спеціальний одяг з різноманітними сенсорами та передавачами. Комп'ютер може бачити людину у просторі самостійно. Також минулого року компанія Intel анонсувала технологію Realsense та лінійку камер F200, R200, SR300. Відрізняются вони лише призначенням, але головна ідея в тому, щоб дати можливість ПК та мобільним пристроям бачити у просторі. Одною із основновних частин Realsense SDK є модуль по відслідковуванню руки та аналізу статичних і динамічних жестів.

Для того щоб зрозуміти важливість цієї проблеми потрібно порівняти функціональність комп'ютерної миші та людської руки. Миша має в основному сенсор положення у 2d просторі, 3 кнопки та колесо прокрутки. Людська рука може знаходитись у 3d просторі та приймати достатньо складні форми (статичні жести) чи робити деякий комплекс рухів у 3d просторі (динамічні жести). Очевидно, що множина станів людської руки більш різнома-

нітна ніж множина станів миші. Саме тому багато дослідників нових способів взаємодії людини та комп'ютера прийшли до того, що можна керувати ПК без тримання в руках яких-небудь пристроїв лише за допомоги жестів. На данний момент спілкування людини та комп'ютера дуже схоже на спілкування людини і сліпої та глухої людини, що виступає в ролі комп'ютера. Дійсно, ПК людину не бачить та не чує взагалі.

Система по локалізації людських рук на відео має бути достатньо простою в плані обчислень так як вона повинна працювати в режимі реального часу та обробляти від 30 до 60 фреймів на секунду. Це зразу відсіює достатньо велику частину підходів які використовуються для локалізації людських рук на фото і обробляють одне зображення довше ніж за одну секунду.

Таким чином, метою цієї роботи є досліждення методів локалізації людської руки на відео.

Для досягнення цих цілей вирішені наступні задачі:

- Проведений аналіз існуючих підходів по локалізації руки на відео та обробки цифрових зображень.
- Реалізовані три методи та проведене дослідження їх роботи.
- Реалізований зручний спосіб навчання Байесовского класифікатора

Об'єктом дослідження є системи по локалізації людської руки у відеопотоці.

Предметом дослідження є методи та алгоритми формування системи детекції людської руки за допомоги веб камери чи камери глибини.

Наукова новизна отриманих результатів полягає у тому, що проведений достатньо обширний аналіз існуючих методів та порівняння.

Практичними результатами роботи є реалізація трьох підходів, що працюють достатньо точно в певних специфічних умовах, та порівняльний їх аналіз.

Робота складається з чотирьох розділів. У першому розділі розглядається постановка задачі дослідження та актуальність проблеми. Другий розділ присвячений критеріям якості рішення задачі та опису алгоритмів локалізації людської руки на відео. У третьому розділі здійснено огляд технологій та алгоритмів, що використовуються в роботі, проведений порівняльний аналіз та

наведено схеми програм. У четвертому розділі розглядається фунціонально-вартісний аналіз програмного продукту.

РОЗДІЛ 1 АНАЛІЗ ЗАДАЧІ РОЗПІЗНАВАННЯ ЛЮДСЬКОЇ РУКИ НА ВІДЕО

1.1 Актуальність поставленої задачі

Розпізнавання людської руки на відео - задача актуальна саме зараз оскільки це основа для більш складного процесу розрізнавання статичних та динамічних жестів.

Розпізнавання жестів насамперед це лише ідея по роботі з жестами і базується на абсолютно різних технологіях: переносимі пристрої (рукавиці чи інші контролери), зафіксовані камери (RGB камери, depth камери чи комбінація RGB камер для отримання стереоскопічного зображення) чи навіть радари (працють по принципу сканування магнітного фону та його зміни у часі).

Застосування можуть мати найрізноманітніші форми:

- Керування ПК - у 2016 році вийшли перші ноутбуки з вбудованою камерою Intel Realsense F200 (замість звичної RGB камери). На данний момент на її базі створюються програмні продукти по керуванню ПК без миші.
- Навчання мові німих - аналізуючи жести можна повністю оцифрувати мові німих та створити програму-вчителя, що буде показувати жести і перевіряти наскільки правильно користувач їх повторює
- Керування віртуальними середовищами - в кінці 2015 року компанія Microsoft презентувала прототип Hololens, що створений саме для реалізації доповненої реальності та за допомоги камер Kinect аналізувати жести користувача для більш звичної взаємодії з віртуальним середовищем оскільки майже 90% фізичної взаємодії з середовищем людина виконує за допомоги рук, а тому слід вважати, що це основний спосіб взаємодії.

Причини складності задачі:

- Нестабільні умови освітленності - освітленність дуже сильно впливає на алгоритми, що базуються на обробці RGB відеопотоку;
- Поява сторонніх об’єктів у полі зору камери - дуже сильно впливає оскільки збільшує ймовірність розпізнати цей об’єкт як той, що потребує подальшого аналізу;
- Недостатня якість камер призводить до того, що в зображенні присутні шуми;
- Прості RGB камери мають неякісні матриці і через це реєструють кольори спотворено ускладнює роботу з колірними ознаками шкіри людини;
- Колір шкіри людей різниться і тому навчання класифікаторів потрібно проводити на достатньо великих вибірках.

1.2 Аналіз існуючих підходів до вирішення задачі

1.2.1 Виділення об’єктів за допомоги віднімання фону

Віднімання фону - процедура, що за умов нерухомості камери, оцінює зміну кожного пікселя зображення відносно фону, зображення якого було отримане раніше, та отримати в результаті бінарне зображення відмінності поточного зображення від зображення фону.

У вигляді математичних формул це виглядає так:

$$P[F(t)] = P[I(t)] - P[B] \tag{1.1}$$

де $P[B]$ - зображення фону, $P[I(t)]$ - поточне зображення, $P[F(t)]$ - різниця між поточним зображенням да фоном.

За умови незмінності фону можна працювати з формулою (1.1).

Для відслідковування рухів динамічних об’єктів також використовується рівняння:

$$\|P[F(t)] - P[F(t+1)]\| > Treshold \tag{1.2}$$

де $Treshold$ - деяка задана порогова величина.

