НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«Київський політехнічний інститут імені Ігоря Сікорського»

Приладобудівний факультет

Кафедра комп'ютерно-інтегрованих технологій виробництва приладів

РОЗРАХУНКОВО–ГРАФІЧНА РОБОТА

з дисципліни цифрова обробка сигналів та зображень

на тему «Розробка програмного забезпечення для автоматичного вирівнювання зображень за розміром, поворотом або нахилом»

Виконав(-ла):

студент(-ка) 1 курсу

групи ПБ-41мп, ПБФ

Литвиненко А. Г.

Перевірила:

Демченко М. О.

Київ 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря Сікорського”

КАФЕДРА КОМП’ЮТЕРНО-ІНТЕГРОВАНИХ ТЕХНОЛОГІЙ ВИРОБНИЦТВА ПРИЛАДІВ

Дисципліна Цифрова обробка сигналів та зображень

Спеціальність Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Курс 5 Група ПБ-41мп Семестр перший

З А В Д А Н Н Я

на розрахунково-графічну роботу студента

Литвиненко Артем Глібович

(прізвище, ім’я, по батькові)

1.Тема роботи Розробка програмного забезпечення для автоматичного вирівнювання зображень за розміром, поворотом або нахилом

2. Дата видачі завдання: 06 вересня 2024 р.

3. Початкові дані до роботи:

Опис методик розв’язання задач, що надані в спеціальній літературі з теорії надійності, методи математичної статистики, характеристики технічного засобу автоматизованої системи

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці) Огляд літературних речей, вивчення методу розв’язання задачі, опис математичних методів і моделей, опис алгоритму програми, розробка програми, тестування і налагодження за контрольним прикладом, опис технічного забезпечення підсистеми, опис лінгвістичного забезпечення підсистеми, розробка методичного забезпечення підсистеми.

5. Перелік графічного матеріалу

6. Строк здачі студентом завершеної роботи 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів розрахункової роботи	Строк виконання етапів роботи	Примітки
1	Огляд літературних джерел та постановка задачі	27 вересня 2024 р.	+
2	Вивчення вибраного методу розв’язання задачі	04 жовтня 2024 р.	+
3	Розроблення алгоритмів розв'язання задачі	18 жовтня 2024 р.	+
4	Розроблення програм розв’язання задачі згідно з вибраним методом	08 листопада 2024 р.	+
5	Відлагодження програм за контрольними прикладами	22 листопада 2024 р.	+
6	Оформлення пояснювальної записки та презентації РГР	06 грудня 2024 р.	+
7	Захист розрахункової роботи	20 грудня 2024 р.	+

Студент _____ Артем ЛИТВИНЕНКО

(підпис) (ім’я, прізвище)

Керівник _____ Марія ДЕМЧЕНКО

(підпис) (ім’я, прізвище)

“ _20_ ” __грудня___2024 р.

Вступ

У сучасному світі обробка зображень займає ключове місце в різноманітних галузях, включаючи медицину, геоінформаційні системи, розваги, безпеку та багато інших. Одним із фундаментальних завдань у цій сфері є автоматичне вирівнювання зображень, яке передбачає приведення двох або більше зображень до спільної геометричної конфігурації. Це необхідно для подальшого аналізу, порівняння або об'єднання інформації з різних джерел.

Автоматичне вирівнювання зображень включає корекцію розміру, повороту та нахилу зображень, що дозволяє точно збігати їхні ключові елементи. Такий процес є критично важливим при створенні панорам, де окремі фотографії об'єднуються в єдине широкоформатне зображення, або при стабілізації відео, де необхідно усунути небажані рухи камери. У медицині вирівнювання зображень дозволяє порівнювати різні знімки пацієнта для виявлення змін або аномалій.

Реалізація алгоритму автоматичного вирівнювання зображень вимагає застосування методів комп'ютерного зору та машинного навчання. Основними етапами цього процесу є виявлення ключових точок на зображеннях, їхнє описання за допомогою спеціальних дескрипторів, знаходження відповідностей між цими точками та обчислення геометричних перетворень, які забезпечують точне вирівнювання. Сучасні бібліотеки, такі як OpenCV, надають потужні інструменти для реалізації цих етапів, роблячи процес більш доступним та ефективним.

Однак автоматичне вирівнювання зображень стикається з низкою викликів, серед яких варто виділити різноманітність умов освітлення, наявність шумів, різницю у масштабах та перспективі зображень. Для подолання цих

труднощів необхідно ретельно підбирати методи обробки та оптимізувати параметри алгоритмів.

Ця робота присвячена детальному опису процесу реалізації алгоритму автоматичного вирівнювання зображень з урахуванням їхнього розміру, повороту та нахилу. Ми розглянемо основні концепції та етапи цього процесу, використовуючи мову програмування Python та бібліотеку OpenCV. Крім того, будуть наведені практичні приклади коду, що ілюструють кожен з етапів алгоритму, а також обговорені можливі напрямки покращення та оптимізації для досягнення найкращих результатів.

Завдяки цій роботі читач зможе отримати глибоке розуміння принципів автоматичного вирівнювання зображень та навчитись ефективно впроваджувати ці методи у своїх проектах, що сприятиме вирішенню широкого спектру задач у сфері обробки зображень.

1. Огляд літературних джерел

Python пропонує величезну кількість бібліотек, які дають змогу зробити життя програмістів простішим і цікавішим. Давайте розглянемо деякі з них та їхнє застосування:

- **NumPy** – для роботи з багатовимірними масивами та виконання математичних операцій;
- **Pandas** – надає структури даних, як-от DataFrame, що дають нам змогу зручно обробляти, аналізувати та маніпулювати таблицями даних;
- **Matplotlib** – якщо ви хочете створити красиві графіки та візуалізації;
- **TensorFlow** – якщо ви хочете зануритися у світ машинного навчання і створення нейронних мереж;
- **Beautiful Soup** – якщо вам потрібно витягти дані з HTML або XML файлів;
- **Pygame** – якщо ви мрієте про створення своєї власної гри;

Це лише деякі з популярних бібліотек Python. Загалом їх набагато більше, і кожна з них пропонує свої унікальні можливості та застосування;

Встановлення бібліотек Python

Встановлення бібліотек Python – досить простий процес:

1. Переконайтеся, що у вас встановлено Python на комп’ютері. Якщо ні, завантажте та встановіть його з офіційного сайту.
2. Відкрийте командний рядок або термінал.

