МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет ракетно – космічної техніки

Кафедра вищої математики та системного аналізу

Пояснювальна записка до дипломної роботи

магістра

(освітньо-кваліфікаційний рівень)

на тему «Автоматизація розпізнавання фінансових документів»

ХАІ.405.463м.124.1404026.19О

Виконав: студент 6 курсу групи № 463м
напряму підготовки (спеціальності)
124 «Системний аналіз і управління»
(шифр і назва напряму підготовки (спеціальності))
Горбатюк Д. В.
(прізвище й ініціали студента)
Керівник: Макарічев В. О.
(прізвище й ініціали)
Рецензент: Романова Т. Є.
(прізвище й ініціали)

Харків 2019

РЕФЕРАТ

Пояснювальня записка до курсовох роботи містить: 43 с., 60 джерел, 16 рисунків.

РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ, PYTHON, OPENCV, ОБРОБКА ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ ЧЕКІВ, КОМП'ЮТЕРНЕ БАЧЕННЯ.

Об'єкт дослідження – обробка зображень.

Предмет дослідження – способи обробки зображень з метою виділення корисної інформації.

На сьогоднішній день компьютерне бачення – це один із найбільш стрімко розвиваючих напрямків у розробці. Він допомагає автоматизувати багато процесів. OpenCV – це відкрита бібліотека, що містить велику кількість працюючих алгоритмів що допомагають у розробці додатків пов'язаних з комп'ютерним зором.

Мета дослідження – створення додатку, який обробляє вхідне зображення, розпізнає текстову інформацію та повертає структуровані дані.

Задачі дипломної роботи:

1. Проаналізувати види обробки зображень
2. Дослідити алгоритми різних видів обробки зображень.
3. Виділити необхідні кроки для обробки зображення чеку.
4. Створити систему, яка зможе виділити з зображення тільки корисну область зображення чеку.
5. Розбити зображення на області
6. Розпізнати області
7. Обробити результат розпізнавання

Розробляється додаток мовою програмування Python з використанням бібліотеки OpenCV, який проводить зчитування зображення, багатокрокову обробку, повертає область що містить необхідну інформацію.

ABSTRACT

Explanatory note to the course work contains: 43 pp., 60 sources, 16 drawings.

PICTURE RECOGNITION, PYTHON, OPENCV, IMAGE PROCESSING, CHECK RECEIPT, COMPUTER VISION.

The object of study is image processing.

The subject of the study - methods of image processing in order to highlight useful information.

Today, computer vision is one of the fastest growing areas of development. It helps automate many processes. OpenCV is an open library containing a large number of working algorithms that help you develop applications related to your computer vision.

The purpose of the study is to create an application that processes the input image, recognizes text information and returns structured data.

Tasks of the thesis:

1. Analyze the types of image processing
2. Explore algorithms for different types of image processing.
3. Select the necessary steps to process the check image.
4. Create a system that can only distinguish from the useful area of the check image.
5. Divide the image into areas
6. Recognize the area
7. Process the result of recognition

An application is developed in Python programming language using the OpenCV library, which performs image reading, multi-step processing, returns an area containing the necessary information.

ЗМІСТ

ВСТУП

Не можливо уявити діяльність будьякого бізнесу без ведення бюджету. Ціль бюджетування – зростання економічної ефективності, укріплення фінансової стійкості підприємства. У той же час бізнес стрімко діджиталізується, всі процеси оцифровуються. Один із таких кроків – це автоматичне ведення бюджету.

Окрім потрібностей бізнесу у веденні бюджету, усе більше і більше звичайних людей зацікавлені в тому щоб мати можливість відслідковувати, куди були витрачені їх гроші. На сьогоднящній день існує дуже багато додатків для десктопів та телефонів для ведення бюджету. Найпопулярніші додатки мають більше 200 тисяч скачувань, що каже про велику популярність додатків даного призначення. Основною проблемою в них є слабка автоматизація процесів, тому користувачеві набридає постійно вводити дані вручну.

Результат цієї роботи повинен вирішити проблему оцифровування товарних чеків. Додаток може стати у нагоді бізнесу для автоматизування одного з початкових етапів ведення бюджету, а також у нагоді для розробників мобільних додатків для ведення бюджету.

1 ЗАПИТ НА ДОДАТОК ДЛЯ РОЗПІЗНАВАННЯ ФІНАНСОВИХ ДОКУМЕНТІВ

1.1 Аналіз фінансових додатків

Для аналізу існуючих додатків вибрано 10 найпопулярніших додатків, критерій вибору – кількість скачувань.

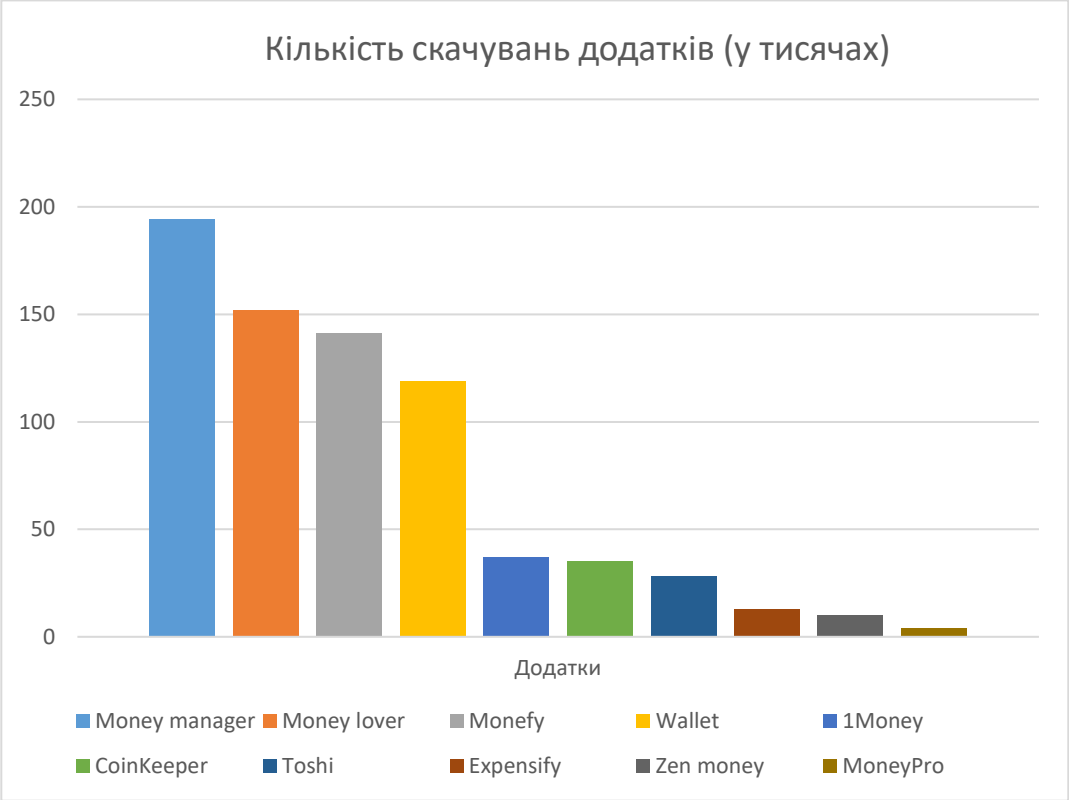

Рисунок 1.1 Кількість скачувань фінансових додатків

Проаналізувавши основні можливості всіх цих додатків було виділено наступні:

1. Ведення витрат за категоріями
2. Синхронізація операція з банком через додаток
3. Розпізнавання смс від банку
4. Сканування QR коду на чеку

Синхронізація операцій з банком допомагає автоматично вносити витрати в цілому за однією транзакцією, але втрачається інформація про індивідуальні позиції у чеку, тому відносити витрати до певної категорії зазвичай важко, наприклад, коли чек містить одночасно їжу, побутову хімію, техніку і т.п. Теж саме можна сказати і про розпізнавання смс від банку. Аналіз таких даних не має сенсу.