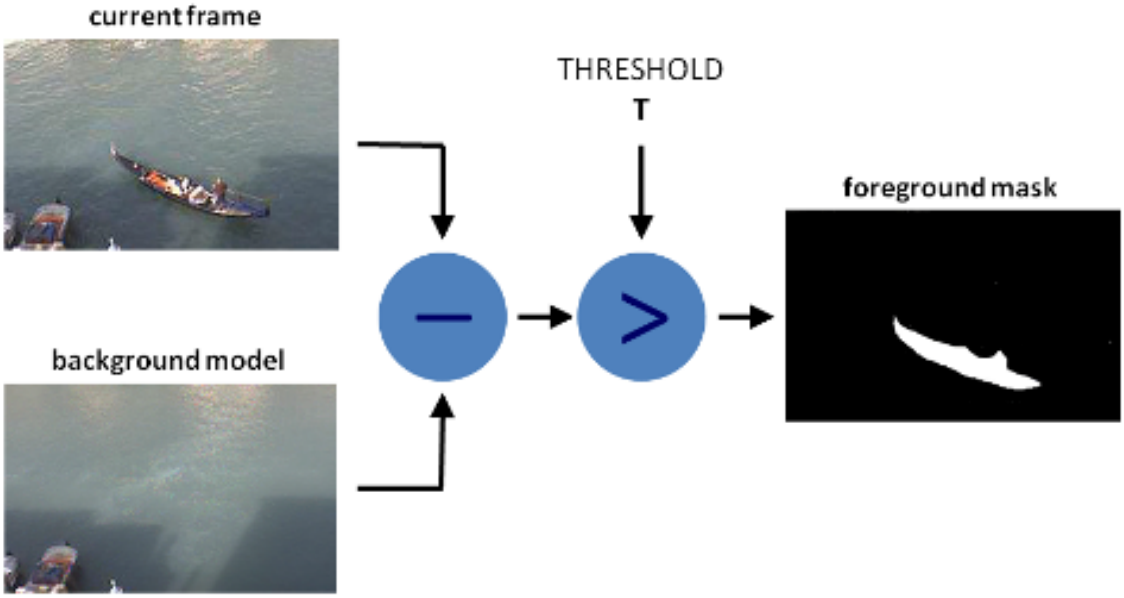

Рисунок 1.1 – Ілюстрація роботи простого алгоритма віднімання фону

Алгоритм дійсний як для кольорових зображень, так і для зображень у відтінках сірого.

Переваги цього алгоритму у тому, що його достатньо просто реалізувати та він не потребує складних обчислень (більшість операцій можна векторизувати та паралелізувати що значно прискорює швидкість обробки). Проте його найбільшим недоліком можна вважати те, що за його допомоги неможливо виділити об’єкт на динамічному фоні, він зпродукує велику кількість областей для подальшого аналізу, який може бути вже не таким тривіальним і система на його базі не зможе працювати в режимі on-line.

1.2.2 Відслідковування руки за допомоги контролерів

У минулих століттях саме цей підхід для роботи з відслідковуванням руки та аналізу жестів був популярним через достатньо прозору та просту програмну реалізацію. За допомоги приблизно 16 сенсорів та вбудованого в рукавицю гіроскопа можна достатньо точно відслідковувати навіть мікрорухи руки.

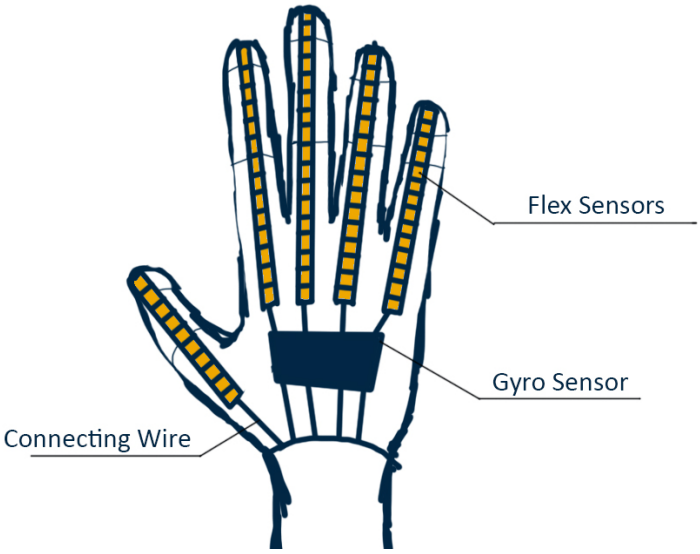

Рисунок 1.2 – Схема рукавиці з сенсорами

З точки зору розробників ПЗ простота була у тому, що на вхід вони вже мали позиції усіх ключових точок руки відносно центру (найчастіше це була лодоня), та положення центру у просторі. Для аналізу статичних жестів достатньо мати лише відносне положення усіх ключових точок відносно центру. Для аналізу динамічних жестів вже потрібні дані зміни положення руки у просторі.

Також були прототипи контролерів, що містили лише гіроскоп, проте відслідковування позиції у просторі руки як одного об’єкта без данних про позиції ключових точок на пальцях виявився нікому непотрібним і контролели швидко вийшли з виробництва.

Переваги цього підходу до відслідковування руки у просторі в тому, що він дає дуже точні результати та з мінімальною затримкою, проте він має дуже серйозний недоліки - потреба в носінні на собі додаткових незручних приладів та складність підтримки і ремонту таких девайсів.

1.2.3 Колірні фільтри для детекції шкіри на зображенні

Оскільки будь-яке цифрове зображення можливо представити у вигляді:

$$M = matrix\, W \times H \times D\ ,\ d = dimD \tag{1.3}$$

де M - цифрове зображення, W - ширина зображення, D - розмір колірного простору. Наприклад для типових колірних просторів RGB, HSV, YCrCb $d = 3$. Для колірного простору градацій сірого $d = 1$.

Колірний простір можна представити так:

$$(x = (x_1, ..., x_d) \in D) \Leftrightarrow (0 <= x_1 <= d_1, ..., 0 <= x_d <= d_d) \tag{1.4}$$

де D - колірний простір, $d_1, ..., d_d$ - обмеження координат колірного простору.

Найчастіше зустрічається випадок коли $x_i \in \mathbb{N} \cup \{0\}, d_i = 2^8, i = 1..d$ оскільки це списується в принцип збереження данних у пам'яті ПК.

1.2.3.1 Прості колірні фільтри

Основний принцип роботи простих колірних фільтрів полягає у тому, що емпірічними методами виділяється гіперкуб, що повністю містить у собі всі можливі значення пікселей, що характеризували певний об'єкт.

Формальний вигляд простого фільтра:

$$H = x \in D \mid a_i <= x_i <= b_i, i = 1..d \tag{1.5}$$

де a_i, b_i - грані гіперкуба.

Прості колірні фільтри хороші тим, що дуже прості в реалізації, швидко працюють та є можливість паралелізації обчислень оскільки фільтр обробляє кожен піксель незалежно.

Результатом роботи колірного фільтра H зображення M є бінарна матриця Q :

$$Q = [q_{ij}],\ q_{ij} = (M[i,j] \in H), i = 1..dimW, j = 1..dimH \tag{1.6}$$

Тобто матриця $q_{ij} \in \{0, 1\} \forall i, j$ і тому можна достатньо просто записати композицію простих фільттрів як поелементні бінарні операції над матрицями.

