Національний технічний університет України
«Київський політехнічний інститут»

Факультет (інститут) ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти Перший(Бакалаврський)
(перший (бакалаврський), другий (магістерський) або спеціаліста)

Спеціальність 7.05010102, 8.05010102 Інформаційні технології проектування
7.05010103, 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2016 р.

ЗАВДАННЯ
на дипломний проект (роботу) студенту

Козинському Євгену Олександровичу
(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Програма розпізнавання цифр в мові жестів,

керівник проекту (роботи) к.т.н., доц. каф. СП ІПСА Бритов О.А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від 12 травня 2016 р. № 50-ст

2. Строк подання студентом проекту (роботи) 09.06.2016

3. Вихідні дані до проекту (роботи):

Форма реалізації – у вигляді програми на мові Python. Можливе використання додаткових бібліотек для роботи із зображеннями.

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити)

1. Розглянути можливі варіанти реалізації пошуку жестів на зображенні.
2. Вибрати алгоритми, які будуть використані при розпізнаванні жестів.
3. Розробити план тестування ядра еволюційних обчислень.
4. Розробити алгоритм знаходження рук на зображенні.

5. Розробити алгоритм розпізнавання жестів.
6. Виконати програмну реалізацію та провести тестування.
7. Проаналізувати роботу програмного продукту на наборі тестових зображень.
8. Виконати економічний аналіз програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів тощо)

1. Логотип OpenCV + Python– плакат.
2. Порівняння зображень за допомогою контрольних точок – плакат.
3. Співставлення двух контурів - плакат.

6. Консультанти розділів проекту (роботи)*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічна частина			

7. Дата видачі завдання 01.02.2016

Календарний план

№ з/п	Назва етапів виконання дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Отримання завдання	01.02.2016	
2	Збір інформації	15.02.2016	
3	Вивчення варіантів реалізації та вибір варіанту для розробки	28.02.2016	
4	Аналіз методів філтраціх	10.03.2016	
5	Розробка агоритму пошуку рук	12.03.2016	
6	Розробка алгоритма розпізнавання жестів	14.03.2016	
7	Виконання програмної реалізації	15.03.2016	
8	Оптимізація рішення	01.04.2016	
9	Аналіз роботи програмного продукту	01.05.2016	
10	Аналіз економічної обґрунтованості	15.05.2016	
11	Оформлення дипломної роботи	31.05.2016	
12	Отримання допуску до захисту та подача роботи в ДЕК	09.06.2016	

Студент _____ Є.О.Козинський
(підпис) (ініціали, прізвище)

Керівник проекту (роботи) _____ О.А.Бритов
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту (роботи).

АНОТАЦІЯ

бакалаврської дипломної роботи Козинського Євгена Олександровича на тему: “Програма розпізнавання цифр в мові жестів”

Дана дипломна робота присвячена розробці програмного додатку, що здатний розпізнавати жести людини на відео.

Приведений огляд існуючих методів аналізу відеозаписів та алгоритмів комп’ютерного зору. Було створено програмне забезпечення, що здатне розпізнавати цифри 1-5 показані людиною за допомогою мови жестів. Даний програмний додаток створений для демонстрації використання існуючих методів обробки зображення.

Загальний обсяг роботи: 73 сторінок, 1 додаток на 9 сторінок, 24 рисунка, 10 таблиць, 21 посилання.

Ключові слова: комп’ютерний зір, мова жестів, аналіз рухів, порівняння зображень, контрольні точки, контурний аналіз.

АННОТАЦИЯ

Бакалаврской дипломной работы Козинского Евгений Александровича на тему: «Программа распознавания цифр в языке жестов»

Данная дипломная работа посвящена разработке программного приложения, которое способно распознавать жесты человека на видео.

Приведён осмотр существующих методов анализа видеозаписей и алгоритмов компьютерного зрения. Было создано програмное обеспечение, которое способно распознавать цифры 1-5 показнные с помощью языка жестов. Данное програмное приложение создано для демонстрании использования сущестующих методов обработки изображения.

Общий объем работы: 73 страниц, 1 приложение на 9 страниц, 24 рисунков, 10 таблиц, 21 библиографических наименований.

Ключевые слова: компьютерное зрение, язык жестов, анализ движений, сравнение изображений, контрольные точки, контурный анализ.

ANNOTATION

a bachelor’s degree work of Eugene Kozynskyi

entitled "Algorithms and Software Tools for Image Processing Based on Genetic Optimization Algorithms"

This project is devoted to the development of software tool for sign language recognition. The aim is to develop an video-processing tool based on image processing algorithms.

The project covers the following: image filtration, motion detection, object recognition; developed algorithm for keypoint matching. The methods of handling the image data and their procs and cons from video camera were analyzed. Localization algorithm was implemented for the system to localize hands on video. Featrue matching algorithms were implemeted for the system to recognize objects.

Total volume of work: 73 pages, 1 appendix for 9 pages, 24 figures, 10 tables, 21 bibliographic titles.

Keywords: computer vision, sign language, motion analysis, image comparation, control points, contour analysis.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КБ – комп’ютерне бачення

CV – Computer Vision

ПЗ – програмне забезпечення

SIFT - Scale Invariant Feature Transform

МЖ – мова жестів

ВСТУП

Комфортна комунікація між людьми, незалежно від їх фізичних вад, повинна забезпечуватися усіма можливими способами. Саме цьому питання забезпечення комунікації є дуже актуальним для будь-якого суспільства. Це питання можна вирішувати на багатьох рівнях, один з яких є рівень інформаційних технологій. Більшість людей сьогодні мають принаймні один комп’ютер та смартфон. Можливість використання даних пристроїв для пом’якшення бар’єру між людьми наштовхує на необхідність дослідження сучасних методів та алгоритмів аналізу візуальної інформації.

Мова жестів — це вид спеціального письма, який дає змогу позначати цілі слова, а також літери алфавіту певними жестами. Мову жестів для обміну інформацією використовують як люди з вадами голосових зв'язок та слуху, так і люди без таких вад. Мова жестів за своїми можливостями не поступаються звуковим мовам, хоча соціально мають нижчий статусу. Інформація у мові жестів передається шляхом відтворення руками спеціальних символів та їх послідовностей.

Аналізом та розпізнаванням образів у зображеннях займається теорія комп’ютерного зору. Вона вважається достатню молодою, багато її методів знаходяться ще на стадії дослідження. Робота із зображеннями займає ключову роль у створенні людиноподібних систем штучного інтелекту.

Зчитування жестів включає в себе такі проблеми, як виділення людини із зображення, виділення її обличчя, тіла, рук. Аналіз положення частин тіла одне відносно іншої, аналіз жестів у часі тощо.

Методи локалізації направлені на знаходження інформативних частин зображення у той час, як методи розпізнавання використовуються для класифікації та подальшого аналізу даних.

Для того, щоб розробити додаток для розпізнавання жестів, треба програмно відповісти на наступні питання:

1. Де знаходяться руки на зображенні;
2. Що показують руки на зображенні.

Мета цієї роботи – дослідити сучасні методи аналізу та обробки відео даних та використати придбанні знання при розробці програмного додатку

Для досягнення поставленої мети необхідно:

- зробити аналіз існуючих методів обробки зображення;
- серед множини доступних рішень вибрати ті методи, які будуть досліджуватися протягом дипломної роботи;
- розробити програму, що використовує вибрані методи;

зробити аналіз готової програми, її недоліків та можливих варіантів покращення.

Необхідно зауважити, що алгоритми направлені на зчитування тіла людини із зображення можуть бути використані у надзвичайно різних сферах життя, починаючи із сфер розваги та закінчуючи військовим промислом.

1. АНАЛІЗ СУЧАСНИХ РІШЕНЬ ТА ІСНУЮЧИХ АЛГОРИТМІВ

1.1 Сучасні рішення

1.1.1 Microsoft Kinect Sign Language Translator

Рисунок 1.1 – головне вікно програми Sign Language Translator [1]

Опис:

Проект знаходиться на етапі дослідження та експериментів. Роботи проводяться в основному в Китаї. Інформації за останній рік знайдено не було.

Переваги:

- Переклад як з МЖ в звичайну мову так и навпаки;
- Підтримка розмовного перекладу.

Недоліки:

- Необхідність у використанні пристроя Microsoft Kinnect;

- Непортативне рішення.

1.1.2 Motionsavvy

Рисунок 1.2 – Рекламний плакат компанії Motionsavvy[2]

Опис:

Компанія створила програмне забезпечення, котре дозволяє розпізнавати жести за допомогою комбінації планшету та ІК-пристрою MotionLeap.

Переваги:

- Переклад як з МЖ в звичайну мову так и навпаки;
- Портативність.

Недоліки:

- Необхідність у використанні пристрою MotionLeap;

- MotionLeap має відомі проблеми зі зчитуванням рук, що перетинаються. Можливо ця проблема існує у даному програмному забезпеченні.

1.2 Методи зчитування даних.

Основні пристрої зчитування даних для розпізнавання жестів:

1. Електронні рукавички. Даний пристрій може зчитати інформацію про позицію та кут повороту рук за допомогою магнітних та інерціальних датчиків.
2. Depth-aware cameras. Певні види камер дають змогу отримати карту глибин об’єкту спостереження. За допомогою цієї карти є можливість відтворити його тривимірну репрезентацію.
3. Сереокамери. Використовуючи дві камери, положення яких одна відносно іншої відомо заздалегідь, можна отримати тривимірне зображення об’єкту спостереження.
4. ІК-пристрої. Деякі пристрої можуть відтворити 3д модель пристрою за допомогою інфочервоних датчиків.

Одиночна камера. Звичайна 2д камера може бути використана для розпізнавання жестів. Вважається, що даний метод отримання інформації нене являється настільки ж ефективним як інші, але він являється найбільш доступним для людей.

1.3 Методи аналізу даних

1.3.1 Фільтрація.

До цієї групи входять методи, які дають змогу виділити на зображеннях необхідні області без їх аналізу. Більша часина цих методів викунує якесь єдине перетворення до всіх точок зображення. На моменті фільтрації аналіз зображення не робиться, але точки, котрі проходять фільтрацію, можна розглядати як області із особливими характерностями.

1.3.1.1 Бінарізація за порогом.

Найбільш просте та розпоширине перетворення – це бінарізація зображення за порогом. Для RGB зображення та зображення у відтінках сірого порогом є значення кольору.

Рисунок 1.3 – Демонстрація бінарізації за порогом. [3]

1.3.1.2 Вейвлети

Вейвлет-аналізом зображення називається пошук довільного патерна на зображенні за допомогою згортки з моделлю цього патерна. Існує набір класичних функцій, які використовуються у вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морле, вейвлет мексиканська шляпа, тощо.[15 16 17 18]

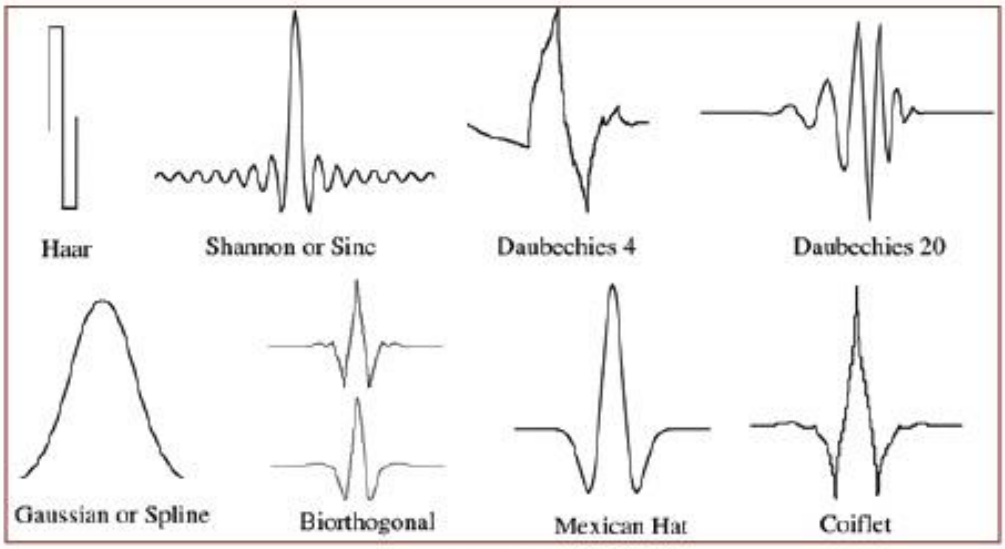


Figure 8

Examples of types of wavelets

Рисунок 1.4 – різновиди вейвлетів[4]

1.3.1.3 Кореляція

Отримання інформації про різність декількох зображень

Рисунок 1.5 – Приклад кореляції зображень [5]

1.3.1.4 Контури

Знаходження контурів об’єктів є ефективним методом переходу від роботи над зображенням до роботи над об’єктом зображення. У випадку, коли об’єкт

має складну геометричну форму, але гарно виділяється на фоні, то, частіш за все, виділення контурів повністю вирішує задачу фільтрації

Існують наступні методи, які вирішують задачу фільтрації контурів:

- Оператор Кенні;
- Оператор Соблея;
- Оператор Лапласа;
- Оператор Прюітт;
- Оператор Робертса.