3. Введіть команду “pip install ім’я_бібліотеки” для встановлення потрібної бібліотеки. Наприклад, “pip install numpy” для встановлення NumPy. Натисніть Enter і дочекайтеся завершення встановлення.
4. Якщо у вас є файл із залежностями (requirements.txt), використовуйте команду “pip install -r requirements.txt”, щоб встановити всі бібліотеки з файлу.
5. Щоб встановити конкретну версію бібліотеки, вкажіть її в команді встановлення. Наприклад, “pip install numpy==1.19.2” встановить версію 1.19.2 NumPy.

Бібліотеки для роботи з графікою

Якщо ви хочете додати вау-ефекти до ваших проєктів або створити вражаючі візуалізації даних, бібліотеки для роботи з графікою в Python допоможуть вам у цьому. Є кілька популярних бібліотек і приклади їхнього використання:

- **Matplotlib** – популярна бібліотека для графіків і візуалізацій у Python. Вона пропонує різноманітні стилі та типи графіків: лінійні, стовпчасті, кругові діаграми, теплові карти та інші. Ви можете налаштовувати кольори, мітки та осі для створення професійних графіків. Наприклад, ви можете відобразити графік продажів за місяцями або візуалізувати розподіл даних. Також можна створити стовпчасту діаграму для відображення доходів різних відділів компанії.

- **Seaborn** – бібліотека, заснована на Matplotlib, пропонує зручні функції для створення стильних та інформативних графіків. Вона включає готові теми оформлення, що роблять графіки привабливими та професійними. Також у Seaborn є спеціальні функції для статистичних графіків, включно з діаграмами розкиду і ящиками з вусами. Використовуючи Seaborn і

ящики з вусами, ви зможете порівнювати доходи між групами клієнтів і виявляти закономірності та відмінності в їхньому розподілі. Це корисно під час ухвалення маркетингових рішень, наприклад, аналізу доходів за сегментами клієнтів і визначення найпривабливіших груп для цільових маркетингових стратегій.

- **Plotly** – бібліотека, що пропонує інтерактивні графіки та візуалізації, які можна вбудовувати у вебсторінки та інтерактивні додатки. Вона підтримує різні типи графіків: лінійні, стовпчасті, розкиду, поверхні тощо. Ви можете додавати інтерактивні елементи, такі як навігація, вибір даних та інструменти масштабування. Наприклад, можна створити інтерактивну карту з точками, що представляють міста, а колір кожної точки відображатиме середню температуру. Користувачі зможуть навести курсор, щоб отримати детальну інформацію про температуру в кожному місті.
- **OpenCV** – чудова бібліотека для роботи з комп'ютерним зором і обробки відео. Вона надає потужні функції для обробки та аналізу зображень. Можна розпізнавати об'єкти, витягувати ознаки, застосовувати фільтри та багато іншого. OpenCV широко використовується в галузі комп'ютерного зору, робототехніки та автоматичного відеоаналізу. Наприклад, для створення системи стеження за рухом можна використовувати OpenCV для аналізу відеопотоку з веб-камери. Можна виявляти рухомі об'єкти, виділяти їхні межі та відстежувати їхнє положення в режимі реального часу. Це корисно для систем відеоспостереження або інтерактивних ігор.

2. Вивчення вибраного методу розв’язання задачі

Морфологічні перетворення - це кілька простих операцій, що застосовуються до елементів зображення. Зазвичай така обробка проводиться на бінарних зображеннях. При проведенні морфологічної обробки вхідними даними є зображення та структурний елемент або ядро, яке визначає характер роботи процедури. В результаті морфологічної обробки отримують деякі атрибути для подальшого аналізу [1].

Зазвичай структурний елемент має розмір 3×3 і має початок у центральному пікселі (рис.7.1). Він переміщається по зображенню, і на кожному пікселі зображення його елементи порівнюються з набором базових пікселів. Якщо два набори елементів відповідають заданій умові, визначеній оператором, піксель під початковою точкою структурного елемента встановлюється на попередній визначене значення (0 або 1 для бінарних зображень). Таким чином, морфологічний оператор визначається його структурним елементом і прикладним оператором.

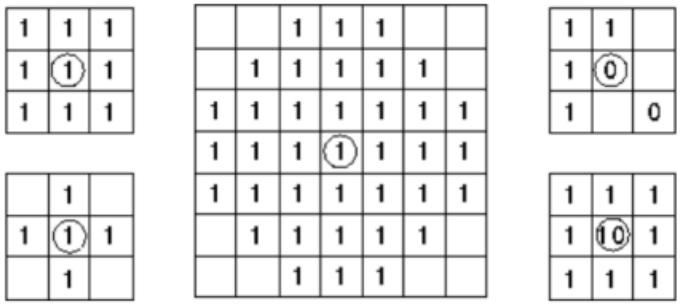

Рис. 1. Приклади структурних елементів [2].

Двома основними морфологічними операторами є ерозія (звуження) та дилатація (розширення). Всі інші операції базуються на логічних операціях над множинами (перетин, об’єднання, включення, доповнення).

Для базових морфологічних операторів структурний елемент містить лише пікселі переднього плану (тобто одиниці) та «фон». Ці оператори, які є комбінацією ерозії та дилатації, часто використовуються для вибору або придушення особливостей певної форми, наприклад , видалення шуму із зображень або вибору об’єктів із певним напрямком [2].
Більш детально опис функцій бібліотеки OpenCV для морфологічної обробки зображень наведено в таблиці 7.1.

Таблиця 7.1. Функції морфологічної обробки [1].