Звісно за допомоги кубів можливо апроксимувати достатньо точно найскладніші фігури, проте це має свої слабкі сторони:

а) апроксимація певного тіла кубами в просторі розмірності d достатньо нетривіальна задача з точки зору математики

б) зі збільшенням точності буде рости кількість простих колірних фільтрів у ланцюгу композицій і це призведе до падіння швидкості роботи

1.2.3.2 Гаусівська модель

Одним з більш складних підходів є припущення, що ймовірність належності кольору пікселя до множини кольорів певного об'єкта є деяка випадкова величина, що розподілена за гаусівським законом розподілу [1].

Також цей підхід враховує змінні умови освітлення шляхом поєднання двух гаусівських моделей. Перша гаусівська модель будується у спеціальних умовах звичайного освітлення, друга - при черезмірному освітленні.

Алгоритм навчання такої моделі - метод вибору математичного сподівання та кореляційної матриці.

Класифікація проводиться згідно з вибраною довірчою ймовірністю. У конкретному випадку колірного простору розміності 2 це буде еліпс.

Основним недоліком цього підходу є те, що на практиці достатньо важко правильно підібрати математичне сподівання та кореляційну матрицю і тому точність методу недостатня.

1.2.3.3 Байесовська модель

Цей підхід зустрічається у літературі не дуже часто і основна формула, що описує модель, на перший погляд незрозуміла:

$$P(s|c) = \frac{P(c|s) * P(s)}{P(c)} \tag{1.7}$$

Це звичайний запис теореми Байеса для знаходження апостеріорних ймовірностей.

$P(s|c)$ - ймовірність того, що піксель належить до множини кольорів шуканого об'єкта за умови що піксель має колір c. $P(s|c)$ в свою чергу має обернене формулювання - ймовірність того, що піксель приймає значення c за умови що він належить до множини кольорів об'єкта. $P(s)$ - ймовірність того, що піксель належить до множини кольорів об'єкта. $P(c)$ - загальна ймовірність того, що піксель має колір c.

Прочитавши такі формулювання складно взагалі зрозуміти як обраховувати ймовірності що стоять по праву сторону від рівності.

Основне припущення цього підходу - кольори окремих пікселей на зображенні незалежні один від одного. За такого припущення можна сказати, що $P(s)/P(c)$ - деяка загальна константа якою можна знехтувати поставивши її рівною 1, а $P(c|s) = m/n$, де m - кількість пікселів кольору c, що належать об'єкту, n - загальна кількість пікселів, що належала об'єкту.

Навчання такої моделі можна розбити на 2 етапи:

а) Обробка зображення для навчання - на зображеннях призначених для навчання замальовуються усі сторонні елементи окрім шкіри і таким чином формується маска зображення.

б) Навчання - підраховується кількість кожного пікселя що потрапляє у область шкіри на зображенні та в кінці навчання ділиться на загальну кількість пікселів, що належали шкірі.

Таким чином побудована функція, що ставить у відповідність елементам колірного простору величину на проміжку $[0, 1]$. У випадках розмірностей 1 чи 2 можливо побудувати графік цієї функції для аналізу навчання моделі.

1.3 Формалізація постановки задачі дослідження

1.3.1 Робочі позначення

1.3.2 Формальний опис задачі

1.3.3 Основні цілі

1.4 Висновки за розділом

РОЗДІЛ 2 ТЕОРЕТИЧНА ЧАСТИНА

2.1 Загальний підхід до задачі розпізнавання людської руки на відео

Загальний підхід до розв'язання поставленої задачі можна розбити на окремі етапи:

а) Отримання поточного зображення з камери - у випадку простої RGB камери це лише отримання поточного зображення, проте у випадку Intel Realsense камери цей етап може потребувати деяких простих обчислень. Оскільки камера від компанії Intel має окрім звичайного RGB сенсору ще й інфрачервоний для отримання карти глибини може виникнути проблема синхронізації двох відеопотоків оскільки фізично ці два сенсори знаходяться на деякій відстані один від одного. Методи для синхронізації це звичайні алгоритми синхронізації відеопотоків 2 чи більше камер зі стереоскопії. Саме такий і реалізований у драйвері librealsense.

б) Попередня обробка зображення - використовується для стабілізації будь-якого цифрового зображеня припускаючи що сенсори не ідеальні та можуть вносити деяку похибку, яку можливо зменшити використавши фільтри зглажування: box фильтр, гаусівський фільтр чи фильтр медіаною.

в) Застосування вибраного алгоритму до зображення - цей етап також можна узагальнити для трьох вибраних підходів оскільки в усіх випадках вхідні дані алгоритму складаються лише з зображення, а вихідні з бінарного зображення з виділеними областями в яких рнймовірно локалізована людська рука.

г) Заключна обробка вихідного результату вибраного алгоритму - отримавши бінарне зображення з виділеними областями потрібно обробити усі області та відфільтрувати ті, що по певним параметрам немає сенсу розглядати. Наприклад, вважаючи, що рука на зображенні повинна мати розмір більший ніж деяка фіксована величина можливо відсіяти велику кількість малих за площею областей. Те саме стосується і занад-

то великих областей, які за деяких причин були помічені алгоритмом як рука, проте вони завеликі для обробки і їх також немає сенсу розглядати.

На виході буде отримано бінарне зображення з областями, в яких ймовірно знаходиться людська. На цьому задача локалізації людської руки завершується і цей результат може подаватися на вхід алгоритмів по аналізу жестів. Які працюють з кожною областю окремо та можуть проводити свою спеціальну перевірку на те, чи знаходиться у цій області людська рука.

2.2 Колірні моделі та простори

Оскільки цифрова електроніка оперує лише дискретною математикою, то над вченими 20 століття постала проблема представлення кольорів у ЕОМ. Основна ідея представлення кольорів прийшла з науки біології та експериментальним шляхом було доведено, що людина є трихроматом [2]. Сітчатка ока трихроматів має 3 види рецепторів, що називаються ковбочками, які відповідають за колірний зір. Кожна з цих ковбочок має як параметр деяку довжину хвилі, на яку вона дає максимальний зворотній сигнал.

За історичних обставин склалося так, що еталонним колірним простором є XYZ. Ця колірна модель була запропонована та прийнята організацією CIE (International Commission on Illumination) у 1931 році. Саме ця модель є базовою для практично усіх інших колірних моделей.

Експерименти, що були проведені Девідом Райтом та Джоном Гілдом у 1930х роках послужили основою для визначення функції колірної відповідності.