У більшості випадків задачу розпізнавання контурів можна вирішити саме оператором Кенні. [19 20 21]

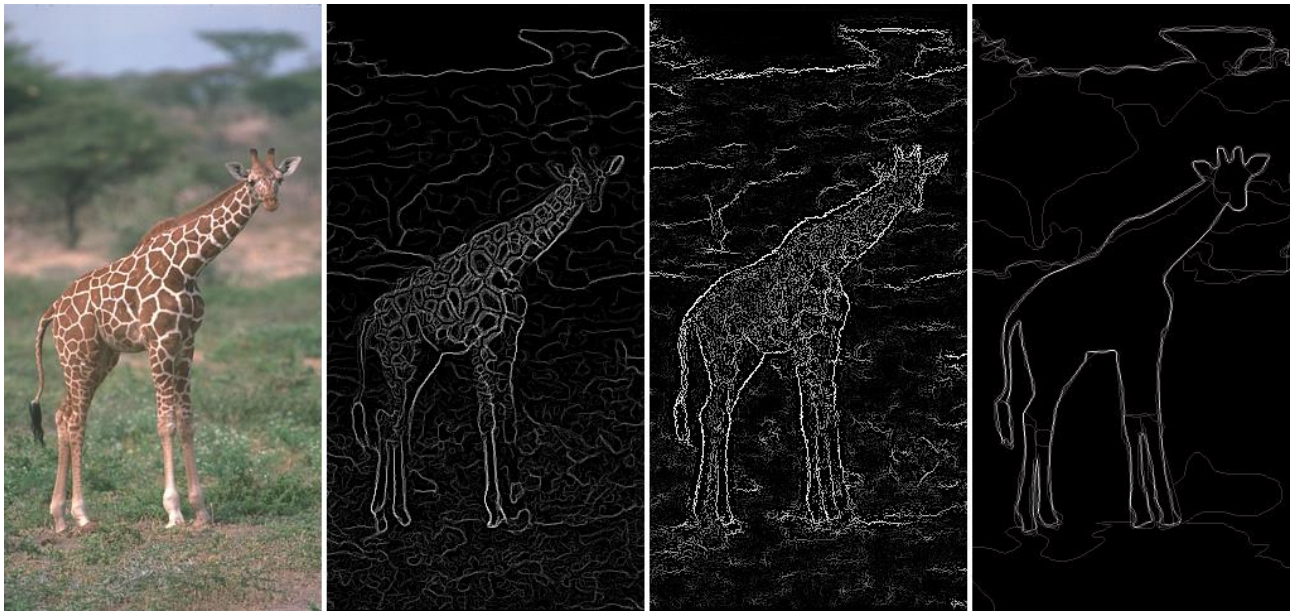


Рисунок 1.6 – приклад алгоритмів розпізнавання контурів [6]

1.3.2 Логічна обробка результатів фільтрації

Результатом фільтрації є набір придатних до обробки даних. Але часто їх важко використовувати без попередньої обробки.

1.3.2.1 Контурний аналіз

Задачею контурного аналізу є отримання контуру зображення із його границь. Контур є унікальною характеристикою об’єкта, що дозволяє вирішити задачу класифікації.

1.3.2.2 Особливі точки

Особливі точки – це унікальні характеристики об’єкта, які дозволяють співставити об’єкт із самим собою або з об’єктами схожого класу. Існують багато способів, які дозволяють виділити такі точки. Деякі з них виділяють ці точки через сусідні кадри, а деякі через великий проміжок часу та зміну освітлення. Деякі алгоритми дозволяють знайти особливі точки, які залишаються такими навіть при поворотах об’єкта. Види особливих точок:

Особливі точки, які залишаються стабільними протягом секунд

Дані точки слугують для супроводження об’єкта у сусідніх кадрах відео, або для склеювання зображення з декількох камер. До таких точок можна віднести локальні максимуми зображення, кути, точки з максимумом дисперсії, визначені градієнти тощо.

Особливі точки, які залишаються стабільними при зміні освітлення та невеликих рухах об’єкта

Дані точки слугують для навчання з подальшою класифікацією типів об’єктів. Наприклад класифікатор обличчя – це продукт системи, створеної саме на таких точках.

Стабільні точки

Дані точки залишаються стабільними навіть при повороті зображення. Алгоритми знаходження таких точок набагато складніше, їх менше, та зачасту, вони запатентовані.

1.3.3 Навчання

Після локалізації об’єкта дослідження та виділення стабільних точок або контурів використовуються методи, які дозволяють приймати рішення. Описати цей процес можна наступним чином:

Знаходиться тестова вибірка, яка має декілька классів об’єктів. Для кожного зображення є набір ознак, котрі були виділені за допомогою деякого алгоритму.

Алгоритм навчання повинен створити таку модель, за допомогою яких він зможе проаналізувати нове зображення та зробити висновок, який із об’єктів знаходиться на зображенні.

До алгоритмів навчання відносяться такі алгоритми, як нейроні мережі, регресії, k-means кластеризатори, AdaBoost класифікатори, тощо.

1.4 Існуючі алгоритми

1.4.1 Локалізація жестів

Першим алгоритмом виділення рук на зображенні було обрано фільтрацію по кольору шкіри. Даний метод полягає в тому, щоб виділити пікселі кольору шкіри із зображення. Дані про колір шкіри можна отримати із кольору обличчя. Результатами дослідження були незадовільні, так як усі тестові данні були у схожих до шкіри тонах.

До стандартних каскадів Хаара також входять класифікатори долоні, кулака, руки тощо, але вони працюють набагато гірше аніж відповідний класифікатор обличчя. Зауважимо, що каскади Хаара дуже чуттєві до кута нахилу зображення, а отже, за допомогою них неможливо зробити класифікатор рук у всіх можливих положеннях.

1.4.1.1 Виділення рук на основі аналізу рухів

Для подальших досліджень було зроблено припущення, що мова жестів за своєю природою є рухом, та саме області руху є найбільш інформативними для аналізу відеоряду.

1.4.1.2 Оптичний потік та лінії Фарнебеку.

Є багато методів аналізу рухів на зображенні. Найбільш інформативний, та складний з точки зору обчислення є підрахунок оптичного потоку кадрів. Основною ідеєю цього методу є обчислення зсуву пікселей у сусідніх зображеннях.

Рисунок 1.7 – приклад знаходження оптичного потоку за допомогою алгоритму Фарнебека[7]

1.4.2 Розпізнавання жестів

1.4.2.1 Метод гістограм кольорів

Ідея методу гістограм кольорів для порівняння зображень зводиться до наступного. Уся множина кольорів розбивається на набір непересічних, повністю охоплюючих зображення підмножин. Для зображення формується

гістограма, що відображає частку кожної підмножини кольорів в кольоровій гамі зображення. Для порівняння гістограм вводиться поняття відстані між ними.

При розбитті RGB-кольорів за яскравістю обчислюється інтенсивність кожного кольору. Отримане значення, вкладене в проміжок між 0 і 255, потрапляє в один з 16 інтервалів, на які розбивається діапазон можливих значень. Як відстань між гістограмами використовується сума модулів різниці відповідних елементів гістограм; деякий удосконалення методу досягається при обчислюванні відстані на підставі поелементного порівняння гістограм з урахуванням сусідніх елементів. Цей метод найбільш ефективний для чорно-білих зображень.

Для кольорових RGB-зображень кращі результати дає інший спосіб - розбиття RGB-кольорів на прямокутні паралелепіпеди. Кольорове RGB-простір розглядається як тривимірний куб, кожна вісь якого відповідає одному з трьох основних кольорів (червоний, зелений або синій. При такому розгляді будь-який колір RGB-зображення може бути представлений точкою куба. Для побудови гістограми кольорів кожна сторона ділиться на 4 рівних інтервалу, відповідно RGB-куб ділиться на 64 прямокутних паралелепіпеда. Гістограма зображення відображає розподіл точок RGB-простору, відповідних кольорам пікселів зображення. Як відстань між гістограмами використовується покомпонентна сума модулів різниці між ними. Незважаючи на граничну простоту підходу, він показує досить стабільні результати. [11]

Основним недоліком методу колірних гістограм є те, що він втрачає інформацію про просторове розташування об'єктів. Абсолютно різні картинки можуть мати подібні гістограми кольорів. Наприклад, зображення осіннього листя може містити багато невеликих плям червоного кольору. Це дасть подібну колірну гістограму із зображенням великого червоного об'єкта.[12]

1.4.2.2 Метод аналізу кореляцій

Традиційна техніка порівняння поточного зображення з еталоном ґрунтується на розгляді зображень як двовимірних функцій яскравості (дискретних двовимірних матриць інтенсивності). При цьому вимірюється або відстань між зображеннями, або міра їх близькості.[8]

Як правило, для обчислення відстаней між зображеннями використовується співвідношення

$$\rho\,(f,g) = [\Sigma\langle x,y\rangle \in X \mid f\,(x,y) - g\,(x,y)|\,\alpha]\hat{}\left(\frac{1}{\alpha}\right) \quad (1)$$

де $f\,(x,y)$, $g(x,y)$ - функції інтенсивності;
X - поле зору.

Величина α ∈ [1, ∞) визначає характеристики використовуваної метрики. Очевидно, що при α = 2 цей вираз описує звичайне евклідова відстань між зображеннями, які розглядаються як вектори, що належать простору L2 (x, y) на поле зору X функцій інтенсивностей з інтегрованим квадратом.

Нехай дано n еталонних зображень $\{fi\}$, i = 1, ..., n, кожне з яких відповідає i-го класу. Віднесення знову висунутого фрагмента зображення g до деякого класу j може здійснюватися, наприклад, за методом мінімальної відстані до відповідного еталона:

$$j \;=\; argmini\rho\,(g,fi) \quad (2)$$

Цей найпростіший метод має два основних недоліки:

- Критерій виявлення залежить від лінійних розмірів зразка і зображення.
- Критерій виявлення не інваріантний навіть до найпростіших фотографічним перетворенням яскравості виду $f' = af + b$

Більш прийнятним тому є використання кореляційної метрики, а саме, нормованого коефіцієнта кореляції,

$$K(f,g) = \frac{\sum_{\langle x,y\rangle \in X}(f(x,y)-f_0)(g(x,y)-g_0)}{\sqrt{\sum_{\langle x,y\rangle \in X}(f(x,y)-f_0)^2} * \sqrt{\sum_{\langle x,y\rangle \in X}(g(x,y)-g_0)^2)}} \quad (3)$$

де $f\ (x,y)$, $g(x,y)$ - функції інтенсивності;

Нормований коефіцієнт кореляції має наступні добре відомими властивостями:

1. $-1 \le K\ (f,g)\ \le 1, \forall f,g$
2. $(K\ (f,g)\ =\ 1)\ \Leftrightarrow\ (g\ =\ af\ +\ b, a >\ 0, \forall b)$
3. $(K\ (f,g)\ =\ -1)\ \Leftrightarrow\ (g\ =\ af\ +\ b, a\ <0, \forall b)$

Останню властивість, як правило, називають "зворотним контрастом". Нехай, як і раніше, дано n еталонних зображень $\{fi\}$, i = 1, ..., n, кожне з яких відповідає i-го класу. Виявлення фрагмента зображення g за методом максимальної кореляційної зв'язку здійснюється тоді за правилом

$$j\ =\ argmaxiK\ (g, fi) \quad (4)$$

Після цього на підставі отриманого значення максимальної кореляції може перевірятися достовірність детектування. Якщо $K\ (g, fi)\ \ge Kmin$, То виявлення визнається достовірним. В іншому випадку об'єкт вважається нерозпізнаним. Теоретичним обґрунтуванням застосування кореляційного методу виявлення є його сувора оптимальність для виявлення детермінованого сигналу в білому шумі з гаусовим розподілом яскравості. [8]

Важливі недоліки кореляційних методів виявлення проявляються в присутності радіометричних і особливо геометричних спотворень поточного зображення в порівнянні з еталонним. Зокрема, спостерігається швидке зменшення кореляційної зв'язку при так званих ракурсних викривлення, наприклад, при поворотах зображень. Присутність спотворень типу "warping" зазвичай вже не дозволяє використовувати кореляційні методи виявлення. У той же час, для цього класу алгоритмів запропоновано велику кількість процедур, що

дозволяють або підвищити їх працездатність, або значно прискорити процес пошуку. На цьому шляху були розроблені ієрархічні кореляційні алгоритми, які зберегли актуальність і на поточний момент. Конструктивна ідея зменшення часу пошуку лежить в основі методу амплітудного ранжирування. Відповідно до цього методу належить аналізувати швидкість росту кореляції в міру обробки поля зору і, якщо ця швидкість недостатня, припиняти обробку поточного фрагмента, переходячи до наступного. Для того щоб домогтися інваріантності кореляційних алгоритмів хоча б до афінних перетворень, були випробувані різні перетворення зображень, наприклад перетворення Мелліна. На жаль, в силу недостатньої стійкості кореляційних алгоритмів до можливих спотворень, вони не знаходять широкого застосування при конструюванні алгоритмів виявлення складно структурованих об'єктів.[8]

Окремий напрямок, що використовує кореляційні методи виявлення, це створення когерентних пристроїв обробки - оптичних корреляторов. Дослідження в цьому напрямку активно тривають, так як навіть незважаючи на значні конструктивні труднощі боротьби з мінливістю еталонів, що реалізується тут фантастична швидкість обробки (швидкість світла) привертає до себе пильну увагу. [9]

1.4.2.3 Метод порівняння контурів

Задача розпізнавання зображення за допомогою порівняння контурів зводиться до двох підзадач:

1. Виділення контурів з зображення.
2. Власне розпізнавання.