Ім’я функції Опис функції	Ім’я функції Опис функції
cv.erode(src, kernel,anchor, interpolation)	Функція ерозії зображення Обов’язковим параметром є ядро, за яким проводиться ерозія. Також можна вказати якір для зміни центру ядра, кількість операцій чи тип рамки.
cv.dilate (src, kernel, ancho, interpolation)	Функція дилатації зображення. Обов’язковим параметром є ядро, за яким проводиться дилатація. Також можна вказати якір для зміни центру ядра, кількість операцій чи тип рамки.
	Функція виконує розширені морфологічні операції над зображенням. Обов’язковими параметром є ядро та тип операції. Також можна вказати якір для зміни центру ядра, кількість операцій чи тип рамки.

cv.morphologyEx (src, op, ancho, kernel, interpolation)	Типи морфологічних операцій: cv.MORPH_ERODE – ерозія cv.MORPH_DILATE – дилатація cv.MORPH_OPEN – відкриття cv.MORPH_CLOSE – закриття cv.MORPH_GRADIENT – градієнт cv.MORPH_TOPHAT – «циліндр» cv.MORPH_BLACKHAT – «чорний капелюх»

Ерозія та дилатація Основна ідея ерозії така ж, як і ерозії ґрунту, вона розмиває межі об’єкта переднього плану. Зазвичай передній план має білий колір. Ядро ковзає по зображенню. Піксель в результаті на вихідному зображенні (або 1, або 0) вважатиметься 1, лише тоді, якщо всі пікселі під ядром дорівнюють 1, інакше він розмивається (обнулюється). Отже, відбувається те, що всі пікселі біля межі будуть відкинуті залежно від розміру ядра. Тому товщина або розмір об’єкта переднього плану зменшується або просто зменшується біла область на зображенні (рис.7.2). Це корисно для видалення невеликих білих шумів, відокремлення двох з’єднаних об’єктів тощо.

Рис. 2. Ерозія зображення [2].

Протилежною до ерозії операцією є дилатація. В даному випадку на результуючому зображенні елемент пікселя дорівнює «1», якщо хоча б один піксель під ядром має значення «1». Тому збільшується біла область на зображенні або збільшується розмір об’єкту переднього плану.

Зазвичай у таких випадках, як видалення шуму, ерозія супроводжується розширенням. Оскільки ерозія видаляє не тільки білі вкраплення шуму, але вона також зменшує наш об’єкт. Тому, щоб відновити його розміри ми використовуємо дилатацію. Це також корисно для з’єднання зламаних частин предмета.

Рис. 3. Дилатація зображення [2]

До розширених морфологічних операцій відносять відкриття, закриття, виділення границь, «циліндр», «чорний капелюх». Всі вони базуються на операціях ерозії та дилатації, а також логічними операціях над множинами. Розглянемо більш детально кожну з них.

Відкриття. Основний ефект відкриття схожий на ерозію, оскільки він полягає у видалені деяких пікселів переднього плану (світлі) з країв областей пікселів переднього плану. Однак він менш руйнівний, ніж ерозія. Відкриття визначається як ерозія з подальшим розширенням *з використанням того самого структурного елемента для обох операцій.*

$$open(src, kernel)=dilate(erode(src, kernel))$$

Закриття. Основний ефект відкриття схожий на дилатацію, оскільки він полягає у збільшенні межі передніх (світлих) областей на зображенні (і зменшенні дірки фонового кольору в таких областях), але воно менш руйнує вихідну форму межі.
Закриття визначається як дилатація з подальшим розмиванням *з використанням того самого структурного елемента для обох операцій.*

$$close(src, kernel)=erode(dilate(src, kernel))$$

Виділення меж (градієнт). Градієнт описує контур зображення. Він може визначатись як різниця між двома зображеннями, наприклад що піддавалось розширенню та ерозії, або різниці вхідного зображення та того, що піддавалось ерозії, та ін. В бібліотеці OpenCV оператор виділення меж описується як різниця між дилатацією та ерозією.

$$morph\_grad(src, kernel)=dilate(src, kernel)-erode(src, kernel)$$

«Циліндр». Визначає різницю між вхідним зображенням і відкриттям зображення.

$$tophat(src,element)=src-open(src,element)$$

«Чорний капелюх». Визначає різницю між закриттям зображення та вхідним зображенням.

$$\boldsymbol{blackhat(src,element)=close(src,element)-src}$$

В бібліотеці OpenCV є функція, яка об’єднує ці оператори: ***cv.morphologyEx (src, op, ancho, kernel, interpolation)***. Саме параметр ***op*** задає тип обробки зображення.

Розглянемо як впливають морфологічні операції різними операторами на структурні особливості зображення зображення. Для цього створимо функцію *morphology (img, list_op, n, m)*, вхідними параметрами якої є саме зображення, список морфологічних операцій та розміри ядра. Створена функція повертає список зображень, що піддаються морфологічній обробці.

3. Опис математичних методів і моделей

Етап опису математичних методів і моделей передбачає опис математичних методів і моделей, які використовуються у процесі програмного забезпечення для автоматичного вирівнювання зображень за розміром, поворотом або нахилом.

Опис методики проєктування

Для коректної роботи програми слід використоувати звичайні формати фотографій такі як JPEG або PNG. Далі важливо обрати конкретне значення ядра і порогу (наприклад, значення ядра – 10, а порогу – 100). Програма надає автоматично знайдені контури фігури, якщо знайдені контури і крайні точки коректні, то потрібно підтвердити подальшу роботу програми, якщо обрані крайні точки і контури не відповідають дійсності, програми запропонує обрати відповідні характеристики знову. Для цього є можливість введення різного значення ядра і порогу, щоб мати можливість коректно знайти відповідні контури фігури.

4. Опис алгоритму проектування

Алгоритм програми представлений у вигляді блок-схем котрі наведені нижче.

Основна програма

Рис. 1. Блок-схема основної програми.

Блок 0 – початок; Блок 1 – завантаження відповідних бібліотек; Блок 2 – виведення запиту на введення (шлях до зображення); Блок 3 – виклик функції transformMain(); Блок 4 – кінець.