Колір у моделі XYZ задається таким чином:

$$\begin{aligned} X &= \int_{380}^{780} I(\lambda) * x(\lambda)\mathrm{d}\lambda \\ Y &= \int_{380}^{780} I(\lambda) * y(\lambda)\mathrm{d}\lambda \\ Z &= \int_{380}^{780} I(\lambda) * z(\lambda)\mathrm{d}\lambda \end{aligned} \tag{2.1}$$

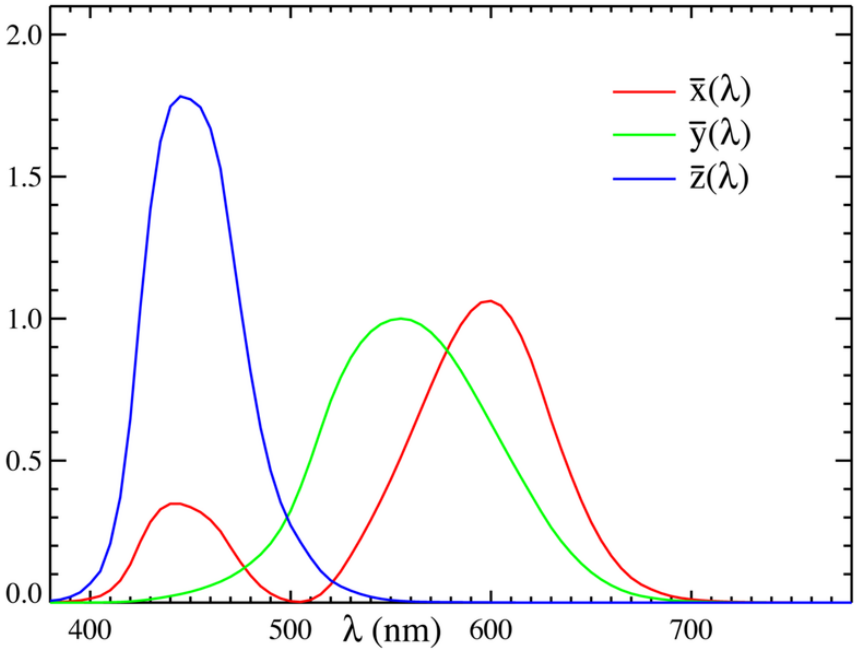

Рисунок 2.1 – Функції колірної відповідності

Саме ця модель задає правила змішування кольорів та задає обмеження що накладаються на спектральні складові, які мають один колір.

Якщо формально побудувати переріз простору XYZ площиною $X + Y + Z = const$, то 2 з 3 координат будуть лінійно незалежні і їх можна записати так:

$$\begin{aligned} x &= \frac{X}{X+Y+Z} \\ y &= \frac{Y}{X+Y+Z} \\ z &= \frac{Z}{X+Y+Z} \end{aligned} \tag{2.2}$$

Такий переріз називається хроматичною діаграмою.

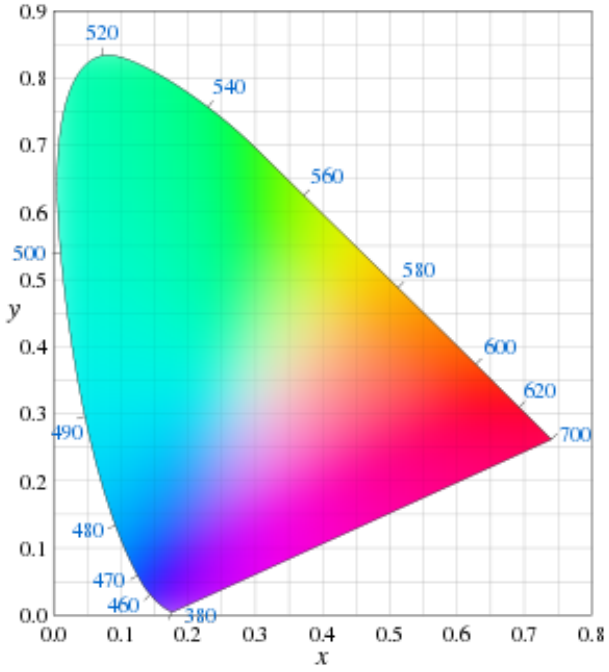

Рисунок 2.2 – Хроматична діаграма з довжинами хвиль кольорів

2.2.1 RGB

RGB (скорочено від англ. Red, Green, Blue — червоний, зелений, синій) — адитивна колірна модель, що описує спосіб синтезу кольору, за якою червоне, зелене та синє світло накладаються разом, змішуючись у різноманітні кольори. Широко застосовується в техніці, що відтворює зображення за допомогою випромінення світла.

У даній моделі колір кодується градаціями складових каналів (Red, Green, Blue). Тому за збільшення величини градації котрогось каналу — зростає його інтенсивність під час синтезу.

Кількість градацій кожного каналу залежить від розрядності бітового значення RGB. Зазвичай використовують 24-бітну модель, у котрій визначається по 8 біт на кожен канал, і тому кількість градацій дорівнює 256, що дозволяє закодувати 2563 = 16 777 216 кольорів.

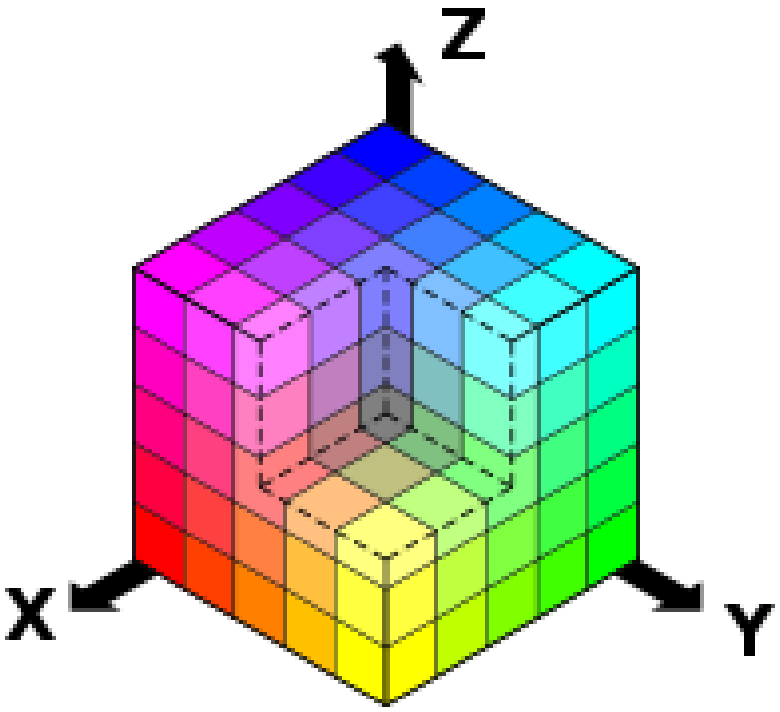

Рисунок 2.3 – Тривимірне представлення моделі RGB

Колірна модель RGB призначена сприймати, представляти та відображати зображення в електронних системах, таких як телебачення та комп’ютери, хоча її також застосовували у традиційній фотографії. Вже до електронного віку, модель RGB мала за собою серйозну теорію, засновану на сприйнятті кольорів людиною.