Сканування QR коду на чеку дає великі можливості, закодовані дані мають усю необхідну інформацію, та нажаль більшість чеків не мають такого QR коду, бо на сьогоднішній день існує дуже мала кількість банків які надають таку інформацію.

1.2 Шляхи вирішення проблеми

Дуже корисною функцією є автоматичне оцифровування усієї інформації з товарного чеку. Для вирішення цієї проблеми існує багато мов програмування та готових рішень. Найпривабливішим серед них є бібліотека OpenCV.

OpenCV (Open Source Computer Vision Library) - 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, відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях. Працювати з бібліотекою можливо на багатьох мовах програмування, а саме: Python, Java, Ruby, Matlab, Lua та інших.

Бібліотека містить понад 2500 оптимізованих алгоритмів, які допомагають у наступних задачах:

1. Аналіз та обробка зображень
2. Системи з розпізнавання обличчя
3. Ідентифікації об'єктів
4. Розпізнавання жестів на відео

5. Відстежування переміщення камери

6. Склеювання зображень між собою, для створення зображень всієї сцени з високою роздільною здатністю

Та готові алгоритми не вирішують поставлене завдання, а є лише інструментом. У процесі дослідження були розглянуті способи обробки зображень, в тому числі, покращення якості. Ряд літератури за цією темою перелічено в джерелах літератури [2-7].

Порівнюються способи корекції яскравості від простої лінійної корекції до нелінійної та гамма корекції. Наступний вид корекції - це корекція кольору. Цей вид обробки фотографій допомагає боротись з такими видами шумів як “сіль та перець”, імпульсний та гаусів шуми. Способи покращення зображень із такими шумами - фільтри [9]. Були розглянуты такі фільтри: лінійний, гаусовий та медіанний та інші.

Інший спосіб виділення важливої інформації на зображенні - це виділення країв. Робота в цьому напряму велася І.Собелем [25], Щарром [31], Д. Кенні [6] та інщі. Всі способи базуються на знаходжені градієнту зображення. Оператор Собеля та Щара згортають зображення за допомогою свого фільтра у той час як оператор Кенні застосовує подвійний поріг фільтрації країв та виділяє їх за допомогою гістерезісу.

2. ПРОЕКТУВАННЯ ДОДАТКУ

2.1 Мова Python

Для зручної роботи із зображеннями існує бібліотека OpenCV. OpenCV (англ. Open Source Computer Vision Library, бібліотека компьютерного зору з відкритим кодом) — бібліотека алгоритмів компьютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека розроблена Intel і нині підтримується Willow Garage та Itseez. Сирцевий код бібліотеки написаний мовою C++ і поширюється під ліцензією BSD. Біндинги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua та інших. Може вільно використовуватися в академічних та комерційних цілях.

Мовою програмування був вибраний Python, тому що він має наступні переваги:

Простота кодування

"Код як звичайний англійський" є основною метою Python. Це дозволяє програмістам зосередитись на дизайні, а не на кодуванні. Ця перевага є дуже вигідною, особливо якщо стикаються зі складними сценаріями.

Швидке прототипування

Оскільки можливо більше зосередитися на дизайні, тепер можливо експериментувати з більшою кількістю ідей дизайну. Python добре підходить для впровадження нових функцій.

Великі бібліотеки для машинного навчання

Python найчастіше використовується для машинного навчання. Навіть науковці можуть вкладати свій час, роблячи свій внесок, оскільки його легко кодувати і безкоштовно. Розробникам не потрібно багато хвилюватися щодо проектів, над якими вони працюють, оскільки більшість їх справ уже охоплені бібліотеками Python.

Безкоштовно

На відміну від MATLAB, який спеціалізується на аналізі даних, дослідженні, візуалізації тощо, але він не безкоштовний, ви можете розпочати роботу з Python, не платячи за нього.

Додаток може бути безпосередньо інтегрован з веб-фреймворками

Зрілі веб-фреймворки (як Django), які спрямовані на швидкий час розробки, акуратний та реалістичний дизайн, також доступні в Python. У Python навіть є мікрофреймворки, які так само функціональні, як і їхні більші аналоги.

Найчастіше використовується

Це означає, що він має більшу громаду. Є багато публікацій у блогах та Інтернет-ресурсів щодо Python + OpenCV.

2.2 Архітектура OpenCV

Весь функціонал бібліотеки розділяється на модулі. Усі модулі умовно можно віднести до 3 категорії:

Алгоритмічні модулі. Включають в себе наступний функціонал: базові структури, математичні функції, генерація псевдовипадкових чисел, зчитування запис в XML, обробка зображень(фільтри, перетворення), методи та моделі машинного навчання, різні дескриптори, аналіз руху та слідкування за об’єктом(оптичний потік, шаблони руху, усунення фону), детектування об’єкту на зображенні(вейвлети Хаара), калібровка камери, елементи обробки тривимірних зображень, бібліотека пошуку найближчих сусідів, устарілий код збережений заради зворотньої сумісності. Назви модулів: core, imgproc, calib3d, video, ml, objdetect, features2d, photo, stitching, videostab, superres, contrib, legacy, nonfree, flann.

GPU модулі. Включають функціонал для прискорення деяких функцій OpenCV за рахунок використання графічних процессорів. Має 2 модулі: gpu, ocl.

Інфраструктурні модулі. Функціонал для роботи з інтерфейсом користувача, завантаження та зберігання зображень та відео, сумісність деякими

мовами програмування. Включає наступні модулі: highgui, world, python, java, androidcamera, ts.

2.3 Типічна обробка зображення

Зазвичай обробка зображення проходить одні і ті самі етапи:

1. Зчитування зображення
2. Пониження кольорових просторів
3. Позбавлення шуму
4. Виділення області
5. Зчитування інформації

Зчитування зображення із файлової системи робиться за допомогою стандартних функцій із пакета OpenCV. Пониження кольорових просторів зазвичай зводиться до переводу кольорового зображення у чорно-біле зображення. У цьому разі зображення з пікселями що мають інформацію про три кольорові канали (наприклад модель BRG) трасформується у зображення з одним каналом. Отримавши зображення з інформацією лише по одному каналу стає можливим розглядати зображення як функцію від двох координат.

У більшості випадків зображення має шум, що буде заважати детектувати об’єкти на зображенні. Зашумлене зображення можно представити наступним рівнянням:

$$g(x, y) = f(x, y) + \eta(x, y)$$

Формула 2.1 Представлення зашумленого зображення

Де $f(x, y)$ - незашумленне зображення, $\eta(x, y)$ - шум, цей доданок невідомий, тому його не можно просто відняти від $g(x, y)$, але можно застосувати фільтрацію.