Функція автоматичного вирівнювання зображення transformMain(arc)

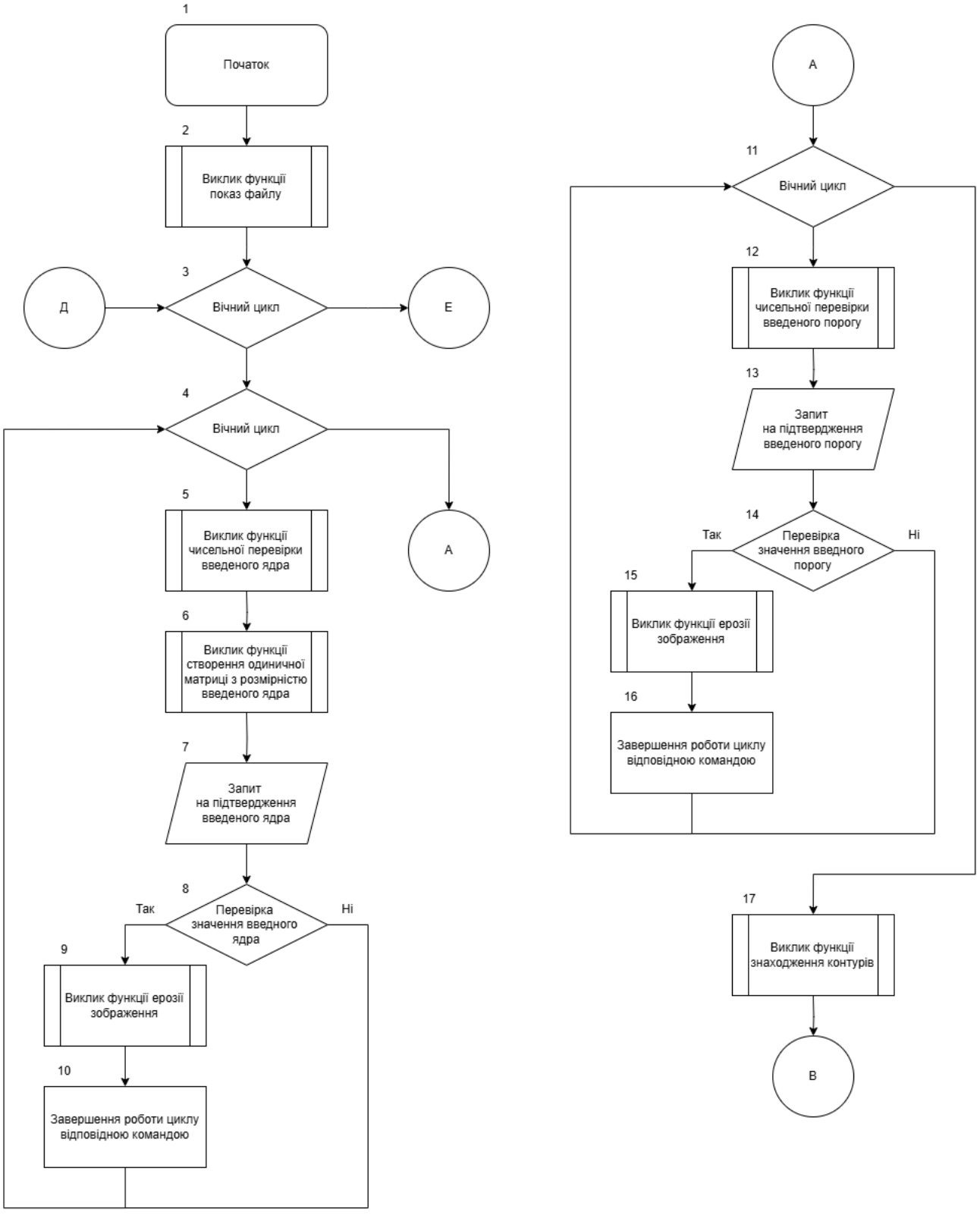

Рис. 2. Блок-схема функції transformMain(arc).

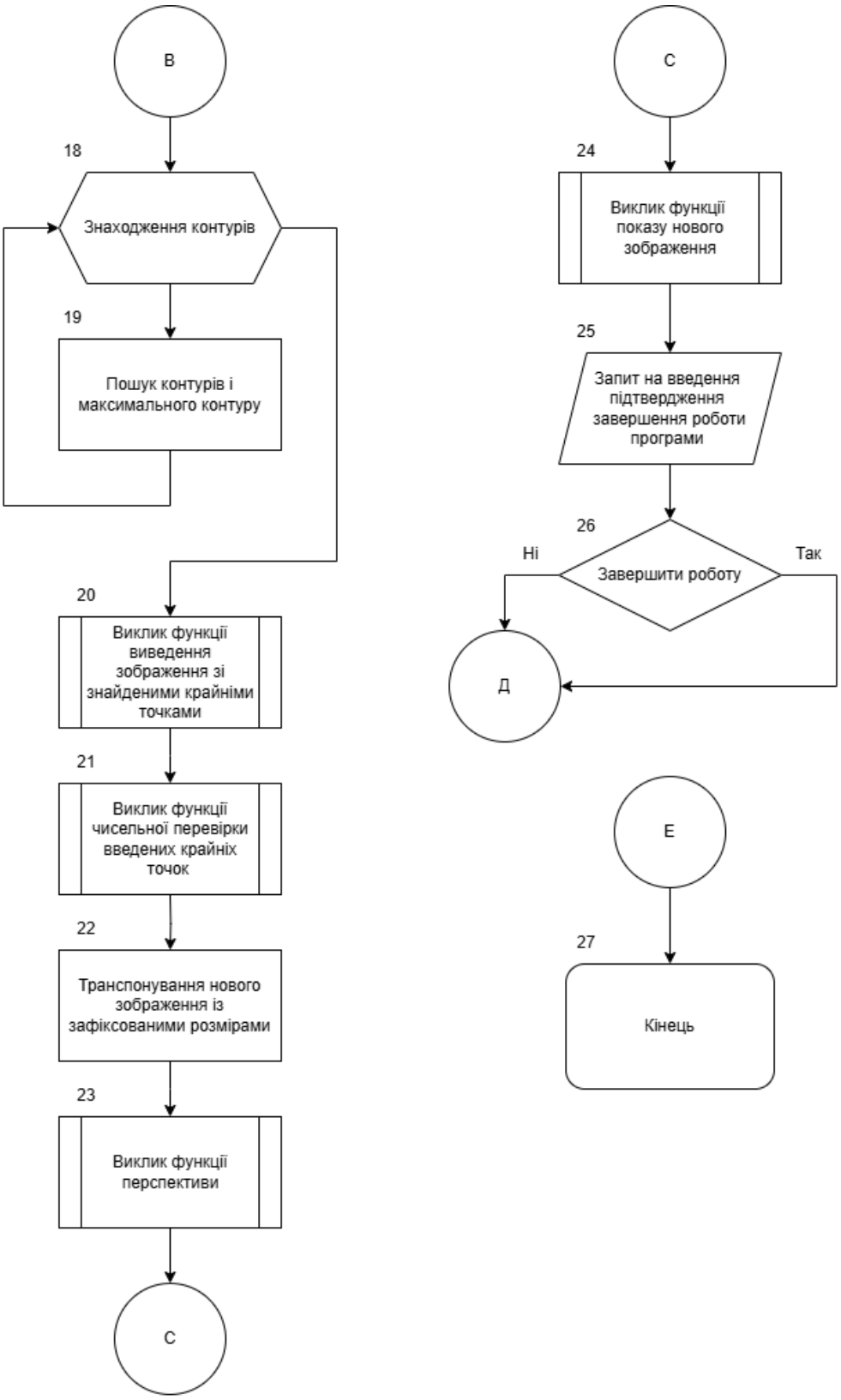

Рис. 3. Продовження блок-схеми функції transformMain(arc).