Типово приладами із RGB-входом є кольоровий телевізор і відеокамера, сканер і цифровий фотоапарат.

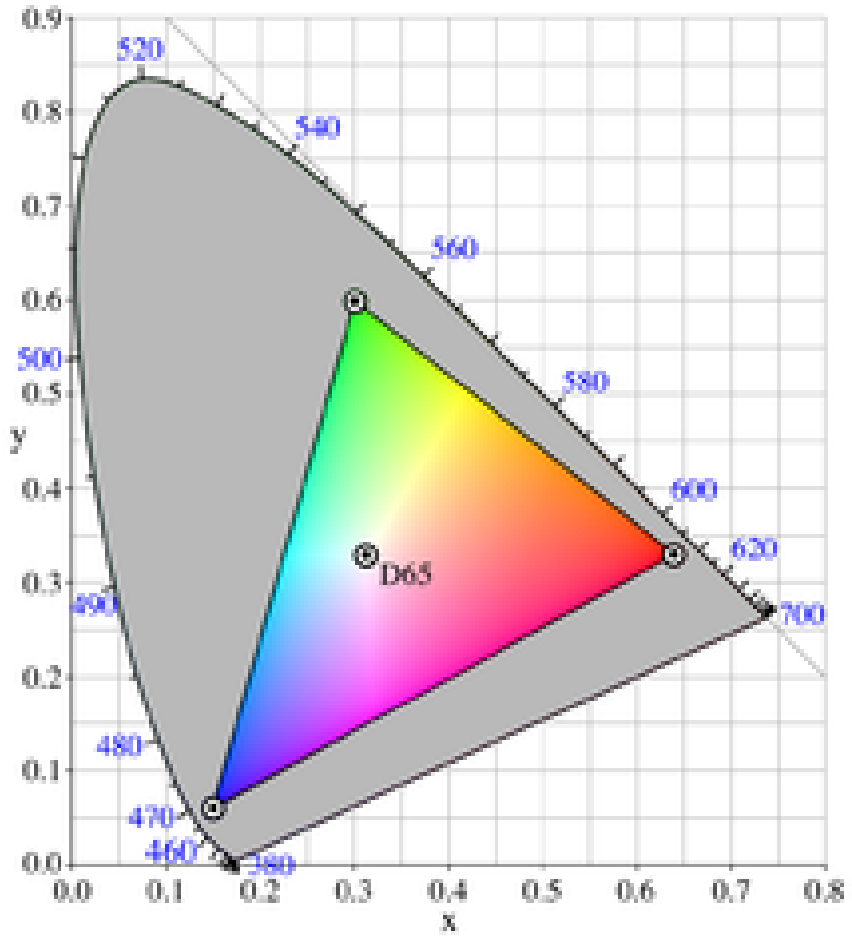

Рисунок 2.4 – Обмеженість моделі по можливості передачі кольору

Переваги моделі:

а) Апаратна близкість із монітором, сканером, проектором, іншими пристроями;
б) Велика кольорова гама, близька до можливостей людського зору;
в) Доступність багатьох функцій обробки зображення (фільтрів) у програмах растрової графіки;
г) Невеликий (порівняно до моделі CMYK) обсяг, проте ширший спектр кольорів.

Недоліки:

а) Обмеженість моделі;
б) Немає явного відділення люмінантної компоненти - для аналізу яскравості пікселя потрібно робити додаткові перетворення.

2.2.2 HSV,HSL

HSB — колірна модель, що використовується тільки для оформлення векторних і текстових об’єктів документа. Описує колірний простір, заснований на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і яскравості (Brightness).

- Hue — колірний тон, (наприклад, червоний, зелений або синьо-блакитний). Варіюється в межах 0-360°, але іноді приводиться до діапазону 0-100 або 0-1. У Windows весь колірний спектр ділиться на 240 відтінків (що можна спостерігати в редакторі палітри MS Paint), тобто тут «Hue» зводиться до діапазону 0-240 (відтінок 240 відсутній, оскільки він дублював би 0).
- Saturation — насиченість. Варіюється в межах 0-100 або 0-1. Чим більший цей параметр, тим «чистіший» колір, тому цей параметр іноді називають чистотою кольору. А чим ближчий цей параметр до нуля, тим ближчий колір до нейтрального сірого.
- Value (значення кольору) або Brightness — яскравість. Також задається в межах 0-100 або 0-1.

Модель була створена Елві Реєм Смітом, одним із засновників Pixar, в 1978 році. Вона є нелінійним перетворенням моделі RGB.

Колір, представлений в HSV, залежить від пристрою, на який він буде виведений, так як HSV — перетворення моделі RGB, яка теж залежить від пристрою. Для отримання коду кольору, не залежного від пристрою, використовується модель Lab.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Результати роботи та аналіз оптимальних умов роботи алгоритмів

3.1.1 Віднімання фону

3.1.2 Байесовський класифікатор

3.1.2.1 Класична реалізація

3.1.2.2 Корекція класифікатора

3.1.3 Камера глибини

3.2 Порівняння алгоритмів

3.3 Обгрунтування вибору платформи та мови програмування

3.4 Висновки до розділу

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для обробки вхідного відеопотоку і локалізації людської руки. Інтерфейс користувача був розроблений за допомогою мови програмування C++ та opencv.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційних систем Windows або Linux.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично, цей метод працює за таким алгоритмом:

- визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.
- для кожної функції визначаються повні річні витрати й кількість робочих часів.

- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.
- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

4.1 Постановка задачі техніко-економічного аналізу

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на персональних комп'ютерах зі стандартним набором компонент;
- забезпечувати високу швидкість обробки великих об'ємів даних у реальному часі;
- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використання його як модуля;
- передбачати мінімальні витрати на впровадження програмного продукту.

4.1.1 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування. Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

F_1 вибір мови програмування;

F_2 інтерфейс користувача;

F_3 вибір камери для роботи.

Кожна з основних функцій може мати декілька варіантів реалізації.

- Функція F1 :

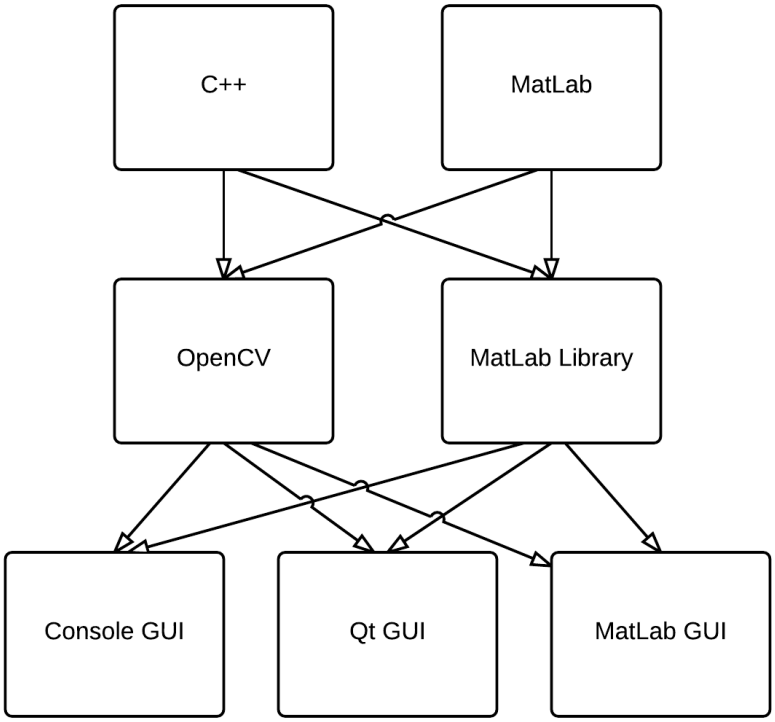

Рисунок 4.1 – Морфологічна карта

а) мова програмування Python;
б) мова програмування C++;

- Функція F2:
  а) інтерфейс користувача, створений на opencv;
  б) інтерфейс користувача як консольний додаток.
- Функція F3:
  а) проста веб камера;
  б) Intel Realsense F200.

4.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (таблиця 4.1).

Морфологічна карта відражає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4.1 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	А	Займає менше часу при написанні коду	Менша швидкість написання коду
	Б	Швидкодійний, кросплатформений	Менша швидкість написання коду
F2	А	Проста розробка	Недостатній функціонал
	Б	Простота інтерфейса	Неможливість динамічної зміни робочих параметрів
F3	А	Доступність	Лише камера RGB
	Б	Потужний SDK, depth камера	Висока ціна

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

4.1.2.1 Функція F_1

Оскільки для нас важлива швидкість роботи, варіант а) має бути відкинутий.

4.1.2.2 Функція F_2

Оскільки для нас важлива простота, варіант а) має бути відкинутий.

4.1.2.3 Функція F_3

Вибір двох різних камер основна ідея проекту і тому вважаємо варіанти а) та b) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

а) $F_{1б} - F_{2б} - F_{3а}$

б) $F_{1б} - F_{2б} - F_{3б}$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.2 Обґрунтування системи параметрів ПП

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

4.2.1 Опис параметрів

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

X_1 - швидкодія мови програмування;

X_2 - об'єм пам'яті для збереження даних;

X_3 - час обробки даних;

X_4 - потенційний об'єм програмного коду.

X_1 відображає швидкодію операцій залежно від обраної мови програмування.

X_2 відображає об'єм оперативної пам'яті персонального комп'ютера, що необхідний для збереження та обробки даних під час виконання програми.

X_3 відображає час, який витрачається на дії.

X_4 показує кількість програмного коду який необхідно створити безпосередньо розробнику.

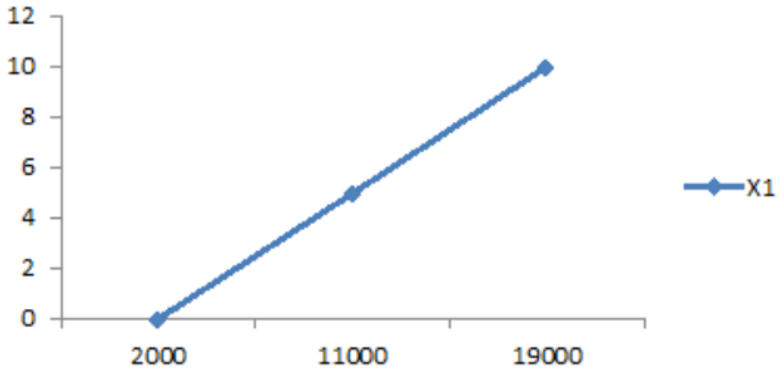

Рисунок 4.2 – Швидкодія мови програмування X_1

4.2.2 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 4.2

Таблиця 4.2 – Основні параметри ПП

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X_1	оп/мс	19000	11000	2000
Об'єм оперативної пам'яті	X_2	мб	32	16	8
Час обробки зображення	X_3	мс	1000	420	60
Об'єм програмного коду	X_4	строк	2000	1500	1000

За даними таблиці 4.2 будуються графічні характеристики параметрів: рис. 4.2 - рис. 4.5