Для боротьби із шумом існує багато фільтрів:

1. Середньоарифметичний.

$$f(x,y) = \frac{1}{MN}\sum g(s,t)$$

- крім того що погано вбирає шум, розмиває багато деталей, але є найпростішим.

2. Середньогеометричний

$$f(x,y) = \exp(\frac{1}{MN}\sum \ln g(s,t))$$

- застосування данного фільтру до зображення, що має шум “перець”, тому що один “нульовий” піксель робить весь добуток нульовим, у результаті шум “перець” буде ще ширший. Зазвичай результат застосування фільтра схожий до результату середньоарифметичного фальтра, але при цьому втрачається менше деталей.

3. Середньогармонічний фільтр

$$f(x,y) = \frac{MN}{\sum \frac{1}{g(s,t)}}$$

- добре працює лише у випадку “білого” імпульсного шуму, але не працює для “чорного” імпульсного шуму; непогано працює для інших типів шумів, таких як гауссів шум.

4. Медіанний фільтр

$$f(x,y) = med\{g(s,t)\}$$

- дуже ефективний проти імпульсного шуму, різкі межі зберігає, вбирає невеликі деталі, зображення стає більш “мультиплікаційним”.

5. Фільтр Гаусса:

$$f(x,y) = \frac{1}{2\pi r^2}\sum_{s,t} e^{\frac{-(s^2+t^2)}{2r^2}} g(x+s, y+t)$$

- менше розмиває маленькі деталі, добре вбирає шум, розмиває границі.

3 РОЗРОБКА ДОДАТКУ

3.1 Первинна обробка зображення

Для початку є вхідне зображення чеку на рисунку 1.

Рисунок 3.1. Вхідне зображення чеку.

Після заміни кольору на чорно-біле зображення отримано зображення на рисунку 2. Заміна робиться вбудованою функцією в бібліотеці PythonCV:

```
cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Рисунок 3.2 Чорно-біле зображення.

З великого переліку фільтрів має сенс використовувати лише медіанний фільтр або фільтр Гаусса. На рисунках 3 та 4 зображена серія чорнобілих зображень чеку до яких були застосовані фільтри Гаусса та медіанний з різними параметрами.

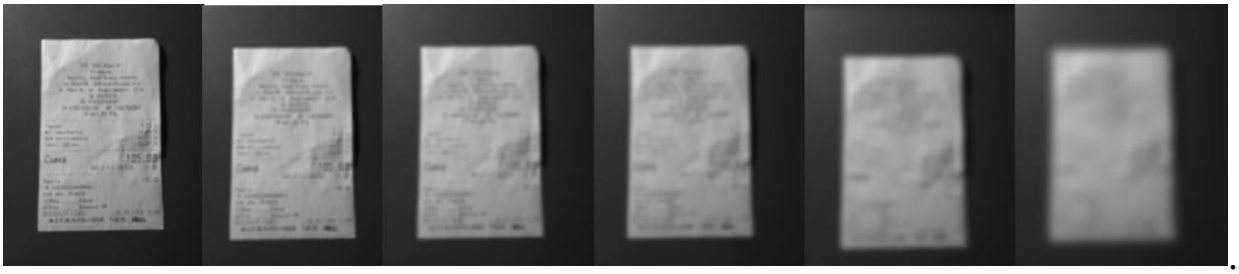

Рисунок 3.3 Результат застосування фільтра Гаусса з різними параметрами.

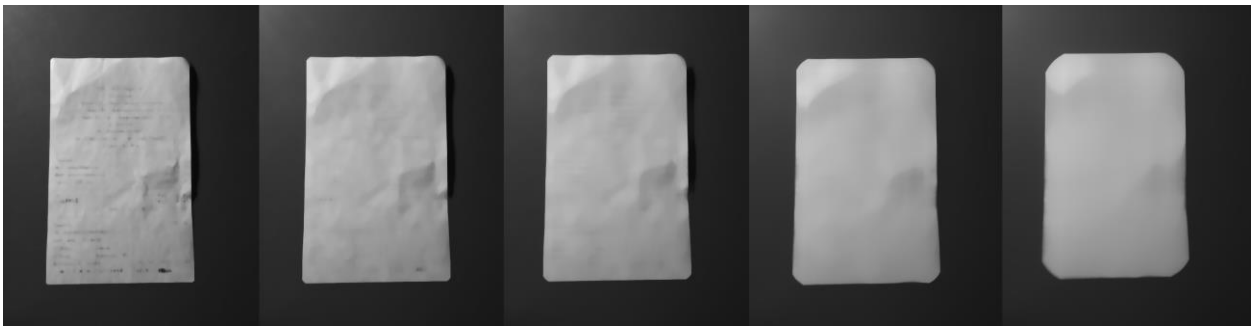

Рисунок 3.4 Результат застосування медіанного фільтра з різними параметрами.

Результатом застосування фільтру на данному етапі повинно бути отримано зображення з чіткою виділеною областю чека, областю зображення, на якому потрібно шукати текстову інформацію. Як бачимо, фільтр Гаусса погано підходить до такої задачі, бо межі стають більш розмитими, а область чека не дуже однорідна, кількість зображень велика, та відрізняється не так сильно. У той час, коли при застосуванні медіанного фільтра результат досягається за меншу кількість проходів, а результуюче зображення містить набагато менше шуму, межі чіткі, а область більш однорідна.

3.2 Робота з колірними просторами

Іншою проблемою є те, що функція, ща застосовує медіанний фільтр, приймає на вхід два параметра: саме зображення, та розмір фільтра. У даному випадку ця змінна виставлялась користувачем, котрий бачив результат застосування фільтра. Але у працюючому додатку такої можливості не має. У данному випадку дослідження на різних фото камерах фотографій чеків довільного розміру показали що похибка при фіксованому значенні є незначною, тому цей параметр можна залишити константою.

Проблема в даному підході полягає в ідеальності вхідних даних. Білий чек на темному фоні дає на чорно-білому зображенню чітку форму, якщо змінити фон на більш світлий - людина все ще зможе виділити межі чеку на чорно-білому зображені, проте значення пікселів будуть доволі близькі, множина значень фону може перетинатися із множиною значень чеку (рисунок 3.5)

Рисунок 3.5 Зображення на світлому фоні, чорно-білий варіант, розмитий варіант.

Вхідне зображення - матриця точок, кожна з котрих має три величини - значення червоного, зеленого та синього кольорів. При переводі зображення у чорно-білий варіант початкова інформація про різність кольорів частково втрачається. Рішенням цієї проблеми є перехід у інщий колірний простір.