Блок 1 – початок; Блок 2 – виклик функції показ файлу; Блок 3 – Запуск нескінченого циклу для повторої роботи програми; Блок 4 – запуск нескіченного циклу для підтвердження обраного ядра; Блок 5 – виклик функції чисельної перевірки введеного значення ядра; Блок 6 – викилик функції kernel(n), яка створює одиничну матрицю з розмірністю введеного ядра;
Блок 7 – запит на підтвердження введеного значення ядра; Блок 8 – перевірка введеного значення ядра, якщо умова виконується, то переходимо до
Блоку 9 – виклик функціх ерозії зображення і далі до Блоку 10 – завершення роботи циклу блоку 4 відповідною командою з відти слідуємо до Блоку 11 – нескічений цикл; Блок 12 – виклик функції чисельної перевірки введеного значення порогу; Блок 13 – запит на підтвердження введеного порогу; Блок 14 – перевірка значення введеного порогу, якщо «Так» то переродимо до блоку 15 – виклик функції ерозії зображення, далі до блоку 16 – завершення роботи циклу. Якщо Блок 14 не викокується, то знову повертаємося до блоку 11.
Блок 17 – виклик функції знаходження контурів; Блок 18 – цикл знаходження контурів далі йде блок 19 – пошук контурів і максимального контуру;
Блок 20 – виклик функції виведення щображення зі знайденими крайніми точками; Блок 21 – Запит на введення відповідних даних крайніх точок;
Блок 22 – транспонування нового зображення із зафіксованими розмірами;
Блок 23 – виклик функції перспективи; Блок 24 – виклик функції показу нового зображення. Блок 25 – запит на введення підтвердження завершення роботи, якщо «Ні» повертаємося до блоку 3, якщо «Так» виходимо з циклу;
Блок 27 – кінець.

Функція чисельної перевірки введених користувачем даних get_positive_integer (prompt):

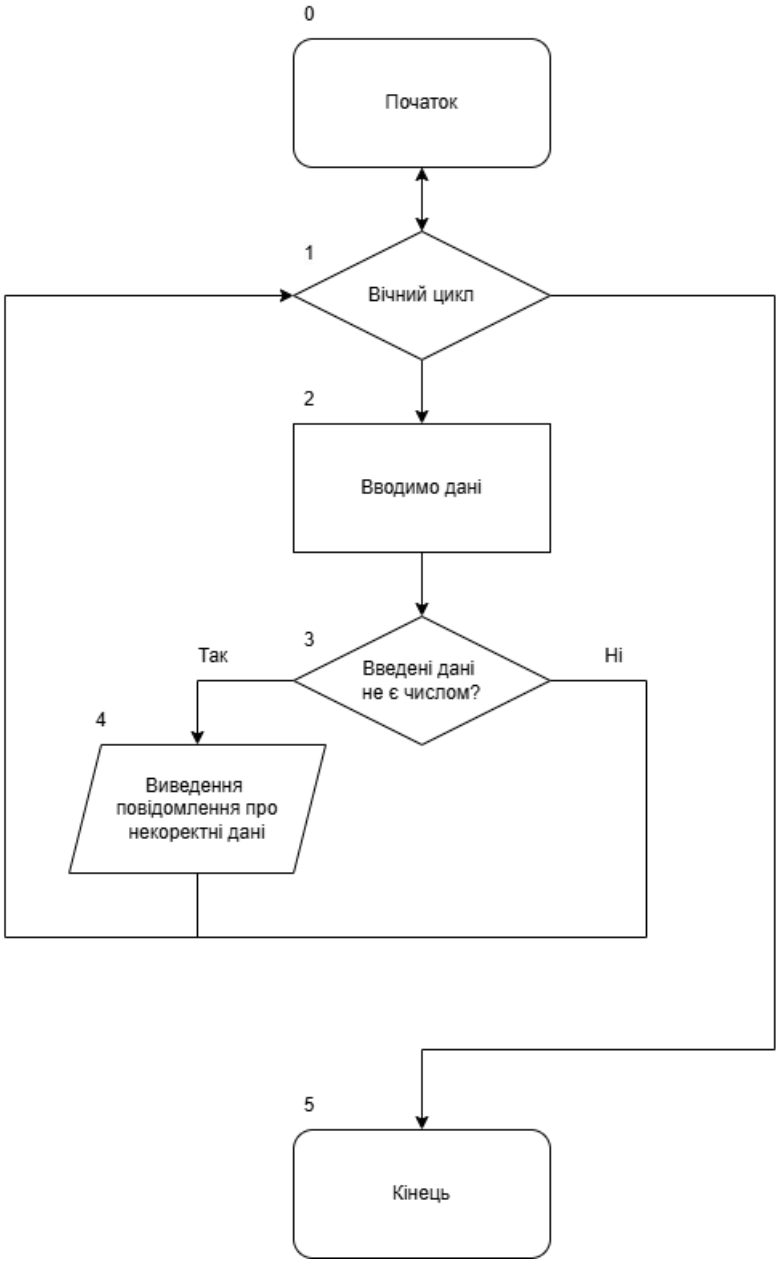

Рис. 4. Блок-схема функції чисельної перевірки.

Блок 0 – початок; Блок 1 – нескінчений цикл; Блок2 – Вводимо дані; Блок 3 – перевірка чи не є введені дані є числом, якщо так то переходимо до блоку 4 – виведення повідомлення, що введені дані не є числом, якщо усова не виконується то цикл завершується; Блок 5 – кінець.