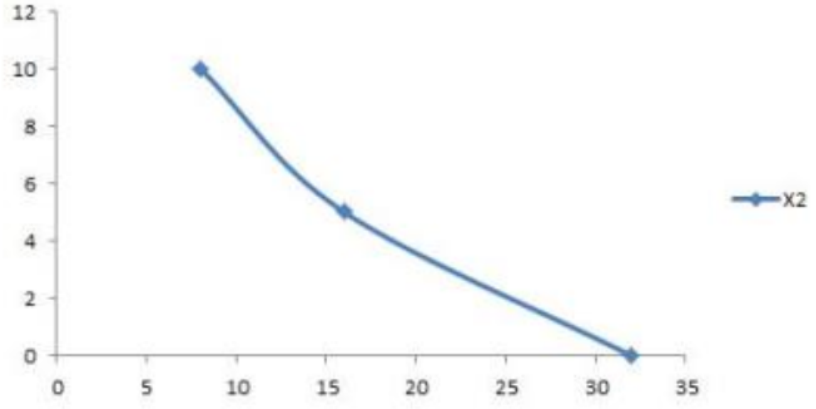

Рисунок 4.3 – Об’єм оперативної пам’яті X_2

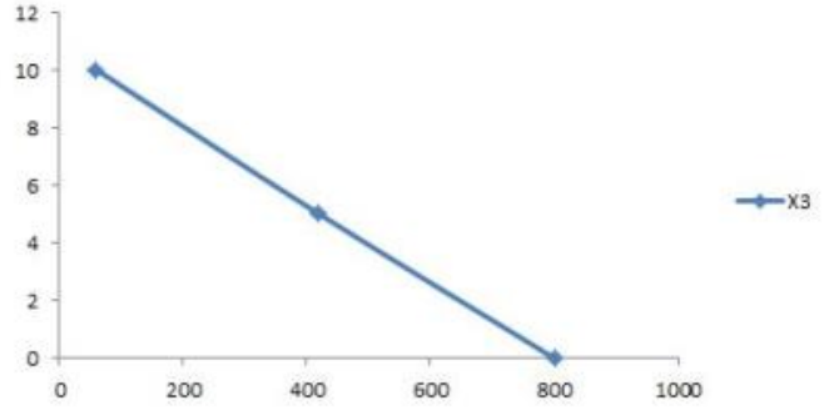

Рисунок 4.4 – Час, який витрачається на дії X_3

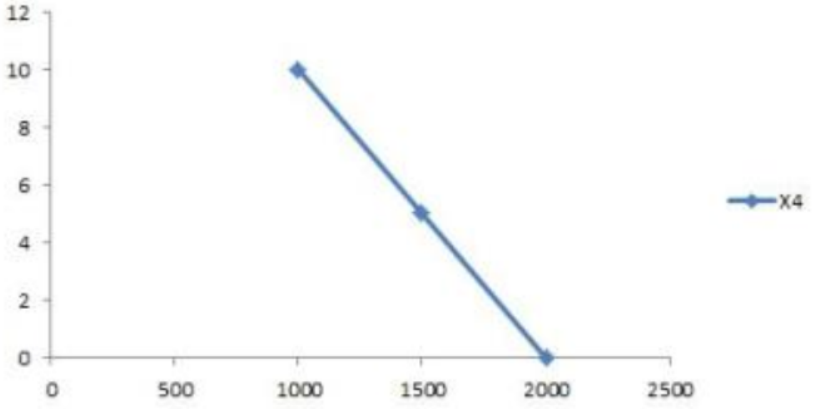

Рисунок 4.5 – Об’єм програмного коду X_4

Таблиця 4.3 – Результати експертного ранжування параметрів

Позначення	Назва	Ранг за оцінкою експерта							Сума рангів R_i	Відх. Δ	Δ^2
		1	2	3	4	5	6	7			
X_1, оп/мс	Швидкодія мови програмування	4	3	4	4	4	4	4	27	0.75	0.56
X_2, мб	Об’єм оперативної пам’яті	4	4	4	3	4	3	3	25	-1.25	1.56
X_3, мс	Час, який витрачається на дії	2	2	1	2	1	2	2	12	-14.25	203.06
X_4, строк	Об’єм програмного коду	5	6	6	6	6	6	6	41	14.75	217.56
Разом		15	15	15	15	15	15	15	105	0	420.75

4.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при оптичному аналізі нотних листів.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R = \sum_{i=1}^{n} R_i = \frac{Nn(n+1)}{2} = 105 \tag{4.1}$$

де n - число оцінюваних параметрів, N - число експертів

б) середня сума рангів:

$$T = \frac{1}{n} \sum_{i,j} R_{ij} = 24.5 \tag{4.2}$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \qquad i = 1 \dots n \tag{4.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta^2 = 35 \tag{4.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12 \cdot S}{N^2(n^3 - n)} \tag{4.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0.67.

Скориставшись результатами ранжування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4

Таблиця 4.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Чисельне значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	>	>	>	>	>	>	>	>	1.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	>	>	>	>	>	>	>	>	1.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Числове значення, що визначає ступінь переваги i–го параметра над j–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5, & X_i > X_j \\ 1.0, & X_i = X_j \\ 0.5, & X_i < X_j \end{cases} \tag{4.6}$$

З отриманих оцінок складемо матрицю $A = ||a_{ij}||$

Для кожного параметра зробимо розрахунок вагомості $K_{\text{в}}^{(i)}$ за наступною формулою:

$$K_{\text{в}}^{(i)} = \frac{b_i}{\sum_{j=1}^{n} b_j} \quad \text{, де} \quad b_i = \sum_{j=1}^{n} a_{ij} \tag{4.7}$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{в}}^{(i)} = \frac{b'_i}{\sum_{j=1}^{n} b'_j} \quad \text{, де} \quad b'_i = \sum_{j=1}^{n} a_{ij} \cdot b_j \tag{4.8}$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Ітерації					1		2		3	
Параметри X_i	X_1	X_2	X_3	X_4	b_i	$K_{\text{в}}^{(i)}$	b_i^1	$K_{\text{в}}^{(i)}$	b_i^2	$K_{\text{в}}^{(i)}$
X_1	1.0	0.5	0.5	1.5	3.5	0.219	22.25	0.216	100	0.215
X_2	1.5	1.0	0.5	1.5	4.5	0.281	27.25	0.282	124.3	0.283
X_3	1.5	1.5	1.0	1.5	5.5	0.344	34.25	0.347	156	0.348
X_4	0.5	0.5	0.5	1.0	2.5	0.156	14.25	0.155	64.75	0.154
Всього:					16	1	98	1	445	1

Таблиця 4.5 – Розрахунок вагомості параметрів

4.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X2(об'єм пам'яті для збереження даних) та X1 (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X3 (час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується за формулою (4.9) Результати наведемо в таблиці 4.6

$$K_K(j) = \sum_{i=1}^{n} K_{\text{в}}^{(i,j)} B^{(i,j)} \tag{4.9}$$

де n - кількість параметрів, $K_{\text{в}}^{(i)}$ - коефіцієнт вагомості i-го параметра, $B^{(i,j)}$ - оцінка i-го параметра в балах.

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
$F_1(X_1)$	А	11000	3.6	0.215	0.774
$F_2(X_2)$	А	16	3.4	0.283	0.962
$F_3(X_3, X_4)$	А	800	2.4	0.348	0.835
	Б	80	1	0.154	0.154

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

За даними з таблиці 4.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + ... + K_{\text{ТУ}}[F_{zk}] \tag{4.10}$$

визначаємо рівень якості кожного з варіантів:

$$\begin{aligned} K_{K1} &= 0.774 + 0.962 + 0.835 = 2.57 \\ K_{K2} &= 0.774 + 0.962 + 0.154 = 1.89 \end{aligned}$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

а) Розробка проекту програмного продукту;

б) Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$\text{Т}_\text{О} = \text{Т}_\text{Р} \cdot \text{К}_\text{П} \cdot \text{К}_\text{СК} \cdot \text{К}_\text{М} \cdot \text{К}_\text{СТ} \cdot \text{К}_\text{СТ.М} \tag{4.11}$$

де $\text{Т}_\text{Р}$ - трудомісткість розробки ПП; $\text{К}_\text{П}$ - поправочний коефіцієнт; $\text{К}_\text{СК}$ - коефіцієнт на складність вхідної інформації; $\text{К}_\text{М}$ - коефіцієнт рівня мови програмування; $\text{К}_\text{СТ}$ - коефіцієнт використання стандартних модулів і прикладних програм; $\text{К}_\text{СТ.М}$ - коефіцієнт стандартного математичного забезпечення.