Одним із зручних колірних просторів є простір HSV (заснована на трьох характеристиках кольору: колірному тоні (Hue), насиченості (Saturation) і значенні кольору (Value), який також називають яскравістю (Brightness)). Розклавши вхідне зображення на три окремі зображення, де в кожному присутня лише одна координата, зобразимо їх у вигляді чорно-білого зображення(рисунок 3.6).

Рисунок 3.6. Розкладене зображення у моделі HSV.

З цього можна зробити висновок, що одна з координат несе найбільш резонансну інформацію, та дає змогу виділити світлу область чеку на нетемному фоні. Але в залежності від зображення найкраща координата може бути одною з

трбьох, тобто не обов’язково перша координата буде нести найважливішу для нас інформацію. В такому випадку для більшої точності є необхідність обробляти одразу 4 зображення – 3 координати із моделі HSV, а також чорно-біле зображення.

3.3 Виділення області

Наступним завданням є виділення контуру самої області. Краї - це такі криві на зображенні, уздовж яких відбувається різка зміна яскравості або інших видів неоднорідностей. Причини виникнення країв: зміна освітленості, зміна кольору, зміна глибини сцени (орієнтації поверхні). Найпопулярнішим методом виділення кордонів є детектор кордонів Кенні. Детектор використовує фільтр на основі першої похідної від Гауссіана.

Межі на зображенні можуть перебувати в різних напрямках, тому алгоритм Кенні використовує чотири фільтра для виявлення горизонтальних, вертикальних і діагональних кордонів. Скориставшись оператором виявлення кордонів виходить значення для першої похідної в горизонтальному напрямку і вертикальному напрямку. Кут напрямку кордону округляється до однієї з чотирьох кутів, що представляють вертикаль, горизонталь і дві діагоналі. Потім йде перевірка того, чи досягає величина градієнта локального максимуму в відповідному напрямку. Після обробки зображенням детектором отримаємо результат на рисунку 7.

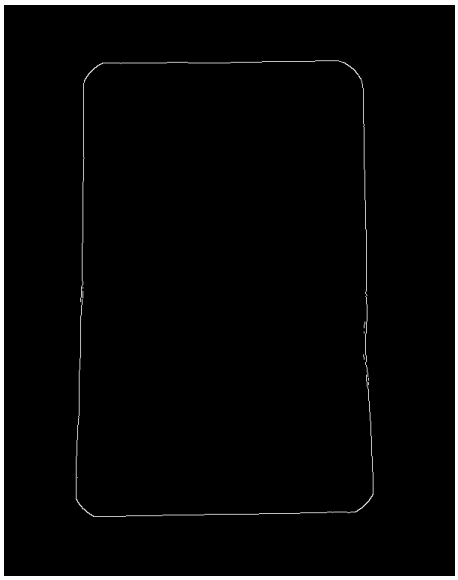

Рисунок 3.7. Зображення після обробки детектором Кенні.

Вже після цього можна скористатися функцією findContours. Серед усіх параметрів найцікавішими є режим угруповання та метод упаковки.

Режим угруповання - один з чотирьох режимів угруповання знайдених контурів:

RETR_LIST - видає все контури без угруповання;

RETR_EXTERNAL - видає тільки крайні зовнішні контури.

RETR_CCOMP - групує контури в дворівневу ієрархію. На верхньому рівні - зовнішні контури об'єкта. На другому рівні - контури отворів, якщо такі є. Всі інші контури потрапляють на верхній рівень.

RETR_TREE - групує контури в багаторівневу ієрархію.

Метод упаковки - один з трьох методів упаковки контурів:

CHAIN_APPROX_NONE - упаковка відсутня і всі контури зберігаються у вигляді відрізків, що складаються з двох пікселів.

CHAIN_APPROX_SIMPLE - склеює все горизонтальні, вертикальні і діагональні контури.

CHAIN_APPROX_TC89_L1, CV_CHAIN_APPROX_TC89_KCOS - застосовує до контурів метод упаковки (апроксимації) Teh-Chin.

Рисунок 3.8. Контур знайденої форми, виділений зеленим.

В нашому випадку потрібно знайти зовшнішній контур та склеїти контури у лінії, тому що розпізнати потрібно лише одну форму, та вона потрібна бути прямокутною. Тобто до функції треба звертатися з параметрами cv.RETR_EXTERNAL та cv.CHAIN_APPROX_NONE. У результаті буде отримані контури, що представлені на рисунку 3.8.

Є велика вірогідність того, що фільтр не зможе убрати всі шуми, тому необхідно провести додаткову фільтрацію контурі. Ідея полягає в тому щоб знайти максимально великий контур за площею, такий, котрий складається з чотирьох точок. Приклад функції, що реалізувати цю ідею наведено на рисунку.