5. Розробка програми проектування

Розробка програми проектування є ключовим етапом в будь-якому процесі створення програмного продукту. Завдяки програмному забезпеченню можна допомогти автоматизувати рутинні та повторювані процеси, що збільшує ефективність та продуктивність роботи, а також, якщо програма розроблена якісно та відповідає потребам користувача, то зможе без проблем вирішити потребу користувача без необхідності глибокого вивчення тематики та алгоритмів, за якими працює програма.

Опис програми

Програма розроблена на мові програмування Python. Оскільки Python не встановлюється автоматично при встановленні операційної системи, тому необхідно додатково його завантажити з офіційного сайту (https://www.python.org/), останню версію та встановити, крім цього, потрібно ще встановити відповідні бібліотеки. Є інший варіант використовувати веб-версію, таку як Google Colab, в цьому середовищі встановлені всі відповідні бібліотеки. Також є варіант використовувати Jupiter Notebook, також встановлено всі необхідні бібліотеки.

Програмний код

```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import cv2 as cv
import imageio
import os

def imageShow(image):
    plt.figure(figsize=(10,20))
    plt.imshow(image)
    plt.show()
#Показ файлу

def kernel(n):
    kernel=np.ones((n),np.uint8)
    return kernel

def get_positive_integer(prompt):
    while True:
        value = input(prompt)
        if not value.isdigit():
            print("Введення таких символів неможливо, дозволено введення цілими додатніми числами.")
        else:
            return int(value)
```

```
def transformMain(filepath):
    imageOrigin = cv.cvtColor(cv.imread(filepath), cv.COLOR_BGR2RGB)
    flagCoreLoop=True
    while(flagCoreLoop):
        imageProcessed = np.copy(imageOrigin)
        print("- - - - -")
        while(flagCoreLoop):
  #          contour=int(input("Введіть розмір ядра: "))
            contour=get_positive_integer("Введіть розмір ядра: ")
            contourKernel=kernel(contour)

            quetion = input("Ви впевнені що хочете обрати цей розмір ядра? Так/Ні ")

            if(quetion=='Так'):
                imgStep1=cv.erode(imageProcessed,contourKernel, 1) #Розмиття
                imgStep2=cv.cvtColor(imgStep1, cv.COLOR_BGR2GRAY)
#Монохромне
                break

        while(flagCoreLoop):

            thresh=get_positive_integer("Оберіть поріг для перетворення: ")

            quetion = input("Ви впевнені що хочете обрати цей поріг перетворення?
Так/Ні ")
            if(quetion=='Так'):
                thresh, imgStep3=cv.threshold(imgStep2, thresh, 255,
cv.THRESH_BINARY) #Бінарне
```

```
            imgStep4=cv.erode(cv.dilate(imgStep3,contourKernel, 1),contourKernel, 1)
#Закриття
            imgStep5=cv.subtract(cv.dilate(imgStep4,contourKernel, 1),
cv.erode(imgStep4,contourKernel, 1)) #Градієнт
            imgStep6=cv.dilate(imgStep5,contourKernel, 1) #Розширення
            break

    #Пошук контурів, і макс. контуру
    foundContours,hierarchy=cv.findContours(imgStep5, cv.RETR_TREE,
cv.CHAIN_APPROX_NONE)
    contourLengthIndex=[]
    for i in range(len(foundContours)):
        contourLength=len(foundContours[i])
        contourLengthIndex.append([i,contourLength])
    index=np.where(contourLengthIndex==np.max(contourLengthIndex))[0][0]

    #Пошук вершин(згідно з практичною)
    perimeter = cv.arcLength(foundContours[index],True)
    points = cv.approxPolyDP(foundContours[index],0.02*perimeter,True)

    contourImage=imageOrigin.copy()
    #Нанесення точок
    colorIndexing=0
    for i in points:
cv.circle(contourImage,(int(i[0][0]),int(i[0][1])),8,(0,0,(50+colorIndexing*50)),-1)
        colorIndexing+=1
    imageShow(contourImage)
```

```
        print("Точки пофарбовані відповідно до їх індексу. Введіть індекс кожної точки до відповідного кута.")
        index1=get_positive_integer("Вкажіть індекс верхнього лівого кута: ")
        index2=get_positive_integer("Вкажіть індекс верхнього правого кута: ")
        index3=get_positive_integer("Вкажіть індекс нижнього лівого кута: ")
        index4=get_positive_integer("Вкажіть індекс нижнього правого кута: ")



        y,x,c = imageOrigin.shape
        imageCoords1 = np.float32([points[index1],points[index2],points[index3],points[index4]])
        imageCoords2 = np.float32([[0,0],[960,0],[0,1280],[960,1280]])
        matrix_aff=cv.getPerspectiveTransform(imageCoords1,imageCoords2)
        imgWarp=cv.warpPerspective(contourImage,matrix_aff,(960,1280))
        imageShow(imgWarp)

        flagQuery=input("Введіть будь-який текст, якщо бажаєте змінити розмір ядра, або Enter, щоб припинити роботу програми: ")
        if(flagQuery==""):
            flagCoreLoop=False

## "photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

6. Тестування і налагодження програм за контрольним прикладом

Тестування та налагодження програм за контрольним прикладом є важливим етапом при створенні програми, оскільки воно дозволяє перевірити правильність роботи програми з відомими вхідними даними та очікуваними результатами.

Опис контрольного прикладу

Тестування будемо проводити за прикладом, а саме отримаємо зображення випускного альбому з відповідним зручним для людського ока розміром, кутом і нахилом розташування на екрані див. рис. 6.1. Також важливо правильно вказати шлях зображення, щоб програма могла знайти його. Введені значення користувачем проходять перевірку їх відповідностей (наприклад, значення ядра є цілим додатнім числом).

Рис. 6.1. Фотографія випускного альбому на загальному плані.

Провереденння тестування за контрольним прикладом

Для проведення тестування, запускаємо програму, програма одразу запитує вказати шлях до зображення див. рис. 6.2. Якщо шлях не вірний система дасть збій.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення
photo11.jpg
```

Рис. 6.2. Запит програми відповідного шляху до зображення.