Розпізнавання конутрів за допомогою hu-моментів зображення.

Порівняння двух контурів можна звести до порявняня їх моментів.

Момент – характерна риса контуру, об’єднана(просумована) з усіма пікселами контуру. Момент(p,q) визначається як:

$$m_{p,q} = \sum_{i=1}^{n} I(x,y) x^p y^q \tag{5}$$

Де p – порядок x, q – порядок y.

Централні моменти можна визначити наступною формулою:

$$\mu_{p,q} = \sum_{i=0}^{n} I(x,y)(x - x_{avg})^p (y - y_{avg})^q \tag{6}$$

Де Xavg, Yavg – центроїди конутру.

Одак, моменти, що дає попередня формула мають суттєві недолики:

- Вони не дозволяють порівняти контури однакової форми, але різного масштабу;
- Вони залежать від системи координат, а значить, не дозволяють визначити перегорнену фігуру.

Описані вище недоліки вирішуються за допомогою нормалізованих інваріантних моментів.

Нормалізовані моменти рахуються за наступною формулою:

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{00}^{(p+q)/2+1}} \tag{6}$$

Hu-інваріантні моменти – це лінійна комбінація центральних моментів. Їх ідея полягає у тому, що комбінує різні центральні моменти можна створити інваріантні уявлення контурів не залежних від масштабу, повороту та відображення.

- hu[0]=η20+η02
- hu[1]=(η20−η02)2+4η211
- hu[2]=(η30−3η12)2+(3η21−η03)2
- hu[3]=(η30+η12)2+(η21+η03)2
- hu[4]=(η30−3η12)(η30+η12)[(η30+η12)2−3(η21+η03)2]+(3η21−η03)(η21+η03)[3(η30+η12)2−(η21+η03)2]
- hu[5]=(η20−η02)[(η30+η12)2−(η21+η03)2]+4η11(η30+η12)(η21+η03)
- hu[6]=(3η21−η03)(η21+η03)[3(η30+η12)2−(η21+η03)2]−(η30−3η12)(η21+η03)[3(η30+η12)2−(η21+η03)2]

де ηXY – нормалізовані моменти зображення.

1.4.2.4 Метод особливих точок

Визначення ознаки зображення

Не існує універсального чи точного визначення, що собою являє ознака, і точне визначення часто залежить від задачі або типу застосування. Враховуючи це, ознака визначається як «цікава» частина зображення, і ознаки використовуються як відправні точки для багатьох алгоритмів комп'ютерного зору. Оскільки ознаки використовуються як відправні точки та основні примітиви для наступних алгоритмів, загальний алгоритм часто буде лише настільки добрим, наскільки добрим є його детектор ознак. Отже, бажаною властивістю детектора ознак є повторюваність: чи буде одну й ту ж ознаку виявлено на двох або більше різних зображеннях однієї й тієї ж сцени, чи ні.

Виявлення ознак є низькорівневою операцією обробки зображень. А саме, вона зазвичай виконується як перша операція на зображенні, і перевіряє кожен піксель, щоби побачити, чи присутня ознака в цьому пікселі. Якщо це є частиною більшого алгоритму, то цей алгоритм зазвичай перевірятиме зображення лише в областях ознак. Як вбудована передумова для виявлення ознак, вхідне

зображення зазвичай згладжується ґаусовим ядром у масштабно-просторовому представленні, й обчислюється одне або декілька зображень ознак, часто виражених в термінах операцій локальних похідних зображень.[13]

Іноді, коли виявлення ознак є обчислювально витратним, і присутні часові обмеження, може застосовуватися алгоритм вищого рівня для скеровування етапу виявлення ознак, так що пошук ознак здійснюватиметься лише деякими частинами зображення.

Багато алгоритмів комп'ютерного зору використовують виявлення ознак в якості першого кроку, так що в результаті було розроблено дуже велику кількість детекторів ознак. Вони сильно різняться за типами ознак, що виявляють, за обчислювальною складністю та повторюваністю.

Особливі точки

Терміни кути та особливі точки використовуються часом як взаємозамінні, й відносяться до точкових ознак зображення, які мають локальну двовимірну структуру. Термін «кут» виник тому, що перші алгоритми спочатку виконували виявлення контурів, а потім аналізували контури для знаходження швидких змін у напрямку (кутів). Ці алгоритми згодом розвинулися до того, що явне виявлення контурів стало більше не потрібним, наприклад, при аналізі високих значень кривини градієнту зображення. Потім було відмічено, що так звані кути також виявлялися на тих частинах зображення, що не є кутами в традиційному розумінні (наприклад, можуть виявлятися невеликі яскраві плями на темному тлі). Ці точки часто називаються особливими точками, але термін «кут» може використовуватись як традиція.

SIFT

SIFT(Scale Invariant Feature Transform) – є одним з найпоширеніших та найточніших алгоритмів знаходження та опису особливих точок. Особлива точка у SIFT є ділянка зображення (ключова точка) з пов’язаним до неї описом.

Ключові точки знаходяться за допомогою SIFT-детектора. Їх опис розраховує SIFT-дескриптор. [9]

SIFT-детектор

SIFT-ключова точка – це кругла область зображення з орієнтацією. Вона описується чотирьома параметрами: координатами центра точки x та y, масштабом (радіусом точки) та орієнтацією. Основною перевагою SIFT-ключивих точок є їх стійкість до геометричних перетворень, а саме перетворень зсуву, повороту чи масштабуваню. [14]

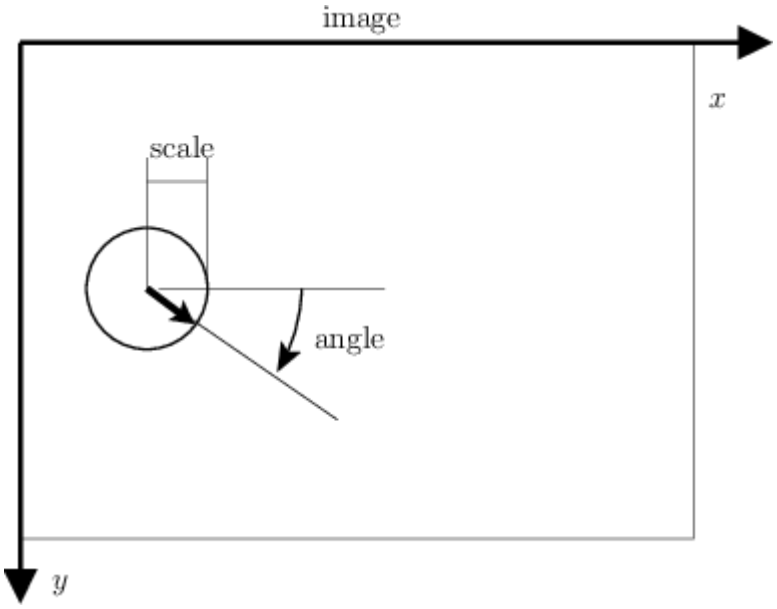


Рисунок 1.8 – графічне зображення опису SIFT-ключової точки[10]

SIFT-дескриптор

Дескриптор SIFT є 3-D просторова гістограма градієнтів. Градієнт в кожному пікселі розглядається як зразок тривимірного елементарного вектора ознак, утвореного положенням піксела та орієнтацією градієнта. Зразки нормуються за допомогою норм градієнта, що формує SIFT дескриптор регіону.

1.5 Засоби реалізації.

1.5.1 Matlab

MATLAB — пакет прикладних програм для числового аналізу, а також мова програмування, що використовується в даному пакеті. Система створена компанією The MathWorks і є зручним засобом для роботи з математичними матрицями, малювання функцій, роботи з алгоритмами, створення робочих оболонок (user interfaces) з програмами в інших мовах програмування. Хоча цей продукт спеціалізується на чисельному обчисленні, спеціальні інструментальні засоби працюють з програмним забезпеченням Maple, що робить його повноцінною системою для роботи з алгеброю.

MATLAB має більше, ніж мільйон користувачів на виробництвах і науковців. Ціна базової комерційної версії без інструментів близько 2000 дол. США і лише 100 дол. США для навчальних закладів з мінімальним набором інструментів.

Застосування:

- MATLAB надає користувачеві велику кількість функцій для аналізу даних, які покривають майже всі області математики, зокрема:
- Матриці та лінійна алгебра — алгебра матриць, лінійні рівняння, власні значення і вектори, сингулярності, факторизація матриць та інше.
- Многочлени та інтерполяція — корені многочленів, операції над многочленами та їх диференціювання, інтерполяція та екстраполяція кривих…
- Математична статистика та аналіз даних — статистичні функції, статистична регресія, цифрова фільтрація, швидке перетворення Фур’є та інші.

- Обробка даних — набір спеціальних функцій, включаючи побудову графіків, оптимізацію, пошук нулів, чисельне інтегрування та інше.
- Диференційні рівняння — вирішення диференційних і диференційно-алгебраїчних рівнянь, диференційних рівнянь із запізнюванням, рівнянь з обмеженнями, рівнянь в часткових похідних та інше.
- Розріджені матриці — спеціальний клас даних пакету MATLAB, що використовується у спеціалізованих додатках.
- Цілочисельна арифметика — виконання операцій цілочисельної арифметики в середовищі MATLAB.

1.5.2 OpenCV

OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим кодом) — бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки 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.

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

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

- Аналіз та обробка зображень
- Системи з розпізнавання обличчя
- Ідентифікації об'єктів
- Розпізнавання жестів на відео
- Відстежування переміщення камери
- Побудова 3D моделей об'єктів
- Створення 3D хмар точок зі стерео камер
- Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю
- Система взаємодії людини з комп'ютером
- Пошуку схожих зображень із бази даних
- Усування ефекту червоних очей при фотозйомці зі спалахом
- Стеження за рухом очей
- Аналіз руху
- Ідентифікація об'єктів
- Сегментація зображення
- Трекінґ відео
- Розпізнавання елементів сцени і додавання маркерів для створення доповненої реальності
- та інші.

1.5.3 Python

Python (рекомендоване прочитання — «Па́йтон», запозичено назвуз британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному

використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується декілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Переваги:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);
- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане на мові Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор).

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ.

Інтерпретатор мови Python і багата стандартна бібліотека (як вихідні тексти, так і бінарні дистрибутиви для всіх основних операційних систем) можуть бути отримані з сайту Python www.python.org, і можуть вільно розповсюджуватися. Цей самий сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

1.6 Вимоги до кінцевого програмного додатку

1. Програмний додаток повинен розпізнавати цифри від 1 до 5 зображені російською мовою жестів у потоковому відео.
2. Додаток повинен вміти працювати як із збереженими даними, так і з даними, отриманими із веб-камери комп’ютера у реальному часі.
3. Додаток має відображати відео-зображення та після кожного жесту виводити текстовий переклад у верхній частині екрану.
4. Після відтворення жесту, не пізніше ніж через 2 секунди після його завершення, переклад повинен з’явитися у верхній частині робочої області.
5. Якість розпізнавання образів вимірюється частотою правильної класифікації жестів і не повинна применшувати 50% в цілому.
6. Додаток має бути розрахованим на подальше розширення «словнику» жестів.
7. Додаток має бути написаний із використанням мови програмування Python та бібліотеки OpenCV.

1.7 Висновки

У цьому розділі було розглянуто сучасні програмні рішення для зчитування жестів. Також було проаналізовано базові алгоритми обробки зображення та доступні засоби розробки програмного забезпечення. Як наслідок, було написано вимоги до кінцевого продукту дипломної роботи, обрано мову програмування, бібліотеку для розробки, та алгоритми для подальшої роботи.

2. ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМИ

2.1 Збір даних для тестування

Для подальших експериментів було знято ряд відеозаписів, де 5 різних людей зображують жести 1-5. Тестові дані містять тіло людини вище ніг на монотонному фоні у монотонному одязі. Данні були зняті камерою високої якості.