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $\text{Т}_\text{Р} = 90$ людино-днів. Поправний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $\text{К}_\text{П} = 1.7$. Поправний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх завдань рівний 1: $\text{К}_\text{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо ц за допомогою коефіцієнта $\text{К}_\text{СТ} = 0.8$. Тоді, за формулою (4.11), загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \quad \text{людино-днів}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $\text{Т}_\text{Р} = 27$ людино-днів, $\text{К}_\text{П} = 0.9$, $\text{К}_\text{СК} = 1$, $\text{К}_\text{СТ} = 0.8$:

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \quad \text{людино-днів}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$\begin{aligned} T_1 = &\quad (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 \quad = 1328.64 \text{ людино-годин} \\ T_2 = &\quad (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 \quad = 1345.52 \text{ людино-годин} \end{aligned}$$

Найбільш високу трудомісткість має варіант 2. В розробці беруть участь два програмісти з окладом 7000 грн., один аналітик-програміст з окла-

дом 9500 грн. Визначимо годинну зарплату за формулою:

$$C_h = \frac{M}{T_m t} \text{ грн} \tag{4.12}$$

де – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$C_h = \frac{7000 + 7000 + 9500}{3 \cdot 21 \cdot 8} = 46.62 \text{ грн} \tag{4.13}$$

Тоді, розрахуємо заробітну плату за формулою:

$$\text{С}_{\text{зп}} = C_h \cdot T_i \cdot \text{К}_{\text{КД}} \tag{4.14}$$

де C_h– величина погодинної оплати праці програміста; T_i – трудомісткість відповідного завдання; $\text{К}_{\text{КД}}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

а) $\text{С}_{\text{зп}} = 46.62 \cdot 1328.64 \cdot 1.2 = 74340.57$ грн

б) $\text{С}_{\text{зп}} = 46.62 \cdot 1345.52 \cdot 1.2 = 75285.04$ грн

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

а) $\text{С}_{\text{від}} = \text{С}_{\text{зп}} \cdot 0.22 = 74340.57 \cdot 0.22 = 16354.92$ грн

б) $\text{С}_{\text{від}} = \text{С}_{\text{зп}} \cdot 0.22 = 75285.04 \cdot 0.22 = 16562.70$ грн

Тепер визначимо витрати на оплату однієї машино-години — $\text{С}_{\text{М}}$

Так як одна ЕОМ обслуговує одного програміста з окладом 7000 грн., з коефіцієнтом зайнятості 0.2 то для однієї машини отримаємо:

$$\text{С}_{\text{Г}} = 12 \cdot M \cdot \text{К}_{\text{З}} = 12 \cdot 7000 \cdot 0.2 = 16800 \text{ грн}$$

З урахуванням додаткової заробітної плати:

$$\text{С}_{\text{зп}} = \text{С}_{\text{г}} \cdot (1 + K) = 16800 \cdot (1 + 0.2) = 20160 \text{ грн}$$

Відрахування на єдиний соціальний внесок:

$$С_{від} = С_{зп} \cdot 0.22 = 20160 \cdot 0.22 = 4435.2 \text{ грн}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 20000 грн.

$$С_а = К_{тм} \cdot К_а \cdot Ц_{пр} = 1.15 \cdot 0.25 \cdot 20000 = 5750 \text{ грн}$$

де $К_{тм}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_а$ – річна норма амортизації; $Ц_{пр}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$С_р = К_{тм} \cdot Ц_{пр} \cdot К_р = 1.15 \cdot 20000 \cdot 0.05 = 1150 \text{ грн}$$

де $К_р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$Т_{еф} = (Д_к - Д_в - Д_с - Д_р) \cdot t \cdot К_в = \\ = (365 - 104 - 8 - 16) \cdot 8 \cdot 0.9 = 1706.4 \text{ годин}$$

де $Д_к$ – календарна кількість днів у році; $Д_в, Д_с$ – відповідно кількість вихідних та святкових днів; $Д_р$ – кількість днів планових ремонтів устаткування; t – кількість робочих годин в день; $К_в$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$С_{ел} = Т_{еф} \cdot N_c \cdot К_з \cdot Ц_{ен} = 1706.4 \cdot 0.156 \cdot 0.9733 \cdot 2.018 = 523.82 \text{ грн}$$

де N_c – середньо-споживча потужність приладу; $К_з$ – коефіцієнтом зайнятості приладу; $Ц_{ен}$ – тариф за 1 КВт-годину електроенергії.

Накладні витрати розраховуємо за формулою:

$$С_н = Ц_{пр} \cdot 0.67 = 20000 \cdot 0.67 = 13400 \text{ грн}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{екс} = C_{зп} + C_{від} + C_{а} + C_{р} + C_{ел} + C_{н}$$

$$C_{екс1} = 20160 + 4435.2 + 5750 + 1150 + 523.82 + 13400 = 45419.02 \text{ грн}$$
$$C_{екс2} = 20160 + 4435.2 + 13586 + 1150 + 472.84 + 13400 = 53204.03 \text{ грн}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{м\text{-}г1} = C_{екс}/T_{еф} = 45419.02/1706.4 = 26.61 \text{ грн/год}$$
$$C_{м\text{-}г1} = C_{екс}/T_{еф} = 53204.03/1706.4 = 31.17 \text{ грн/год}$$

В нашому випадку всі роботи пов'язані з розробкою програмного продукту ведуться на ЕОМ. Витрати на оплату машинного часу розраховуються за наступною формулою:

$$C_{м} = C_{м\text{-}г} \cdot T$$

В залежності від обраного варіанта реалізації, витрати на оплату машинного часу складають:

а) $C_{м} = 26.61 \cdot 1328.64 = 35355.11$ грн
б) $C_{м} = 31.17 \cdot 1345.52 = 41939.85$ грн

Накладні витрати складають 67% від заробітної плати:

$$C_{н} = C_{зп} \cdot 0.67$$

а) $C_{н} = 74340.57 \cdot 0.67 = 49808.18$ грн
б) $C_{н} = 75285.04 \cdot 0.67 = 50440.98$ грн

Отже, вартість розробки ПП за варіантами становить:

$$C_{пп} = C_{зп} + C_{від} + C_{м} + C_{н}$$

а) $C_{пп} = 74340.57 + 27335.02 + 35355.11 + 49808.18 = 186838.88$ грн
б) $C_{пп} = 75285.04 + 27682.31 + 41939.85 + 50440.98 = 195348.18$ грн

4.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{терj} = K_{Кj} / C_{Фj} \tag{4.15}$$

а) $K_{тер1} = 2.57/186838.88 = 1.37 \cdot 10^{-5}$;
б) $K_{тер2} = 1.89/195348.18 = 0.96 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{тер1} = 1.37 \cdot 10^{-5}$.

4.6 Висновки до розділу

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{тер} = 0.16 \cdot 10^{-4}$.

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – C++;
- інтерфейс користувача у консольному варіанті;
- камера RGB;

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал та швидкодію.

СПИСОК ЛІТЕРАТУРИ

1. Jian-Hua Zheng Chong-Yang Hao Y.-Y. F., Zang X.-Y. Adaptive skin detection under unconstrained lightning conditions using a bigaussian model and illumination estimation // Image Anal Stereol. 2005.
2. Eric Kandel James Schwartz T. J. Principles of Neural Science. 4 edition. McGraw-Hill, New York, 2000. P. 577–80.