Далі програма запитує вказати значення ядра див. рис. 6.3. значення ядра може бути цілим додатнім числом, в інших випадках буде виведено повідомлення див. 6.4.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення photo11.jpg
- - - - -

Введіть розмір ядра: 10
```

Рис. 6.3. Введення значення ядра.

```
# "photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: ааа
Введення таких символів неможливо, дозволено введення цілими додатніми числами.

Введіть розмір ядра:
```

Рис. 6.4. Введення некоректного значення ядра.

Якщо значення коректне, але не відповідає тому значенню яке хотів ввести користувач, програма виводить повідомлення про підтвердження відповідного значення. В іншому випадку програма перезавантажить запит про введення значення ядра див. рис. 6.5.

```
# "photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10

Ви впевнені що хочете обрати цей розмір ядра? Так/Ні
Так
```

Рис. 6.5. Підтвердження введеного користувачем значення ядра.

Наступним кроком потрібно ввести значення порогу відображення див. рис. 6.6.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10
Ви впевнені що хочете обрати цей розмір ядра? Так/Ні Так

Оберіть поріг для перетворення:
100
```

Рис. 6.6. Введення значення порогу.

Аналогічна перевірка є для введеного значення користувачем значення порогу відображення див. рис. 6.7.

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10
Ви впевнені що хочете обрати цей розмір ядра? Так/Ні Так
Оберіть поріг для перетворення: аааaа
Введення таких символів неможливо, дозволено введення цілими додатніми числами.

Оберіть поріг для перетворення:
```

Рис. 6.7. Введення некоректного значення порогу.

Так само, потрібно підтвердити введене значення порогу перетворення див. рис. 6.8.

```
1  # "photos/photo11.jpg"
2  arc=input("Введіть шлях до зображення ")
3  transformMain(arc)
4
5
```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10
Ви впевнені що хочете обрати цей розмір ядра? Так/Ні Так
Оберіть поріг для перетворення: аааaа
Введення таких символів неможливо, дозволено введення цілими додатніми числами.
Оберіть поріг для перетворення: 100
Ви впевнені що хочете обрати цей поріг перетворення? Так/Ні Ні
Оберіть поріг для перетворення: 100
Ви впевнені що хочете обрати цей поріг перетворення? Так/Ні Так
```

Рис. 6.8. Введення некоректного значення порогу.

Потім програма виводить завантажене зображення зі знайденими контурами і крайніми точками див рис. 6.9.

Рис. 6.9. Зображення зі знайденими контурами і крайніми точками.

Програма виводить запит про введення індексів пофарбованих точок які потрібно ввести аналізуючи рис. 6.9. див. рис. 6.10.

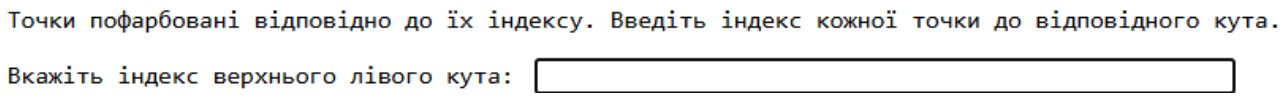

Рис. 6.10. Зображення зі знайденими контурами і крайніми точками.

Значення індексів є цілими додатніми цілими числами, починаються з 0 до 3. Тому, впроваджено перевірку коректного введеного значення індексу користувачем див. рис. 6.11.

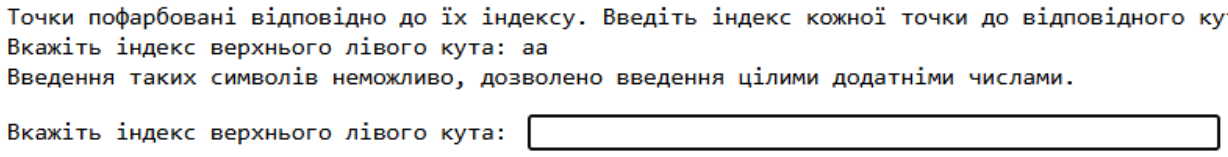

Рис. 6.11. Перевірка введеного користувачем значення індексу.

Так само, перевіряються всі необхідні індекси див. рис. 6.11.

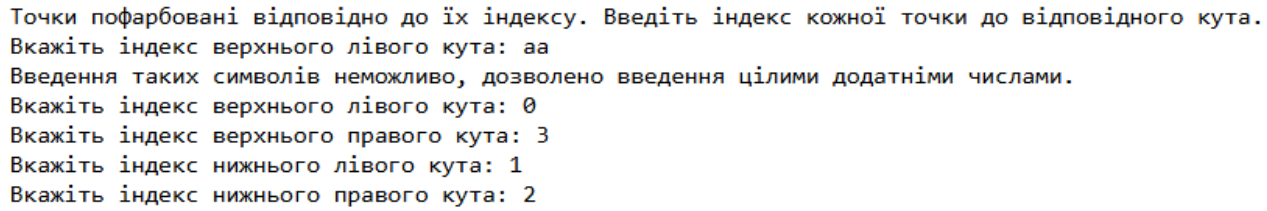

Рис. 6.11. Введення користувачем значення індексів.

Після введення всіх необхідних даних, програми виводить результат роботи див. рис. 6.12. Також виводить запит чи бажає користувач змінити відповідні характеристики або написнути “Enter” щоб припинити роботу програми.

Введіть будь-який текст, якщо бажаєте змінити розмір ядра, або Ent
er, щоб припинити роботу програми:

Рис. 6.12. Результат роботи програми.

7. Опис технічного забезпечення підсистеми

Технічне забезпечення системи автоматизованого проектування включає в себе сукупність всіх технічних (апаратних) засобів, які повинні бути задіяними при функціонуванні автоматизованої системи для ефективного виконання завдання проектування.

Тому, задля забезпечення швидкодії та всіх вимог для виконання програми, встановлено мінімальні вимоги до технічного забезпечення, а саме:

• процесор: Intel® Core™ i5 processor 4300M на частоті 2.60 ГГц (2 ядра, 2 потоки);

• простір на диску: 64 Гб;

• оперативна пам’ять: 4 Гб;

• графіка: вбудована в процесор графічна система або дискретна відеокарта з 1 Гб відеопам’яті;

• операційна система: Windows 10, Linux та macOS;

• Python останньої доступної версії;

• монітор: мінімум 800 х 600 пікселів; 59

• клавіатура;

• миша.