```
def findContourWithMaxArea(contours):
    contours = sorted(contours, key=cv.contourArea, reverse=True)[:10]
    for c in contours:
        peri = cv.arcLength(c, True)
        approx = cv.approxPolyDP(c, 0.02 * peri, True)
        if len(approx) == 4:
            return approx
```

Рисунок Пошук конутуру за наійбільшою площею.

3.4 Порігова обробка зображення

Детектор Кені дуже добре знаходить контури на зображенні і для деяких зображень це може стати проблемою. Частини зображення до пониження кольорових просторів можуть виглядати як достатньо рівномірний фон. Після пониження вони могуть виглядати як окремі обїекти на зображені, тому будуть створювати додаткові контури, та що гірше, можуть влинути на головний контур, та змінють його параметри відносно шуканої області.

На даному єтапі обробки зображення було вирішено скористатися пороговою обробкою зображення. З огляду на те, що ми маємо фото чеку для розпізнавання, він повинен бути головним об’ектом на зображенні. Беручи за увагу білий колір чеку, його область на зображенні має бути найбільш яскравою. OpenCV має декілька функцій для порігової обробки:

Simple threshold

Приймає значення порігу, всі значення зображення, що менше заданого порігу стають 0, інщі приймають максимальне значення. Дуже простий алгоритм не працює в багатьох випадках через нерівномірне освітлення частин зображення. Приклад вхідних зображень та вихідних зображень показано на рисунку.

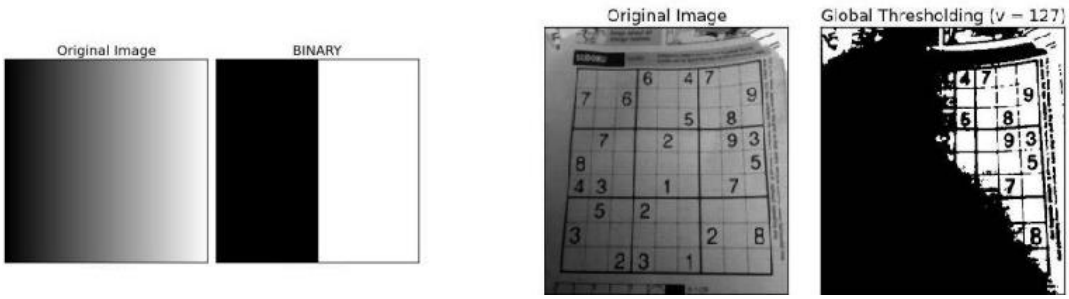

Рисунок 3.9 Робота алгоритму simple threshold

Adaptive threshold

Адаптивний алгоритм не приймає в якості параметра значення порогу, тому що він вираховує для кожного пікселя на зображенні своє значення в залежності від значення сусідніх пікселів. Результат порігової обробки менше залежить від якості освітленності зображення. Приклади роботи наведені на рисунку.

Рисунок 3.10 Робота алгоритму adaptive threshold

Після обробки чотирьох зображень буде отримано до 4 найкращих контурів(з урахуванням того, що на найменш відповідних зображеннях не буде знайдено ні одного контура, тобто кількість контурів може бути менше 4).

Рисунок 3.11 Кращі контури для трьох каналів зображення

3.5 Опрацювання кращих контурів зображення

Для подальшої роботи над зображенням потрібно мати лише один контур, тому із чотирьох найркащих контурів треба виділити кращий з кращих. Перший

контур, або випадковий контур взяти не вийде. Попри те, що було вибрано найращі контури на зображеннях, ще не є доказом того, що всі ці контури відпоівдають реальному контуру чека, на зображенні.

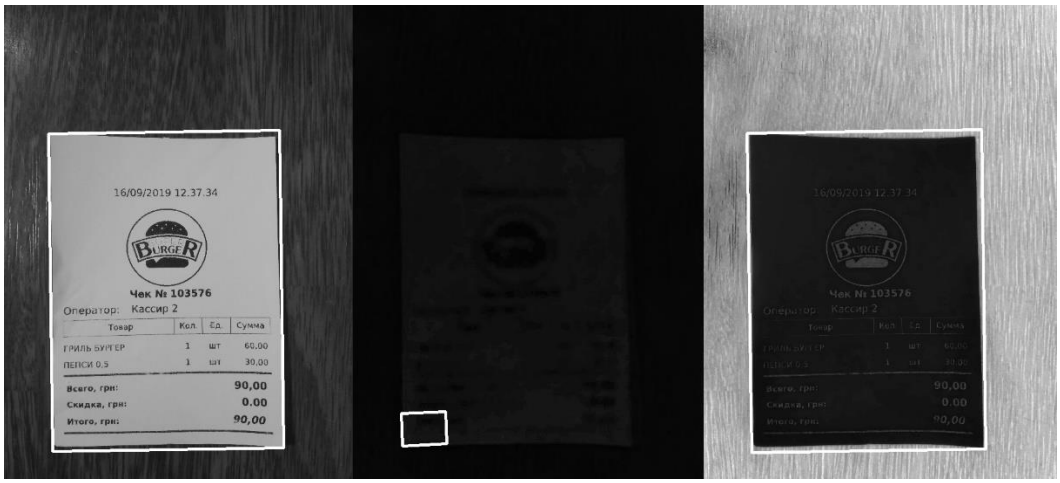

Рисунок 3.12 Приклад невірного контуру серед вірних

Досліди показали що 4 зображення для знаходження одного контуру є перебільшенням, тобто мінімум на двох зображенях з чотирьох буде знайдений вірний контур. Для знаходження кращого контуру рахуються два параметри – близкість центру контору до центру зображення та схожість з іншими контурами. Серед двох контурів які найбільш схожі між собою береться той, у котрого менша площа. Порівнюючи ці два параметри для всіх контурів знаходиться найкращий контур.

Враховуючи те, що знайдено 4 контури можливого зображення є можливість вибрати найкращий контур користувачу.

Владіючи інформацією про контур легко знайти 4 точки – кути шуканого чеку, після цього вирізаємо шукане зображення за допомогою функції warpPerspective, що допомагає врахувати перспективу та зробити рівномірною ширину та висоту шуканого зображення.

Рисунок Результат обрізання зображення за контуром

3.6. Вирізання інформаційної частини зображення

На данному етапі зображення має великий відсоток площі зайнятий шуканим чеком, проте більшість зображень, які пройшли цей єтап обробки, мають також задній план, частини якого можуть погіршити якість. Оскільки шукана інформація на чеку – це символи, тому треба отримати зображення, котре буде мати мінімальну площу, проте все ще буде мати всі символи для подальшого розпізнавання.

Проведемо схожу обробку зображення, а саме:

Зробимо зображення чорнобілим(на даному етапі значення фону не є таким важливим, тому зрізи зображення у просторі HSV не э обов'язковими)

Проведемо операцію adaptive threshold

Застосуємо фільтр medianBlur

У результаті отримаємо зображення з відчутно помітними символами, контуром та деяким шумом як зображено на рисунку 3.12.

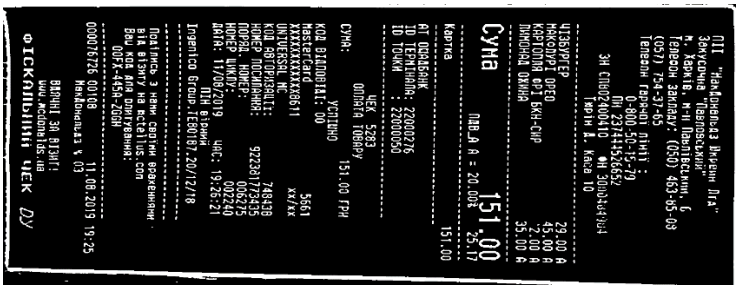

Рисунок 3.12 Зображення з високою контрастністю символів

Застосувавши до зображення функцію для знаходження контурів, отримаємо інформацію про контури всіх об’єктів на зображенні.

Проаналізувавши деяку кількість зображень чеків що пройшли цей єтап, змогли умновно розділити всі контури на декілька груп. Відношення до групи визначається функцією:

```
def getKMeansValue(contour):
    _, width, height, square = getContourProperties(contour)
    value = square
    if width != 0:
        value += (height / width) ** 2
    if height != 0:
        value += (width / height) ** 2
    return value
```

Рисунок 3.13 Функція відображення контуру до значення групування

Основними показником є площа контуру, проте, щоб значення контурів звичайнойї літери, та тонкої та великої лінії не могли бути близькими, додано суму квадратів відношень висоти до ширини контуру.

Аналіз більшості чеків дає результат – вся інформація записана шрифтами двох – трьох величин. Тобто згідно цієї функції усі контури можна розділити на 3 группи. Додамо одну группу для дуже малого шуму, а також одну групу дуже великого шуму. У результаті отримаємо 5 груп. За допомогою алгоритму К-середніх групуємо всі контури, та продовжуємо роботу лише з трьома групами, що за припущенням відповідають контурам надрукованих символів. Об’єднавши отфільтровані контури можна знову скористатися функцією findCoutnours, а потім використати вже написану функцію вирізання контуру із зображення з

урахуванням перспективи. Результати вирізання у більшості випадків повністю виключають фон із зображення, що дуже допомагає у подальшій обробці.

Рисунок 3.14 Зображення до та після обрізання за контурами символів

3.7 Знаходження слів

На даному кроці маємо зображення із мінімальною кількістю сторонніх об’єктів, тому розпочинаємо виділяті корисну інформацію. Снайшовши контури всіх символів формуємо слова за допомогою алгоритму найближчих сусідів враховуючи прості правила -літери в словах знаходяться на одній лінії та не далі деякої відстані. Оскільки лінії символів на зображенні не ідеально рівні, тому робимо припущення що кут нахилу прямої з’єднуючої центри літер до горизонталі повинен бути меньше 45 градусів, а відстань між літерами повинна бути менше половини ширини літери. Результати роботи показали що в цілому більшість сформованих слів відповідають дійсності, проте такі символи як «і» та «1» мають дуже маленьку ширину, отже при роботі алгоритму «широка» літера може приєднати до себе «і» або «1», але ці символи не можуть цього зробити,

тому декілька одиниць в ряд будуть відноситися до різних слів. Як побачимо далі, ця похибка не дуже впливає при подальшій обробці.

Отримані слова сортуємо у рядках та отримуємо координати цілих рядків. За координатами вирізаємо рядки та передаємо на розпізнавання.

3.8 Розпізнавання інформації

Для розпізнавання тексту із оброблених зобржень існує вільна програма Tesseract. Плюси використання полягають у тому, що программа розпізнає символи на багатьох мовах прямо з коробки без додаткового навчання.

Tesseract має декілька режимів роботи налаштованих на розпізнавання цілого тексту, рядка тексту, або окремих слів. Для кращого результату розпізнавання Tesseract треба виділити декілька порад:

1. Мінімальний кут нахилу текста

   Тессеракт зазвичай успішно виправляє перекос до 5 градусів. Однак найкраще виправити обертання зображення, перш ніж передавати його в OCR.

2. Мінімум 20% ширина рамки тексту

   Tesseract очікує, що на зображенні будуть порожні поля (фонового кольору) навколо тексту.

3. Правильне використання режимів роботи

   Це впливає на те, як Tesseract розбиває зображення на рядки тексту та слова.

4. Подальша обробка тексту

   Треба розглядати вихід OCR як необроблені дані. Щоб отримати хороші результати, все-таки потрібно реалізувати припущення та знання конкретної проблемної області. Якщо ви очікуєте, що в полі чи одному слові є цифри чи літери, застосуйте правильну підстановку для неоднозначних символів. Наприклад: 5- S,

   1-I, 0-O, 2-Z, 4-A, 8-B.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Опис програмного продукту

Дана глава присвячена кількісній оцінці матеріальних витрат на виробництво представленого в дипломній роботі додатоку. Додаток розроблений для розпізнавання фінансових документів, а саме товарних чеків. Він виділяє з зображення інфомрацію щодо місця здійснення покупки, дати покупки, загальної суми та назви і суми окремих товарів.

Під програмним продуктом (ПП) розуміється програмне забезпечення (ПЗ) як результат людської діяльності, виставлений на ринку масового покупця в якості товару і має ненульову споживчу вартість.

Продукт розроблений в програмному середовищі PyCharm.

4.2 Аналіз ринків збуту

Дослідження ринку збуту виявило, що потенційні покупці програмного продукту можуть бути:

1. Розробники додатків, яким потрібен програмний продукт як бібліотека
2. Компанії, що процюють з великим обігом фінансових документів та потребують автоматизації

4.2 Визначення трудовитрат робіт

Метою економічного розділу є розрахунок собівартості і вартості програмного продукту.

Для того щоб оцінити вартість розроблюваного програмного продукту необхідно:

• скласти перелік робіт, які слід виконати, потім розрахувати трудовитрати на їх виконання;

• розрахувати заробітну плату розробників;

• розрахувати витрати на матеріали, комплектуючі та машинний час;

• відрахування на соціальні заходи.

У витрати на розробку програмного продукту також входять: вартість малоцінних і швидкозношуваних предметів, вартість оренди комп'ютера, відрахування з заробітної плати і т.д.

До переліку етапів робіт, які необхідно виконати входить:

• формулювання постановки задачі;

• проектування програмного продукту;

• розробка програмного продукту;

• впровадження продукту.

Тривалість кожного етапу визначається за формулою 4.1.

$$T = \frac{t}{n}, \tag{4.1}$$

де T – тривалість етапу в робочих днях;

t – трудовитрати етапу;

n – кількість виконавців, одночасно зайнятих на певному етапі роботи.

4.3 Перелік робіт для створення програмного продукту

Для керівництва ходом робіт і ведення всього проекту в цілому необхідна посаду керівника. Для проектування підсистеми, її подальшої налагодження і введення в експлуатацію необхідно участь програміста.

Розрахуємо тривалість розробки продукту за видами робіт. Результати розрахунків містяться в таблиці 4.1.

Таблиця 4.1– Перелік робіт

№ етапу	Найменування етапу	Тривалість, дн.		Кількість, лд.		Трудовитрати, лд.-дн.
		Керівник	Програміст	Керівник	Програміст	
Розробка технічного завдання						
1	Розробка технічного завдання на постановку задачі	7		1		7
Разом:		7		1		7
Постановка задачі						
	Розробка математичної моделі і алгоритмів	1		1		1
	Розробка опису завдання і технічного завдання	3		1		3
Разом:		4		1		4
Розробка програмного продукту						
1	Розробка алгоритмів		1		1	1
3	Розробка програми		7		1	7
4	Тестування		3		1	3
Разом:			11		1	11
Всього:		5	11	1	1	16

В кінцевому підсумку, ми отримали, що термін створення програмного продукту – 16 днів.

4.4 Склад виконавців роботи та розрахунок заробітної плати

Дані про посадові оклади і склад виконавців роботи занесені в таблицю 4.2. Тривалість робочого місяця будемо вважати 22 дня. Робочий день – восьмигодинний.

Таблиця 4.2 – Склад виконавців роботи

Посада	Посадовий оклад, грн.	
	Місячний	Добовий
Керівник	8800	400
Програміст	7200	327,2

Заробітна плата – винагорода за працю залежно від кваліфікації працівника, складності, кількості, якості та умов виконуваної роботи, а також компенсаційні виплати і стимулюючі виплати.

До витрат на заробітну плату праці відносяться основна і додаткова заробітна плата персоналу, зайнятого безпосередньо при виконанні конкретної теми: науковці, науково-технічний, науково-допоміжний персонал і виробничі робітники.

Основна заробітна плата ($ЗП_{осн}$) складається з суми середньої добової заробітної плати керівника та програміста, помноженої на тривалість їх праці відповідно і розраховується за формулою 4.2.

$$ЗП_{осн} = ЗП_{добкер} * Т_{кер} + ЗП_{добпрог} + Т_{прог}, \quad (4.2)$$

де $ЗП_{добкер}$, $ЗП_{добпрог}$ – добова заробітна плата керівника і програміст відповідно (таблиця 4.2);

$Т_{кер}$, $Т_{прог}$– тривалість праці керівника і програміста відповідно (таблиця 4.1).

Таким чином, основна заробітна плата, розрахована за формулою (4.2) дорівнює:

$$ЗП_{осн} = 400 * 5 + 327{,}2 * 11 = 2317{,}8 + 2863{,}8 = 5599{,}2 (грн).$$

Далі проводиться розрахунок додаткової заробітної плати ($ЗП_{дод}$), яка становить 20% від основної заробітної плати і розраховується за формулою 4.3.

$$ЗП_{дод} = ЗП_{осн} * Н_{дод}, \quad (4.3)$$

де $Н_{дод}$ – коефіцієнт додаткової зарплати, рівний 20%.

Отже, за формулою (4.3) додаткова заробітна плата дорівнює:

$$ЗП_{дод} = 5599{,}2 * 0{,}2 = 1119{,}84 (грн).$$

Разом, загальний фонд заробітної плати становить:

$$ЗП = ЗП_{осн} + ЗП_{дод} = 5599{,}2 + 1119{,}84 = 6719{,}04 (грн).$$

Нарахування на заробітну плату (єдиний соціальний внесок – $ЗП_{соц}$) складають 22% і розраховується за формулою 4.4.

$$ЗП_{соц} = ЗП * Н_{соц}, \quad (4.4)$$

де $Н_{соц}$– коефіцієнт єдиного соціального внеску, рівний 22%.

Отже, за формулою (4.4) нарахування на заробітну плату становлять:

$$ЗП_{соц} = 6719{,}04 * 0{,}22 = 1478{,}18 (грн).$$

4.5 Перелік необхідного обладнання для створення програмного продукту

Вартість необхідного обладнання, а також їх призначення і необхідну кількість представлені в таблиці 4.3.

Таблиця 4.3 – Вартість обладнання

№ п/п	Найменування матеріалу	Призначення	Кількість, шт.	Ціна за одиницю, грн.
1	Ноутбук ACER APSIRE E5 - 511	Робота за програмним середовищем	1	17000
Разом				17000

Проведемо розрахунок амортизації (АМ) обладнання, в нашому випадку для ноутбука ACER APSIRE E5 - 511. Амортизацію обладнання як елемент собівартості даної продукції пропонується розраховуватимуть за такою методикою:

З урахуванням первісної вартості обладнання і річної норми амортизації розраховується річна сума амортизаційних відрахувань по даному виду обладнання за формулою 4.5.

$$АМ_{jрік} = С_j * Н_{ам}, \quad (4.5)$$

де $Н_{ам}$– коефіцієнт річної норми амортизації, рівний 25%;

$С_j$ – первісна вартість обладнання,;

$АМ_{jрік}$ – сума річної амортизації обладнання j виду.

Визначається величина амортизаційних відрахувань у розрахунку на одну годину роботи обладнання даного виду за формулою 4.6.

$$\text{АМ}_{\text{jгод}} = \frac{\text{АМ}_{\text{jрік}}}{\text{T}_\text{j}}, \quad (4.6)$$

де $\text{АМ}_{\text{jгод}}$ – величина амортизації обладнання даного виду протягом однієї години його використання;

T_j – річний фонд роботи даного обладнання, рівний 1844 годинам.

В залежності від часу використання обладнання в процесі виготовлення продукту розраховується розмір амортизаційних відчислень, пов'язаних з виробництвом одиниці даного виду за формулою 4.7.

$$\text{АМ}_j = \text{АМ}_{\text{jгод}} * t_j, \quad (4.7)$$

де АМ_j – амортизація обладнання даного виду при виготовленні продукту; t_j – час використання обладнання даного виду при виготовленні продукту, рівний 128 годинам.

Якщо використовувати різне обладнання, то величина амортизації як елемент собівартості продукту повинна бути визначена шляхом підсумовування амортизаційних відрахувань по окремих видах обладнання за формулою 4.8.

$$\text{АМ} = \sum_{i=1}^{m} \text{АМ}_j, \quad (4.8)$$

деАМ – сума амортизації по продукту;

m – кількість видів обладнання.

Отже, за формулою (4.5) річна сума амортизаційних відрахувань для ноутбука ACER APSIRE E5 - 511:

$$\text{АМ}_{\text{ноутрік}} = 17000 * 0{,}25 = 4250 \text{ (грн)}.$$

Тоді, за формулою (4.6) величина амортизаційних відрахувань на одну годину для ноутбука ACER APSIRE E5 - 511:

$$\text{АМ}_{\text{ноутгод}} = 4250 / 1844 = 2{,}3 \text{ (грн)}.$$

За формулою (4.7) величина амортизаційних відрахувань залежно від часу використання ноутбука ACER APSIRE E5 - 511:

$$\text{АМ}_{\text{ноут}} = 2{,}3 * 128 = 294{,}4 \text{ (грн)}.$$

За формулою (4.8) сума амортизації по продукту:

$$\text{АМ} = 294{,}4 \text{(грн)}.$$

Вартість технологічної електроенергії обчислюється за формулою 4.9.

$$S_Э = T_{ар} * T * W, \quad (4.9)$$

де $T_{ар}$ – тариф електроенергії за один кВт, рівний 2,01 грн.;

T – кількість годин роботи;

W – споживана технологічна потужність (W = 0,032 кВт).

Отже, за формулою (4.9) вартість технологічної електроенергії:

$$S_Э = 2{,}01 * 128 * 0{,}032 = 8{,}23(\text{грн}).$$

Вартість освітлювальної електроенергії розраховується за формулою 4.10.

$$S = T_{ар} * T * W, \quad (4.10)$$

де W – споживана потужність освітлювальним прибором (W = 0,1 кВт).

Отже, за формулою (4.10) вартість освітлювальної електроенергії:

$$S = 2{,}01 * 128 * 0{,}1 = 25{,}7 \text{ (грн)}.$$

4.6 Розрахунок собівартості та повної вартості програмного продукту

Собівартість – це вартісна оцінка використовуваних в процесі виробництва продукції (робіт, послуг) природних ресурсів, сировини, матеріалів, палива, енергії, основних фондів, трудових ресурсів та інших втрат на її виробництво і реалізацію. Собівартість дорівнює сумі всіх витрат на розробку проекту і розраховується за формулою 4.11.

$$C = ЗП + ЗП_{соц} + АМ + S_Э + S, \quad (4.11)$$

Отже, з формули (4.11) собівартість становить:

$$C = 6719{,}04 + 1478{,}18 + 294{,}4 + 8{,}23 + 25{,}7 = 8525{,}55 \text{ (грн)}.$$

Калькуляційні розрахунки на розробку програмного забезпечення представлені в таблиці 4.4.

Таблиця 4.4 – Статті калькуляції на розробку програмного продукту

№	Стаття калькуляції	Витрати, грн.
1	Основна заробітна плата	5599, 2
2	Додаткова заробітна	1119,84
3	Загальна заробітна плата	6719,04
4	Єдиний соціальний внесок	1478,18
5	Амортизація	294,4
6	Вартість технологічної електроенергії	8,23
7	Вартість освітлювальної електроенергії	25,7
8	Загальна собівартість розробки	8525,55

4.7 Висновок

- В даному розділі було:
- визначено трудовитрати робіт створення програмного продукту;
- призначено виконавців роботи, а також проведено розрахунки заробітної плати та соціального внеску, які склали 6719,04 грн. та 1478,18 грн. відповідно;
- складено перелік робіт для кожного з виконавців;
- складено перелік необхідного обладнання, для яких розраховані витрати на амортизацію у розмірі 294,4 грн.;
- проведено розрахунок вартості електроенергії: технологічної та освітлювальної. Витрати склали 8,23 грн. та 25,7 грн. відповідно.
- проведено розрахунки собівартості(8525,55 грн.).

ВИСНОВКИ

В даній роботі був розроблений python додаток для розпізнавання фінансових документів, а саме :

1. Розроблен алгоритм роботи додатку
2. Спроектована архітектура додатку
3. Розділена функціональність на роботу в декількох режимах
4. Протестована функціональність на достатньому об’ємі даних
5. Виділено особливості зображення що покращують результат
6. Розрахована ефективність роботи за таким додатком
7. Представлені результати розробки додатку
8. Підрахована вартість розробки додатку

В подальшому рекомендується поліпшити результати роботи додатку шляхом навчання OCR за допомогою тренувальних даних. Також все ще залищаються шляхи для оптимізації роботи додатка та зменшення середнього часу обробки зображення.

В данній роботі додаток розроблявся базуючись на порівняно невеликій базі вхідних даних, тому деякі вхідні дані можуть значно відрізнятись від очікуваних і таким чином точність розпізнавання буде вкрай низькою.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Wei Xu and Jane Mulligan. Performance evaluation of color correction approaches for automatic multi-view image and video stitching. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 263–270. IEEE, 2010.
2. Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008.
3. Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. Pattern recognition, 13(2):111–122, 1981.
4. Josef Bigun. Vision with direction. Springer, 2006.
5. Gunilla Borgefors. Distance transformations in digital images. Computer vision, graphics, and image processing, 34(3):344–371, 1986.
6. John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, (6):679–698, 1986.
7. M. Danelljan, F.S. Khan, M. Felsberg, and J. van de Weijer. Adaptive color attributes for real-time visual tracking. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 1090–1097, June 2014.
8. Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In Computer Vision (ICCV), 2013 IEEE International Conference on, pages 1841–1848. IEEE, 2013.
9. Р. Гонсалес, Р. Вудс Цифровая обработка изображений. М.: Техносфера, 2005
10. Eduardo SL Gastal and Manuel M Oliveira. Domain transform for edge-aware image and video processing. In ACM Transactions on Graphics (TOG), volume 30, page 69. ACM, 2011.
11. N Guil, José María Gonzalez-Linares, and Emilio L Zapata. Bidimensional shape detection using an invariant approach. Pattern Recognition, 32(6):1025–1038, 1999.

12. Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In Computer Vision–ECCV 2010, pages 1–14. Springer, 2010.
13. J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In proceedings of the European Conference on Computer Vision, 2012.
14. Joseph J Lim, C Lawrence Zitnick, and Piotr Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 3158–3165. IEEE, 2013.
15. Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. Computer Vision and Image Understanding, 78(1):119–137, 2000.
16. Lukáš Neumann and Jiří Matas. Text localization in real-world images using efficiently pruned exhaustive search. In in Document Analysis and Recognition, 2011 International Conference on. IEEE, 2011, pages 687–691.
17. Neumann, Matas L., and J. Scene text localization and recognition. pages 3538–3545. IEEE, 2012.
18. Wayne Niblack. An introduction to digital image processing. Strandberg Publishing Company, 1985.
19. Guang-Zhong Yang, Peter Burger, David N Firmin, and SR Underwood. Structure adaptive anisotropic image filtering. Image and Vision Computing, 14(2):135–145, 1996.
20. HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. Image and Vision Computing, 8(1):71–77, 1990.
21. C. Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In ECCV, 2014.
22. Zoran Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, volume 2, pages 28–31. IEEE, 2004.

23. И.С. Грузман, В.С. Киричук, В.П. Косых, Г.И. Перетягин, and А.А. Спектор. Цифровая обработка изображений в информационных системах. 2000.
24. Sobel I., Feldman G. «A 3x3 Isotropic Gradient Operator for Image Processing», 1968
25. Duda R., Hart P. Pattern Classification and Scene Analysis. — John Wiley and Sons, 1973. — P. 271—272
26. Solomon C.J., Breckon T.P. Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab. — Wiley-Blackwell, 2010. — ISBN 978-0470844731
27. Wilhelm Burger, Mark J. Burge. Digital Image Processing: An Algorithmic Approach Using Java. — Springer, 2007. — ISBN 978-1-84628-379-6.
28. Tim Morris. Computer Vision and Image Processing. — Palgrave Macmillan, 2004. — ISBN 978-0-333-99451-1.
29. Basim Alhadidi, Mohammad H. Zu'bi, Hussam N. Suleiman. Mammogram Breast Cancer Image Detection Using Image Processing Functions // Information Technology Journal. — 2007. — Т. 6, вып. 2. — С. 217–221.
30. Scharr, Hanno, 2000, Optimal Operators in Digital Image Processing.
31. B. Jähne, H. Scharr, and S. Körkel. Principles of filter design. In Handbook of Computer Vision and Applications. Academic Press, 1999.
32. Р. Гонсалес, Р. Вудс С. Эддинс Цифровая обработка изображений в среде Matlab. М.: Техносфера, 2006.
33. В.Т. Фисенко, Т.Ю. Фисенко, Компьютерная обработка и распознавание изображений: учеб. пособие. - СПб: СПбГУ ИТМО, 2008. –192 с.
34. Marko Tkalciˇ c, Jurij F. Tasi ˇ c, Colour spaces - perceptual, historical and applicational background, Faculty of electrical engineering University of Ljubljana
35. Burchett, K. E. (2002). "Color harmony". Color Research and Application, 27 (1), pp. 28–31.
36. Hard, A. & Sivik, L. (2001). "A theory of colors in combination – A descriptive model related to the NCS color-order system". Color Research

37. Sharma, G. (2003). Digital Color Imaging Handbook. Boca Raton, FL: CRC Press. ISBN 978-0-8493-0900-7.Agoston, Max K. (2005). Computer Graphics and Geometric Modeling: Implementation and Algorithms. London: Springer. pp. 300–306. ISBN 978-1-85233-818-3.
38. Kuehni, Rolf G. (2003). Color Space and Its Divisions: Color Order from Antiquity to the present. New York: Wiley. ISBN 978-0-471-32670-0.
39. Fairchild, Mark D. (2005). Color Appearance Models (2nd ed.). Addison Wesley
40. Компьютерное зрение в промышленности. Лекция в Яндексе 2018 [Електронний журнал]: https://habr.com/ru/company/yandex/blog/422087/
41. Машинное зрение. Что это и как им пользоваться? Обработка изображений оптического источника 2018 [Електронний журнал]: https://habr.com/ru/post/350918/
42. Системы компьютерного зрения: современные задачи и методы 2016 [Електронний журнал]: https://controlengrussia.com/innovatsii/sistemy-komp-yuternogo-zreniya-sovremenny-e-zadachi-metody/
43. A Gentle Introduction to Computer Vision [Електронний журнал]: https://machinelearningmastery.com/what-is-computer-vision/
44. ComputerVision 2018 [Електронний журнал]: https://habr.com/ru/company/funcorp/blog/351638/
45. Computer vision [Електронний журнал]: https://en.wikipedia.org/wiki/Computer_vision
46. David Marr (1982). Vision. W. H. Freeman and Company. ISBN 978-0-7167-1284-8.
47. Azriel Rosenfeld; Avinash Kak (1982). Digital Picture Processing. Academic Press. ISBN 978-0-12-597301-4.
48. Richard Hartley and Andrew Zisserman (2003). Multiple View Geometry in Computer Vision. Cambridge University Press. ISBN 978-0-521-54051-3.
49. Gösta H. Granlund; Hans Knutsson (1995). Signal Processing for Computer Vision. Kluwer Academic Publisher. ISBN 978-0-7923-9530-0.

50. Computer Vision for Beginners 2019 [Електронний журнал]: https://towardsdatascience.com/computer-vision-for-beginners-part-1-7cca775f58ef
51. 16 OpenCV Functions to Start your Computer Vision journey (with Python code) 2019 [Електронний журнал]: https://www.analyticsvidhya.com/blog/2019/03/opencv-functions-computer-vision-python/
52. Computer Vision with Python 2017 [Електронний журнал]: https://www.udemy.com/course/computer-vision-with-python/
53. Computer Vision platform using Python. [Електронний журнал]: http://simplecv.org/
54. Start here: Learn computer vision & OpenCV [Електронний журнал]: https://www.pyimagesearch.com/start-here-learn-computer-vision-opencv/
55. Computer Vision Using OpenCV [Електронний журнал]: https://dzone.com/articles/opencv-python-tutorial-computer-vision-using-openc
56. Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV".
57. Adrian Kaehler; Gary Bradski (14 December 2016). Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media. pp. 26ff. ISBN 978-1-4919-3800-3.
58. "Intel Acquires Computer Vision for IOT, Automotive | Intel Newsroom". Intel Newsroom. Retrieved 2018-11-26.
59. Peter N. Belhumeur, João P Hespanha, and David Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(7):711–720, 1997.
60. L Berger, V A Raghunathan, C Launay, D Ausserré, and Y Gallot. Coalescence in 2 dimensions: experiments on thin copolymer films and numerical simulations. The European Physical Journal B - Condensed Matter and Complex Systems, 2(1):93–99, 1998.