Рисунок 2.1 – Демонстраційні кадри з тестових відеозаписів

2.2 Фільтрація зображення

Першим етапом у реалізації програми розпізнавання мови жестів є виділення інформативних одиниць із відеозапису. Інформативною одиницею у даній задачі є місцеположення людини, її обличчя та рук.

Скориставшись засобами бібліотеки OpenCv було проведено обробку зображення за допомогою порогової, адаптивної фільтрації та виділення контурів зображення. Результати даних алгоритмів приведено нижче:

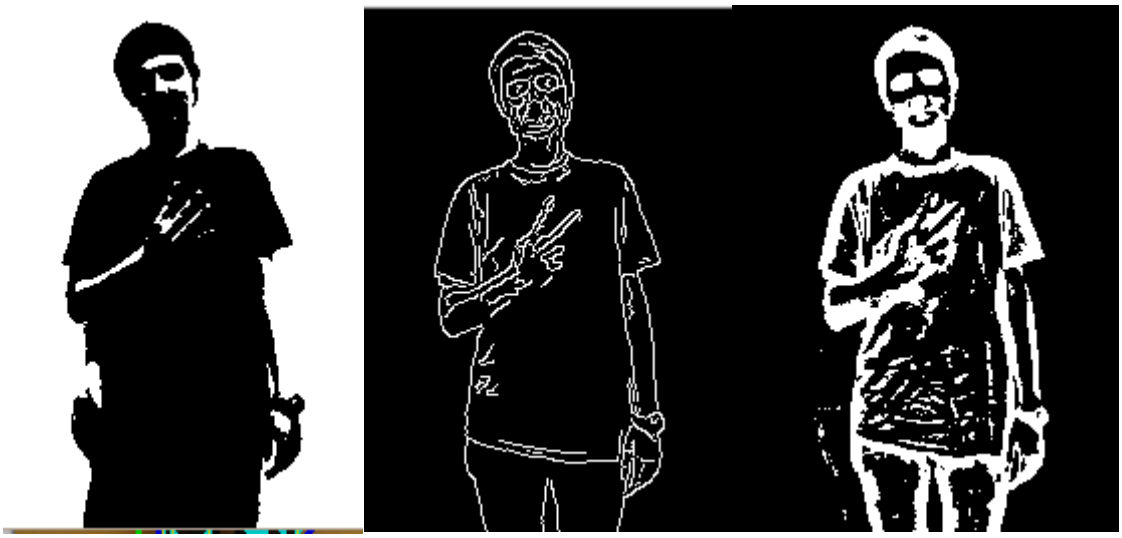


Рисунок 2.2 – результати: а) порогової фільтрації; б) аналізу контурів; с) адаптивної фільтрації.

Порогова фільтрація дуже чутлива до кольорів зображення та занадто нестабільна, у той час як адаптивна фільтрація та виділення контурів фільтрують зображення достатньо якісно.

Для боротьби з шумами та стабілізації зображення було використано стандартні морфологічні засоби бібліотеки OpenCV erode() delite(), результати роботи яких наведено нижче:

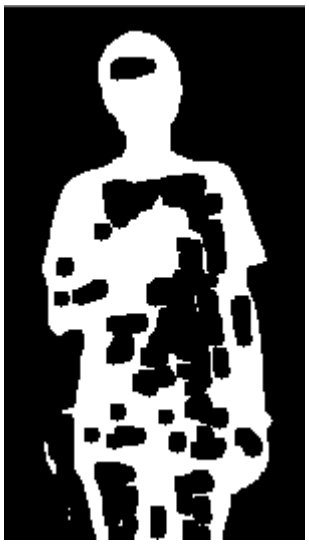


Рисунок 2.3 Результат адаптивної фільтрацій після морфологічних операцій

На базі отриманого зображення за дотопомогою алгоритмів пошуку найбільшого контуру була знайдена випукла оболонка людини та бітова маска для виділення «цікавих місць» відеозапису.

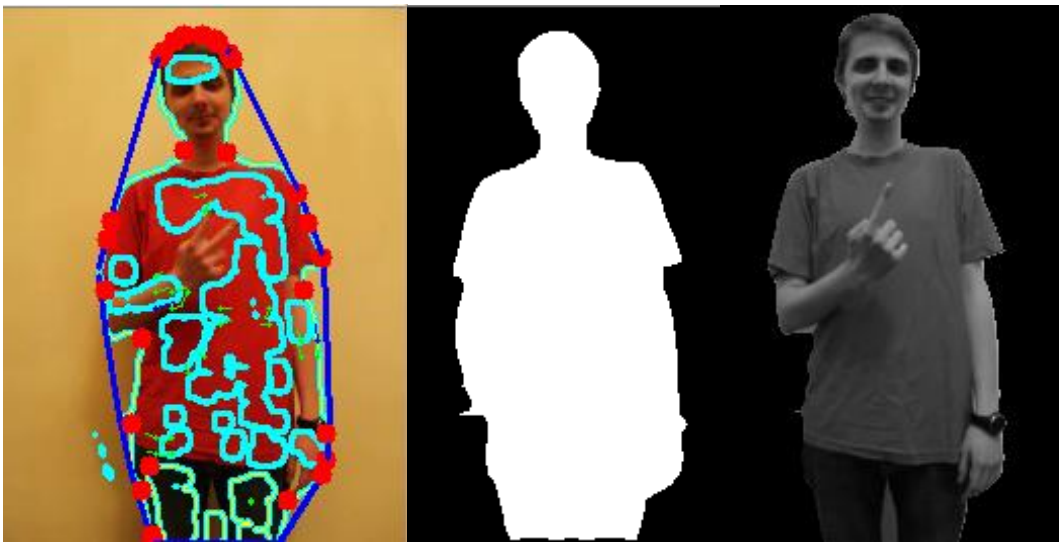


Рисунок 2.4 – а) підрахунок випуклої оболонки за допомогою аналізу контурів; б) бітова маска зображення знайденого контуру; в) початкове зображення оброблене за допомогою бітової маски

Як бачимо, за допомогою даних алгоритмів можна ефективно та достатньо швидко виділити геометричне положення точок, що відносяться до людини шляхом відсіювання фону. Слід зауважити, що методи фільтрації є відносно нестабільним, адже фільтрація виконується при умові монотонного фону. Окрім цього, дані методи зовсім не полегшують задачу виділення рук та обличчя із зображення.

2.3 Спроби виділення обличчя та рук

Описані вище методи хоч і являються ефективними для виділення людини, вони є не ефективними у знаходженні більш малих об’єктів на зображені. Ті самі операції проведені на обробленому зображенні не дають змоги виділити ані руки, ані обличчя.

2.3.1 Виділення обличчя із відеоряду

На відміну від зчитування жестів, виділення обличчя на зображення є дуже популярною проблемою та має багато ефективних методів вирішення. Для знаходження обличчя було обрано алгоритм каскадів Хаара. Даний метод відноситься до алгоритмів машинного навчання, та може бути використаний для пошуку будь-якого предмету на зображення. Для цього потрібно підібрати тестові данні та провести навчання. Слід зауважити, що використаний класифікатор для знаходження обличчя не є зроблений власноруч та знаходиться у вільному доступі в пакеті OpenCV.

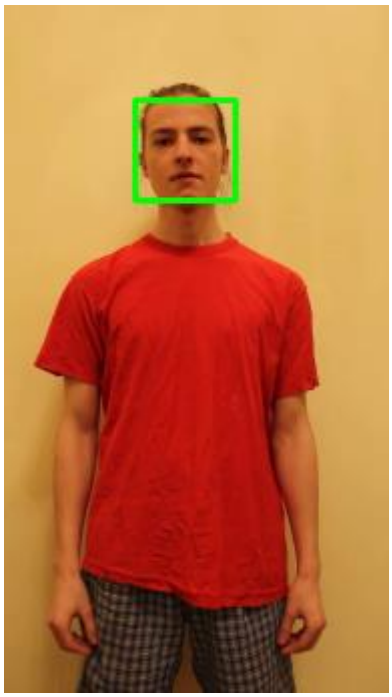


Рисунок 2.5 – демонстрація роботи алгоритму пошуку обличчя на зображенні

2.3.2 Виділення рук за допомогою аналізу оптичного потоку

Для реалізації ліній Фарнебека було використано функцію бібліотеки OpenCV *calcOpticalFlowFarneback.*

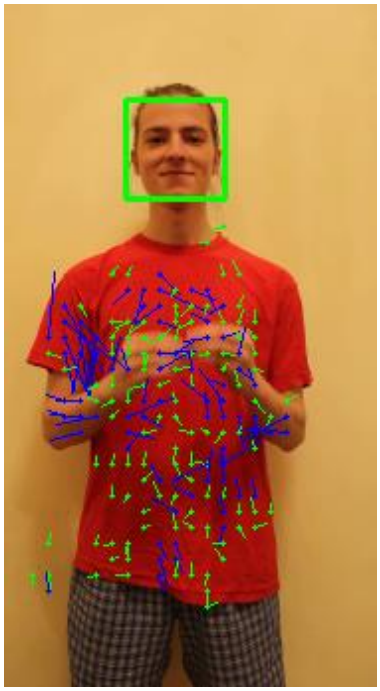


Рисунок 2.6 – приклад графічного відображення ліній Фарнебека (більшість векторів відфільтровано для наочності)

Аналізуючи оптичний потік з точки зору інтенсивності векторів у певних точках можна отримати бінарну карту рухомих пікселей. Емпіричним шляхом було знайдено порогове значення довжини вектору руху. Результат наведено нижче:

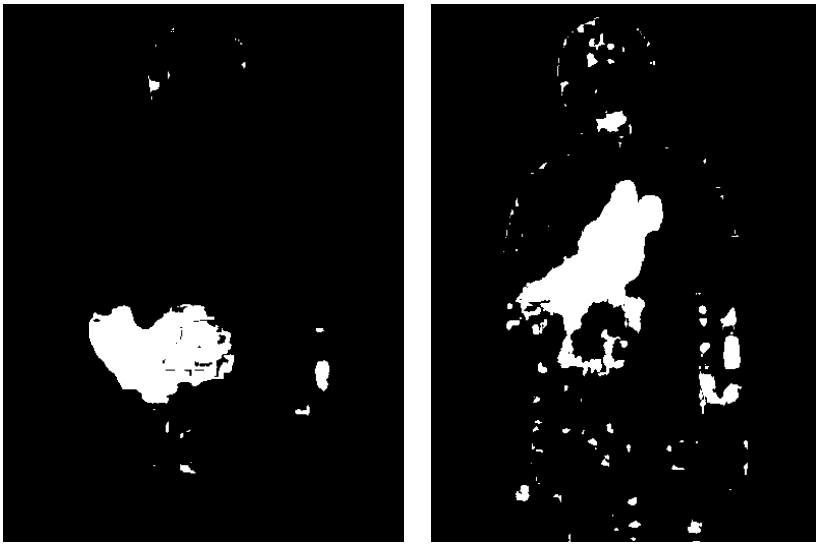


Рисунок 2.7 – бінарне відтворення оптичного потоку векторів

Після відсіювання шумів за допомогою морфологічних функцій *erode* та *dilate* було отримано результуюче зображення:

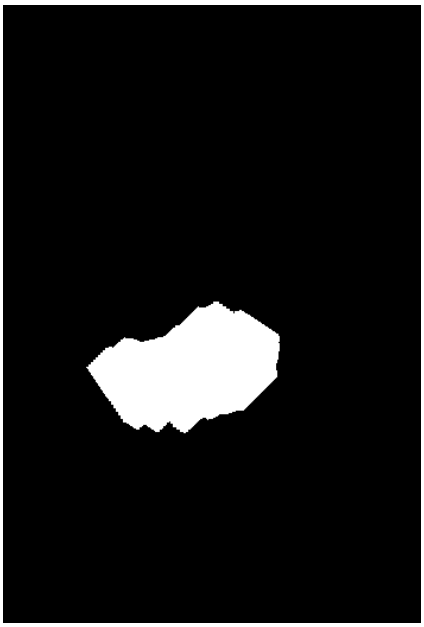


Рисунок 2.8 – оброблені бінарні відтворення оптичного потоку

Виконуючи контурний аналіз отриманого вище зображення, та алгоритми знаходження обмежуючого прямокутника було локалізовано рух.

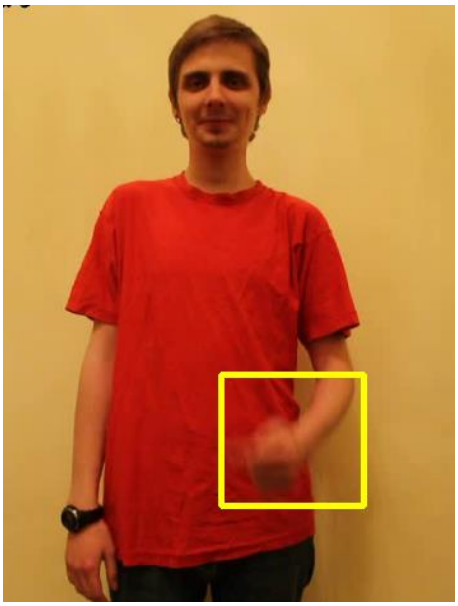


Рисунок 2.9 – обмежуючий прямокутник рухомих точок

Зауважимо, що даний метод відслідковує усі рухи на зображені. Це дозволяє відслідковувати руки окремо одне від одного.

2.4 Розпізнавання жестів

2.4.1 Контрольні точки

Для розпізнавання жестів було знято шаблони розпізнавання:

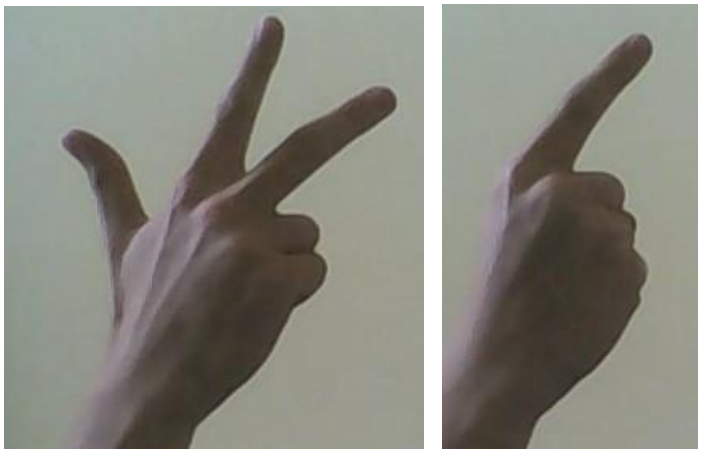


Рисунок 2.10 – приклад шаблонів для розпізнавання

Знаходження контрольних точок на зображенні було виконано за допомогою бібліотеки OpenCV використовуючи функцію *detectAndCompute*

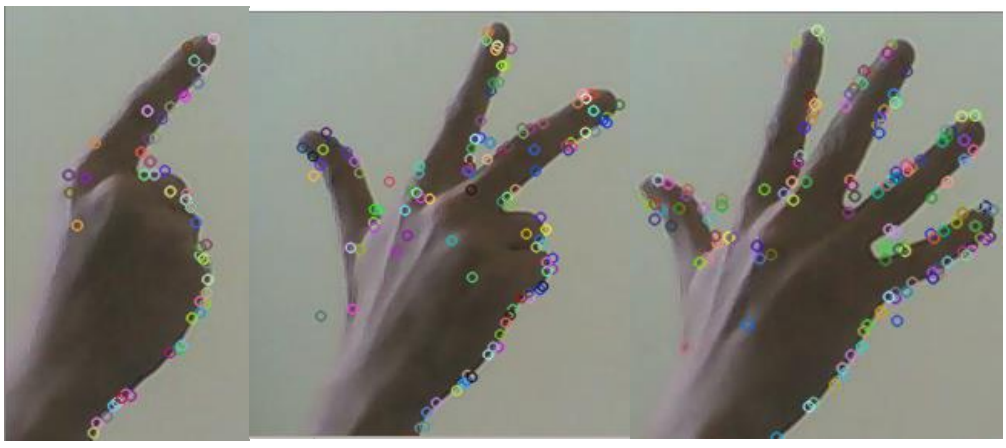


Рисунок 2.11 – контрольні точки шаблонних зображень

Задля порівняння шаблонного та фактичного зображень було використано метод «співвідношення випробувань». Даний метод залучається у знаходженні двух найкращих збігів контрольних точок, та їх фільтрацію, якщо співвідношення «якості» збігу менше 0.75:

$$\frac{M_1}{M_2} \geq 0.75 \tag{7}$$

де M_1 – найкращий збіг, M_2 - другий найкращий збіг.

Дане випробування відкидає ті точки, які мають більш ніж 1 вірогідний збіг.

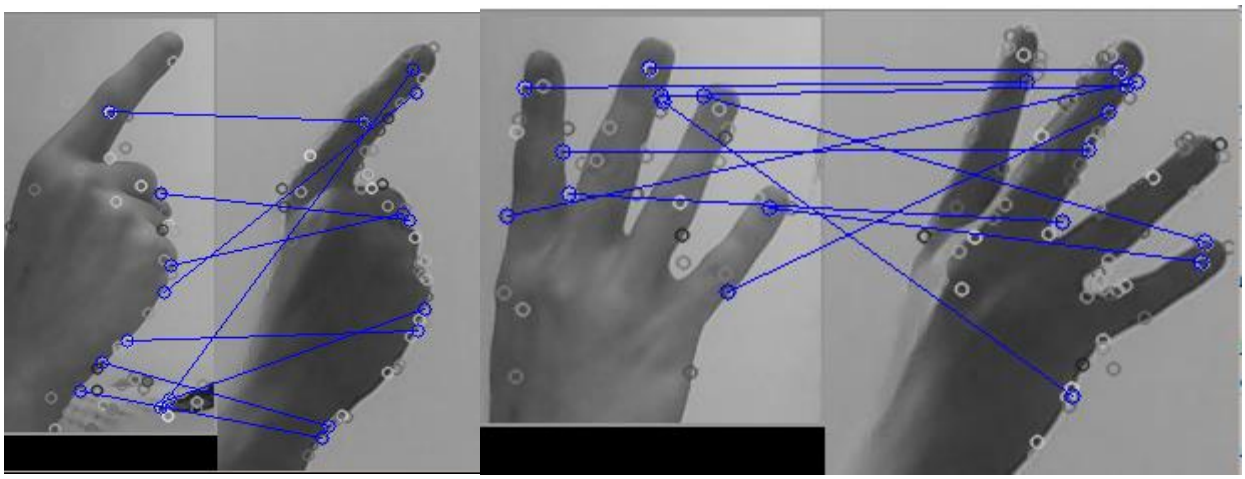


Рисунок 2.12 – порівняння зображень за допомогою контрольних точок

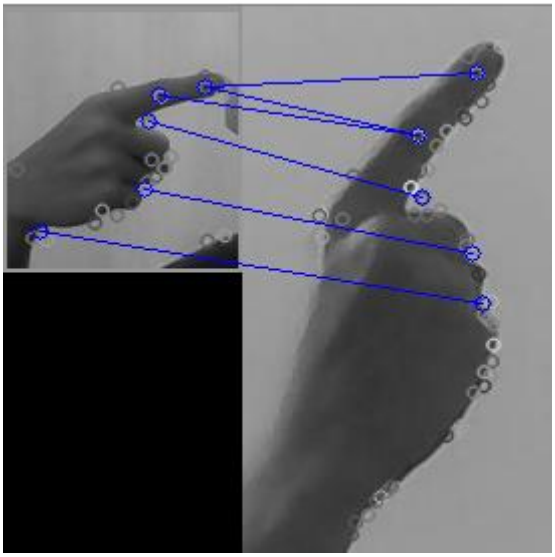


Рисунок 2.13 – стійкість алгоритму до афінних перетворень

Оптимізація методу порівняння

Розрахувавши попарну схожість шаблоних зображень між собою було отримано наступну таблицю:

Таблиця 2.1 - Попарне порівняння особливих точок

Попарне порівняння особливих точок					
	1	**2**	**3**	**4**	**5**
1	1,00	0,43	0,40	0,34	0,29
2	0,35	1,00	0,44	0,37	0,45
3	0,26	0,25	1,00	0,27	0,28
4	0,17	0,29	0,28	1,00	0,54
5	0,19	0,23	0,32	0,35	1,00

Аналізуючи таблицю 2.1 можна зробити висновок про надмірну схожість сусідніх шаблонів. Задля зменшення ступеня схожості зображень було виконано ітераційне видалення *загальних точок* (тобто таких, що мають однакові характеристики для більшості зображень, тобто не несуть інформативності для задачі класифікації).

Таблиця 2.2

Попарне порівняння особливих точок					
	1	**2**	**3**	**4**	**5**
1	1	0,24	0,09	0,11	0,04
2	0,27	1	0,15	0,16	0,10
3	0,20	0,29	1	0,23	0,11
4	0,16	0,21	0,16	1	0,18
5	0,11	0,23	0,13	0,29	1

2.4.2 Контурний аналіз

За допомогою пакету OpenCV було виділено контури:

Рисунок 2.14 – Виділений контури руки з зображення

Порівнюючи знайдені контури за допомогою функції matchContour() можна отримати цілком задовільні результи.

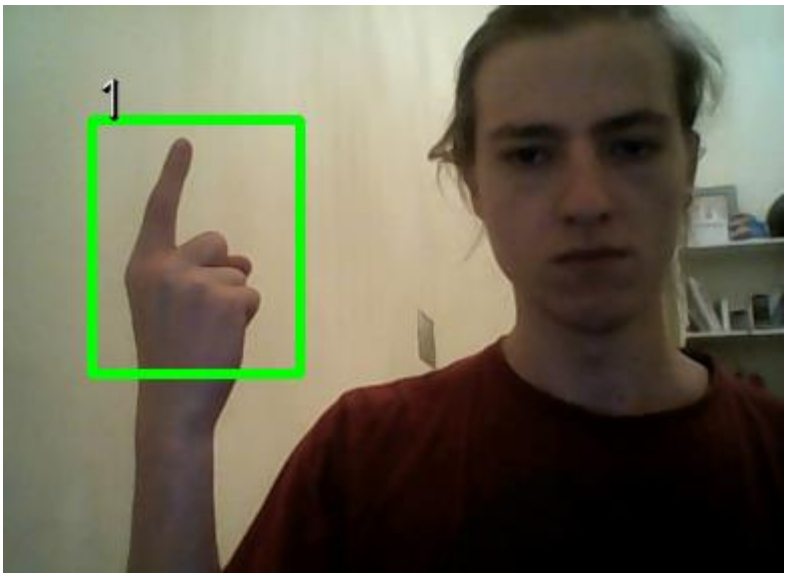


Рисунок 2.15 – результуючий інтерфейс програми

2.5 Висновки

У цьому розділі були проаналізовані та програмно відтворені основні методи для роботи із зображенням. Було реалізовано алгоритми знаходження рук на відео та алгоритми розпізнання жестів. Після цього, створене рішення було покращене шляхом оптимізації використаних алгоритмів. Внаслідок цього було отримано робочу програму розпізнання жестів, яка відповідає поставленим у дипломній роботі цілям.

3. ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для аналізу нелінійних нестаціонарних процесів. Інтерфейс користувача був розроблений за допомогою мови програмування Python.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Windows/Linux/Mac.

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 наоснові 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.

3.1 Обґрунтування функцій програмного продукту

Виходячи з конкретних цілей, які реалізуються програмою, виділимо її основні функції:

- F1 – зчитування відеоданих
- F2 – занаходження та відслідковування рук на зображені
- F3 – виділення з відеопотоку жестів
- F4 – розпізнавання жестів
- F5 – вивід перекладу

Кожна з основних функцій може мати декілька варіантів розв’язання.

Для F1:

а) зчитування даних з камери або файла.

Для F2:

а) відслідковування та локалізація рухів;

б) знаходження рук за допомогою алгоритмів машинного навчання.

Для F3:

а) виділення жестів шляхом аналізу інтенсивності рухів.

Для F4:

а) розпізнавання жестів за допомогою шаблонів;

б) розпізнавання жестів за допомогою контрольних точок.

Для F5:

а) відображення інформації у текстовому вигляді.

За розглянутими варіантами будуємо морфологічну карту

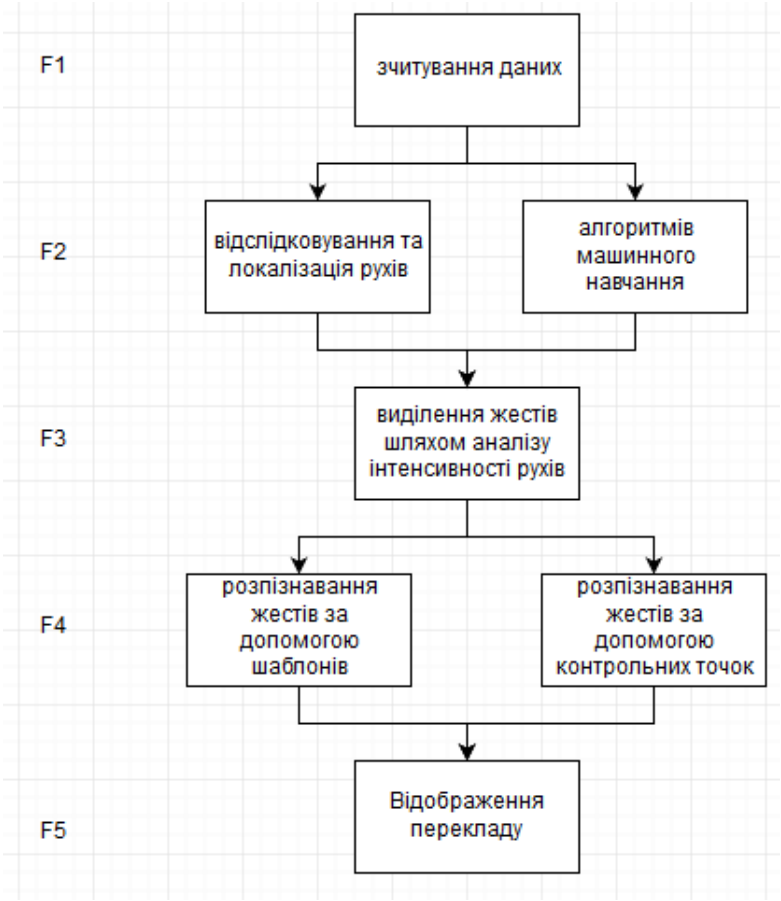


Рисунок 3.1 – Морфологічна карта

На основі цієї карти будують позитивно-негативну матрицю.

Таблиця 3.1

Позитивно-негативна матриця			
Основна функція	**Варіант реалізації**	**Переваги**	**Недоліки**
F1		Ефективність	
F2	а	Простота реалізації	Менша стійкість до перешкод
	б	Більша стійкість	Складність реалізації
F3		Простота реалізації	немає
F4	а	Простота реалізації	Менша стійкість до поворотів рук
	б	Більша стійкість	Складність реалізації
F5		Ефективність	

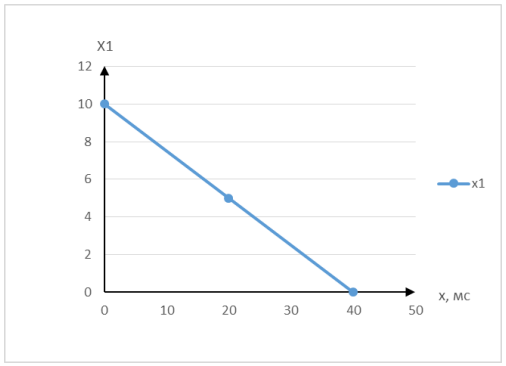
3.2 Обґрунтування системи параметрів

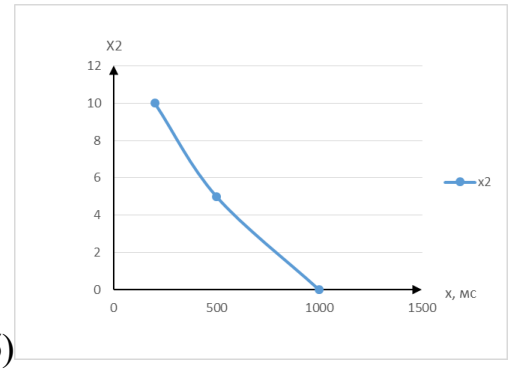
Для характеристики розроблюваної програми можна використати такі параметри:

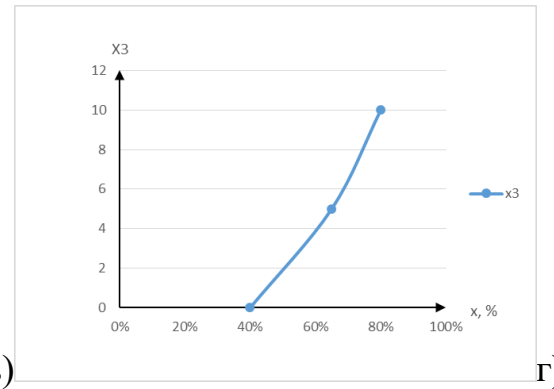
- X1 – час на обробку кадра зображення;
- X2 – час на розпізнавання жеста;
- X3 – точність розпізнавання жеста.

Таблиця 3. 2

Основні параметри програми				
Параметр	**Позначення параметра**	**Гранично допустиме значення**	**Значення параметра**	
			Середнє отримане	**Якого треба досягти**
Відносна затримка на обробку одного кадру зображення, мс	X1	40	20	0
Час на розпізнавання жеста, мс	X2	1000	500	200
Точність розпізнавання жеста, %	X3	60%	40%	80%
Об’єм використаної памяті, МБ	X4	100	60	20

а) 

б) 

в) 

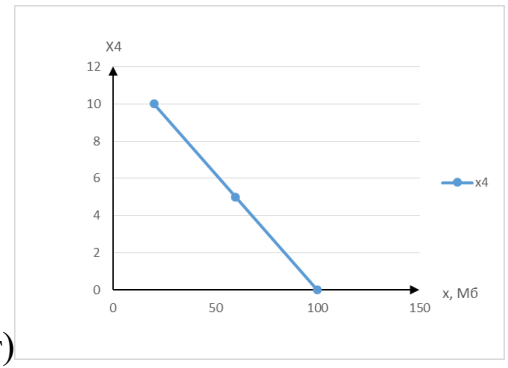
г) 

Рисунок 3.2 – оцінка основних параметрів програми. а) відносна затримка на обробку одного кадру зображення; б) час на розпізнавання жеста; в) Точність розпізнавання жеста; г) об’єм використаної памяті.

Вагомість параметрів визначають методом попарного їх порівняння, використовуючи результати ранжирування експертами.

Таблиця 3. 2 - Коефіцієнти конкордації експертних оцінок.

			Ранг параметра по оцінці експерта							Сума рангів,		Відхилення,	Квадрат відхилення,
Параметри	Назва параметра	Одиниці вимірювання									Т		
			1	2	3	4	5	6	7				
x1	час на обробку кадра зображення	Мс	2	3	2	3	2	3	1	16	4	-1,5	2,25
x2	час на розпізнавання жеста	Мс	3	2	3	2	3	2	3	18	4,5	0,5	0,25
x3	точність розпізнавання жеста	Р	4	4	4	4	4	4	4	28	7	10,5	110,25
x4	пам'ять	Мб	1	1	1	1	1	1	2	8	2	-9,5	90,25
Усього			10	10	10	10	10	10	10	70	17,5	0	203

Таблиця 3.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
x1 та x2	>	<	>	=	<	>	=	>	1,5
x1 та x3	<	=	<	<	<	=	>	>	1,5
x1 та x4	>	>	=	>	>	>	>	>	1,5
x2 та x3	<	<	>	>	<	=	>	=	1
x2 та x4	<	>	=	>	=	>	>	>	1,5
x3 та x4	>	=	<	=	>	>	>	>	1,5

За даними табл. 3.3 обчислюємо коефіцієнт конкордації за формулою:

$$W = \frac{12S}{N^2(n^3 - n)} \quad (8)$$

W = 0,828571

Оскільки розрахункове значення коефіцієнта конкордації більше від нормативного, то можна використовувати результати опитування експертів для наступних розрахунків. Розрахунок вагомості параметрів ПП ($К_{вi}$) наведено в табл. 30.

Таблиця 3.5 - **Р**озрахунок вагомості параметрів ПП

	Параметри xj				**Перший крок**		**Другий крок**		**Третій крок**	
Параметри xi	x1	x2	x3	x4	b_i	$К_{вi}$	b_i	$К_{вi}$	b_i	$К_{вi}$
x1	1	1,5	0,5	1,5	4,5	0,28125	13,5	0,291892	182,25	0,249893
x2	0,5	1	0,5	1,5	3,5	0,21875	7	0,151351	49	0,067187
x3	1,5	1,5	1	1,5	5,5	0,34375	22	0,475676	484	0,663639
x4	0,5	0,5	0,5	1	2,5	0,15625	3,75	0,081081	14,0625	0,019282
Загалом:					16	1	46,25	1	729,3125	1

Таблиця 3.6 - Розрахунок показників якості варіантів реалізації основних функцій ПП

Основна функція	**Варіант реалізації**	**Абсолютне значення параметру**	**Бальна оцінка параметру**	**Коефіцієнт вагомості параметру**	**Коефіцієнт якості**
F(X1,X4)	а)	300	3,60	0,229462	0,82606232
	б)	175	1,30	0,229462	0,29830028
F3(X1)	б)	64	5,00	0,229462	1,14730878
F4(X2,X3)	а)	10	1,80	0,273	0,4914
	б)	24	2,60	0,725212	1,88555241

Визначимо рівень якості кожного з варіантів:

- К1 = 0,82606232 + 1,14730878 + 0,4914 = 2.45;
- К2 = 0,29830028 + 1,14730878 + 0,4914 = 1.92;
- К3 = 0,82606232 + 1,14730878 + 1,88555241 = 3.856;
- К4 = 0,29830028 + 1,14730878 + 1,88555241 = 3.33.

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

3.3 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе 3 окремих завдання:

1. Зчитування даних F1.
2. Виділення жестів шляхом аналізу інтенсивності рухів F3.
3. Відображення інформації у текстовому вигляді F5.

Таблиця 3.7

		ступінь новизни	складність алгоритму	Норми часу	Поправочні коефіцієнти
F1		Г	3	8	0,36
F2	**а**	Б	2	27	1,08
	б	А	1	90	1,7
F3		Б	2	27	1,08
F4	**а**	Б	1	36	1,021
	б	Б	1	36	1,021
F5		Г	3	8	0,36

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

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

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (9)$$

де T_P – трудомісткість розробки ПП; $K_П$ – поправочний коефіцієнт; $K_{СК}$ – коефіцієнт на складність вхідної інформації; K_M – коефіцієнт рівня мови програмування; $K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

$K_{СТ}$ беремо 0.8 для всіх завдань, так як усі вони створюються за допомогою програмних бібліотек.

T_1 = 8*0.36*0.8 = 2,5 людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Таблиця 3.8

Трудомісткість		
F1		2,304
F2	**а**	23,328
	б	122,4
F3		23,328
F4	**а**	29,4048
	б	29,4048
F5		2,304

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

$T_{(F1,F2a,F3,F4(ab),F5)}$ = (2,304 + 23,328 + 23,328 + 29,4048 + 2,304) · 8 = 645.3504 людино-годин;

$T_{(F1,F2b,F3,F4(ab),F5)}$ = (2,304 + 23,328+ 122,4 + 29,4048 + 2,304) · 8 = 1437.9264 людино-годин;

Найбільш високу трудомісткість має варіант ІІ.

В розробці беруть участь один програмісти з окладом 20000 грн. Визначимо зарплату за годину за формулою:

$$С_Ч = \frac{М}{T_m \cdot t} \text{ грн.} \quad (10)$$

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

$$С_Ч = \frac{20000}{20 * 8} = 125 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$С_{ЗП} = С_Ч \cdot T_i \cdot К_Д$,

де $С_Ч$– величина погодинної оплати праці програміста;T_i – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$С_{ЗП}$ = 125 · 645.3504 · 1.2 = 96 802.5 грн.

ІІ. $С_{ЗП}$ = 125 · 1437.9264 · 1.2 = 215 688 грн.

Відрахування на єдиний соціальний внесок становить 22%:

$С_{ВІД} = С_{ЗП} \cdot 0.22$ = 21296.55 грн.

$С_{ВІД} = С_{ЗП} \cdot 0.22$ = 47451.36 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 20000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З = 12 \cdot 20000 \cdot 0,2 = 48000$грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 57600$грн.

Відрахування на єдиний соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.22 = 17280 \cdot 0,22 = 12672$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 15000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 15000 = 4312.5$ 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:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 15000 \cdot 0.05 = 862.5$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ 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:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,105 \cdot 0,8 \cdot 1,6848 = 241.495$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 15000 \cdot 0,67 = 10\ 050$грн.

Тоді, річні експлуатаційні витрати будуть:

$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$

$С_{ЕКС} = 57600 + 12672 + 4312.5 + 862.5 + 241.495 + 10\ 050 = 85594.26$ грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М\text{-}Г} = С_{ЕКС} / Т_{ЕФ} = 85594.26/1706,4 = 50.17$ грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$С_М = С_{М\text{-}Г} \cdot Т$

I. $С_М = 50.17 \cdot 645.3504 = 32377.21$ грн.;

II. $С_М = 50.17 \cdot 1437.9 = 72139.443$ грн.;

Накладні витрати складають 67% від плати:

$С_Н = С_{ЗП} \cdot 0,67$

I. $С_Н = 96\ 802.5 \cdot 0,67 = 64857.34$грн.;

II. $С_Н = 215\ 688 \cdot 0,67 = 144510.96$ грн.;

Отже, вартість розробки ПП за варіантами становить:

$С_{ПП} = С_{ЗП} + С_{ВІД} + С_{М} + С_{Н}$

I. $С_{ПП} = 96\ 802.5 + 21296.55 + 32377.21 + 64857.34 = 215\ 332$ грн.;

II. $С_{ПП} = 215\ 688 + 47451.36 + 72139.443 + 144510.96 = 437\ 182$ грн.;

3.4 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{Кj} / С_{Фj},$$

$К_{ТЕР1} = 2.45 / 215\ 332 = 0{,}11 \cdot 10^{-4}$;

$К_{ТЕР2} = 1.92 / 437\ 182 = 0{,}44 \cdot 10^{-5}$;

$К_{ТЕР3} = 3.856 / 215\ 332 = 0{,}17 \cdot 10^{-4}$;

$К_{ТЕР4} = 3.33 / 437\ 182 = 0{,}76 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є третій варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР3}$ рішенню

F1, F2a, F3, F4b, F5

3.5 Висновки

В першій частині 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.

Після проведення функціонально вартісного аналізу був зроблен висновок, що найбільш вигідним з точки зору економіки є наступний варіант реалізації ПП:

1. зчитування даних з камери або файла
2. відслідковування та локалізація рухів для знаходження рук
3. виділення жестів шляхом аналізу інтенсивності рухів
4. розпізнавання жестів за допомогою контрольних точок
5. відображення інформації у текстовому вигляді

Вартість якого складає 215 332 грн.

ВИСНОВКИ

В результаті виконання роботи було розглянуто основні методи обробки відеозапису. Було проаналізовано методи фільтрації зображень та методи обробки відфільтрованих даних. Було створено програму, що розпізнає цифри 1-5 показані мовою жестів у реальному часі.

У ході досліджень було визначено, що найдешевше у плані швидкодії та швидкості написання коду було розпізнавання жестів за допомогою контрольних точок та за допомогою порівняння контурів. Обидва підходи є достатньо стабільними до геометричних перетворень, легко розширюваними, дають досить точні результати. На відміну від контрольних точок, порівняння контурів достатньо сильно залежить таких факторів як освітленість, стабільність зображення, чіткість зображення та величину образу розпізнавання. Підхід, пов’язаний з контрольними точками потребує більш ґрунтовних досліджень та експериментів.

Основною не вирішеною задачею дипломної роботи є локалізація рук людини на зображенні з високою точністю. Це не неможливо, але потребує більш стійких алгоритмів, або додаткового обладнання.

Для збільшення швидкості роботи алгоритму необхідно провести додаткові досліди по аналізу оптичного потоку зображення.

Для збільшення точності роботи алгоритму розпізнавання необхідно розробити більш стійкі методи порівняння контрольних точок. Порівнювати їх необхідно у контексті їх розташування одна відносно одної. Можливо також об’єднання метода контрольних точок з порівнянням контурів для отримання більш стабільної роботи програми. Такий метод є теоретично перспективним, але потребує подальшого дослідження.

В якості демонстрації проведеної роботи маємо робочу систему, що здатна розпізнавати цифри 1-5 показані мовою жестів на відеокамеру. Досліджені алгоритми можуть бути використані у будь-яких системах людино-машинної інтеграції.

Отже, внаслідок проведеного дослідження стало зрозуміло, що потрібно спрямувати подальші дослідження та експерименти на розробку потужніших алгоритмів розпізнавання зображень. Не зважаючи на те, що якість вхідних даних можна суттєво підвищити за допомогою додаткового обладнання (стерео-камери, іч-камери), ці обладнання не є засобами широкого використання та не можуть бути використані з такою легкістю, як звичайна відеокамера. Розвиток сучасних технологій комп’ютерного зору направлений як раз в цю сторону, тому, почавши ці дослідження, можна побудувати конкурентоспроможну систему, здатну до самонавчання, яка має великий попит та соціальне значення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sign Language Translator – Режим доступа: http://research.microsoft.com/en-us/collaboration/stories/kinect-sign-lang_screen.jpg. Дата доступа: 06.04.2016.
2. Motion Savvy: Рекламний постер – Режим доступа: https://images.indiegogo.com/file_attachments/931256/files/20141014180025-features.png?1413334825. − Дата доступа: 25.04.2016.
3. Приклад фільтрації за порогом. – Режим доступа: https://habrastorage.org/getpro/habr/post_images/b22/e12/c6d/b22e12c6d33eb898e34b373475be35d8.jpg. Дата доступа: 10.04.2016.
4. Continuous Wavelet Transform – Режим доступа: http://www.satmagazine.com/cgi-bin/display_image.cgi?1114554895. Дата доступа: 10.04.2016.
5. Matlab: Корреляція зображень. – Режим доступа: http://www.mathworks.com/help/images/ref/imshowpair_ex1_diff.png. Дата доступу: 10.04.2016.
6. Boundary Detection with Sketch Tokens - Режим доступа http://cs.brown.edu/courses/cs143/proj5/teaser.png. − Дата доступа: 13.04.2016.
7. Farneback optical flow – Режим доступа: https://1.bp.blogspot.com/-SYVmZsXoYuA/Vs6-nFguzBI/AAAAAAAABr4/T1Iwd8B-q3o/s1600/opticalflow.png - Дата доступа: 18.03.2016.
8. T.J. Keating "An Improved Method of Digital Image Correlation" Photogrammetric Engineering and Remote Sensing \ T.J. Keating, P.R. Wolf, F.L. Scarpace, 1975 . – С. 993-1002.
9. SIFT: Офіційний сайт – Режим доступа: http://sift.jcvi.org/- Дата доступа: 20.03.2016.
10. SIFT: Опис ключової точки. – Режим доступа: http://www.vlfeat.org/api/sift-frame.png. – Дата доступа: 03.03.2016.

11. Байгарова Н.С. Методы индексирования и поиска изображений. – Режим доступа: http://web.ihep.su/library/pubs/aconf00/dconf00/ps/030.pdf. \ Байгарова Н.С., Бухштаб Ю.А, Горный А.А. Дата доступа: 10.05.2016.
12. Башков Е.А. Поиск изображений в больши бд с использованием корреляции цветовых гистограм.\ Башков Е.А., Шозда Н.С. – Режим доступа: http://graphicon.ru/html/2002/pdf/Bashkov_Zhozda_Re.pdf. Дата доступа: 10.05.2016.
13. Броневич А.Г. Анализ неопределенности выделения информативных признаков и представление изображений \ АГ Броневич, 2013 . – С. 65-69.
14. D. G. Lowe. Distinctive image features from scale-invariant keypoints.\ D. G. Lowe. IJCV, 2004. – С. С 91-110,
15. Messom, C.H., «Fast and Efficient Rotated Haar-like Features Using Rotated Integral Images» \ Messom, C.H, Barczak, A.L.C Australian Conference on Robotics and Automation, 2006. – С. 10-56,
16. Papageorgiou, Oren, «A general framework for object detection», International Conference on Computer Vision, 1998. – С. 298
17. Viola, «Rapid object detection using a boosted cascade of simple features», Computer Vision and Pattern Recognition, 2001. – С. 110-120
18. Lienhart, R, «An extended set of Haar-like features for rapid object detection», ICIP02, 2002. – С. 900-903
19. T. Pajdla Hlavac "Surface discontinuities in range images," in Proc IEEE 4th Int. Conf. Comput. Vision,1993. . – С. 524-528
20. M. H. Asghari, "Edge detection in digital images using dispersive phase stretch," International Journal of Biomedical Imaging, Vol. 2015, Article ID 687819, 2015. – С. 169-189
21. M. H. Asghari, "Physics-inspired image edge detection," IEEE Global Signal and Information Processing Symposium (GlobalSIP 2014), paper: WdBD-L.1, Atlanta, December, 2014. – С. 65

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
import cv2
import numpy as np
import tools
from GuestureRecognizer import GestureRecognizor
import ctypes
import mylib
import math
import operator
from collections import defaultdict
from flowWorker import FlowWorker

#feature detection
dim = None;

def main():

    global dim
    video_name = 'kir.avi'
    video_name = 0
    result_number = str(1);
    images = []
    cap = cv2.VideoCapture(video_name)
    flowWorker = FlowWorker();
    ret, frame = cap.read()
    constant = 400.0
    r = constant / frame.shape[1]
    dim = (int(constant), int(frame.shape[0] * r))
    frame = resize_img(frame);
    prev_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    flow = np.zeros_like(prev_gray)
    has_been_moved = np.zeros_like(prev_gray)
    cv2.namedWindow('magwindow')
    cv2.createTrackbar('minamplitude', 'magwindow', 400, 1000, nothing)
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (6,6))
    kernel2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2, 2))
    saved_cntrs = []
    previous_rects = []

    face_cascade = cv2.CascadeClassifier('haar/face.xml')
    table = None
    recognizor = GestureRecognizor()

    cont = 0;
    moved_clear_step = 5;
    moved_step = 0;
    isStopped = False
    while cap.isOpened():
        ret, frame = cap.read()
        if (frame is None):
            cap = cv2.VideoCapture(video_name)
            continue

        frame = resize_img(frame)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        flow = flowWorker.calc_farneback(prev_gray, gray)
```

```
        magnitudes = flowWorker.to_squared_mag_array(flow)

        cv2.imshow('moved', has_been_moved)
        minamplitude = float(cv2.getTrackbarPos('minamplitude',
'magwindow'))/100
        moved_step += 1
        #converting amplitudes to binary image
        ret, magnitudes = cv2.threshold(magnitudes.astype('uint8'),
minamplitude, 255, cv2.THRESH_BINARY)


        img = cv2.erode(magnitudes, kernel, iterations=4)
        img = cv2.dilate(img, kernel, iterations=8)
        cv2.imshow('erode-dilate cycles', img)

        #kp = fast.detect(frame, None)
        faces = face_cascade.detectMultiScale(frame, 1.3, 5)
       # frame = cv2.drawKeypoints(frame, kp, color=(255, 0, 0))
        for (x,y,w,h) in faces:
            #cv2.rectangle(frame, (x,y), (x+w, y+h), (255,0,255), 4)
            for i, yl in enumerate(xrange(y,np.size(frame,0),h)):
                if 1 == i or i == 4:
                    lcolor = (255,255,255)
                    #cv2.line(frame, (0,yl), (np.size(frame,1), yl), lcolor,1)

        cntr, hierarchy = cv2.findContours(img, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)


        if len(cntr) == 0:
            #cv2.waitKey(10000)
            cntr = saved cntrs
            clr = (0,255,0)
            isStopped = True
        else:
            clr = (0,255,255)
            saved_cntrs = cntr
            if isStopped:
                isStopped = False
                has_been_moved *= 0

        new_rects = []
        for countour in saved_cntrs:
            newrect = cv2.boundingRect(countour)
            (x,y,w,h) = newrect
            #white rect
            #cv2.rectangle(frame, (x, y), (x + w, y + h), (255,255,255), 1)
            for rect in previous_rects:
                if(inside(rect, newrect)):
                    if(square(newrect) < 1.5*square(rect)):
                        newrect = rect
            new_rects.append(newrect)
        previous_rects = new_rects

        if table is None:
            table = defaultdict(lambda: 0)

        for (x,y,w,h) in new_rects:
            if not isStopped:
                if(len(table) > 0):
                    pass
                    #table1 = [(x,y/cont) for (x,y) in table.iteritems()]
```

```
                    #print(table1)
                    #result_number = max(table1, key=operator.itemgetter(1))[0]

                table = defaultdict(lambda: 0)
                cont = 0
            if (isStopped):

                part = frame[y:y + h, x:x + w]
                images.append(part)
                if(len(images) > 5):
                    #name = recognizor.recognize(images)
                    name = recognizor.recognize(images)
                    if name is not None:
                        for k,v in name:
                            table[k] += v;
                        cont+=1

                        if (len(table) > 0):
                            table1 = [(x, y / cont) for (x, y) in
table.iteritems()]
                            print(table1)
                            result_number = max(table1,
key=operator.itemgetter(1))[0]
                            isStopped = False
                    else:
                        result_number = "no number"
                        isStopped = False
                    images = []
                pass
                #cv2.imshow('features', features)
        for (x, y, w, h) in new_rects:
            cv2.rectangle(frame, (x,y),(x+w,y+h),clr, 3)

        cv2.imshow('magwindow', magnitudes)
        #frame = cv2.flip(frame, 1)

        if len(new_rects) > 0:
            r = new_rects[-1]
            #r = cv2.flip(r, 1)
            (x, y, w, h) = r
            tools.draw_str(frame, (x, y), result_number)
        cv2.imshow('main', frame)

        prev_gray = gray.copy()
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break


dim = None


def inside((x1,y1,w1,h1), (x2,y2,w2,h2), delta = 10):
    x1, y1, w1, h1 = x1 - delta, y1 -delta, w1+delta, h1 + delta
    return x2 >= x1 and y2 >= y1 and \
        x2+w2 < x1+w1 and y2+h2 < y1 + h1

def square((x,y,w,h)):
    return w*h

def resize_img(img):
    global dim
    return cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
```

```
def nothing(x):
    pass

def init_window():
    cv2.namedWindow('canny')
    cv2.createTrackbar('1st', 'canny', 50, 255, nothing)
    cv2.createTrackbar('2nd', 'canny', 93, 255, nothing)


def drawCanny(gray):
    f = cv2.getTrackbarPos('1st', 'canny')
    s = cv2.getTrackbarPos('2nd', 'canny')
    edges = cv2.Canny(gray, f, s)
    cv2.imshow('canny', edges)
    return edges

if __name__ == '__main__':
    main()
```

GuestureRecognizor

```
from os import listdir
from os.path import isfile, join
import cv2
import tools
from collections import defaultdict
from features.dumper import KeypointDumper
import math
import numpy as np
from PIL import Image
from matplotlib import pyplot as plt

class GestureRecognizor:

    def __init__(self):
        self._templates = {}
        self._tpltimgs = {}
        self._hists = {}
        index_params = dict(algorithm=0, trees=10)
        index_params2 = dict(algorithm=6,
                             table_number=12,  # 12
                             key_size=20,  # 20
                             multi_probe_level=2)  # 2
        search_params = dict(checks=500)
        self._matcher = cv2.FlannBasedMatcher(index_params, search_params)
        self._matcher = cv2.BFMatcher(normType=cv2.NORM_L2)
        self._feature_extractor = cv2.SIFT()#nfeatures = 25)
        self._haar_cascades = {}
        dir = 'templatesk'
        onlyfiles = [f for f in listdir(dir) if isfile(join(dir, f)) and
f.endswith('.jpg')]
        kpDumper = KeypointDumper()
        for f in onlyfiles:
            image = cv2.imread(join(dir, f))
            self._tpltimgs[f] = image
            kp, desc =  kpDumper.load(join(dir,f)+'.bin')

            kp, desc = self._feature_extractor.compute(image, kp)
            #kp, desc = self._feature_extractor.detectAndCompute(image, None)
            self._templates[f] = (kp, desc)
```

```
            temp = image.copy()
            cv2.drawKeypoints(image, self._templates[f][0], temp)
            #cv2.imshow(f, temp)

    def similarness2(selfs, image, images):
        # bw = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # bws = [cv2.cvtColor(i, cv2.COLOR_BGR2GRAY) for i in images]
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        im2 = cv2.cvtColor(images[2], cv2.COLOR_BGR2GRAY)
        hist = cv2.calcHist([image.astype('float32')],  # <-- convert to float32
                    channels=[0],
                    mask=None,  # <-- convert to uint8
                    histSize=[256],
                    ranges=[0, 256])

        hist2 = cv2.calcHist([im2.astype('float32')],  # <-- convert to float32
                             channels=[0],
                             mask=None,  # <-- convert to uint8
                             histSize=[256],
                             ranges=[0, 256])

        return cv2.compareHist(hist, hist2, cv2.cv.CV_COMP_CORREL)




    def similarness(self, image1, image2):
        """
    Return the correlation distance be1tween the histograms. This is
'normalized' so that
    1 is a perfect match while -1 is a complete mismatch and 0 is no match.
    """
        # Open and resize images to 200x200

        i1 = Image.fromarray(image1).resize((64, 64))
        i2 = Image.fromarray(image2).resize((64, 64))
        # Get histogram and seperate into RGB channels
        i1hist = np.array(i1.histogram()).astype('float32')
        i1r, i1b, i1g = i1hist[0:256], i1hist[256:256 * 2], i1hist[256 * 2:]
        # Re bin the histogram from 256 bins to 48 for each channel
        i1rh = np.array([sum(i1r[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        i1bh = np.array([sum(i1b[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        i1gh = np.array([sum(i1g[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        # Combine all the channels back into one array
        i1histbin = np.ravel([i1rh, i1bh, i1gh]).astype('float32')

        # Same steps for the second image
        i2hist = np.array(i2.histogram()).astype('float32')
        i2r, i2b, i2g = i2hist[0:256], i2hist[256:256 * 2], i2hist[256 * 2:]
        i2rh = np.array([sum(i2r[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        i2bh = np.array([sum(i2b[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        i2gh = np.array([sum(i2g[i * 16:16 * (i + 1)]) for i in
range(16)]).astype('float32')
        i2histbin = np.ravel([i2rh, i2bh, i2gh]).astype('float32')
        return cv2.compareHist(i1histbin, i2histbin, cv2.cv.CV_COMP_INTERSECT)
```

```
    def findKeypoint(self, keypoint, keypointlist, eps):
        (l_x, l_y) = keypoint.pt
        for kp in keypointlist:
            (x,y) = kp.pt
            if abs(l_x - x) <= eps and abs(l_y - y) <= eps:
                return kp
        return None

    def kps_for_image_set(self, images):
        kps = []
        kp_hight_prior = []
        for i, img in enumerate(images):
            kp = self._feature_extractor.detect(img)
            kps.append(kp)
        delta = 5
        min =  0.2*len(images)
        result = {}
        for kp in kps:
            for k in kp:
                kpf = self.findKeypoint(k, result.keys(),5)
                if kpf is not None:
                    result[kpf] += 1
                else:
                    result[k] = 1
        return [kp[0] for kp in result.iteritems() if kp[1] > min]

    def recognizeHist(self, images):
        image = images[2]
        result = {}
        for k, i in self._tpltimgs.iteritems():
            sim = self.similarness2(i, images)
            result[k] = sim
        return [x for x in sorted(result.iteritems(), key=lambda x: x[1],
reverse=True)]

    def recognizeHaar(self, images):
        if len(self._haar_cascades) == 0:
            #loading haars
            infos = tools.get_files_and_path('haar\guestures')
            for (name, location) in infos:
                self._haar_cascades[name] = cv2.CascadeClassifier(location)
        result = []
        for entity in self._haar_cascades.iteritems():
            cascade = entity[1]
            faces = cascade.detectMultiScale(images[2], 1.3, 5)
            if len(faces) > 0:
                result.append(entity[0])

        return result

    def recognize(self, images):
        #image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        max = 10;

        # if isinstance(images, list):
        #     i_pts = self.kps_for_image_set([f for f in images])
        # else:
        #     i_pts = self._feature_extractor.detect(images)

        image = images[2]
```

```
        i_pts, i_desc = self._feature_extractor.detectAndCompute(image.copy(),
None);
        if(len(i_pts) < 1):
            print 'no points'
            return

        result = defaultdict(lambda: 0)
        result_match = {}
        for key in self._templates:
            (points, descr) = self._templates[key]
            if descr is None or i_desc is None or len(descr) == 0 or len(i_desc)
== 0:
                continue
            # matches = self._matcher.knnMatch(descr, i_desc, k=2)
            matches = self._matcher.knnMatch(i_desc, descr, k = 2)
            good = tools.get_good_matches(matches, coef=0.8) #TODO -0.7
            #good = [f[0] for f in matches if len(f) > 0]
            # good = tools.get good matches(matches, coef=0.6)
            result_match[key] = good

            if(len(good)>0):

result[key.split('.')[0]]+=float(len(good))#/(math.log(len(points)))

        kv = list(result.items())
        if len(kv) == 0:
            print('no matches')
            return None

        kv = sorted(kv,key=lambda x: x[1],reverse=True)
        # key = kv[0][0]
        # points = self._templates[key][0]
        # t1 = self. tpltimgs[key].copy()
        # t2 = image.copy()
        # t1 = cv2.drawKeypoints(t1, points, (255,255,255))
        # t2 = cv2.drawKeypoints(t2, i_pts, (255,255,255))
        # t1 = cv2.cvtColor(t1, cv2.COLOR_BGR2GRAY)
        # t2 = cv2.cvtColor(t2, cv2.COLOR_BGR2GRAY)
        #
        # res = tools.drawMatches( t2, i_pts,t1, points, result_match[key])
        # cv2.imshow('r', res)
        return kv[:3]
```

feature_cmp.py

```
import tools
import cv2
import numpy as np
from features.dumper import KeypointDumper

def main(dir):
    files = [f for f in tools.get_files(dir,True) if f.endswith("jpg")]
    fextractor = cv2.SIFT()
    index_params = dict(algorithm=0, trees=5)
    search_params = dict(checks=50)
    fmatcher = cv2.FlannBasedMatcher(indexParams=index_params,
searchParams=search_params)
    fmatcher = cv2.BFMatcher()

    k_d = []
    imgs = []
    for file in sorted(files):
        img = cv2.imread(file)
```

```
            imgs.append(img)
            kp_d = fextractor.detectAndCompute(img, None)
            #result = cv2.drawKeypoints(img, kp_d[0])
            #cv2.imshow('t', result)
            #cv2.waitKey()
            k_d.append(kp_d)

        print_paired_matches(imgs, fmatcher, k_d)
        same_dots = np.zeros((len(imgs), len(imgs)))
        max_sim = 0.3
        result_f_d = {}
        for i in reversed(range(len(imgs)-1)):
            for j in reversed(range(i+1, len(imgs))):
                print('%d:%d'%(i,j))
                (f2,d2) = k_d[i]
                (f,d) = k_d[j]
                matches = fmatcher.knnMatch(d2, d, k = 2)
                good = tools.get_good_matches(matches)
                similarity, _ = get_similarity(d2,d,fmatcher)
                while similarity > max_sim:
                    (f,d,good) = remove_best(f,d, good)
                    f, d = fextractor.compute(imgs[j], f)
                    similarity, good = get_similarity(d2,d,fmatcher)

                similarity2, _ = get_similarity(d, d2, fmatcher)
                while similarity2 > max_sim:
                    (f2, d2, good2) = remove_best(f2, d2, good)
                    f2, d2 = fextractor.compute(imgs[j], f2)
                    similarity2, good2 = get_similarity(d, d2, fmatcher)

                same_dots[i,j] = float(len(good))/len(f)
                same_dots[j,i], _ = get_similarity(d2,d, fmatcher)
                result f d[files[j]] = (f,d)
                if i == 0:
                    result_f_d[files[i]] = (f2,d2)

        dumper = KeypointDumper()
        for file in files:
            img = cv2.imread(file)
            kp, d = result_f_d[file]

            cv2.drawKeypoints(img, kp, img,(255,255,255))
            cv2.imshow('test', img)
            code = cv2.waitKey() & 0xFF
            if code == ord('q'):
                continue
            elif code == ord('s'):
                dumper.dump((kp, d), file + ".bin")



        print 'pair matches'
        print(same_dots)


    def get_similarity(desc1, desc2, fmatcher):
        matches = fmatcher.knnMatch(desc1, desc2, k=2)
        good = tools.get_good_matches(matches)
        return float(len(good))/len(desc1), good

    def remove_best(points, desc, matches, n = 1):
        sorted_m = sorted(matches, key=lambda x:x.distance)
```

```
    to_remove = sorted_m[:n]
    to_stay = sorted_m[n:]
    remove_idx = map(lambda x: x.trainIdx, to_remove)
    stay_idx = [f for f in range(len(points)) if f not in remove_idx]
    p = np.take(points, stay_idx)
    d = np.take(desc, stay_idx, axis=0)
    return p,d,to_stay

def print_paired_matches(imgs, fmatcher, k_d):
    same_dots = np.zeros((len(imgs), len(imgs)))
    for i in range(len(imgs)):
        for j in range(i, len(imgs)):
            (f, d) = k_d[i]
            (f2, d2) = k_d[j]
            matches = fmatcher.knnMatch(d, d2, k=2)
            good = tools.get_good_matches(matches)
            same_dots[i, j] = float(len(good)) / len(f)

            matches = fmatcher.knnMatch(d2, d, k=2)
            good = tools.get_good_matches(matches)
            same_dots[j, i] = float(len(good)) / len(f2)
    print(same_dots)


if __name__ == '__main__':
    main('D:\diploma\python\\test\\templatesk')

CreateFeature
import tools
import cv2
import pickle
import numpy as np
from dumper import KeypointDumper

feature_extractor = cv2.SIFT()
global_kps = []

def mouse_handler(event,x,y,flags,param):
    if event == cv2.EVENT_LBUTTONUP:
        global_kps.append(cv2.KeyPoint(x,y, _size=4))
        pass

    if event == cv2.EVENT_RBUTTONDBLCLK:
        pass

def process_img(img):
    global global_kps
    #global_kps = feature_extractor.detect(img)
    while True:
        result = img.copy()
        cv2.drawKeypoints(result, global_kps, outImage=result,
color=(255,255,255))
        cv2.imshow('window', result)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            kp_desc = feature_extractor.compute(img, global_kps)
            global_kps = []
            return kp_desc
    pass

def main(dir):
    kdumper = KeypointDumper()
    cv2.namedWindow('window')
    cv2.setMouseCallback('window', mouse_handler)
```

```
    files = [f for f in tools.get_files(dir, includePath=True) if
f.endswith('jpg')]
    for file in files:
        img = cv2.imread(file)
        kp_info = process_img(img)
        kdumper.dump(kp_info, file+'.bin')
```