8. Опис лінгвістичного забезпечення

Лінгвістичне забезпечення підсистеми, яку було створено, складається з мови програмування.

Мова програмування, яку було використано для написання даної підсистеми, є мовою «Python». Ця мова програмування відома своєю простотою та легкістю вивчення, тобто навіть початківець з мінімальними знаннями про мову зможе модифікувати програму за потреби, що зробило її популярною серед розробників програмного забезпечення. Python був обраний для реалізації підсистеми через його широкі можливості та високу продуктивність.

Інтерфейс програми повністю реалізований українською мовою.

9. Розробка методичного забезпечення підсистеми

Методичне забезпечення має на меті стандартизацію процесу розробки програмного забезпечення, розробку методик та алгоритмів для вирішення конкретних задач, а також розробку інструкцій з експлуатації системи.

Інструкція з експлуатації

Запускаємо програмний код, див. рис. 9.1.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

Рис. 9.1. Запуск головної функції.

Програма запрошує вказати шлях з фотографією, тому заздалегіть збережіть в доступному середовищі файл див. рис. 9.2.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)
```

```
Введіть шлях до зображення
photo11.jpg
```

Рис. 9.2. Перший запит програми.

Далі вводимо значення ядра див. рис. 9.3.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)


```

```
Введіть шлях до зображення photo11.jpg
- - - - -

Введіть розмір ядра: 10
```

Рис. 9.3. Введення значення ядра.

Потім програма запитує підтвердження обраного значення ядра див. рис. 9.4.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)


```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10

Ви впевнені що хочете обрати цей розмір ядра? Так/Ні
Так
```

Рис. 9.4. Підтвердження введеного значення ядра.

Наступним кроком буде введення значення порогу перетворення див. рис. 9.5.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)


```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10
Ви впевнені що хочете обрати цей розмір ядра? Так/Ні Так

Оберіть поріг для перетворення:
100
```

Рис. 9.5. Введення порогу перетворення.

Програма запитує підтвердження обраного значення порогу див. рис. 9.6.

```
"photos/photo11.jpg"
arc=input("Введіть шлях до зображення ")
transformMain(arc)


```

```
Введіть шлях до зображення photo11.jpg
- - - - -
Введіть розмір ядра: 10
Ви впевнені що хочете обрати цей розмір ядра? Так/Ні Так
Оберіть поріг для перетворення: 100

Ви впевнені що хочете обрати цей поріг перетворення? Так/Ні
Так
```

Рис. 9.6. Підтвердження введеного значення порогу перетворення.

Виведення знадених контурів і крайніх точок зображення див. рис. 9.7.

Рис. 9.7. Виведення зображення зі знайденими контурами і крайними точками.

Вводимо значення послідовно знайдених крайніх точок (темніший віддіток є 0 значенням, а найбільш світліший віддтінок відповідає значенню 3) див. рис. 9.8.

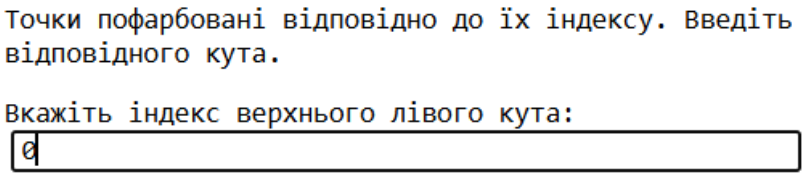

Рис. 9.8. Введення значення вернього лівого кута.

Такі самі маніпуляції проводимо з іншими відповідними крайніми точками див. Рис. 9. 9.

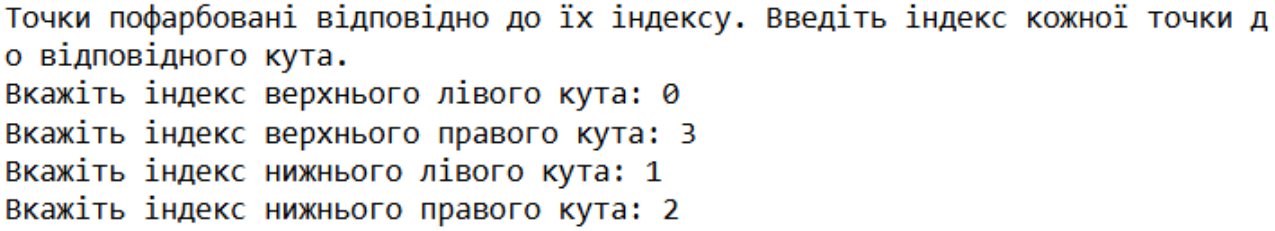

Рис. 9.9. Введення значень відповідних крайніх точок.

Результат роботи програми див. рис. 9.10.

Рис. 9.10. Результат роботи програми.

Також перевірка на коректність роботи програми див. 9.11.

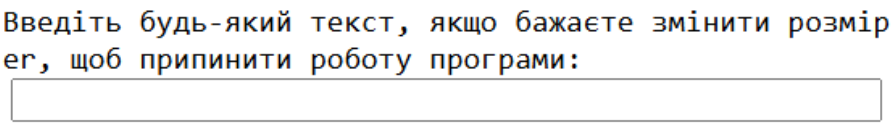

Рис. 9. 11. Підтвердження завершення роботи або редагування характеристик.

Висновок

У результаті виконання розрахункової-графічної роботи було розроблено програмне забезпечення для автоматичного вирівнювання зображень за розміром, поворотом або нахилом. Також, аналізовано літературні джерела, обрано метод вирішення задачі, описано математичні методи і моделі, спроектовано алгоритм програми у вигляді блок-схем, розроблено програмний код мовою програмування Python, проведено тестування та налагодження програми за контрольним прикладом, вказано технічне забезпечення підсистеми, описано лінгвістичне забезпечення підсистеми, а також розроблено методичне забезпечення підсистеми (інструкція з експлуатації). Програма виконує автоматичне вирівнювання зображення за розміром, поворотом і нахилом.

Список використаної літератури

1. Офіційний сайт бібліотеки OpenCV: – Режим доступу: https://opencv.org/

2. Image processing learning resources. – Режим доступу: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm