НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

«На правах рукопису»
УДК (004.932)

«До захисту допущено»
В.о. завідувача кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2023 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою
«Математичні методи криптографічного захисту інформації»

зі спеціальності: 113 Прикладна математика
на тему: **«Відновлення форми і розмірів об'єктів, зображених на розлінованих аркушах»**

Виконав:
студент II курсу, групи ФІ-12МН
Кузнєцов Павло Сергійович _____

Керівник:
в.о. зав. каф. ММЗІ НН ФТІ, доц. каф. ММЗІ НН ФТІ, к.т.н
Яковлєв Сергій Володимирович _____

Консультант: м.н.с Кригін Валерій Михайлович
_____

Рецензент: к.т.н. Кийко Володимир Михайлович
_____

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

Студент _____

Київ — 2023

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
Навчально-науковий фізико-технічний інститут
Кафедра математичних методів захисту інформації

Рівень вищої освіти — другий (магістерський)
Спеціальність — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Сергій ЯКОВЛЄВ
«__» _____ 2023 р.

ЗАВДАННЯ
на магістерську дисертацію

Студент: Кузнєцов Павло Сергійович

1. Тема роботи: «*Відновлення форми і розмірів об'єктів, зображених на розлінованих аркушах*», науковий керівник дисертації: в.о. зав. каф. ММЗІ НН ФТІ, доц. каф. ММЗІ НН ФТІ, к.т.н Яковлєв Сергій Володимирович,

затверджені наказом по університету №__ від «__» _____ 2023 р.

2. Термін подання студентом роботи: «__» _____ 2023 р.

3. Об'єкт дослідження: зображення розлінованих аркушів.

4. Предмет дослідження: модель відновлення форми і розмірів об'єктів, зображених на розлінованих аркушах.

5. Перелік завдань:

1) дослідити існуючу літературу за тематикою обробки зображень документів;

2) дослідити необхідні для цієї роботи застосування проективної геометрії та лінійної алгебри;

3) побудувати алгоритм ректифікації для аркушів з розміткою в клітинку, косу та лінію;

4) побудувати алгоритм пошуку всіх ліній розмітки на ректифікованому зображенні, видалити розмітку та визначити формат;

5) провести експерименти, порівняти результати з результатами комерційних застосунків;

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: Презентація доповіді.

7. Орієнтовний перелік публікацій: доповідь на Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених. 11 травня 2023 р.

8. Дата видачі завдання: 10 вересня 2020р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Узгодження теми роботи із науковим керівником.	01-15 вересня 2020р.	Виконано
2	Дослідження існуючої літератури за тематикою обробки зображень документів.	вересень-жовтень 2020р.	Виконано
3	Дослідження необхідного для цієї роботи застосування проективної геометрії та лінійної алгебри.	жовтень - листопад 2020	Виконано
4	Розробка алгоритму ректифікації зображень аркушів з розміткою в клітинку та косу.	листопад-січень 2020р.	Виконано
5	Розробка алгоритму ректифікації зображень аркушів з розміткою в лінію.	січень - березень 2022р.	Виконано
6	Розробка алгоритму пошуку всіх ліній розмітки на ректифікованому зображенні, видалення розмітки, визначення формату аркуша.	березень-квітень 2023р.	Виконано
7	Проведення експериментів та порівняння результатів з результатами комерційних застосунків.	квітень-травень 2023р.	Виконано

Студент _____ Павло КУЗНЄЦОВ

Керівник _____ Сергій ЯКОВЛЄВ

РЕФЕРАТ

Кваліфікаційна робота містить: 90 сторінок, 39 рисунків, 25 джерел.

Метою роботи є побудова алгоритмів, що ректифікують зображення аркушів з розмітками у клітинку, косу та лінію. А також алгоритмів, що знаходять всі лінії розмітки на ректифікованому аркушу, видаляють розмітку та визначають формат аркушу.

Об'єктом дослідження є зображення розлінованих аркушів.

Предметом дослідження є модель відновлення форми і розмірів об'єктів, зображених на розлінованих аркушах.

В ході дослідження було застосовано RANSAC для пошуку зникомих точок, побудовано ректифікацію зображень аркушів з розміткою за двома та однією зникомою точкою, розроблено алгоритм пошуку всіх ліній розмітки на ректифікованому зображенні та видалено розмітку аркушу. Розроблено алгоритм визначення формату аркушу.

КОМП'ЮТЕРНИЙ ЗІР, ВИРІВНЮВАННЯ ЗОБРАЖЕНЬ ДОКУМЕНТІВ, СКАНУВАННЯ ДОКУМЕНТІВ, ПРОЕКТИВНА ГЕОМЕТРІЯ, КОМП'ЮТЕРНА ГРАФІКА, ОБРОБКА ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ ФОРМИ

ABSTRACT

The qualification work contains: 90 pages, 39 drawings, 25 sources.

The purpose of the work is the construction of rectifying algorithms of squared, oblique and lined notebooks with markup property. As well as algorithms that find all lines of markup of rectified image, remove markup and determine the sheet format.f

The object of the research are images of sheets with markup.

The subject of the study is a model for restoring the shape and size of objects depicted on sheets with markup.

In the course of the study, a modification of RANSAC was created for the search of vanishing points, a rectification based on two and one vanishing points was built, an algorithm for searching for all lines of markup on the rectified image was developed, and the sheet markup was removed. An algorithm for determining the sheet format has been developed.

COMPUTER VISION, DOCUMENT IMAGE ALIGNMENT, DOCUMENT SCANNING, PROJECTIVE GEOMETRY, COMPUTER GRAPHICS, IMAGE PROCESSING, SHAPE RECOGNITION

ЗМІСТ

ВСТУП

Актуальність дослідження. Актуальність дослідження полягає в необхідності зменшення витрат часу на обробку зображень робіт студентів.

Через перехід значної частини освітнього процесу на дистанційне навчання та незалежну від цього необхідність швидко оброблювати велику кількість зображень актуальним є створення програмного забезпечення для викладачів та студентів, що зменшує часові витрати на обробку зображень.

Метою дослідження є побудова алгоритмів, що вирівнюють зображення аркушів у клітинку, косу та лінію так, наче вони відскановані, а також алгоритмів, що знаходять відсутні лінії, видаляють розмітку та визначають формат арушу. Для досягнення мети необхідно розв'язати задачу, котра полягає у побудові перетворень зображень для відповідних розміток та застосуванні методів комп'ютерного зору для пошуку відсутніх ліній, визначення формату і видалення розмітки. Для розв'язання задачі необхідно вирішити такі завдання:

1) дослідити існуючу літературу за тематикою обробки зображень документів;

2) дослідити необхідні для цієї роботи застосування проективної геометрії та лінійної алгебри;

3) побудувати алгоритм ректифікації для аркушів з розміткою в клітинку, косу та лінію;

4) побудувати алгоритм пошуку всіх ліній розмітки на ректифікованому зображенні, видалити розмітку та визначити формат;

5) провести експерименти, порівняти результати з результатами комерційних застосунків;

Об'єктом дослідження є зображення розлінованих аркушів.

Предметом дослідження є модель відновлення форми і розмірів об'єктів, зображених на розлінованих аркушах.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи лінійної алгебри, проективної геометрії, методи комп'ютерного зору.

Наукова новизна отриманих результатів полягає в створенні алгоритмів, котрі відновлюють форму і розмір об'єктів, зображених на розлінованих аркушах. Було застосовано методи комп'ютерного зору для побудови алгоритмів пошуку всіх ліній розмітки ректифікованого аркушу, визначення формату та видалення розмітки. **Практичне значення** результатів полягає у тому, що викладачі та студенти зможуть використовувати запропоновані в цій роботі рішення для прискорення обробки зображень робіт.

1 ДОСЛІДЖЕННЯ ІСНУЮЧОЇ ЛІТЕРАТУРИ ЗА ТЕМАТИКОЮ ОБРОБКИ ЗОБРАЖЕНЬ ДОКУМЕНТІВ

В цьому розділі проведено огляд робіт, що описують алгоритми вирівнювання зображень документів. Окрім цього розглянуто роботу з вилучення заповнених даних із кольорових форм, ідеї з котрої необхідні для побудови алгоритму видалення розмітки аркушу.

В нашій роботі нас цікавить ректифікація перспективними перетвореннями, тому розглянемо декілька робіт стосовно цієї теми.

1.1 Оцінювання положення зникомих точок на зображеннях міських кварталів

В роботі [1] В.Ю. Сдобніков та Б.Д. Савчинський розглянули методи знаходження зникомих точок та запропонували метод, що використовує EM алгоритм. Сутність методу полягає в тому, що пошук зникомих точок та класифікація прямих відбувається одночасно і кількість помилок мінімізується. Алгоритми, розглянуті в цій роботі, є прикладами застосування RANSAC та EM алгоритмів.

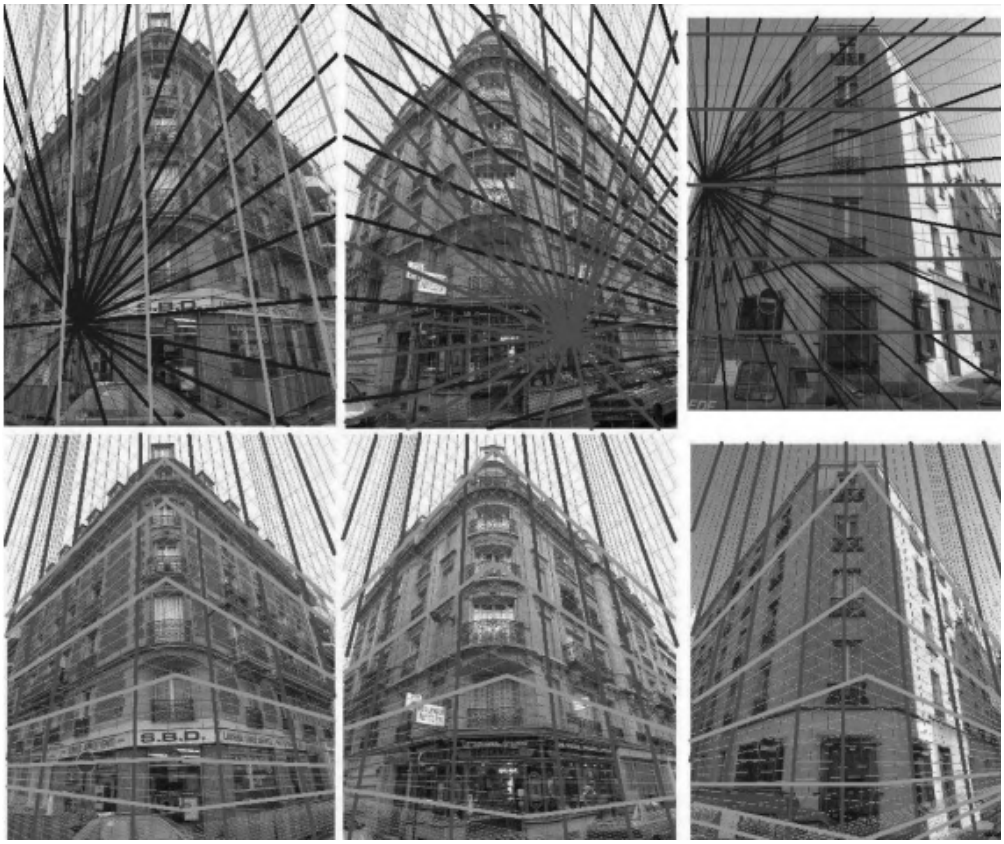

Рисунок 1.1 – Верхній ряд — результати алгоритму [2]; нижній ряд — результати роботи алгоритму, описаного у статті [1].

1.2 Автоматизована ректифікація фотографій користувачів Google

В роботі [3] автори пропонують RANSAC алгоритм для ректифікації зображень користувачів, спотворення котрих є незначними. Замість ліній вони використовують інформацію про знайдені детектором Гарріса [4] краї і зберігають її в множині E, котру називають еджелетом. $E = \{\vec{x}, \vec{d}, s\}$, де $\vec{x}$ — координата краю в однорідних координатах, $\vec{d}$ — напрям краю в однорідних координатах, s — інтенсивність кольору пікселя. Кожному еджелету відповідає лінія $\vec{l_E}$, що проходить через $\vec{x}$ та є паралельною до $\vec{d}$.

Автори використовують RANSAC алгоритм, котрий використовує

еджелети та відповідні їм лінії для голосування та вибору кращої моделі. Обирається зникома точка. Відокремивши не-викиди з множини еджелетів, процедура застосовується для пошуку іншої зникомої точки.

Маючи дві зникомі точки $\vec{v_1}$, $\vec{v_2}$ та відповідну лінію, що їх перетинає $\vec{l_{12}} = \vec{v_1} \times \vec{v_2} = (l_a,l_b,l_c)^T$, будується матриця H, котра перенесе зникомі точки на нескінченність та відновить паралельність ліній на зображенні.

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_a & l_b & l_c \end{bmatrix}$$

$$H\vec{v_1} = \begin{bmatrix} \\ ... \\ \\ ... \\ \\ (v_1 \times v_2)v_1 \end{bmatrix} = \begin{bmatrix} ... \\ \\ ... \\ \\ 0 \end{bmatrix}, H\vec{v_2} = \begin{bmatrix} \\ ... \\ \\ ... \\ \\ (v_1 \times v_2)v_2 \end{bmatrix} = \begin{bmatrix} ... \\ \\ ... \\ \\ 0 \end{bmatrix}$$

Будується матриця повороту R, для котрої необхідно знайти кут між віднесеною на нескінченність зникомою точкою $\vec{v} = (v_x,v_y,0)^T$ та віссю $\vec{Y} = (0,1,0)^T$.

$$\theta = cos^{-1}\left(\frac{\vec{v} \cdot \vec{Y}}{|\vec{v}||\vec{Y}|}\right) = cos^{--1}\left(\frac{v_y}{\sqrt{v_x^2 + v_y^2}}\right)$$

$$R = \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

До початкового зображення застосовується $T = RH$. Алгоритм ректифікував 15.7 % з 2199 зображень користувачів. Недоліком цього

методу є потреба в двох зникомих точках, відповідні лінії котрих є ортогональними.

Рисунок 1.2 – Застосування алгоритму [3].

1.3 Застосування CNN до задачі ректифікації документів

В роботі [5] Хуррам Джавед та Фейсал Шафаїт використовують згорткову нейронну мережу для пошуку положення аркушу на зображенні. Цей метод має високу точність — 94% на датасеті [6].

Згорткова нейронна мережа з подібною до AlexNet архітектурою [7] передбачає чотири кути аркушу. Рекурсивне застосування неглибокої нейронної мережі покращує кожне передбачення.

Для пошуку країв аркушу застосовується 8-шарова, глибока, згорткова, нейронна мережа з двома прихованими, повнозв'язковими шарами та вісьмома регресійними нодами. Перед поданням до нейронної мережі зображення було зменшено до 32×32.

Зображення поділяється на чотири зображення розміром 32×32 та подається у другу згорткову нейронну мережу з двома регресійними нодами.

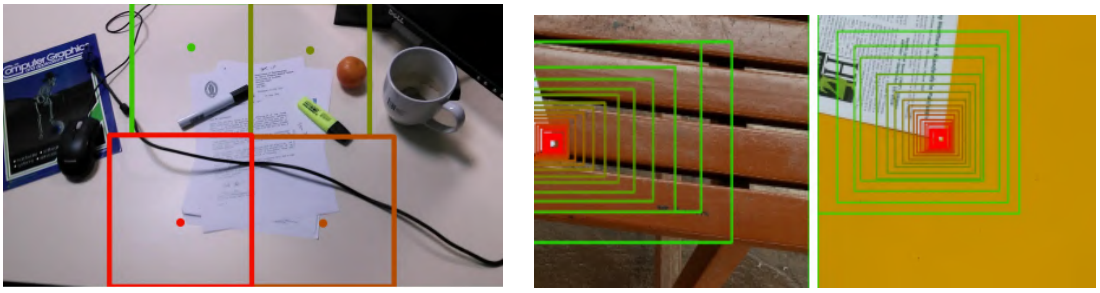

Рисунок 1.3 – Знаходження кутів аркушу. Рекурсивне уточнення на правому.

Якісного результату можна досягнути лише за наявності повного аркушу на зображенні, а тренування нейрнної мережі потребує часу.

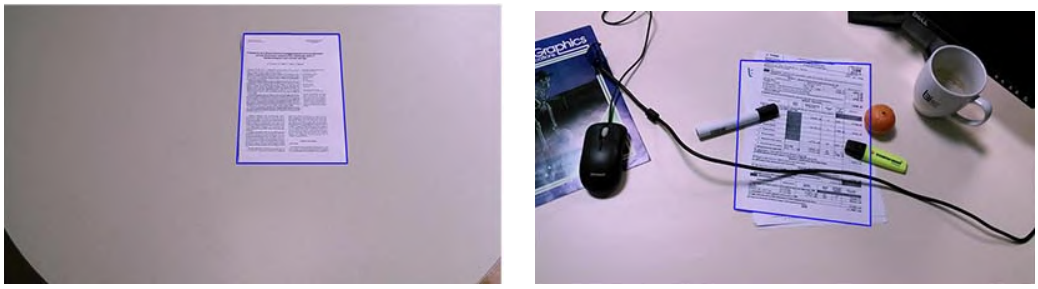

Рисунок 1.4 – Результат роботи нейронної мережі [5].

1.4 Ректифікація зображень документів за текстом на ньому

В роботі [8] Пол Кларк та Маджид Мірмехді вирівнюють зображення, використовуючи текст для пошуку зникомих точок. Рядки тексту вони позначають як лінії, та визначивши тип параграфу знаходять зникомі точки. Вертикальна зникома точка шукається через відстань між горизонтальними лініями, котра змінюється через перспективу.

Запропонований авторами спосіб усуває необхідність знання фокусної відстані камери, отже цей алгоритм застосовується до зображень, зроблених камерами з невідомими внутрішніми параметрами.

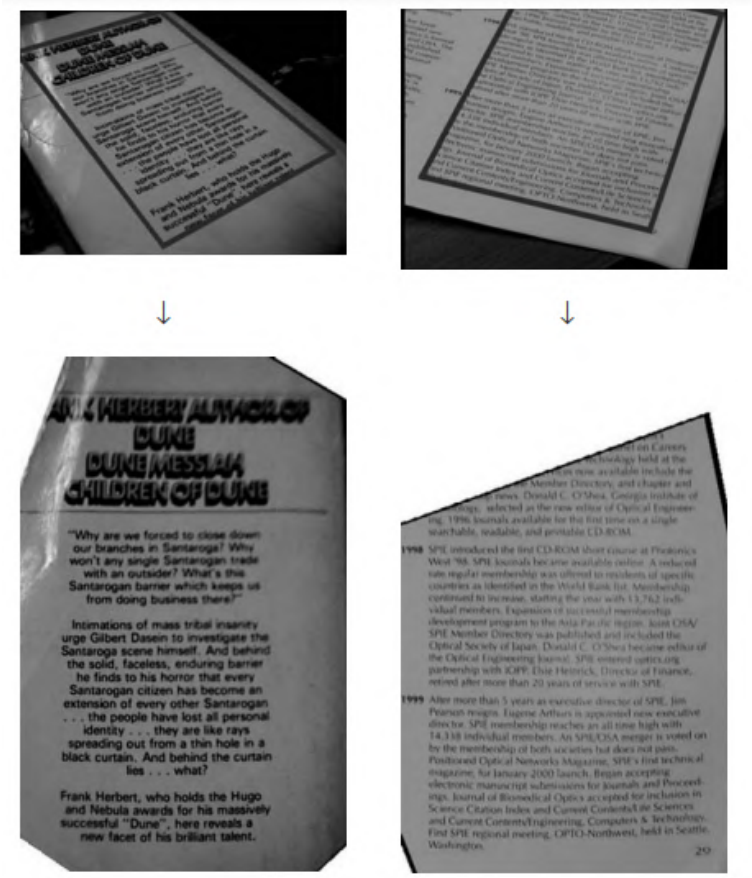

Рисунок 1.5 – Результат роботи методу [8].

В контексті нашої задачі робота має деякі обмеження. Аркуш може не містити достатньо тексту або текст може бути написаний під кутом.

Робота [8] містить ідею щодо пошуку вертикальної зникомої точки через відстані між горизонтальними лініями (рис.1.6). В нашій роботі ми пропонуємо схожий метод, однак не шукаємо вертикальну зникому точку, а користуємось відстанями між лініями для пошуку кута між площинами аркуша та сенсора.

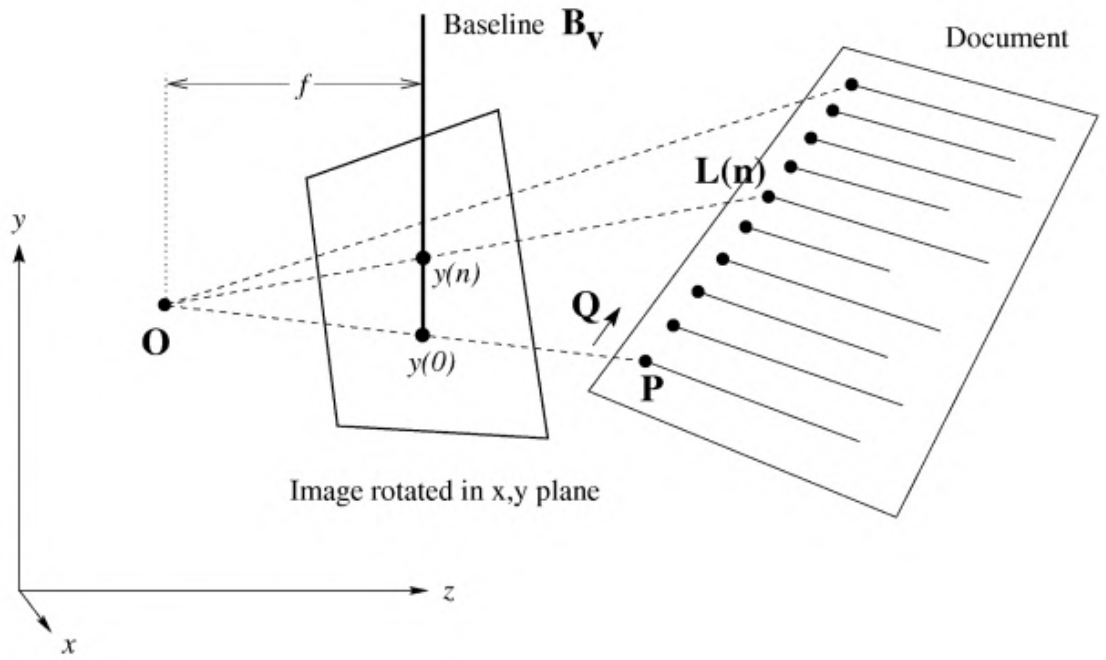

Рисунок 1.6 – Пошук вертикальної зникомої точки [8].

1.5 Алгоритми комерційних застосунків

Алгоритми комерційних застосунків спочатку шукають координати країв аркушу, а потім застосовують перспективне перетворення. Така реалізація присутня у Dropbox, PDF-scanner, Google Drive, Adobe Scan. Недоліки у таких алгоритмів такі самі як і в роботі [5] — необхідність у якісному вхідному зображенні з усіма кутами аркушу, однак на відміну від методу [5] вони не знайдуть аркуш, що лежить над іншим аркушем.

Прикладом такого рішення є застосунок Dropbox [9], в котрому розробники спочатку шукають лінії на бінаризованому зображенні алгоритмом Хафа, після цього визначають кути між попарними перетинами ліній та роблять ймовірнісну оцінку коректності кута.

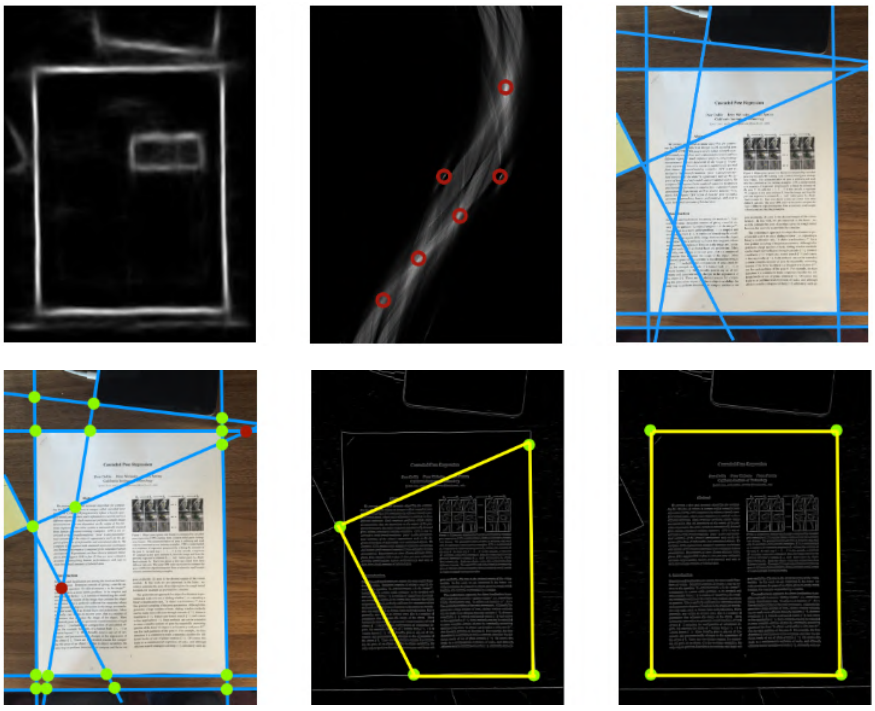

Рисунок 1.7 – Робота застосунку Dropbox [9].

1.6 Пошук зникомих точок за критерієм відстані від фіксованої точки, що перевіряється, до точок що її оточують

В роботі [10] Марія Бондар використовує RANSAC алгоритм для пошуку зникомих точок. В якості оцінки коректності зникомої точки вимірюється відстань від перевіряємої зникомої точки до набору точок що її оточують та обирається в якості найкращої, якщо оточуючих точок найбільша кількість.

В роботі [10] множина вертикальних ліній позначається V. Тоді зникома точка v_1 знаходиться як:

$$v_1 \in \arg \max_{\substack{p=\frac{\ell\times\ell'}{\|\ell\times\ell'\|} \\ \forall \ell,\ell' \in V}} \sum_{\ell^* \in V} \left[\left| \cos^{-1}\left(p \cdot \frac{\ell \times \ell^*}{\|\ell \times \ell^*\|} \right) \right| < \varepsilon \right],$$

та зберігається множина ліній L_1, що до неї збігаються.

$$L_1 = \{l : \left| cos^{-1}\left(v_1^T \frac{l}{||l||}\right)\right| < \epsilon\}$$

Використовуючи отримані значення автор будує перспективне перетворення для ректифікації та обрізає зображення (рис.1.8).

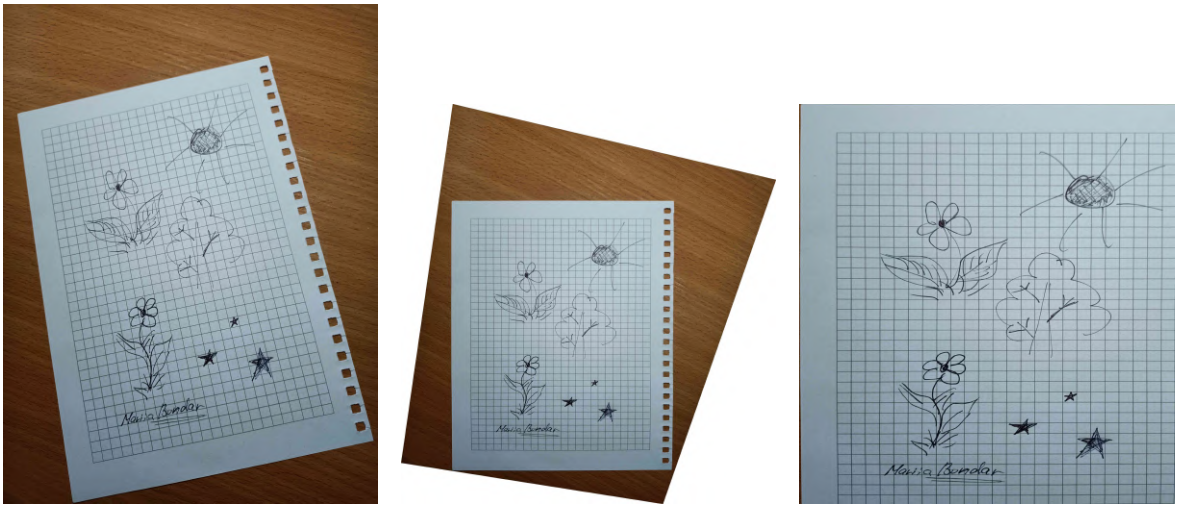

Рисунок 1.8 – Вхідне зображення, вирівнювання, обрізка [10].

В своїй роботі ми також використовуємо RANSAC алгоритм, однак критерієм перевірки відстані є паралельність ліній. Наш метод потребує більше часу, однак є більш надійним, оскільки перевіряє ту властивість, котру ми вимагаємо від результуючого перетворення.

1.7 Вилучення заповнених даних із кольорових форм

В роботі [11] автори побудували систему вилучення заповнених даних із кольорових форм.

На вхід алгоритму подається порожній бланк та заповнений. Задача полягає в відокремленні тексту з заповненого бланку так, щоб текст був чорним на білому фоні.

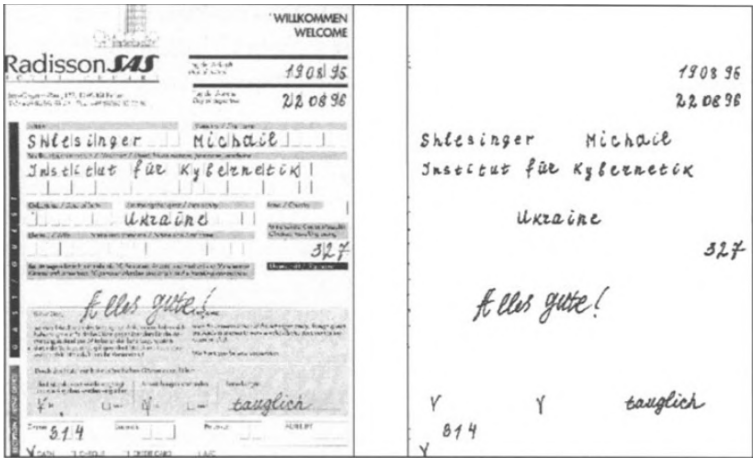

Рисунок 1.9 – Відокремлення тексту з форми [11].

В місцях де рамки форми та написи перетинаються може виникнути пошкодження тексту при їх відокремленні. Це можливо у разі однаковості кольорів рамки та тексту. Спочату видаляється шум на верхній та нижній границі лінії. Після цього визначається товщина лінії та напрями сегментів лінії що дотикаються до верхньої та нижньої меж цієї лінії. В подальшому знайдені параметри використовуються для визначення та використання сусідніх пікселів, з яких формується заповнення проміжку (рис. 1.10).

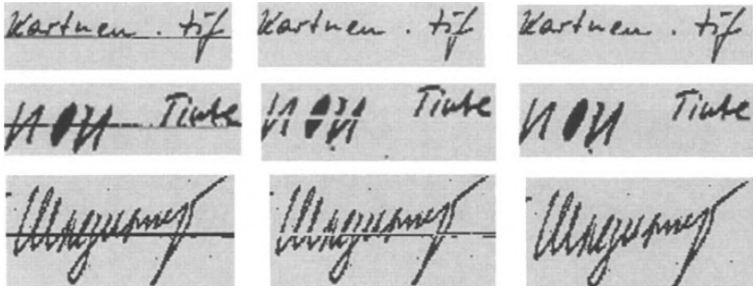

Рисунок 1.10 – Відновлення пошкодженого тексту [11].

Для нашої роботи елементи відновлення тексту є необхідними, оскільки демонструють як потрібно працювати з відновленням кольору лінії.

Висновки до розділу 1.

В розділі розглянуто існуючі методи ректифікації зображень паперових документів, кожен з яких має свої переваги та недоліки. Було розглянуто конкретні обмеження алгоритмів, а саме: необхідність у наявності цілого аркушу на зображенні, якісні вхідні данні, неможливість ректифікації за наявності не ортогональних ліній, що відповідають зникомим точкам. Якщо це відновлення за текстом, то необхідність в достатній кількості написів і т.д

Наша робота орієнтована на позбавлення від зазначених вище обмежень, оскільки ми не маємо потреби у наявності цілого аркушу та використовуємо розмітку для вирівнювання. Також можемо працювати з будь-яким кутом між відповідними наборами ліній.

2 ЕЛЕМЕНТИ ЛІНІЙНОЇ АЛГЕБРИ, ПРОЕКТИВНОЇ ГЕОМЕТРІЇ ТА МЕТОДИ КОМП'ЮТЕРНОГО ЗОРУ

В цьому розділі розглянуто необхідні математичні об'єкти та інструменти, а саме: точки та прямі у проективному просторі, їх властивості, спеціалізації проективного перетворення [19].

Зроблено огляд використаних у роботі алгоритмів комп'ютерного зору: перетворення Хафа [12], детектор Кенні [13], бінаризація Оцу [14], локальна бінаризація [15], морфології розширення та розкриття [16], медіанна фільтрація [17], RANSAC [18].

2.1 Елементи проективної геометрії

В проективній геометрії досліджуються властивості проективних площин та проективного простору, використовуються однорідні координати [20].

Означення 2.1. Точка у двовимірній евклідовій площині позначається як (x,y). Для того щоб представити цю точку у проективній площині необхідно дописати третю координату 1 в кінці $(x,y,1)$. Рівномірне масштабування не має значення, тому точка $(x, y, 1)$ є тою самою точкою що й $(\alpha x,\alpha y,\alpha)$, $\forall\alpha \neq 0$.

Тобто

$$(x,y,w) = (\alpha x, \alpha y, \alpha w), \forall\alpha \neq 0.$$

Через відсутність впливу масштабу координати (x,y,w) називаються однорідними.

Означення 2.2. В евклідовому просторі рівняння прямої на площині визначається як $ax + by + c = 0$. Користуючись фактом про те, що масштабування не впливає на рівняння лінії, запишемо рівняння лінії

у проективному просторі $aX + bY + cW = 0$. У формі скалярного добутку це виглядатиме як $u^T p = p^T u = 0$, де $u = [a,b,c]^T$ — коефіцієнти лінії, а $p = [X,Y,W]^T$ — точка на цій лінії. Звідси можна бачити, що точка та пряма представляються однаково. Коефіцієнти лінії можна інтерпретувати наступним чином: $-a/b$ — кут нахилу, $-c/a$ — точка перетину з віссю x, $-c/b$ — точка перетину з віссю y.

Означення 2.3. Точки, що мають третю нульову координату називаються точкою на нескінченності, або ж ідеальною точкою. Кожному напряму на площині відповідає власна ідеальна точка. Наприклад горизонтальному напряму відповідає (1,0,0), а вертикальному (0,1,0). Ідеальні точки мають такі самі властивості як і решта точок в P^2. Всі ідеальні точки належать ідеальній лінії, що має координати (0,0,1)

Для того щоб зробити перехід від проективної до декартової системи координат необхідно поділити перші дві координати на третю $(x,y) = (x/w, y/w), w \neq 0$. За необхідності замість 0 можна використовувати значення близькі до 0.

Точку перетину двох ліній в проективному просторі можна записати як $p = u_1 \times u_2 = (b_1c_2 - b_2c_1, a_2c_1 - a_1c_2, a_1b_2 - a_2b_1)$, де $u_1 = (a_1,b_1,c_1)$, $u_2 = (a_2,b_2,c_2)$. Якщо дві лінії паралельні, то $-a_1/b_1 = -a_2/b_2$ і точкою перетину є $(b_1c_2 - b_2c_1, a_2c_1 - a_1c_2, 0)$, що відповідає напряму з кутом нахилу $-a1/b1$.

Аналогічно визначається лінія, що проходить через дві точки $u = p_1 \times p_2$.

За необідності визначити чи належать точки p_1,p_2,p_3 одній лінії потрібно порахувати визначник $\det[p_1,p_2,p_3] = 0$ або $p_3^T(p_1 \times p_2) = 0$ — перевірка належності точки до побудованої прямої. Аналогічно перевіряємо чи перетинаються лінії в просторі $\det[u_1,u_2,u_3] = 0$.

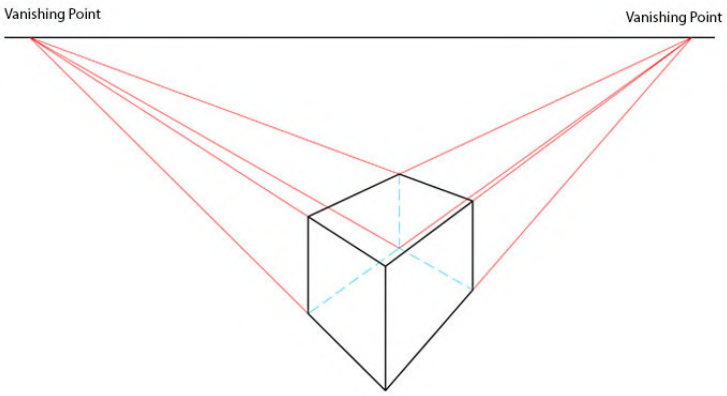

Рисунок 2.1 – Дві ідеальні точки на ідеальній прямій [21].

2.2 Спеціалізації проективного перетворення

Розглянемо в цьому розділі спеціалізації проективного перетворення. Проективні перетворення формують проективну лінійну групу. Спеціалізації проективного перетворення формують її підгрупи.

Означення 2.4. $GL(n)$ — група оборотних матриць $n \times n$ з дійсними елементами. Є загальною лінійною групою розмірності n.

Означення 2.5. $PL(n)$ — проективна лінійна група, матриці котрої пов'язані скалярним множником. $PL(n)$ є фактор групою $GL(n)$. У разі проективних перетворень площини $n = 3$.

Важливими підгрупами $PL(3)$ є афінна група, матриці котрої містять останній рядок (0,0,1). Евклідова група, що є підгрупою афінної групи, котра містить ортогональну 2×2 матрицю у лівому верхньому куту. Або орієнтована Евклідова група, визначник верхньої лівої матриці 2×2 котрої дорівнює 1.

2.2.1 Ізометрії

Ізометрії є перетвореннями площини $\mathbb{R}^2$, що зберігають Евклідову відстань. Представимо це перетворення.

$$\begin{pmatrix} x^{'} \\ y^{'} \\ 1 \end{pmatrix} = \begin{pmatrix} \psi cos(\theta) & -sin(\theta) & t_x \\ \psi sin(\theta) & cos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$\psi = \pm 1$. Якщо $\psi = 1$, то ізометрія зберігає орієнтацію і є Евклідовим перетворенням. Якщо $\psi = -1$, то ізометрія змінює орієнтацію на протилежну. Евклідові перетворення моделюють рух твердого тіла.

Для зручності запишемо перетворення Евкліда в такій формі.

$$x^{'} = H_E x = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} x$$

R — матриця повороту 2×2 (ортогональна матриця, така що $R^T R = R R^T = I$), t — вектор переносу, 0^T — вектор з двох нулів. Якщо потрібно зробити тільки поворот, то t — нульовий, якщо потрібно зробити тільки переніс, то $R = I$.

Планарне евклідове перетворення має три ступені свободи, один для обертання і два для переносу. Інваріантами є довжина, кут та площа.

Ізометрія зберігає орієнтацію, якщо верхній лівий кут — матриця 2×2 має визначник 1. Ізометрії, що зберігають орієнтацію утворюють групу, зі зміною орієнтації — ні.

2.2.2 Перетворення подібності

Перетворення подібності є ізометрією, що поєднана з ізотропічним масштабуванням. У випадку Евклідового перетворення, поєднанного з масштабуванням, перетворення має наступний вигляд.

$$\begin{pmatrix} x^{'} \\ y^{'} \\ 1 \end{pmatrix} = \begin{pmatrix} scos(\theta) & -ssin(\theta) & t_x \\ ssin(\theta) & scos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x^{'} = H_S x = \begin{pmatrix} sR & t \\ 0^T & 1 \end{pmatrix} x$$

s — представляє ізотропічне масштабування. Таке перетворення зберігає форму, та має чотири ступені свободи. Одне для масштабу та три такі ж як і в Евклідового перетворення.

Враховуючи масштабування, інваріанти можна сконструювати з евклідових інваріантів.

2.2.3 Афінне перетворення

Це несингулярне лінійне перетворення з переносом.

$$\begin{pmatrix} x^{'} \\ y^{'} \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$x^{'} = H_A x = \begin{pmatrix} A & t \\ 0^T & 1 \end{pmatrix} x$$

A — несингулярна матриця 2×2. Планарне афінне перетворення має шість ступенів свободи, що відповідають шести елементам матриці.

Матрицю A завжди можна розкласти.

$$A = R(\theta)R(-\phi)DR(\phi), D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

Інваріантами є паралельність ліній, відношення довжин паралельних відрізків, співвідношення площ.

Перетворення зберігає орієнтацію в залежності від $det(A)$. Якщо $det(A) = \lambda_1\lambda_2$, то орієнтація визначатиметься знаком масштабування.

2.2.4 Проективне перетворення

Проективне перетворення — загальне несингулярне лінійне перетвоерння однорідних координат. Це узагальнює афінне перетворення, що є поєднанням загального несингулярного перетворення неоднорідних координат та переносу.

$$x^{'} = H_p x = \begin{pmatrix} A & t \\ v^T & c \end{pmatrix} x$$

$v = (v_1,v_2)^T$. Інваріантом є перехресне співвідношення чотирьох колінеарних точок.

2.2.5 Гомографія

Означення 2.6. Гомографія — це бієктивне відображення $H : P^2 \to P^2$ таке, що три точки x_1, x_2, x_3 лежать на одній прямій тоді і тільки тоді, коли $H(x_1), H(x_2), H(x_3)$ — лежать на одній прямій.

Матриця гомографії декомпозується на перетворення наведені вище та застосовується до зображення.

2.3 RANSAC

RANSAC(RANdom SAmple Consensus) — це ітеративний алгоритм оцінки параметрів моделі, що працює з даними, котрі містять велику кількість викидів [18].

Algorithm 2.1 RANSAC

1: Повторити п. 2—5 N разів.

2: Випадковим чином обирається необхідна кількість точок для визначення параметрів моделі.

3: Обраховуються параметри моделі.

4: Визначається множина не-вибросів, що задовільняє параметрам моделі.

5: Найпотужніша множина не-вибросів зберігається з параметрами моделі.

Кількість ітерацій N обирається таким чином, щоб ймовірність відсутності викидів хоча б у одній випадковій виборці була близькою до 1 ($p = 0.99$). Позначимо u — ймовірність того, що будь-який елемент початкового набору є не-вибросом, $v = 1 - u$ — ймовірність того, що будь-який елемент є вибросом, m — кількість елементів у виборках для визначення моделі.

$$1 - p = (1 - u^m)^N$$

$$N = \frac{log(1-p)}{log(1-(1-v)^m)}$$

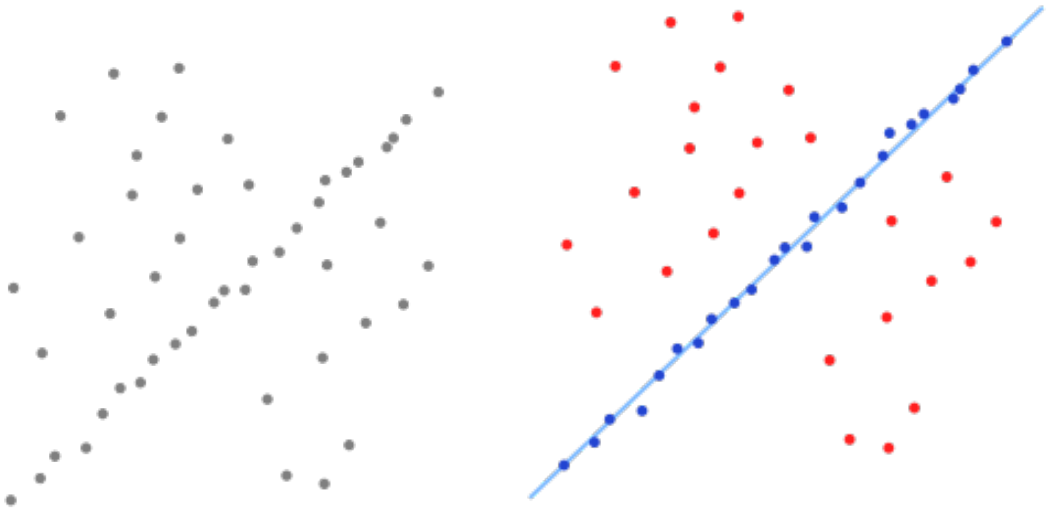

Рисунок 2.2 – Знайдена множина не-викидів [22].

2.4 Перетворення Хафа

Перетворення Хафа [12] це алгоритм пошуку ознак, що може бути параметризований для пошуку різних об'єктів. В цій роботі застосовується для пошуку ліній.

Алгоритм працює наступним чином.

1. Для зручності будемо працювати з лініями записаними в полярних координатах (r,θ).

$$y = \left(-\frac{cos(\theta)}{sin(\theta)} \right) x + \left(\frac{r}{sin(\theta)} \right)$$

$$r = xcos(\theta) + ysin(\theta)$$

2. Для кожної точки (x_0,y_0) визначаємо набір ліній, що проходять крізь неї.

$$r_\theta = x_0 cos(\theta) + y_0 sin(\theta)$$

$$r > 0, 0 < \theta < 2\pi$$

Тобто кожна пара (r_θ, θ) відповідає лінії, що проходить через (x_0,y_0).

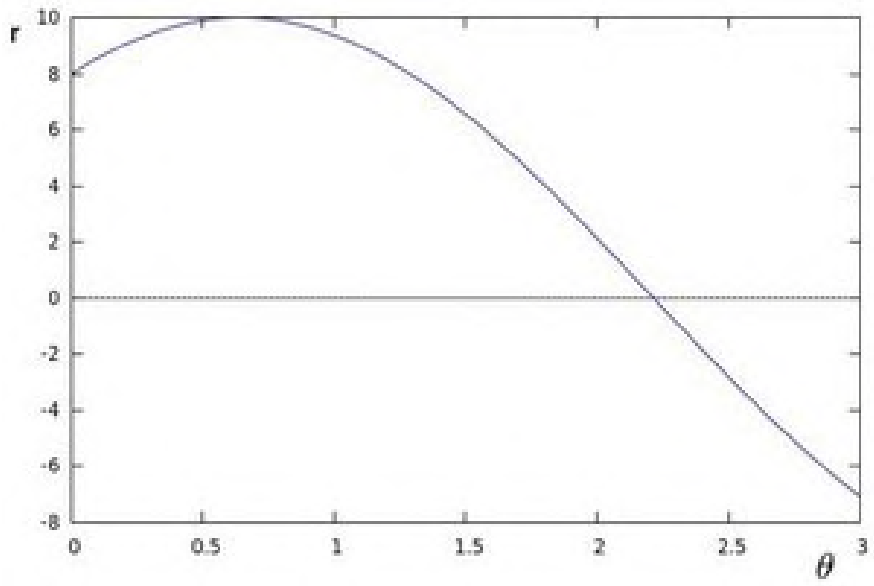

Рисунок 2.3 – Представлення ліній, що проходять через $x_0 = 8, y_0 = 6$ [23].

3. Такі набори параметрів (r,θ) треба побудувати для кожної перевіряємої точки. Точки в котрих ці набори перетинаються знаходиться лінія на зображенні.

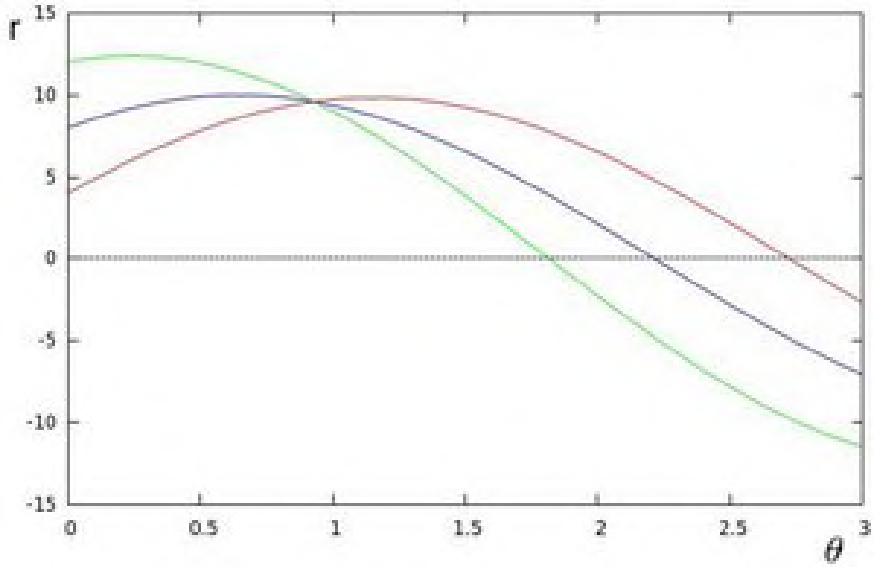

Рисунок 2.4 – Три точки належть одній лінії [23].

Рисунок 2.5 – Перетворення Хафа для пошуку ліній та кіл.

2.5 Детектор Кенні

Детектор країв Кенні [13] це алгоритм, що використовується для пошуку країв на зображенні.

Процес пошуку країв містить 4 етапи.

1) Гаусів фільтр.

2) Пошук градієнтів інтенсивностей пікселів зображення.

3) Знаходження пікселів з максимальним значенням інтенсивності за напрямом країв.

4) Подвійне порогування та перетворення слабких пікселів на сильні.

2.5.1 Гаусів фільтр

На цьому етапі застосовується Гаусів фільтр для позбавлення від шумів. Ефект згладжування залежить від розміру ядра та σ. Чим менше ядро тим менший ефект.

$$H_{ij} = \frac{1}{2\pi\sigma^2} \exp\left(- \frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2} \right)$$
$$1 \leqslant i, j \leqslant 2k+1$$

2.5.2 Пошук градієнтів інтенсивностей пікселів зображення

Інтесивність та напрямок країв можна знайти розрахувавши градієнт зображення за допомогою операторів детекції країв.

На згладженому зображені шукаємо похідні I_x,I_y використовуючи оператори Собеля K_x, K_y відповідно горизонтальному та вертикальному напрямам.

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Після цього вираховуємо величину G та кут θ.

$$|G| = \sqrt{I_x^2 + I_y^2}$$

$$\theta(x,y) = \arctan(\frac{I_y}{I_x})$$

Після цього етапу на зображенні можуть залишитись товсті краї.

2.5.3 Знаходження пікселів з максимальним значенням інтенсивності за напрямом країв

На зображенні отриманому на попередньому кроці присутні товсті краї. Якщо залишити лише максимальні значення інтенсивності за напрямами країв, то ми залишимо лише тонкі краї.

1) Для кожного пікселю матриці інтенсивності визначаємо нарпям та інтенсивність.

2) Оброблюємо кожен її піксель.

3) Якщо на напрямі обраного пікселю є пікслель з більшою інтенсивністю, то обраний піксель приймає значення 0. Інакше зберігає своє значення.

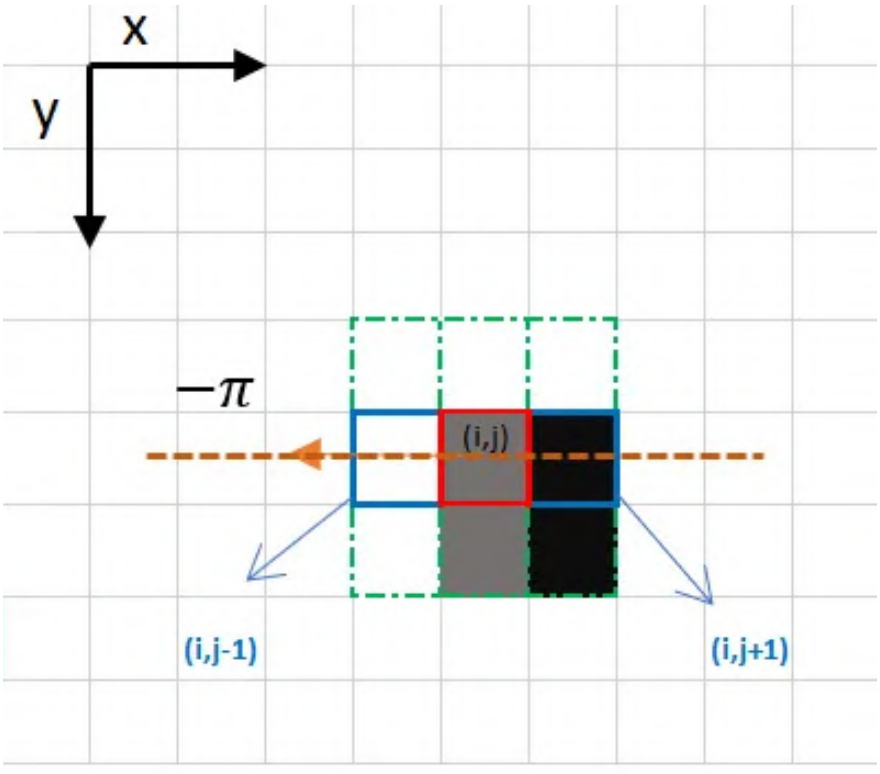

Рисунок 2.6 – Пікслель (i,j) отримає значення 0, оскільки піксель $(i,j-1)$ має більшу інтенсивність на напрямі.

2.5.4 Подвійне порогування та перетворення слабких пікселів на сильні

Подвійне порогування визначає три типи пікселів: сильні, слабкі та нерелевантні. Сильні пікселі мають найбільшу інтенсивність, слабкі пікселі не можуть бути класифіковані як сильні або нерелевнтні, всі інші пікселі є слабкими.

Після подвійного порогування слабкі пікселі перетворюються на сильні шляхом наявності сусідства із сильними.

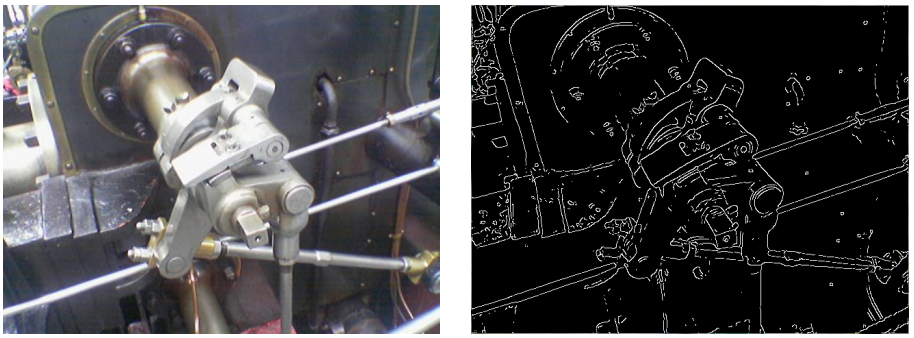

Рисунок 2.7 – Приклад роботи детектора Кенні.

2.6 Бінаризація Оцу

Задача алгоритму Оцу[14] полягає в пошуку такого мінімального порогового значення t, котре мінімізує внутрішньокласову(intra-class variance) дисперсію $\sigma_w^2(t)$.

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

$$q_1(t) = \sum_{i=1}^{t} P(i), \quad q_2(t) = \sum_{i=t+1}^{I} P(i)$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)}, \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P_i}{q_1(t)}, \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P_i}{q_2(t)}$$

q_1,q_2 — ваги або ймовірності класів, $\mu_1(t)$,$\mu_2(t)$ — середнє класів, $\sigma_1^2(t), \sigma_2^2(t)$ — дисперсії класів, I — максимальна інтенсивність пікселю на зображенні або к-ть комірок на гістограмі.

Також замість мінімізації внутрішньокласової дисперсії можна максимізувати міжкласову дисперсію (inter-class variance) $\sigma_b^2 = q_1(t)q_2(t)[\mu_1 - \mu_2]^2$.

2.7 Локальна бінаризація

В нашій роботі потрібно використовувати методи локальної бінаризації [15], оскільки якість освітлення на зображенні не є достатньою. Для вирішення цієї проблеми порогове значення $T(x,y)$ може бути обране як середнє значення вікна сусідніх пікселів з відніманням константи. Або ж як зважена сума вікна сусідніх пікселів (крос-кореляція з вікном Гауса).

2.8 Морфологія

В роботі ми використовуємо декілька морфологій: розширення, відкриття [16]. Для нижчезазначених морфологій потрібно створювати структуруючий елемент.

Функція для розширення(dilation).

$$dst(x,y) = \max_{(x',y'):element(x',y')\neq 0} src(x + x',y + y')$$

Функція для розмиття(erodion).

$$dst(x,y) = \min_{(x',y'):element(x',y')\neq 0} src(x+x',y+y')$$

Функція для відкриття(opening)

$$dst = open(src,element) = dilate(erode(src,element))$$

2.9 Медіанна фільтрація

На етапі бінаризації ми використовуємо медіанний фільтр [17], котрий кожен піксель зображення замінює на медіанне значення з вікна сусідів.

Висновки до розділу 2

В цьому розділі розглянуто всі необхідні математичні означення, перетворення та методи комп'ютерного зору, що будуть застосовуватись у наступних розділах. Наведення детального опису алгоритмів в цьому розділі необхідно для чіткого розуміння того, що саме ми використовуємо, коли користуємось OpenCV [25] та skimage [24].

3 РЕКТИФІКАЦІЯ ЗОБРАЖЕННЯ РОЗЛІНОВАНОГО АРКУШУ ЗА ДВОМА ЗНИКОМИМИ ТОЧКАМИ

В цьому розділі буде запропоновано спосіб ректифікації аркушу, розмітка котрого містить два набори ліній. Це можуть бути аркуші в клітинку, що містять вертикальні та горизонтальні лінії а також аркуші в косу, набір ліній котрих розташований під кутом до горизонтальних.

3.1 Бінаризація зображення

Для якісного пошуку ліній на зображенні необхідно його бінаризувати. Коректна бінаризація дозволить нам шукати точки ліній серед країв і зменшить кількість дублікатів лінії.

В цій роботі до вхідних кольорових зображень можуть застосовуватись різні комбінації порогування, фільтрів, морфологій та детекторів країв. Розглянемо одну з таких комбінацій застосовану до вхідного зображення.

На рис.3.1 до вхідного зображення послідовно зостосовано: переведення у відтінки сірого, локальне порогування з пороговим значенням $T(x,y)$, де T середнє значення від вікна пікселів з відніманням константи, морфологія відкриття з еліптичним структуруючим елементом, детектор країв Кенні та морфологія розширення.

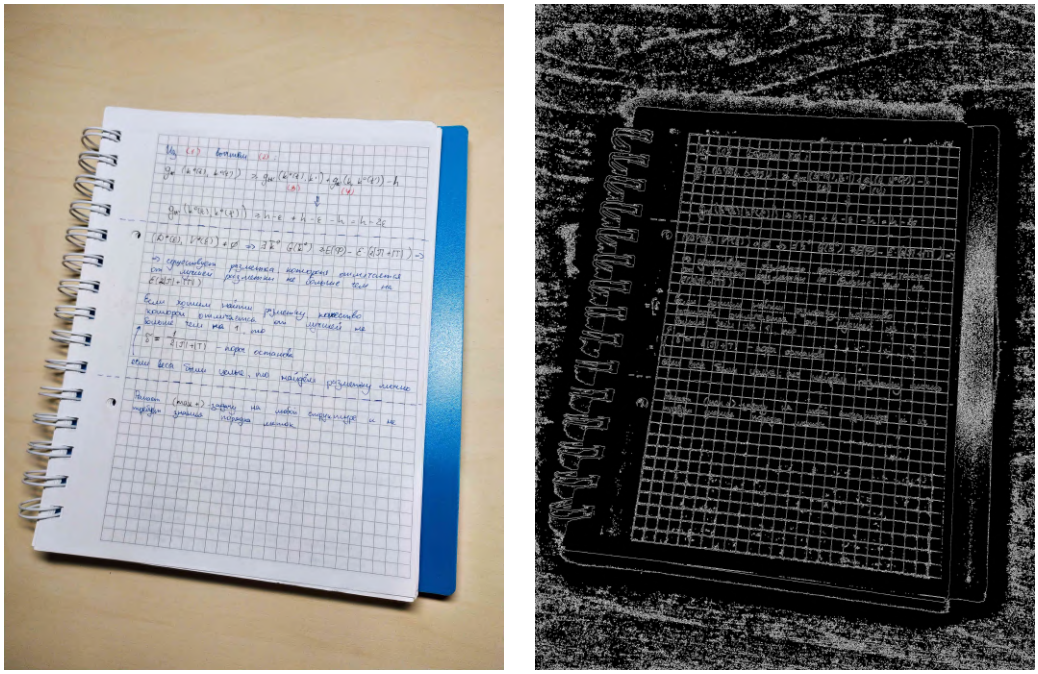

Рисунок 3.1 – Вхідне та бінаризоване зображення.

3.2 Пошук ліній на зображенні

В цій роботі ми використовуємо алгоритм пошуку ліній Хафа. Пошук порогового значення акумулятору для визначення лінії в цій роботі не досліджується. Тому до бінаризованого зображення на рис.3.2 застосуємо $threshold = 1050$.

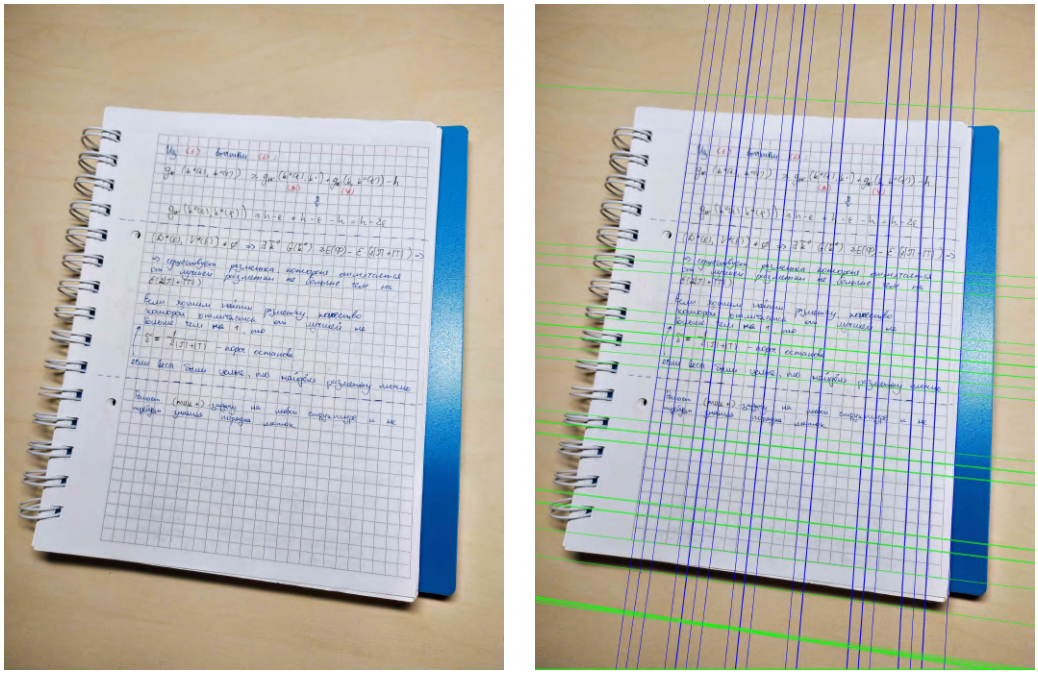

Рисунок 3.2 – Застосування алгоритму Хафа.

3.3 Кластеризація ліній та пошук зникомих точок

Сутність цього підходу полягає в тому, щоб на кожній ітерації RANSAC обиралась пара ліній, із зникомої точки котрої будується матриця, котра перетворює весь набір ліній на паралельні. Паралельність ліній оцінюється і обирається найкраща зникома точка.

Позначимо через l,l' — лінію та відповідну їй паралелізовану. M — точка, що належить лінії l. A — матриця, що переносить зникому точку на нескінченність.

$$Ml = 0 \Leftrightarrow AMl' = 0$$

$$lM = 0 \Leftrightarrow l'AM = 0$$

$$l'A = l \Leftrightarrow l' = lA^{-1}$$

Тепер запишемо RANSAC алгоритм, що шукає найкращу зникому точку p_1. V — множина вертикальних ліній. Через $slope(l) = -l_x/l_y$

запишемо функцію, що вираховує кутовий коефіцієнт лінії у однорідних координатах.

$$p_1 \in \underset{\substack{p=\dfrac{l \times l'}{\|l \times l'\|} \\ \forall l,\ l' \in V}}{\arg\max} \sum_{\forall l^* \in V} \left[\left| slope(l^* A^{-1}) - slope(l A^{-1}) \right| < \epsilon \right]$$

Де A — це матриця, що вираховується через точку перетину p.

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\dfrac{p_z}{p_x} & 0 & 1 \end{pmatrix}$$

Зберігаємо всі лінії, що задовольняють критерію паралельності. Множина ліній, з котрих потім будується обмежувальна рамка.

$$L = \{ l^* : \left| slope(l^* A^{-1}) - slope(l A^{-1}) \right| < \epsilon \}$$

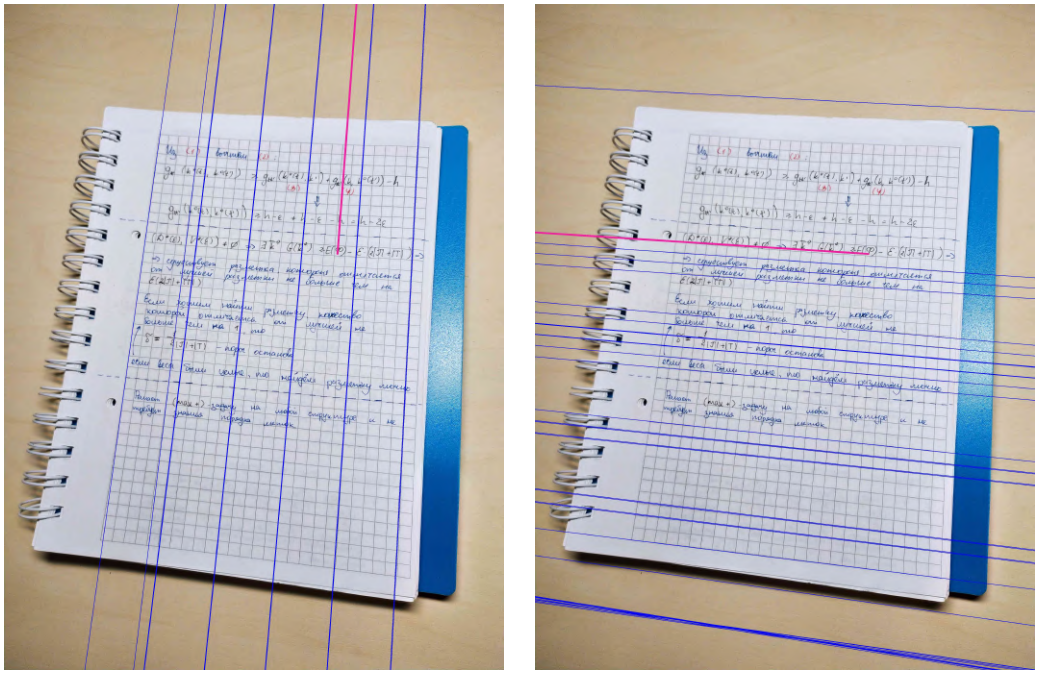

Рисунок 3.3 – Вертикальні та горизонтальні не-викиди та відповідні їм зникомі точки.

Як зображено на рис.3.3 алгоритм застосовується і до набору горизонтальних ліній.

3.4 Перспективне перетворення

З двох зникомих точок ми можемо побудувати перетворення, що переведе зображення у площину з камерою над ним. Маємо зникомі точки v_1,v_2 — у правій системі координат. Також знаємо, що перетворення повинно зробити лінії у наборах пралельними, зникомі точки зробити ідеальними. Тобто вертикальна зникома точка перейде у точку $(x,0,0)$, а горизонтальна у точку $(0,y,0)$. Для цього побудуємо таку матрицю повороту R, що $Rv_1 = (x,0,0)^T$, $Rv_2 = (0,y,0)^T$.

$$R(v_1, v_2) = \begin{pmatrix} v_2 \times (v_1 \times v_2) \\ (v_1 \times v_2) \times v_1 \\ (v_1 \times v_2) \end{pmatrix}$$

точка v_1 перпендикулярна до векторів, що відповідають другому і третьому рядкам матриці R, а v_2 першому і третьому.

Дана матриця має недолік: оскільки описує лінійне перетворення і $rang(R) = 3$, а ядро повнорангового лінійного оператора є множина, що складається з нульового вектора, то точка зображення з координатами $(0, 0)$ після застосування оператора R залишиться без викривлень, а чим далі точки знаходяться від початку координат, тим далі вони будуть знаходитися від своїх початкових положень після перетворення R. Аби зменшити нерівномірність викривлення зображення, перенесемо початок координат у його центр перед застосуванням оператора, а після трансформацій повернемо у початкову точку.

Отже, повне перетворення для перенесення зображення у фронтально-паралельну площину має вигляд $H = T^{-1}R(v_1,v_2)T$, де T — це матриця зсуву центра зображення у початок координат.

$$T = \begin{pmatrix} 1 & 0 & -w/2 \\ 0 & 1 & -h/2 \\ 0 & 0 & 1 \end{pmatrix}$$

Даний підхід можна легко розширити до випадку вирівнювання аркушу в косу лінію, де кут між групами ліній є відомий кут $\alpha = 60$ градусів. Для цього візьмемо точку v_2, яка є зникомою точкою косих ліній, повернемо її на кут $\frac{\pi}{2} - \alpha$ проти годинникової стрілки і позначимо точкою $v_2^{'}$.

$$v_2^{'} = \begin{pmatrix} cos(\frac{\pi}{2} - \alpha) & sin(\frac{\pi}{2} - \alpha & 0 \\ -sin(\frac{\pi}{2} - \alpha) & cos(\frac{\pi}{2} - \alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} v_{2_x} \\ v_{2_y} \\ v_{2_z} \end{pmatrix}$$

Таким чином отримуємо прямий кут між точками v_1 та $v_2^{'}$ (рис.3.4). Після цього треба застосувати до кожного пікселя зображення перспективне перетворення $H^{'} = T^{-1}R(v_1, v_2^{'})T$. Деформація зображення за допомогою H або $H^{'}$ відновлює паралельність прямих та правильний кут між двома групами ліній.

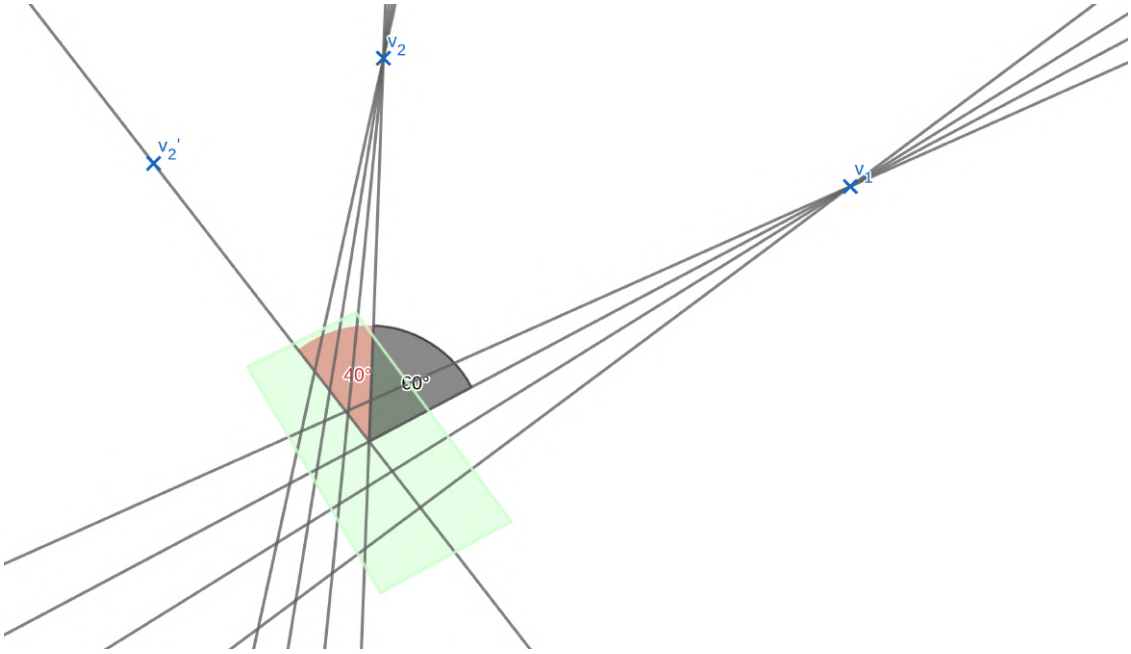

Рисунок 3.4 – Поворот точки v_2.

3.5 Знаходження обмежувальної рамки

Для того, щоб отримати зображення саме аркушу перетвореного зображення, знайдемо його обмежувальну рамку (bounding box). Для цього застосуємо відображення H або H' до координат кутів зображення:

$$H(0,0,1)^T = HA = A'$$

$$H(0,h,1)^T = HB = B^{'}$$

$$H(w,h,1)^T = HC = C'$$

$$H(w,0,1)^T = HD = D^{'}$$

Де A,B,C,D - координати кутів пікселів зображення до перетворення, $A^{'}, B^{'}, C^{'}, D^{'}$ - координати кутів зображення після перетворення, де w,h - ширина і висота початкового зображення, відповідно.

Тепер запишемо координати обмежувальної рамки.

$$x_{max} = max(\frac{A^{'}_x}{A^{'}_z}, \frac{B^{'}_x}{B^{'}_z}, \frac{C^{'}_x}{C^{'}_z}, \frac{D^{'}_x}{D^{'}_z}), \quad x_{min} = min(\frac{A^{'}_x}{A^{'}_z}, \frac{B^{'}_x}{B^{'}_z}, \frac{C^{'}_x}{C^{'}_z}, \frac{D^{'}_x}{D^{'}_z})$$

$$y_{max} = max(\frac{A^{'}_y}{A^{'}_z}, \frac{B^{'}_y}{B^{'}_z}, \frac{C^{'}_y}{C^{'}_z}, \frac{D^{'}_y}{D^{'}_z}), \quad y_{min} = min(\frac{A^{'}_y}{A^{'}_z}, \frac{B^{'}_y}{B^{'}_z}, \frac{C^{'}_y}{C^{'}_z}, \frac{D^{'}_y}{D^{'}_z})$$

$$P_1 = (x_{min}, y_{min}), \quad P_2 = (x_{min}, y_{max})$$

$$P_3 = (x_{max}, y_{max}), \quad P_4 = (x_{max}, y_{min})$$

$$w' = x_{max} - x_{min}, h' = y_{max} - y_{min}$$

Варто зазанчити, що величини $w^{'}$, $h^{'}$ мають бути цілими для побудови саме цієї обмежувальної рамки. В подальшому в роботі можуть виникати малі обмежувальні рамки і округлення величин $w^{'}$,$h^{'}$ - може призвести до помилок.

$$O_1 = (0, 0), \quad O_2 = (0, h^{'})$$

$$O_3 = (w^{'}, h^{'}), \quad O_4 = (w^{'}, 0)$$

Далі за допомогою функції з OpenCV getPerspectiveTransform ми будуємо матрицю гомографії G. Для цього нам необхідно подати до неї точки P_1,P_2,P_3,P_4 та відповідні ним O_1,O_2,O_3,O_4 і за допомогою Гаусівського усунення з обраним оптимальним опорним елементом ми отримаємо необхідну матрицю 3×3.

Тепер до вхідного зображення треба застосувати перетворення GH або $GH^{'}$. Перетворення $H(H')$ зробить поворот початкового зображення, а G перенесе обмежувальну рамку в лівий верхній кут.

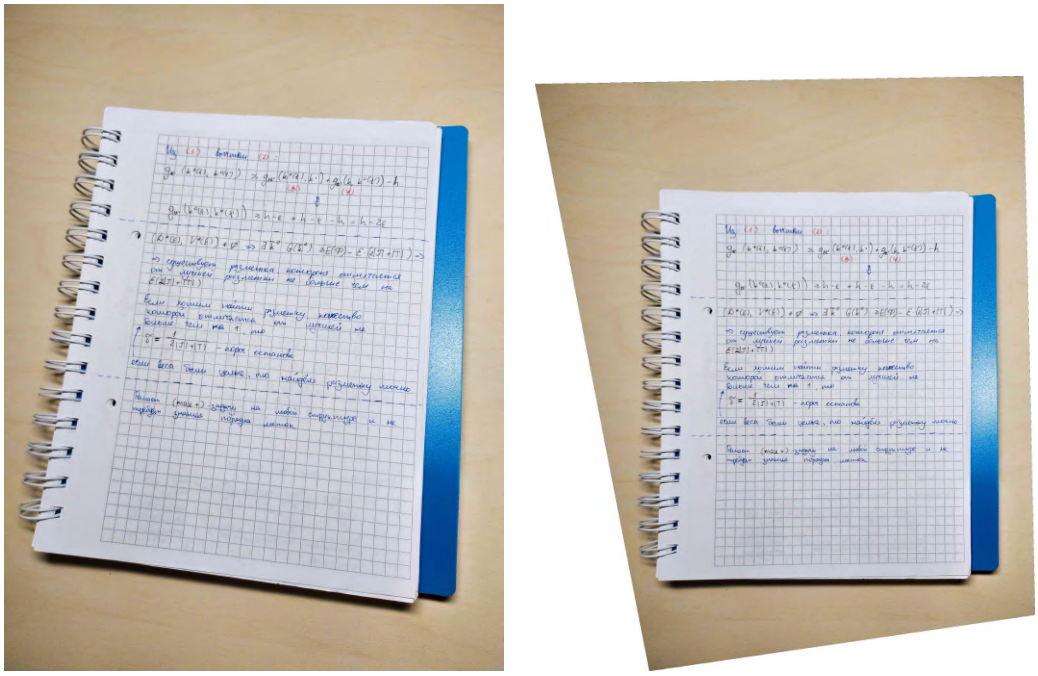

Рисунок 3.5 – Поворот зображення з розміткою в клітинку.

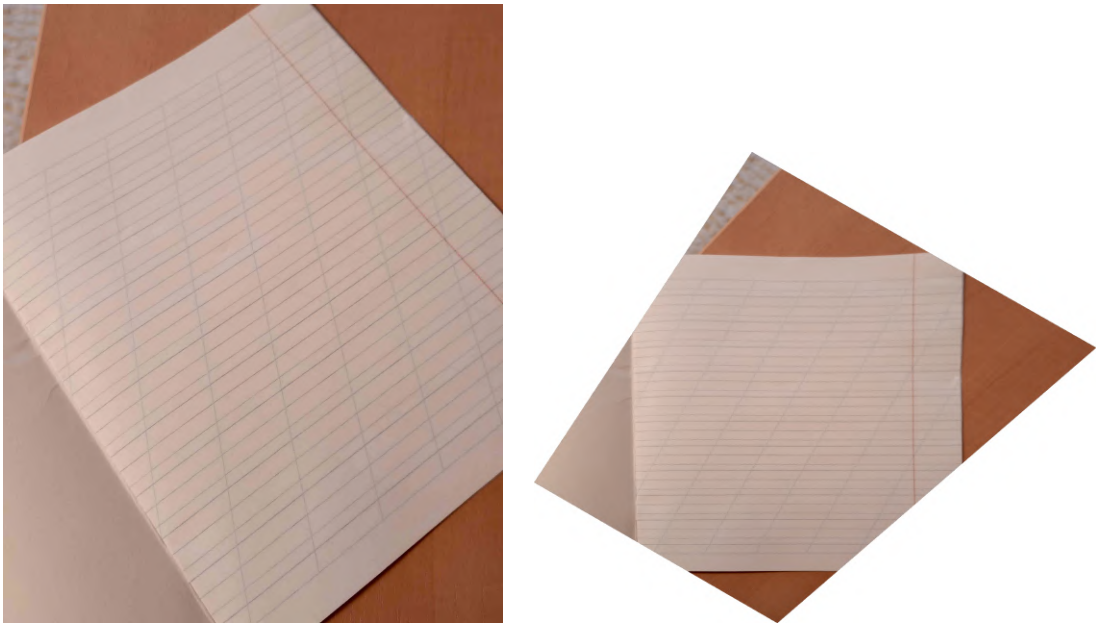

Рисунок 3.6 – Поворот зображення з розміткою в косу.

3.6 Пошук кутів аркушу

Наявність ліній на зображенні надає нам можливість знайти чотири точки аркушу, та відокремити його від фону. Для цього скористаємось множинами L_1, L_2 з котрих дістанемо бокові лінії.

$$l_{top} = \underset{l \in L_1}{\operatorname{argmin}} \left(- \frac{l_z}{l_y} \right), l_{left} = \underset{l \in L_2}{\operatorname{argmin}} \left(- \frac{l_z}{l_x} \right)$$

$$l_{bot} = \underset{l \in L_1}{\operatorname{argmax}} \left(- \frac{l_z}{l_y} \right), l_{right} = \underset{l \in L_2}{\operatorname{argmax}} \left(- \frac{l_z}{l_x} \right)$$

Запишемо точки перетину як векторний добуток ліній.

$$A = l_{top} \times l_{left}, B = l_{left} \times l_{bot}$$

$$C = l_{bot} \times l_{right}, D = l_{right} \times l_{top}$$

Теперь до отриманих точок застосовуємо перетворення GH і отримуємо точки A',B',C',D' з котрих будуємо ще чотири точки і створюємо матрицю гомографії.

Для аркушів з розміткою в косу положення точок A',B',C',D' не утворюють прямокутник. Виправити це можна скориставшись властивістю положення косих ліній. Для цього оберемо серед точок A',B',C',D' ті, що мають мінімальну та максимальну координату x.

$$P_1 = \underset{p \in \{A',B',C',D'\}}{\operatorname{argmin}} \left(\frac{p_x}{p_z}\right),\ P_2 = \underset{p \in \{A',B',C',D'\}}{\operatorname{argmax}} \left(\frac{p_x}{p_z}\right)$$

Тепер запишемо нові точки A'',B'',C'',D'', що сформують прямокутник.

$$A'' = (P_{1_x}, A'_y, A'_z),\ B'' = (P_{1_x}, B'_y, B'_z)$$

$$C'' = (P_{2_x}, C'_y, C'_z),\ D'' = (P_{2_x}, D'_y, D'_z)$$

В подальшому точки A'',B'',C'',D'' використовуються для побудови матриці гомографії так як це робилось раніше.

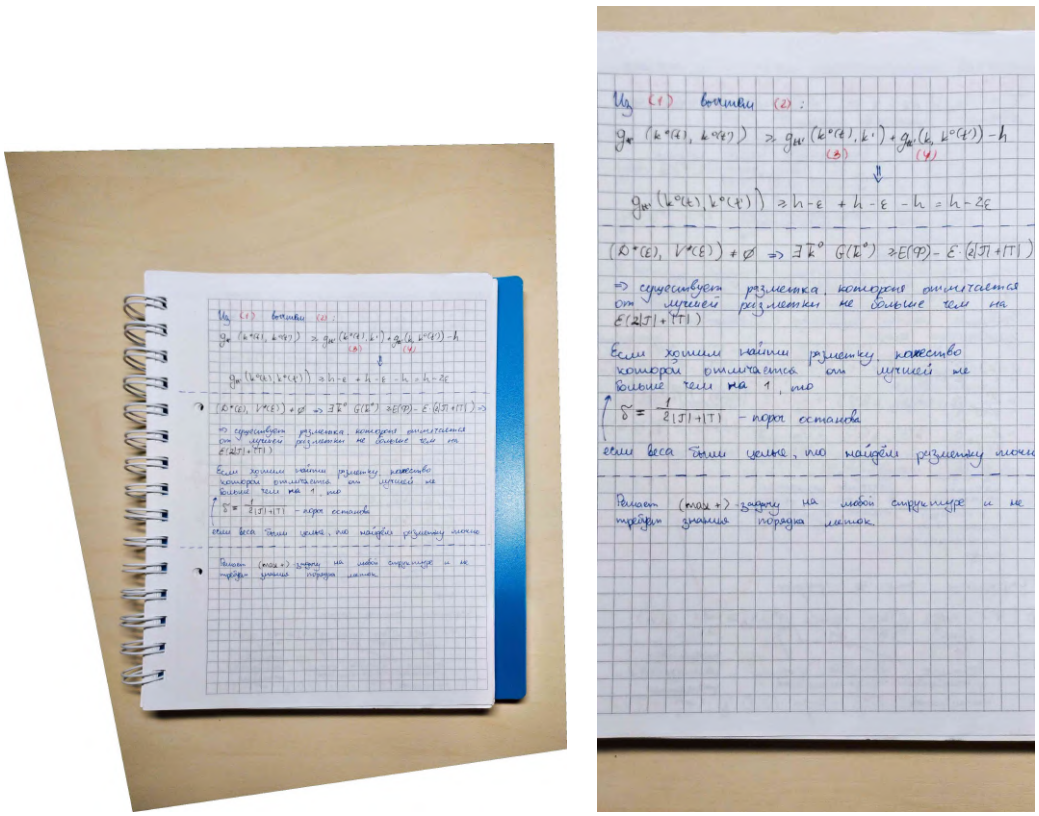

Рисунок 3.7 – Виділення аркушу з розміткою в клітинку.

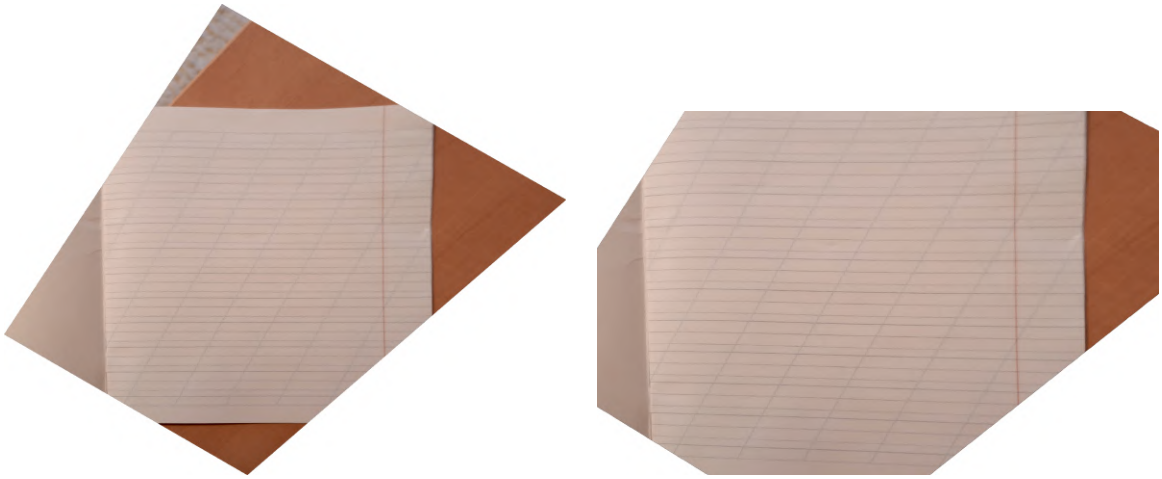

Рисунок 3.8 – Виділення аркушу з розміткою в косу.

Висновки до розділу 3

В цьому розділі було запропоновано алгоритм пошуку зникомих точок та ректифікації зображення. Для цього було розглянуто як бінаризувати та шукати лінії на зображенні, створювати матрицю повороту за двома зникомими точками, оброблювати зникому точку за необхідністю, будувати обмежувальну рамку та виділяти зображення аркушу з фону.

4 РЕКТИФІКАЦІЯ ЗОБРАЖЕННЯ РОЗЛІНОВАНОГО АРКУШУ ЗА ОДНІЄЮ ЗНИКОМОЮ ТОЧКОЮ

В цьому розділі буде запропоновано спосіб ректифікації аркушу, розмітка котрого містить один набір ліній. Тобто буде запропоновано алгоритм обробки аркушу з розміткю в лінію.

Бінаризація зображення, пошук ліній та пошук зникомої точки є такий самий як і в попередньому розділі.

4.1 Знаходження та усунення матриці внутрішніх параметрів камери

Запишемо модель камери обскури як $P = K[R|t]$, де

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}, R = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix}, t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$

K — матриця внутрішніх параметрів камери (calibration matrix). (c_x,c_y) — оптичний центр. $f_x = \frac{F}{p_x}, f_y = \frac{F}{p_y}$ — фокусні відстані в пікселях. F — фокусна відстань в мм. p_x, p_y — розміри пікселів в мм. $s = f_x tan(\alpha)$ — коефіцієнт скосу.

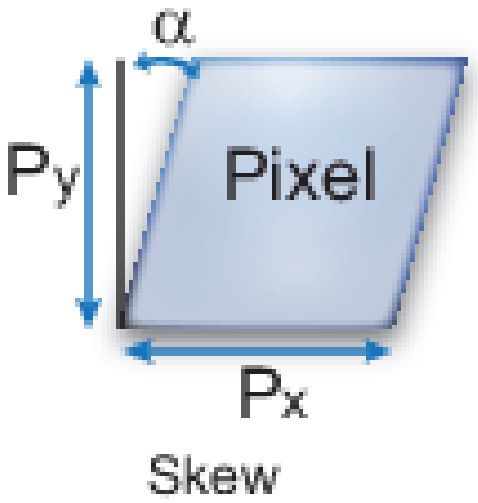

Рисунок 4.1 – Модель фізичного пікселя.

Тобто перетворення координат реального світу у координати зображення можна записати як $x^{'} = Px$.

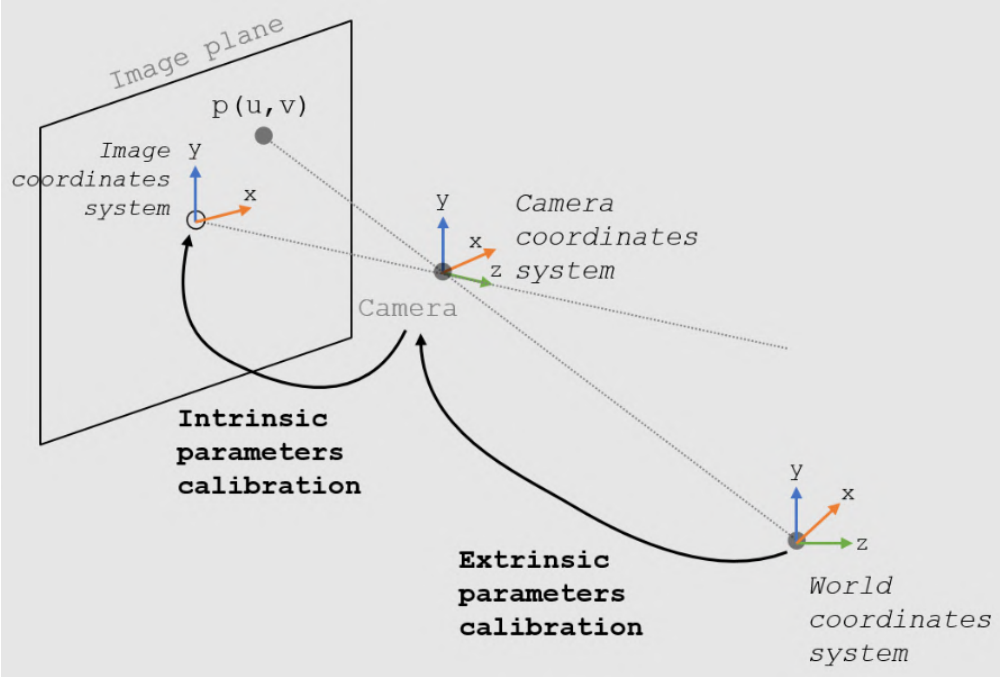

Рисунок 4.2 – Модель внутрішніх та зовнішніх параметрів камери.

Такі параметри як розмір пікселя в мм необхідно знаходити для кожної камери окремо. Матрицю внутрішніх параметрів камери можна знайти для кожної камери окремо, або відкалібрувати камеру так, як це

реалізовано в OpenCv рис.4.3. На вхід аглоритму подається набір зображень з фіксованою камерою та різним положенням шахматної дошки.

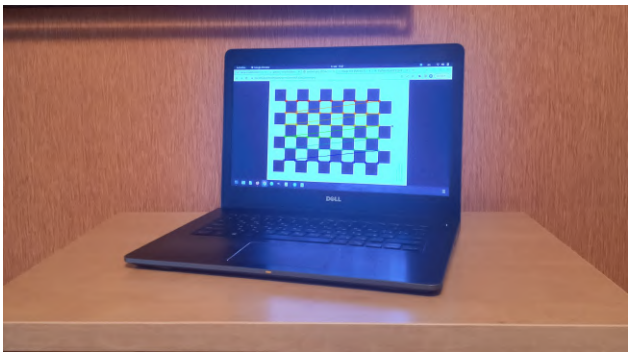

Рисунок 4.3 – Калібрування камери з OpenCV.

В подальших прикладах будемо використовувати наступну матрицю внутрішніх параметрів K та обернену їй K^{-1}.

$$K = \begin{pmatrix} 6.63063710e+03 & 0.00000000e+00 & 1.94171924e+03 \\ 0.00000000e+00 & 5.89316597e+03 & 1.53093796e+03 \\ 0.00000000e+00 & 0.00000000e+00 & 1.00000000e+00 \end{pmatrix}$$

Також будемо вважати, що фізичний піксель є квадратним і довжина його сторони є 0.00112 мм.

4.2 Горизонталізація та паралелізація ліній

В цій роботі ми припускаємо, що усунення матриці внутрішніх параметрів має призвести до усунення скосу та інших спотворень. Точки зображення без матриці K запишемо наступним чином.

$$K^{-1}x' = K^{-1}K[R|t]x = [R|t]x$$

Тепер побудуємо таку матрицю, що зробить горизонтальні лінії паралельними. Для цього застосуємо горизонтальну зникому точку v_2 та точку $v_1 \in \{(0,0,1),(0,1,0),(1,0,0)\}$.

Перед побудовою матриці повороту потрібно застосувати K^{-1} до v_2. Тобто $v_2^{'} = K^{-1}v_2$. Цей крок зроблений для того, щоб матриця повороту працювала з зображенням, котре перетворене K^{-1}.

Зазначимо, що точки v_1,$v_2^{'}$ треба нормувати перед побудовою матриці H. А також треба нормувати рядки матриці H.

$$H = \begin{pmatrix} \frac{v_2^{'}}{|v_2^{'}|} \\ \frac{v_2^{'} \times v_1}{|v_2^{'} \times v_1|} \\ \frac{v_2^{'} \times (v_2 \times v_1)}{|v_2^{'} \times (v_2 \times v_1)|} \end{pmatrix}$$

Варто зазначити, що знак рядків матриці H може змінюватись в залежності від застосованих у ній векторів. В подальших роботах визначення знаку рядків автоматизується.

Перетворення HK^{-1} застосовується до зображення так само, як і перетворення в попередніх розділах. Тобто HK^{-1} застосовується до країв зображення з котрих будується обмежувальна рамка і перспективне перетворення G. Застосувавши GHK^{-1} ми отримаємо зображення малого розміру, тому для того щоб продемонструвати як працюють ці перетворення застосуємо матрицю масштабування S. Для прикладів в цій роботі використаємо наступну S.

$$S = \begin{pmatrix} 10000 & 0 & 0 \\ 0 & 10000 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

SHK^{-1} застосовуємо до країв зображення, а з перетворених країв будуємо G. Перетворення $GSHK^{-1}$ дасть нам доступне для огляду

зображення з горизонталізованими та паралелізованими лініями рис.4.4.

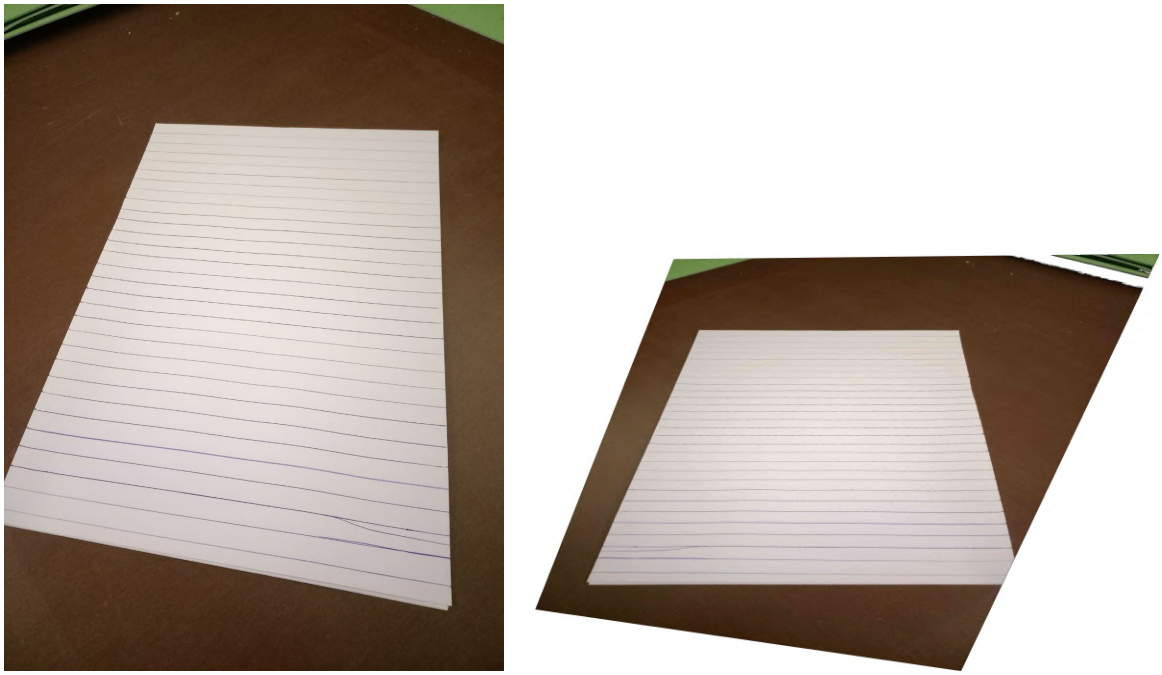

Рисунок 4.4 – Застосування $GSHK^{-1}$.

4.3 Пошук кута між площиною сенсора та площиною аркушу

Як ми можемо бачити на рис.4.4 перетворення HK^{-1} робить лінії горизонтальними та паралельними, однак відстань між лініями не є однаковою, а зменшується з ростом перспективи. Тож використаємо цю властивість для побудови другого перетворення, що зробить відстані між лініями однаковими.

На рис.4.5 точки P_i позначають горизонтальні лінії на сенсорі камери, відстань між котрими не однакова. Точки $S+i\vec{\tau}$ позначають відповідні лінії на площині аркушу.

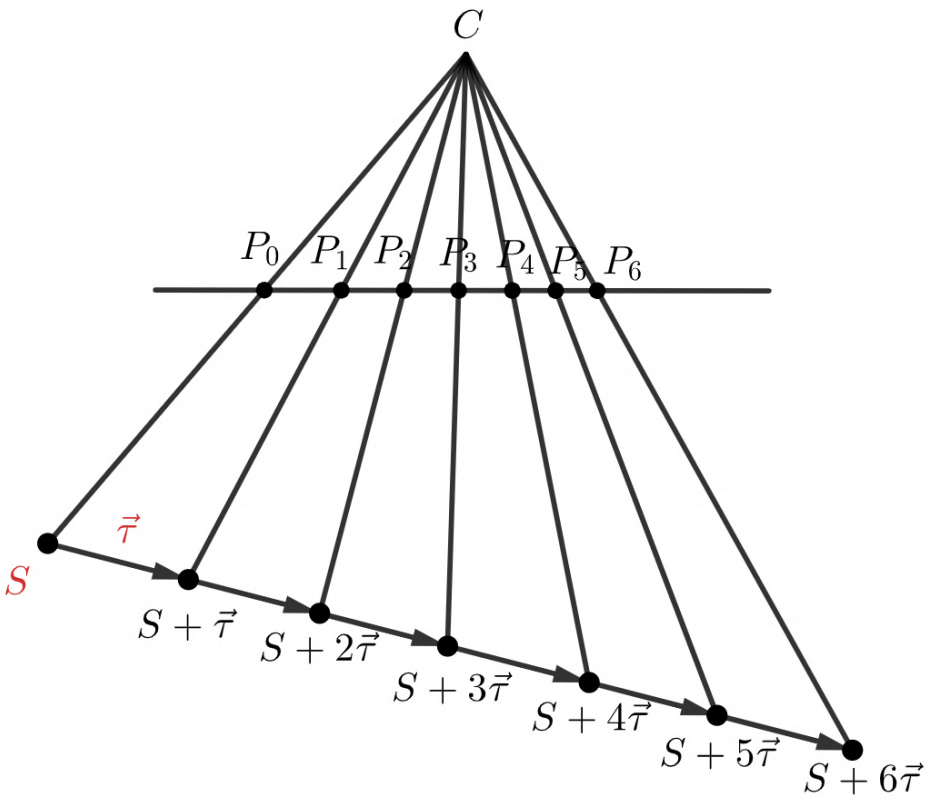

Рисунок 4.5 – Площина сенсору та площина аркушу.

Побудуємо систему рівнянь, вирішивши котру знайдемо точку S та вектор $\vec{\tau}$.

$$P_{0_y} = \frac{S_y}{S_z}, P_{1_y} = \frac{S_y + \tau_y}{S_z + \tau_z}, P_{2_y} = \frac{S_y + 2\tau_y}{S_z + 2\tau_z}, ...$$

$$P_{i_y} = \frac{S_y + i\tau_y}{S_z + i\tau_z},\ P_{i_y}S_z + P_{i_y}i\tau_z - S_y - i\tau_y = 0$$

$$\begin{cases} P_{0_y}S_z - S_y = 0 \\ P_{1_y}S_z + P_{1_y}\tau_z - S_y - \tau_y = 0 \\ P_{2_y}S_z + P_{2_y}2\tau_z - S_y - 2\tau_y = 0 \\ \tau_y^2 + \tau_z^2 - d^2 = 0 \end{cases}$$

d — відстань між лініями в пікселях. Для обрахунку d необхідно

знати розмір фізичного пікселю та відстань між реальними горизонтальними лініями(7-8мм). Розв'язком є величини S_z,S_y,τ_z,τ_y.

$$S_y = 2P_{0_y} d \left(-P_{1_y} + P_{2_y}\right) \sqrt{\frac{1}{c_1}}$$

$$c_1 = P_{0_y}^2 P_{1_y}^2 - 4P_{0_y}^2 P_{1_y} P_{2_y} + 4P_{0_y}^2 P_{2_y}^2 + P_{0_y}^2 + 2P_{0_y} P_{1_y}^2 P_{2_y} - 4P_{0_y} P_{1_y} P_{2_y}^2 - 4P_{0_y} P_{1_y} +$$
$$+ 2P_{0_y} P_{2_y} + P_{1_y}^2 P_{2_y}^2 + 4P_{1_y}^2 - 4P_{1_y} P_{2_y} + P_{2_y}^2$$

$$S_z = 2d \left(-P_{1_y} + P_{2_y}\right) \sqrt{\frac{1}{c_2}}$$

$$c_2 = P_{0_y}^2 P_{1_y}^2 - 4P_{0_y}^2 P_{1_y} P_{2_y} + 4P_{0_y}^2 P_{2_y}^2 + P_{0_y}^2 + 2P_{0_y} P_{1_y}^2 P_{2_y} - 4P_{0_y} P_{1_y} P_{2_y}^2 - 4P_{0_y} P_{1_y} +$$
$$+ 2P_{0_y} P_{2_y} + P_{1_y}^2 P_{2_y}^2 + 4P_{1_y}^2 - 4P_{1_y} P_{2_y} + P_{2_y}^2$$

$$\tau_y = d \left(P_{0_y} P_{1_y} - 2P_{0_y} P_{2_y} + P_{1_y} P_{2_y}\right) \sqrt{\frac{1}{c_3}}$$

$$c_3 = P_{0_y}^2 P_{1_y}^2 - 4P_{0_y}^2 P_{1_y} P_{2_y} + 4P_{0_y}^2 P_{2_y}^2 + P_{0_y}^2 + 2P_{0_y} P_{1_y}^2 P_{2_y} - 4P_{0_y} P_{1_y} P_{2_y}^2 -$$
$$- 4P_{0_y} P_{1_y} + 2P_{0_y} P_{2_y} + P_{1_y}^2 P_{2_y}^2 + 4P_{1_y}^2 - 4P_{1_y} P_{2_y} + P_{2_y}^2$$

$$\tau_z = -d \left(P_{0_y} - 2P_{1_y} + P_{2_y}\right) \sqrt{\frac{1}{c_4}}$$

$$c_4 = P_{0_y}^2 P_{1_y}^2 - 4P_{0_y}^2 P_{1_y} P_{2_y} + 4P_{0_y}^2 P_{2_y}^2 + P_{0_y}^2 + 2P_{0_y} P_{1_y}^2 P_{2_y} - 4P_{0_y} P_{1_y} P_{2_y}^2 -$$
$$- 4P_{0_y} P_{1_y} + 2P_{0_y} P_{2_y} + P_{1_y}^2 P_{2_y}^2 + 4P_{1_y}^2 - 4P_{1_y} P_{2_y} + P_{2_y}^2$$

В процесі дослідження розглядалось два сопосби вирівняти зображення так, щоб відстань між горизонтальними лініями стала однаковою. В першому випадку можна знайти кут між полщиною сенсору та площиною аркушу. В другому ми можемо побудувати рівняння площини аркушу та перенести кожен піксель сенсору на знайдену площину. Другий спосіб наразі потребує поліпшень, котрі будуть

розглянуті в окремому розділі, тому знайдемо кут між вектором τ, що лежить у площині аркушу та площиною сенсору рис.4.6.

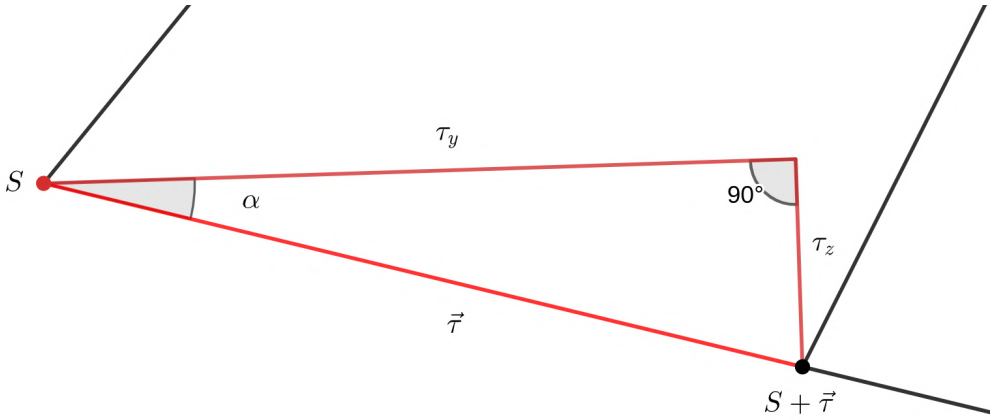

Рисунок 4.6 – Кут α між площинами.

$$\alpha = \arctan(\frac{\tau_z}{\tau_y})$$

Розглянемо приклад пошуку такого кута. Для цього нам необхідно знайти три точки P_0,P_1,P_2, що належать зображенню, котре було отримано застосуванням HK^{-1} до початкового зображення. За визначенням точки P_i належать вертикальній прямій, що перетинає горизонтальні. Наразі маємо декілька способів знайти ці точки. Одним з таких є застосування HK^{-1} до ліній, що перетинаються у горизонтальній зникомій точці та пошук вертикальної прямої у відповідній площині. Також можна використати перетворене $GSHK^{-1}$ зображення — знайти на ньому лінії та належні точки, а потім до точок застосувати $(GSHK^{-1})^{-1}$. Таким чином отримаємо точки на початковому зображенні, до котрих потім застосуємо HK^{-1}.

Зауважемо, що задача вибору трьох точок P_i та відстанню між ними в цій роботі не досліджується, тому їх вибір є довільним.

Для зручності скористаємось другим способом пошуку точок. Для цього відповідні точки на зображенні з горизонталізованими лініями

перетворимо у точки початкового зображення. Застосування цієї процедури до всіх точок зображення показано на рис.4.7.

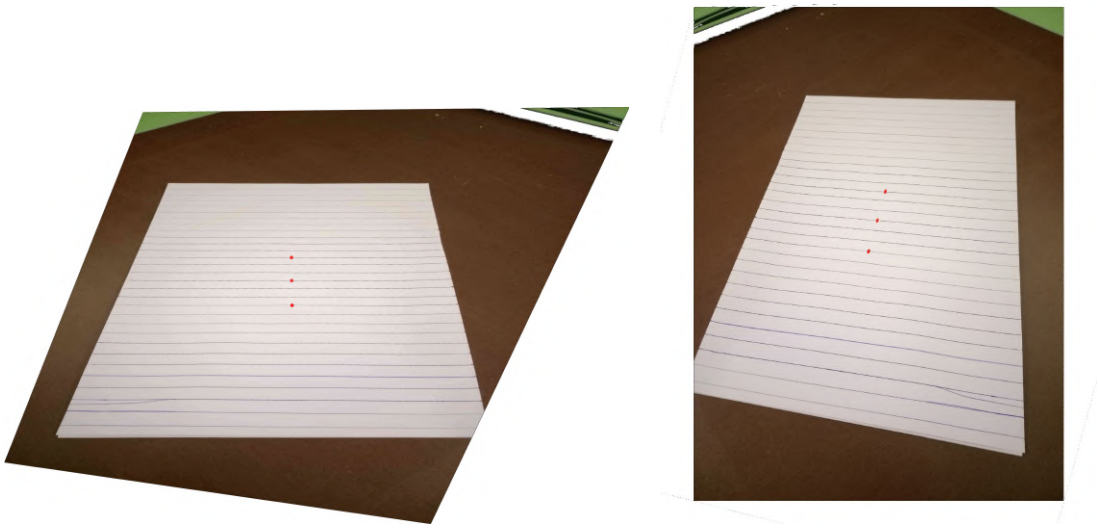

Рисунок 4.7 – Перетворення зображення в початкове положення.

До отриманих точок на початковому зображені застосовуємо HK^{-1} та знаходимо координати.

Для прикладу на рис.4.7 точка S та вектор $\vec{\tau}$ матимуть наступні значення.

$$S_y = -18738.05882183737, S_z = -503595.9365012859,$$

$$\tau_y = 24156.64707938238, \tau_z = 11572.851492959762$$

Відповідно $\alpha = 25.5$.

4.4 Поворот аркушу

Для повороту необхідно перевести координати зображення із $2D$ в $3D$. Побудуємо матрицю $P23$, що зробить таке перетворення.

$$P23 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$P23_{13}$,$P23_{23}$ — координати навколо котрих відбувається поворот.

Запишемо матрицю R трьохвимірного повороту навколо Ox.

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Запишемо матрицю T, що зсуне зображення по Oz. Для прикладів в цій роботі використаємо $T_{34} = 400$, а для загального випадку треба шукати оптимальне значення.

$$T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 400 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Запишемо матрицю $P32$, що переведе зображення із $3D$ в $2D$. Значення $P32_{11}, P32_{22}$ задають масштаб зображення.

$$P32 = \begin{pmatrix} 400 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Також введемо матрицю M, що зсуватиме зображення в центр. Цей крок необхідний для зменшенння спотворення. w,h — ширина та висота зображення.

$$M = \begin{pmatrix} 1 & 0 & -\frac{w}{2} \\ 0 & 1 & -\frac{h}{2} \\ 0 & 0 & 1 \end{pmatrix}$$

Результуюче перспективне перетворення H.

$$H = M^{-1} \cdot P32 \cdot T \cdot R \cdot P23 \cdot M$$

Застосування перетворення H зображено на рис.4.8, $\alpha = 25.5$.

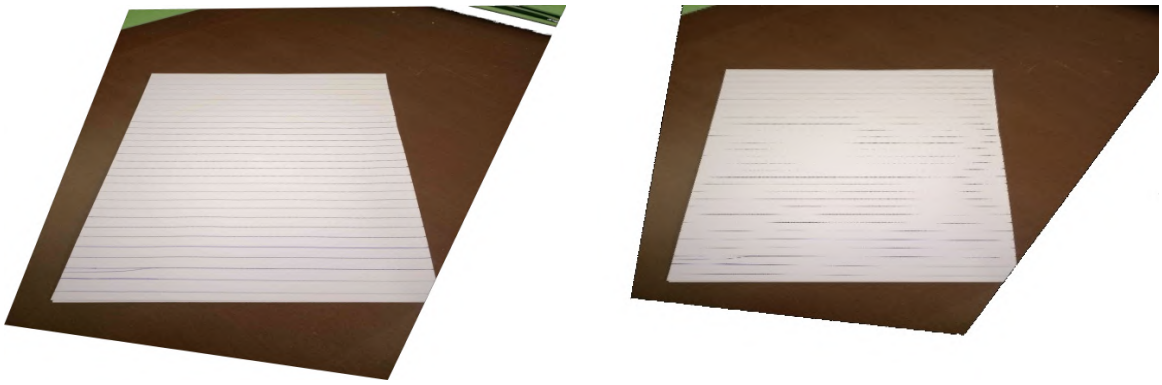

Рисунок 4.8 – Поворот аркушу А4 з горизонталізованими лініями.

Перетворення H може містити матриці масштабування та зсуву. Для

їх визначення необхідна інформація від користувача застосунку.

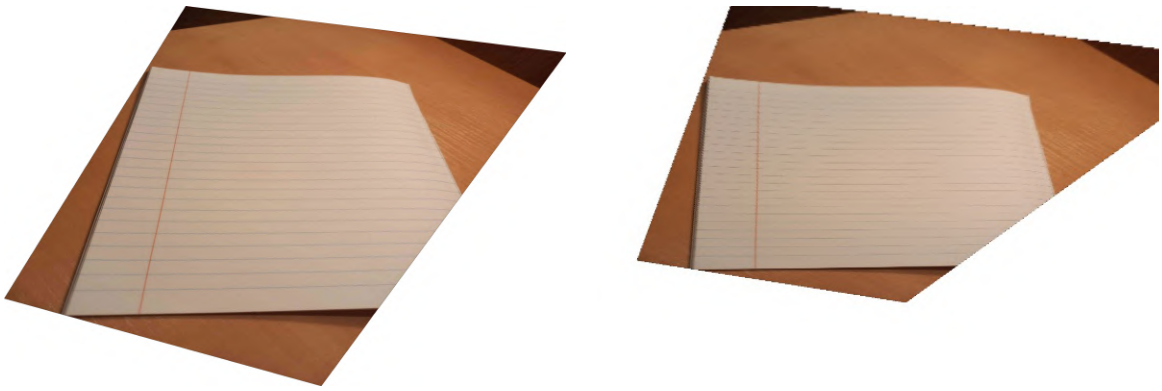

Рисунок 4.9 – Поворот аркушу А5 з горизонталізованими лініями.

4.5 Поворот аркушу через співставлення точок площин

Окрім повороту через кут між площинами можна побудувати рівняння площини та співставити відповідні точки.

Запишемо базис.

$$\vec{x} = (1,0,0)$$

$$\vec{\tau} = (0, \frac{\tau_y}{||\tau_y||}, \frac{\tau_z}{||\tau_z||}) = (0, \cos(\alpha), \sin(\alpha))$$

$$S = (S_x, S_y, S_z)$$

Рівняння площини знайдемо через визначник.

$$\begin{vmatrix} x - S_x & y - S_y & z - S_z \\ 1 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) \end{vmatrix} =$$

$$= -\sin(\alpha)y + sin(\alpha)S_y + \cos(\alpha)z - cos(\alpha)S_z = 0$$

$$Ax + By + Cz + D = 0$$

$$A = 0, B = -\sin(\alpha) = -\tau_z$$

$$C = \cos(\alpha) = \tau_y$$

$$D = \sin(\alpha)S_y - \cos(\alpha)S_z = \tau_z S_y - \tau_y S_z$$

Рівняння площини, котрій належить аркуш.

$$-\tau_z y + \tau_y z + \tau_z S_y - \tau_y S_z = 0$$

Розглянемо найпростіший спосіб вирішення цієї задачі. Для цього через кожну точку на зображенні та точку камери проведемо пряму, котра перетне площину аркушу рис.4.10.

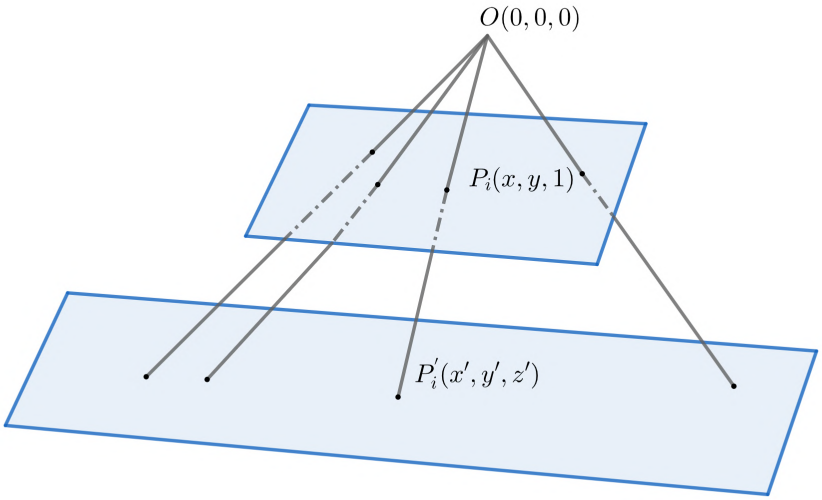

Рисунок 4.10 – Проекція з площини зображення на площину аркушу.

Рівняння прямої, що проходить крізь (0,0,0) та $P_i = (x_i,y_i,1)$.

$$\frac{x-0}{x_i-0} = \frac{y-0}{y_i-0} = \frac{z-0}{1-0}$$

$$y = zy_i, x = zx_i$$

$$-\tau_z z y_i + \tau_y z + \tau_z S_y - S_z \tau_y = 0$$

$$z_i^{'} = z = \frac{S_z \tau_y - \tau_z S_y}{\tau_y - \tau_z y_i}$$

$$y_i^{'} = zy_i = \frac{S_z \tau_y - \tau_z S_y}{\tau_y - \tau_z y_i} y_i$$

$$x_i^{'} = zx_i = \frac{S_z \tau_y - \tau_z S_y}{\tau_y - \tau_z y_i} x_i$$

Такий підхід має два недоліки. Перший — це ітеративна обробка кожного пікселю. Другий — між точками $P_i^{'}$ є проміжки, котрі треба зафарбовувати.

Тому спробуємо побудувати інший спосіб, що орієнтований на використання перетворень площин. Як зображено на рис.4.11 для кожного кута K_i в площині горинталізованого зображення знайдемо відповідну точку $K_i^{'}$ в площині аркушу та побудуємо перетворення G, що розтягне побудовану на $K_i^{'}$ обмежувальну рамку, однак застосовувати таке перетворення потрібно до зображення, що вже знаходиться в площині аркушу. Тому задача полягає в тому, щоб побудувати таке перетворення H.

$$H = G \cdot H_2 \cdot H_1 \cdot K^{-1}$$

Перетворення H_2 мусить перенести точки горизонталізованої площини в площину аркушу. Як показано в попередньому методі, при застосуванні декартових координат точки площини аркушу явно залежать від точок площини зображення, тому побудова такого перетворення буде досліджуватись в проективному просторі.

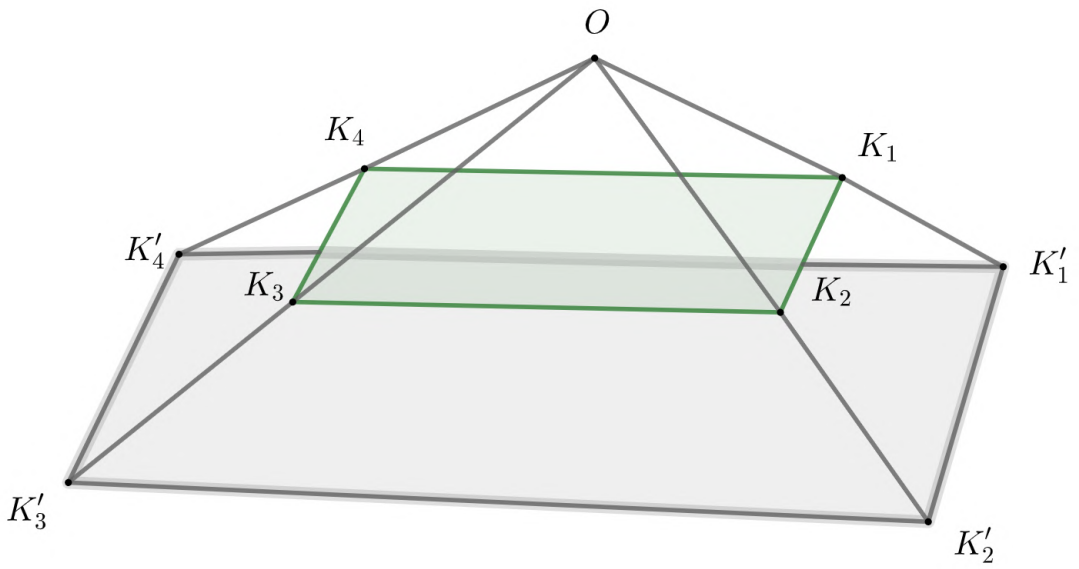

Рисунок 4.11 – Проекція кутів зображення на кути аркушу.

Висновки до розділу 4

В цьому розділі розглянуто спосіб ректифікації зображення за однією зникомою точкою. Для цього ми розглянули як знаходити матрицю внутрішніх параметрів K, будувати перетворення, що горизонталізує лінії, оцінювати параметри площини аркушу та кут між нею та площиною зображення. Зробили поворот аркушу за знайденим кутом та сформулювали задачу повороту перетворенням площин.

5 ВИЗНАЧЕННЯ ФОРМАТУ АРКУШУ ТА ВИДАЛЕННЯ РОЗМІТКИ.

В цьому розділі розглянемо алгоритми обробки розмітки: пошук відсутніх ліній, визначення формату аркушу, видалення розмітки.

5.1 Пошук відсутніх ліній

Щоб визначити формат необхідно знайти всі лінії на аркушу. Відсутність деяких може бути пояснена роботою алгоритму Хафа як на початковому зображенні так і на вирівняному. Проблему зображено на рис.5.1.

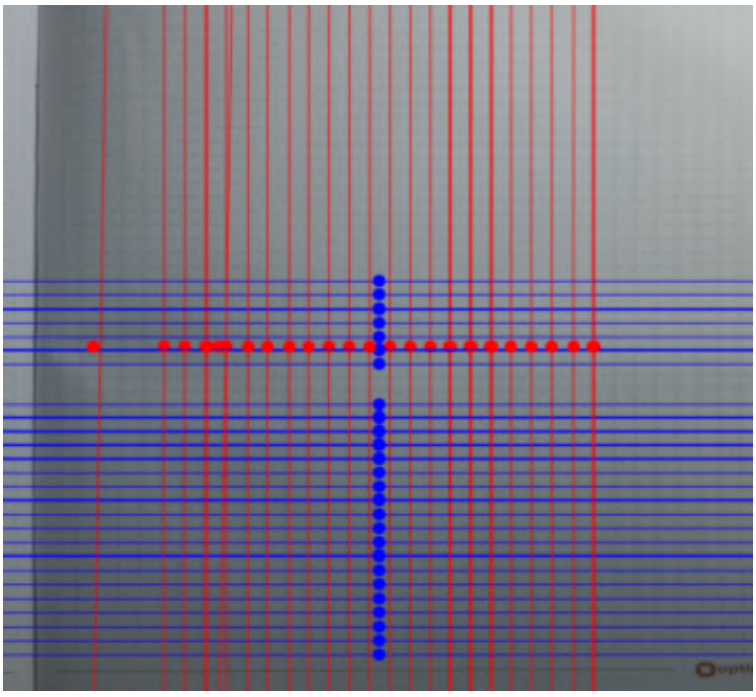

Рисунок 5.1 – Частина ліній знайдена алгоритмом Хафа.

До ректифікованого зображення застосовуємо раніше зазанчені бінаризації та запускаємо алгорим Хафа. Оскільки лінії паралельні, то в подальшому для обчислень будемо використовувати координати точок,

що ним належать. Розглянемо випадок доповнення вертикальних ліній. Через V позначимо множину координат x вертикальних ліній. w,h — ширина та висота зображення. Найкращу відстань d_1 між двома лініями знаходимо наступним чином.

$$d_1 \in \underset{\substack{d=\left|x_1-x_2\right| \\ \forall x_1,\ x_2 \in V \\ z=\frac{w}{d}}}{\arg\max} \frac{\sum\limits_{\delta \in L}\Big[\sum\limits_{x \in V}\big[x \in \delta\big] > 0\Big] + \sum\limits_{\delta \in R}\Big[\sum\limits_{x \in V}\big[x \in \delta\big] > 0\Big]}{z}$$

$$L = \big\{[x-\epsilon, x+\epsilon] : x = x_1 - nd, x \geq 0, n \in N\big\}$$

$$R = \big\{[x-\epsilon, x+\epsilon] : x = x_1 + nd, x \leq w, n \in N\big\}$$

Зберігаємо множини координат $Detected$ та $Missed$, що містять знайдені та пропущені координати ліній відповідно.

$$Detected = \{m : \forall \delta \in L \cup R,\ m = \frac{\sum\limits_{\forall x \in V} x \mathbb{1}\big[x \in \delta\big]}{\sum\limits_{\forall x \in V} \mathbb{1}\big[x \in \delta\big]},\ \sum_{\forall x \in V} \mathbb{1}\big[x \in \delta\big] > 0\}$$

$$Missed = \{\delta_c : \forall \delta \in L \cup R,\ \sum_{\forall x \in V} \mathbb{1}\big[x \in \delta\big] = 0\}$$

Множина $Detected$ складається з середніх значень координат, що потрапляють у відрізки множин L,R. А множина $Missed$ містить центри таких відрізків, оскільки до них не потрапляє жодна координата. У випадку з розміткою у клітинку запишемо кількість вертикальних клітинок $m = |Detected| + |Missed| - 1$. Аналогічно шукаємо кількість горизонтальних клітинок n.

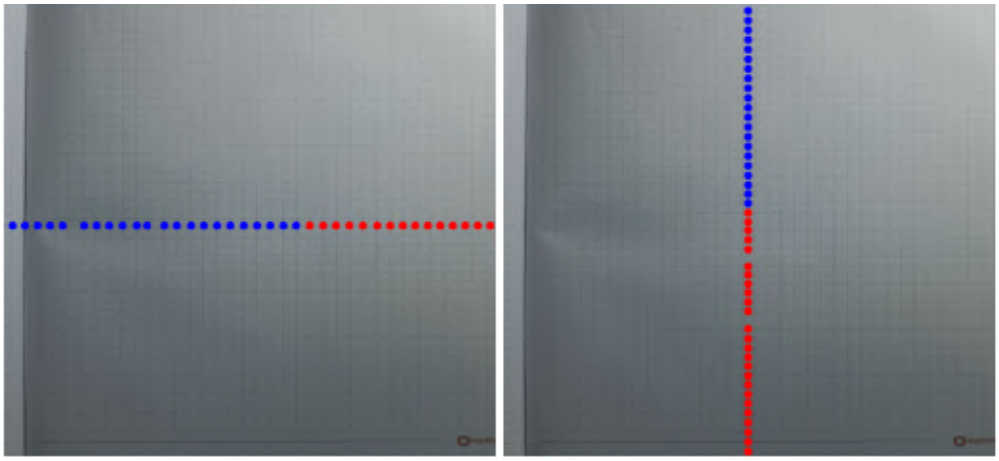

Рисунок 5.2 – Знайдені відсутні лінії.

Відстані d_1, d_2 можуть не співпадати, з цього випливає, що клітинка на зображенні не квадратна. Це можна виправити змінивши розмір зображення. На рис.5.3 ми повторили алгоритм на зображенні з квадратною клітинкою.

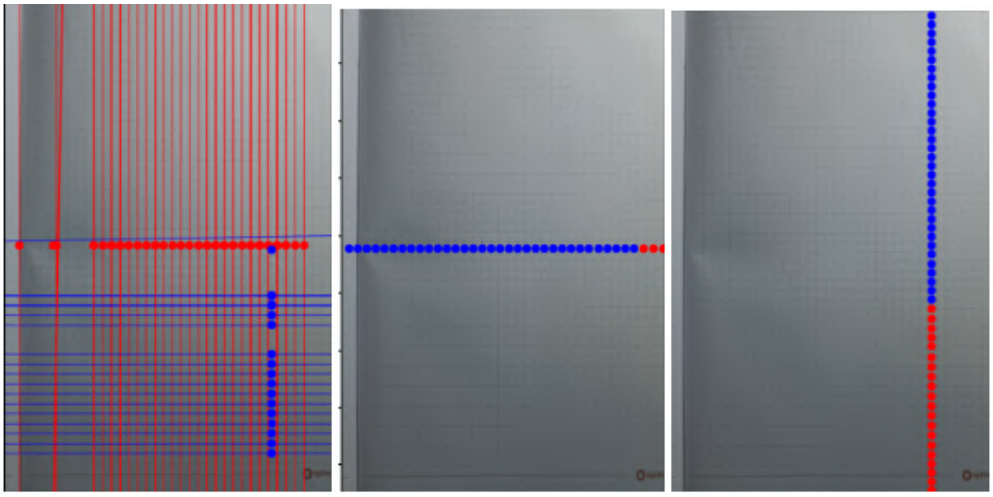

Рисунок 5.3 – Повторення алгоритму на зображенні зі зміненим розміром.

5.2 Визначення формату

Для визначення формату аркушу будемо користуватись тим фактом, що розмір сторони клітинки може бути $5, 7, 10$ мм. Площа аркушу $A4$, $S_{A4} = 210 * 297$ мм, а площа $A5$, $S_{A5} = 148 * 210$ мм.

Припустимо, що площа аркушу може бути $S_5 = 5^2mn$, $S_7 = 7^2mn$, $S_{10} = 10^2mn$. Для кожної площі вираховуємо її відношення до реального розміру аркушу формату $A4$ та $A5$.

$$R_{5A4} = S_5/S_{A4}, R_{5A5} = S_5/S_{A5}$$

$$R_{7A4} = S_7/S_{A4}, R_{7A5} = S_7/S_{A5}$$

$$R_{10A4} = S_{10}/S_{A4}, R_{10A5} = S_{10}/S_{A5}$$

Використовуючи отримані відношення знайдемо розмір клітинки.

$$D_5 = |R_{5A4} - 1| + |R_{5A5} - 1|$$

$$D_7 = |R_{7A4} - 1| + |R_{7A5} - 1|$$

$$D_{10} = |R_{10A4} - 1| + |R_{10A5} - 1|$$

Найменший D_i відповідатиме розміру клітинки i. Визначивши розмір клітинки знайдемо формат аркушу.

$$P_{A4} = |R_{iA4-1}|, P_{A5} = |R_{iA5} - 1|$$

Серед двох P_{Ak} обираємо найменший та відповідний йому формат аркушу.

5.3 Пошук координат розмітки

Для подальшої роботи з розміткою необхідно знати координату кожного пікселю, що їй належить або утворює її "скелет". Через різні деформації аркушу знайдені Хафом прямі можуть проходити повз лінії документу рис.5.4

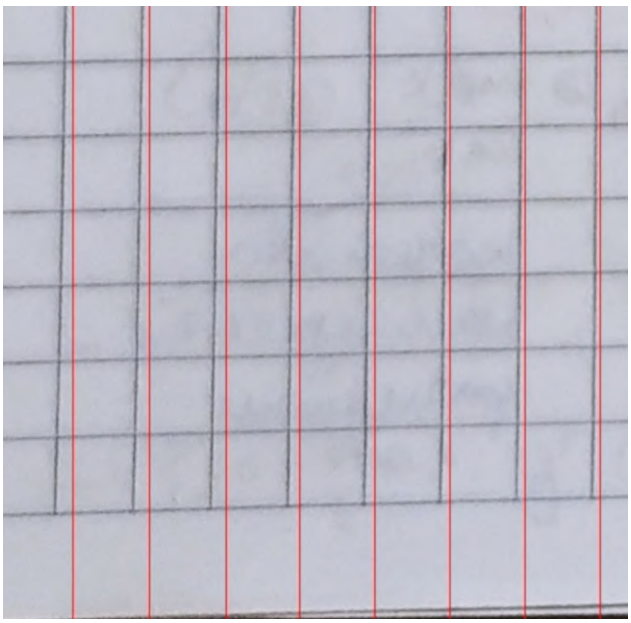

Рисунок 5.4 – Прямі на деформованому аркушу.

Такі деформації можуть виникати через фізичні спотворення аркушу або через вплив оптики.

Для подальшої роботи нам знадобиться бінаризація ректифікованого зображення. В найкращому випадку буде достатньо звичайного порогування рис.5.5.

Розглянемо випадок для вертикальних ліній. Через P позначимо множину всіх координат пікселів вертикальної прямої, що має фіксовану координату x', котра обирається з точки перетину вертикальної та горизонтальної прямої у точці $(x', \frac{h}{2})$.

$$P = \{(x', y) : y = \overline{0, h}\}$$

За допомогою бінаризованого зображення відокремимо координати пікселів, котрі є коректними та знаходяться на лінії від тих що потребують обробки. Посередині рис.5.5 синім кольором позначено коректні координати, зеленим позначено некоректні.

Позначимо $P_b \subset P$ множина координат всіх чорних пікселів на прямій бінаризованого зображення. $P_w = P \setminus P_b$ — множина координат білих пікселів лінії, котрі не належать лінії та потребують корекції.

Розіб'ємо множину P_w на множини P_i.

$$P_i = \{(x,y) : (x,y) \in P_w, y = \overline{(i-1) * [\frac{h}{k}], i * [\frac{h}{k}]}\}, i = \overline{1,k}$$

k це кількість частин лінії, кожну з яких необхідно окремо обробити. В цій роботі не розглядається пошук оптимального k, тому використаємо $k = 8$. Це розбиття зробленно для більш якісного пошуку коректних координат. h — висота зображення у пікселях.

Введемо функції d_i, $i = \overline{1,k}$.

$$d_i(x,x'') = \begin{cases} |x - x''| & , |x - x''| \leqslant \delta_i \\ \delta_i & , |x - x''| > \delta_i \\ \delta_i & , \nexists x'' \end{cases}$$

Де x це відома координата білого пікселю з множини P_i, котру необхідно виправити, x'' — координата найближчого чорного пікселю зліва або справа від x. Кожній множині P_i відповідає функція d_i. Пошук оптимальних δ_i в цій роботі не розглядається, тому при $k = 8$ використаємо

$\delta_1 = 25, \delta_2 = 20, \delta_3 = 15, \delta_4 = 10, \delta_5 = 10, \delta_6 = 15, \delta_7 = 20, \delta_8 = 25.$

Опишемо множини D_i, $i = \overline{1,k}$.

$$D_i = \{(l,r) : l = d_i(x,x_1), r = d_i(x,x_2), \forall (x,y) \in P_i\},$$

де x_1, x_2 — координати чорних пікселів зліва та справа від x.

Для того щоб обрати сторону, в котру треба зсувати координату введемо m'_i, m''_i, $i = \overline{1,k}$.

$$m'_i = \#\{l : l = min(l,r), \forall (l,r) \in D_i\}$$

$$m''_i = \#\{r : r = min(l,r), \forall (l,r) \in D_i\}$$

А також функції s_i, $i = \overline{1,k}$

$$s_i(l,r) = \begin{cases} -l & m'_i > m''_i \\ r & m'_i < m''_i \end{cases}$$

де l та r перше та друге значення пар $(l,r) \in D_i$.

$$P'_i = \{(x + s_i(l,r), y) : l = d_i(x,x_1), r = d_i(x,x_2), \forall (x,y) \in P_i\}$$

де x_1,x_2 — координати чорних пікселів зліва та справа від x.

Запишемо результуючу множину усіх коректних координат прямої.

$$P' = P_b \cup \bigcup_{i=1}^{k} P'_i$$

Для горизонтальних ліній алгоритм аналогічний.

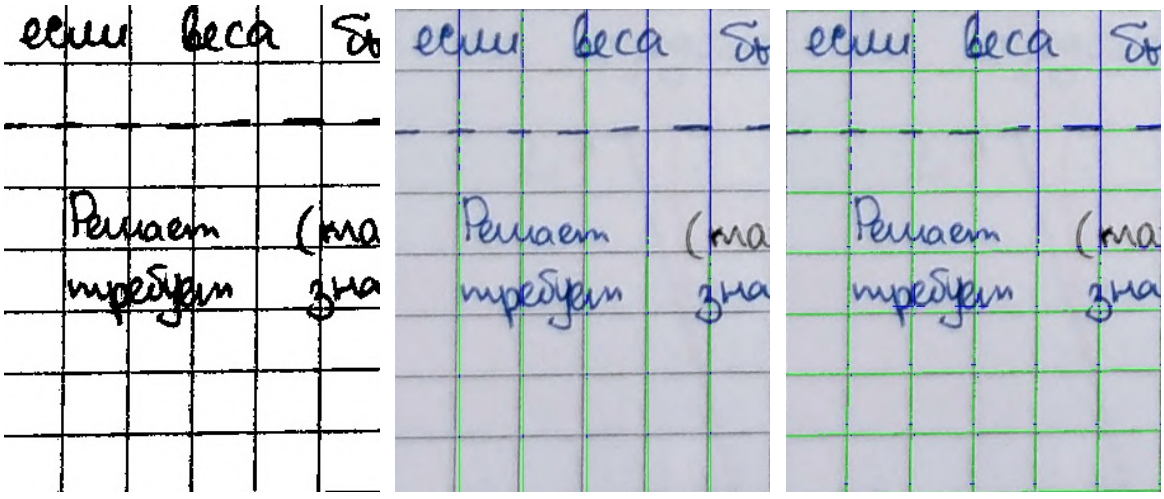

Рисунок 5.5 – Етапи пошуку координат розмітки.

5.4 Видалення розмітки

Під видаленням розмітки треба розуміти, що мова йдеться про її зафарбовування у певні кольори. Спираючись на ідеї роботи [11] запропонуємо алгоритм вибору цих кольорів.

Будемо використовувати множину коректних координат $P^{'}$ знайдену у попередньому розділі. Також необхідною є ширина лінії, однак пошук її оптимального значення в цій роботі не розлядається. Тому ширина лінії n буде приймати непарне значення від 3 до 6 пікселів.

$\forall(x,y) \in P^{'}$ оберемо координати пікселів, котрі будуть впливати на вибір кольору лінії.

$$a_1 = (x - \frac{n-1}{2} - 3, y - 1),\ a_2 = (x - \frac{n-1}{2} - 2, y - 1)$$

$$a_3 = (x - \frac{n-1}{2} - 1, y - 1),\ a_4 = (x - \frac{n-1}{2} - 4, y)$$

$$a_5 = (x - \frac{n-1}{2} - 3, y),\ a_6 = (x - \frac{n-1}{2} - 2, y)$$

$$a_7 = (x - \frac{n-1}{2} - 1, y),\ a_8 = (x - \frac{n-1}{2} - 3, y + 1)$$

$$a_9 = (x - \frac{n-1}{2} - 2, y + 1),\ a_{10} = (x - \frac{n-1}{2} - 1, y + 1)$$

$$a_{11} = (x + \frac{n-1}{2} + 1, y - 1),\ a_{12} = (x + \frac{n-1}{2} + 2, y - 1)$$

$$a_{13} = (x + \frac{n-1}{2} + 3, y - 1),\ a_{14} = (x + \frac{n-1}{2} + 1, y)$$

$$a_{15} = (x + \frac{n-1}{2} + 2, y),\ a_{16} = (x + \frac{n-1}{2} + 3, y)$$

$$a_{17} = (x + \frac{n-1}{2} + 4, y),\ a_{18} = (x + \frac{n-1}{2} + 1, y + 1)$$

$$a_{19} = (x + \frac{n-1}{2} + 2, y + 1),\ a_{20} = (x + \frac{n-1}{2} + 3, y + 1)$$

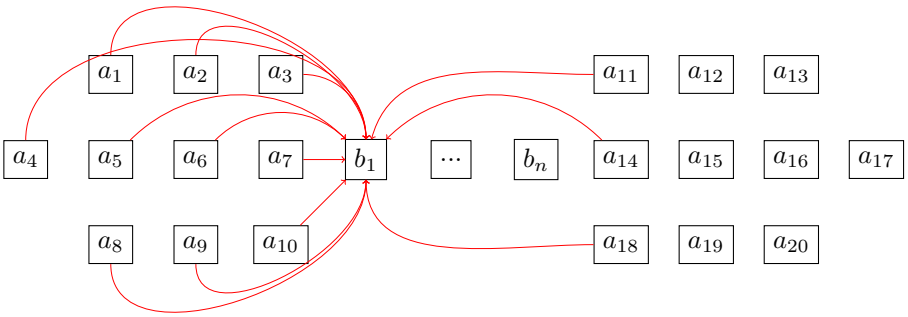

Рисунок 5.6 – Координати a_i, що формують b_1.

Координати пікселів, котрі необхідно зафарбувати.

$$b_1 = (x - \frac{n-1}{2}, y), ..., b_n = (x + \frac{n+1}{2}, y)$$

Введемо множини N_i, котрі будуть містити координати пікселів, що формують відповідні пікселі з координатами b_i.

$$N_1 = \{a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_{10},a_{11},a_{14},a_{18}\}$$

$$N_n = \{a_{11},a_{12},a_{13},a_{14},a_{15},a_{16},a_{17},a_{18},a_{19},a_{20}, a_3,a_7,a_{10}\}$$

$$N_j = \{a_2,a_3,a_6,a_7,a_9,a_{10},a_{11},a_{12},a_{14},a_{15},a_{18},a_{19}\}, j = \overline{2,n-1}$$

На рис.5.6 зображено вибір елементів множини N_1.

$N_i^{'} \subset N_i$ — множина координат пікселів кольору котрих більше на бінаризованому зображенні.

З $N_i^{'}$ обираємо ту координату пікселю яскравість якого має медіанне значення серед пікселів з координатами множини $N_i^{'}$. Позначимо її t_i.

Тепер запишемо у піксель з координатою b_i колір пікселю з координатою t_i.

Для горизонтальних ліній алгоритм аналогічний.

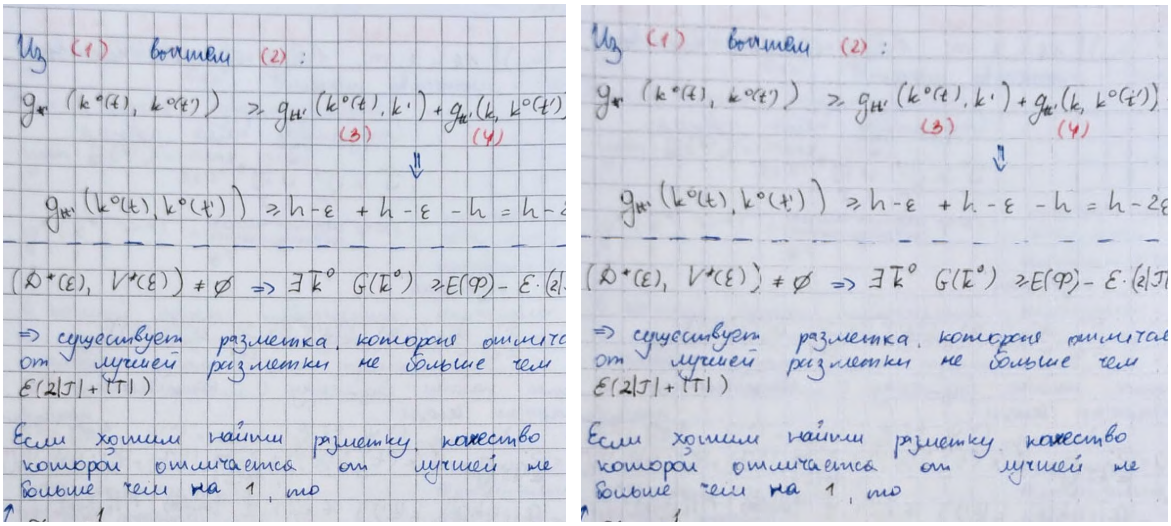

Рисунок 5.7 – Результат роботи алгоритмів пошуку та видалення розмітки.

Висновки до розділу 5

В цьому розділі запропоновано алгоритми пошуку відсутніх ліній, визначення формату, пошуку координат розмітки та її видалення. Зазначимо, що в алгоритмах застосовувались такі параметри як ширина лінії, обмеження зсуву, кількість елементів розбиття множини некоректних координат P_w. В подальшому необхідно використовувати оптимальні значення цих параметрів.

В роботі розглянуто обробку горизонтальних та вертикальних ліній, в подальшому необхідно зробити пошук та видалення ліній що проходять під кутом відмінним від прямого.

6 ЕКСПЕРИМЕНТИ ТА ОБМЕЖЕННЯ АЛГОРИТМУ.

В цьому розділі буде розглянуто результати застосування наведених в роботі алгоритмів. Там, де це можливо, зроблено порівняння з комерційними застосунками. Результат нашого алгоритму порівнюється з результатом застосунку Dropbox та Microsoft Lens.

6.1 Зображення з контрастним фоном

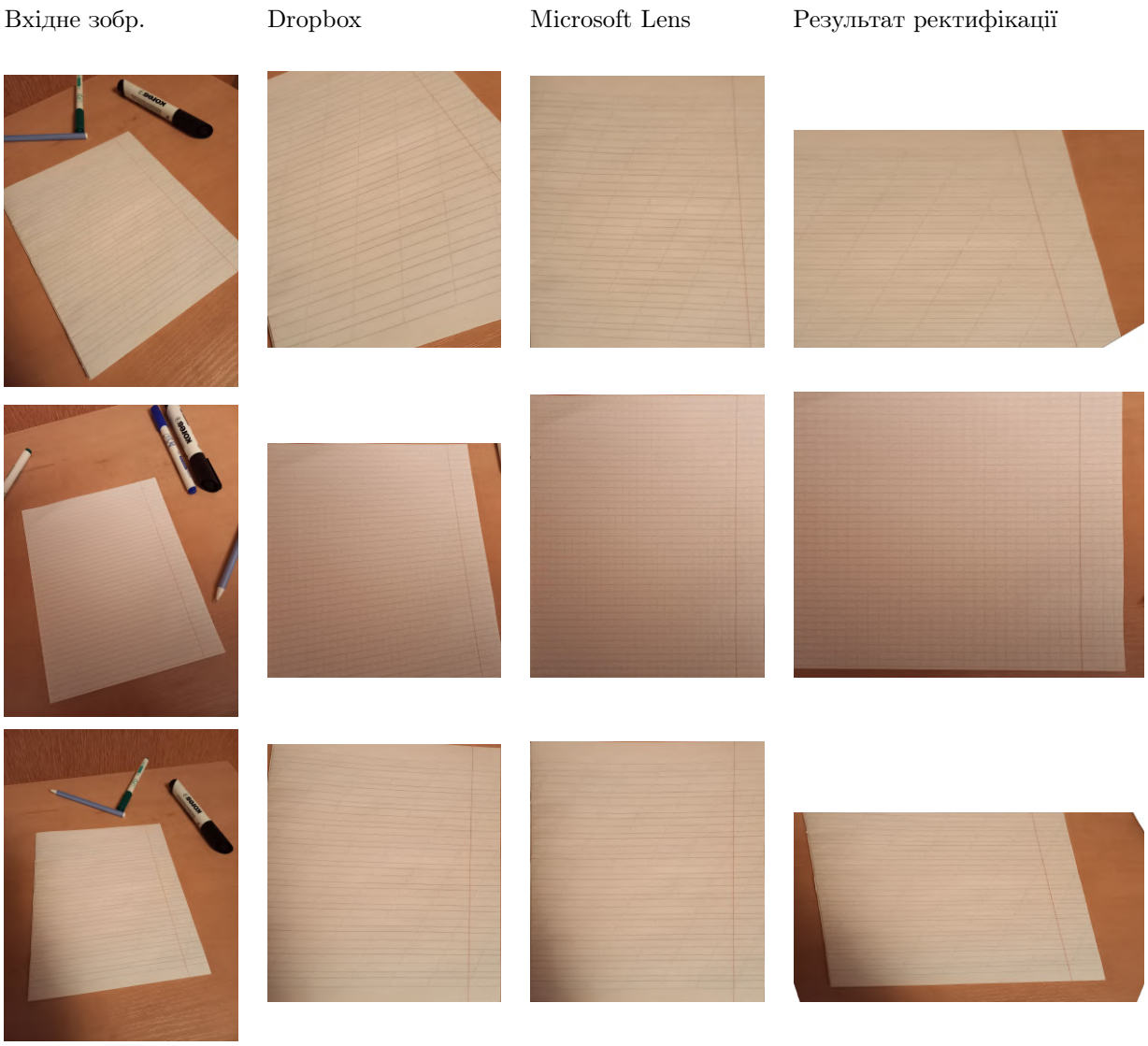

Рисунок 6.1 – Результат застосування алгоритмів до зображень з контрастним фоном.

Для зображень з контрастним фоном алгоритми Dropbox та Microsoft Lens дають приблизно однакові результати, однак є невеликі похибки. Наш алгоритм спрацьовує коректно на таких зображеннях.

6.2 Зображення із замаскованим фоном

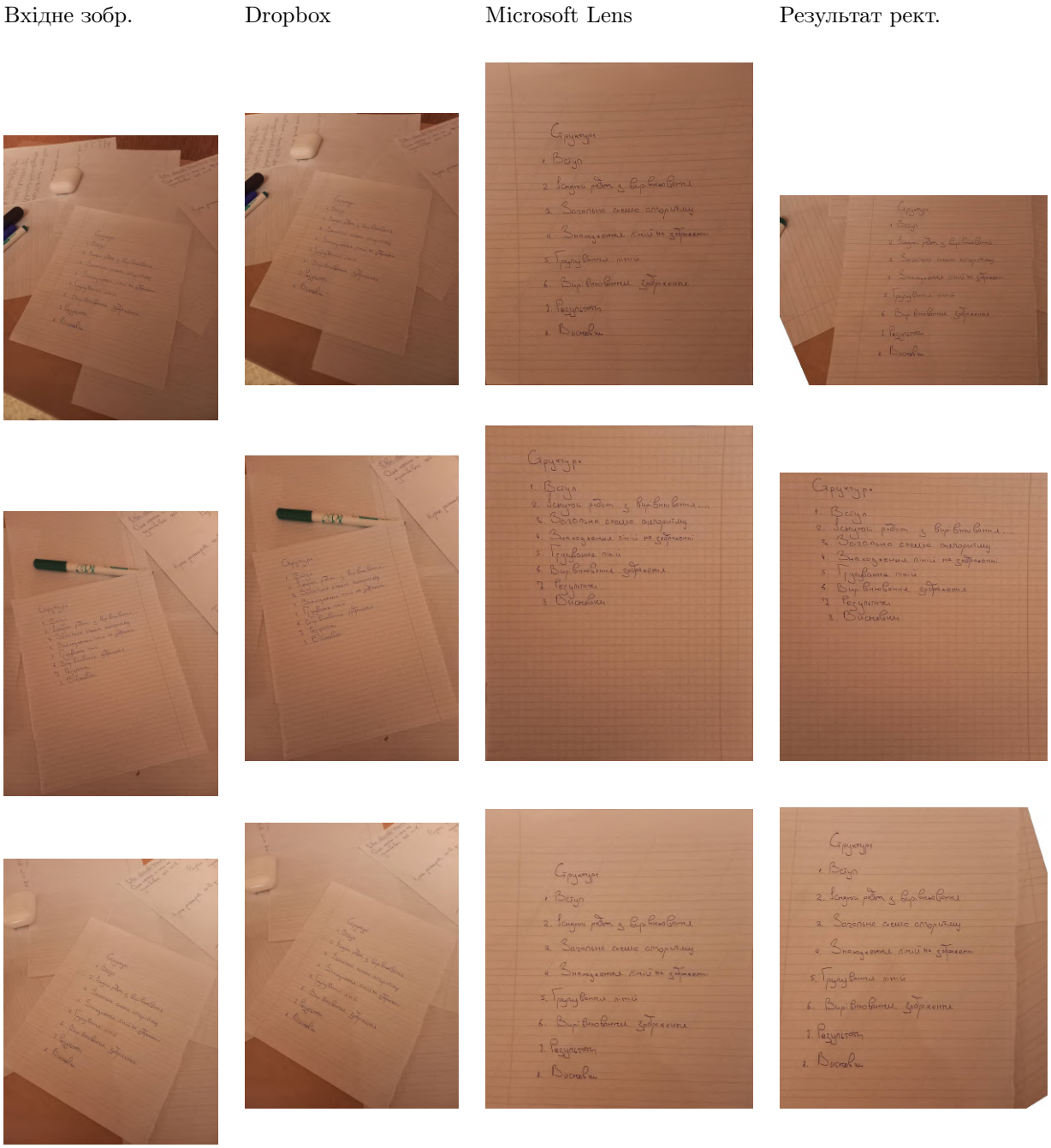

Рисунок 6.2 – Результат застосування алгоритмів до зображень із замаскованим фоном.

Зображення із замаскованим фоном якісно оброблюються нашим алгоритмом та Microsoft Lens. Алгоритм застосунку Dropbox не знайшов краї на зображенні, тому результат не відрізняється від входу.

6.3 Зображення з частковою наявністю документа

Рисунок 6.3 – Результат застосування алгоритмів до зображень з частковою наявністю документа.

Серед зображень із не повним аркушем наш алгоритм надає найкарщий результат. Це пояснюється тим, що ми застосовуємо розмітку

аркушу. Алгоритм застосунку Dropbox не обробив фото, а Microsoft Lens обрізає текст.

6.4 Зображення з неконтрастним фоном

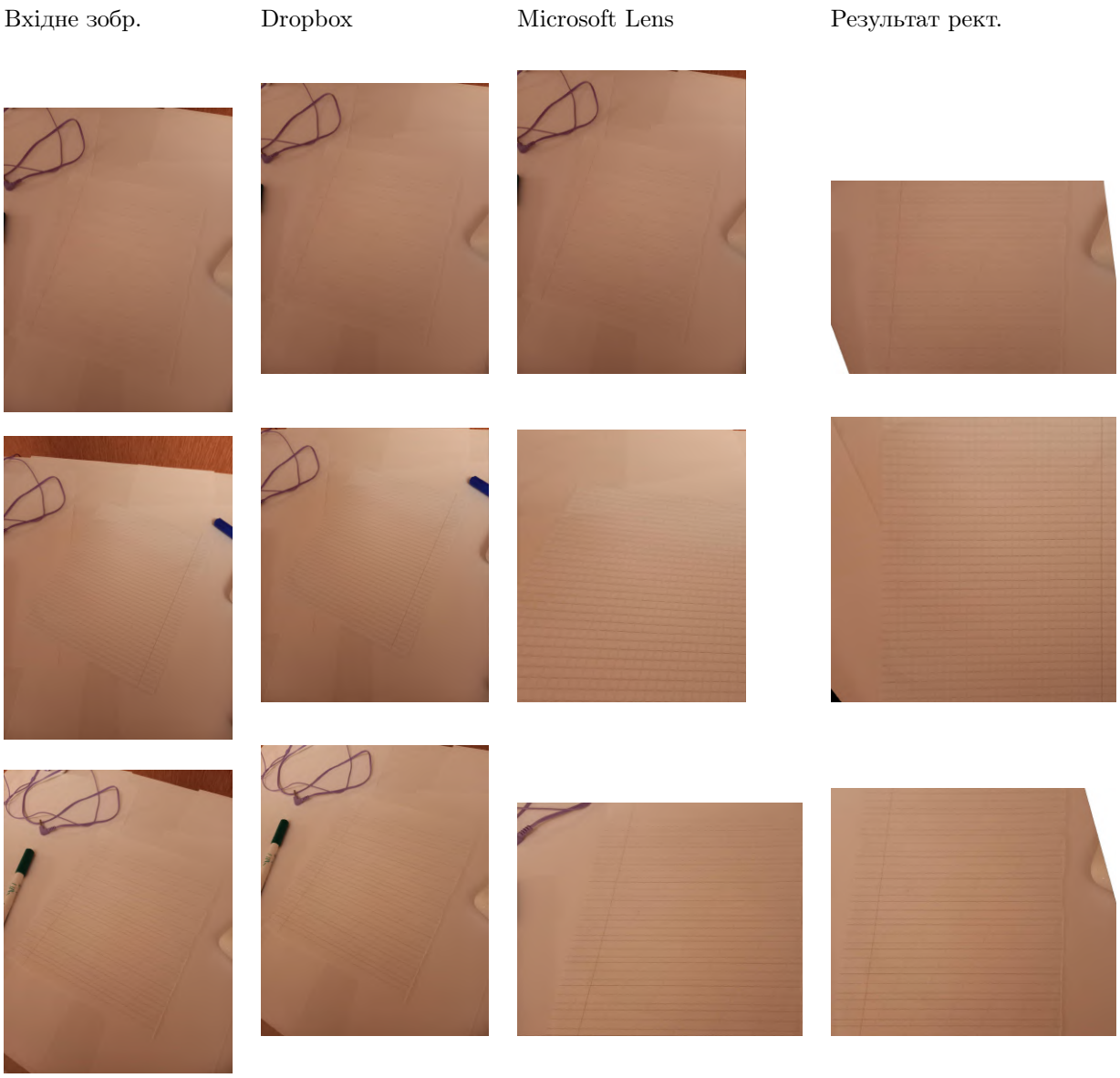

Рисунок 6.4 – Результат застосування алгоритмів до зображень з неконтрастним фоном.

На відміну від алгоритмів Dropbox та Microsoft Lens наш алгоритм якісно спрацював на зображеннях із неконтрастним фоном. Це пояснюється коректною бінаризацією та пошуком ліній.

6.5 Зображення з накладеними один на одного аркушами

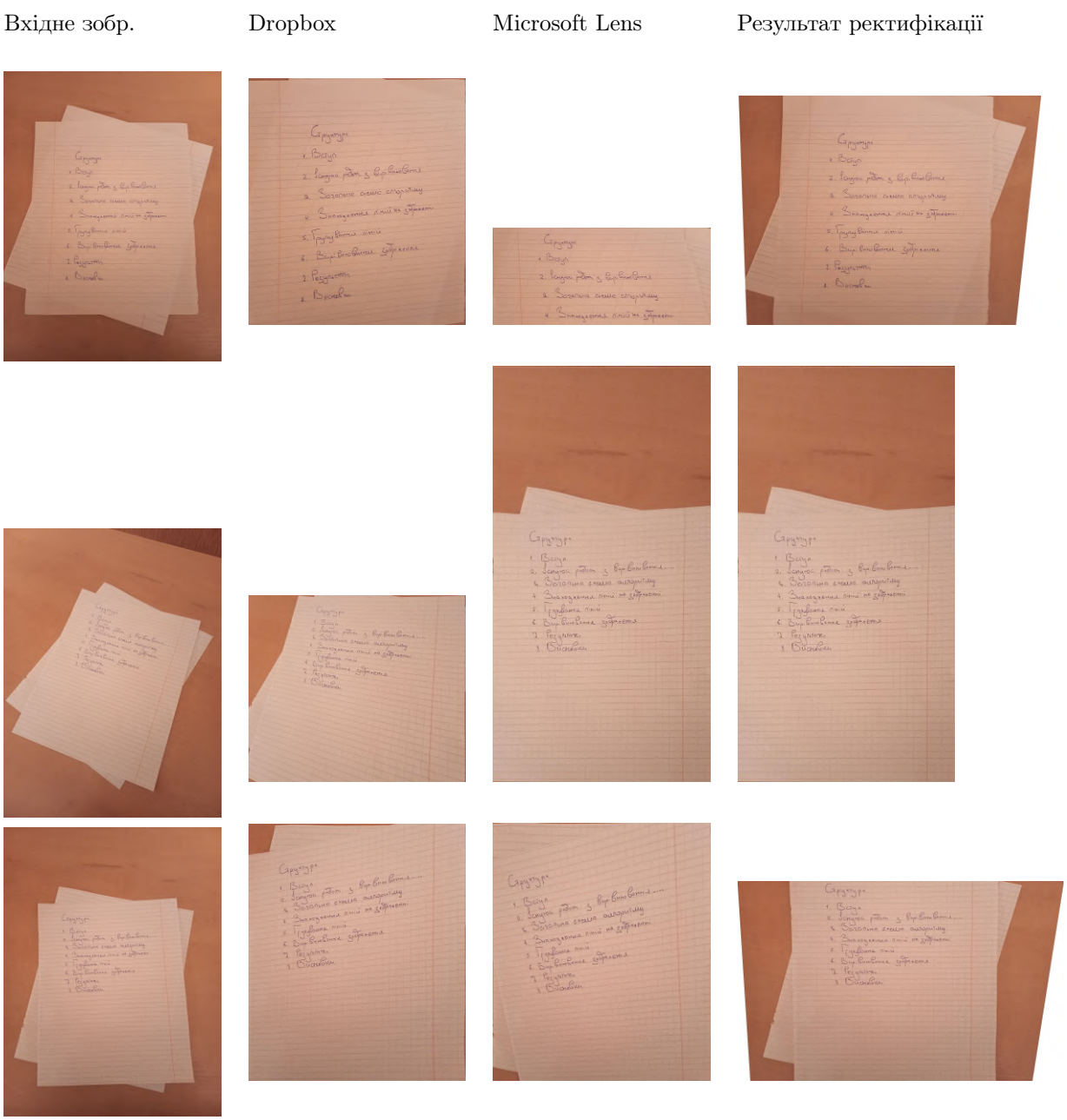

Рисунок 6.5 – Результат застосування алгоритмів до зображень з накладеними один на одного аркушами.

Застосунки, що шукають краї на зображенні можуть знаходити краї різних аркушів, тому їх результат може бути некоректним. Наш алгоритм

шукає зникомі точки ліній і наявність такого дефекту в нього відсутня.

6.6 Вирівнювання аркушу в лінію

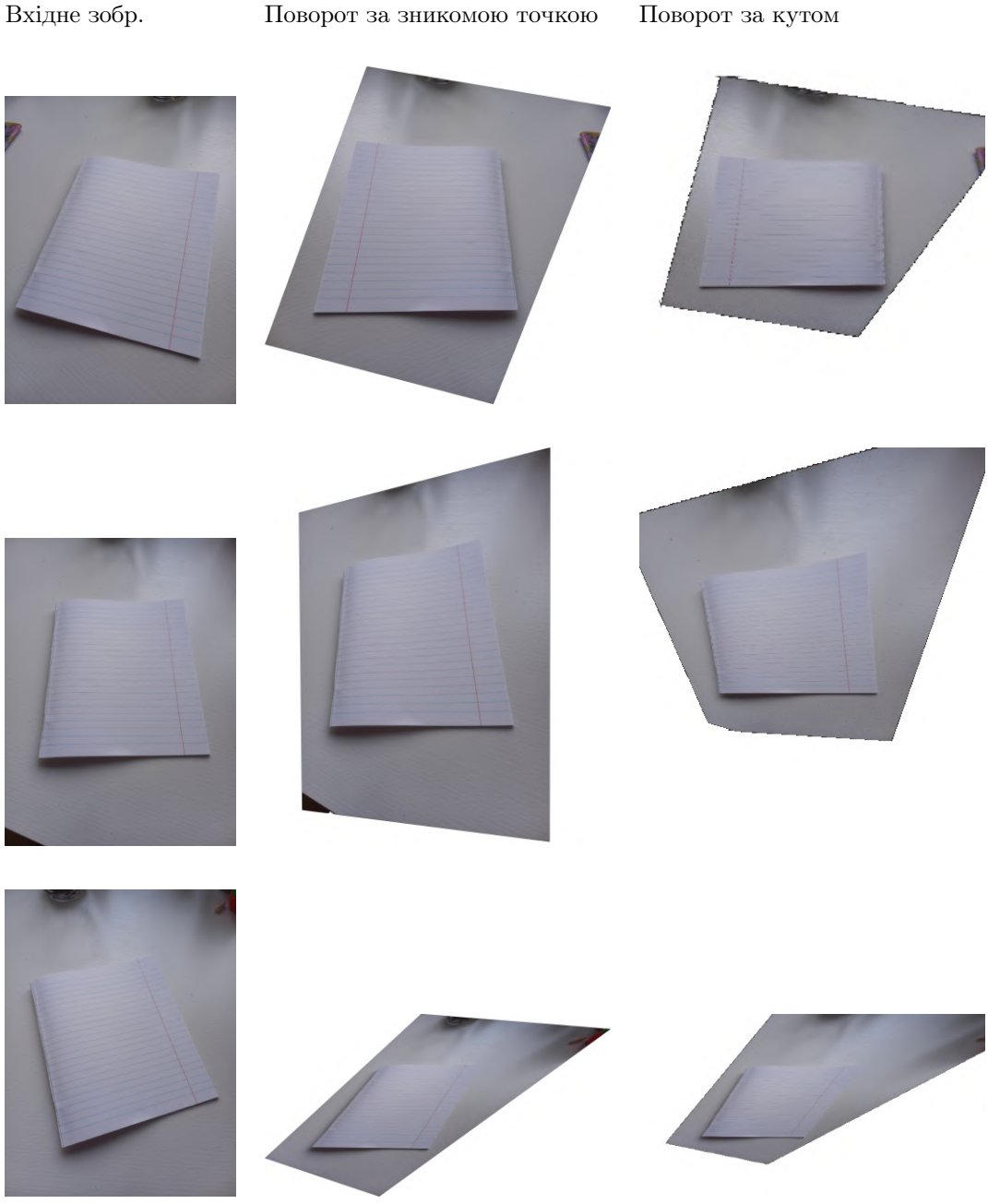

Рисунок 6.6 – Результат застосування алгоритмів до зображень аркушів в лінію.

Вирівнювання аркушів в лінію відбувається за два кроки: горизонталізація та поворот. Також від користувача може потребуватись додаткова інформація про скос та масштабування. Цей недолік зображено на останньому рядку.

Висновки до розділу 6

В цьому розділі порівняно результати роботи нашого алгоритму та алгоритмів Dropbox і Microsoft Lens. Переважно наш алгоритм спрацьовує добре в тих умовах, в котрих написано вище. Варто зазначити, що використання розмітки для ректифікації має такі недоліки як вигнуті аркуші та наявність ліній користувача документу, що може бути знайдена як розмітка аркушу. Також алгоритм не розрахований на пошук орієнтації аркушу.

До вирівнювання аркушів в лінію варто додати, що необхідною є розробка алгоритму вірівнювання за переносом точок між площинами та побудови оптимального вибору ліній з котрих вираховуються $\vec{\tau}$ та S.

ВИСНОВКИ

В цій роботі досліджено та запропоновано алгоритми ректифікації розлінованих аркушів. Для цього в оглядовому розділі було розглянуто актуальні роботи та застосунки, котрі вирішують проблему. Було виявлено, що сучасним рішенням необхідні певні умови для ректифікації документів і вони не придатні для масової обробки документів. Окрім ректифікації в роботі запропоновано алгоритми обробки розмітки, тому роботи з цієї теми також присутні в оглядовому розділі.

В окремий розділ винесено перелік алгоритмів комп'ютерного зору та необхідних елементів проективної геометрії та лінійної алгебри.

В роботі запропоновано алгоритми ректифікації аркушів за двома та однією зникомою точкою. Зникомі точки знаходяться модифікованим алгоритмом RANSAC, котрий на кожній ітерації перевіряє паралельність ліній. З двох зникомих точок будується матриця повороту і до зображення застосовується перспективне перетвоерння. З однієї зникомої точки будується перетворення, що зробить лінії горизонтальними, після цього ми оцінюємо кут між площиною зображення та площиною аркушу і повертаємо початкове зображення на цей кут. Окрім цього сформульовано задачу повороту через співставлення точок площин аркушу та сенсору.

В наступному розділі запропоновано алгоритми обробки розмітки. Для цього на ректифікованому зображенні знаходиться кожна лінія та кожна її координата, після цього проходженням "вікна"з сусідніх пікселів лінія перефарбовується. Цей етап створений для встановлення власної розмітки, створення котрої буде розглянуто в подальших роботах.

Робота потребує багато автоматизацій та пошуку отпимальних параметрів. Необхідно автоматизувати пошук бінаризації зображень, пошук порогових значень RANSAC, оцінку положення векторів матриці повороту, фільтрацію ліній перед їх застосуванням, пошук отпимальних

значень в алгоритмах пошуку та видалення розмітки і т.д.

Для ректифікації аркушів з розміткою в лінію необхідно побудувати пошук трьох найкращих точок для оціки кута між площиною сенсору та площиною аркушу. Окрім цього треба вирішити задачу повороту через співставлення точок площин.

Варто зазначити, що першочерговою ціллю цієї роботи є автоматизована обробка великої кількості зображень аркушів. На даному етапі ми потребуємо від користувача інформацію про внутрішні параметри його камери та надаємо йому власноруч виправляти перекіс.

ПЕРЕЛІК ПОСИЛАНЬ

1. Оценивание положения точек схода на изображениях городских кварталов / В.Ю. Сдобников, Б.Д. Савчинский // Управляющие системы и машины. — 2009. — № 6. — С. 12-18, 29. — Бібліогр.: 7 назв. — рос.

2. Schaffalitzky F., Zisserman A. Planar grouping for auto- matic detection of vanishing lines and points, IVC(18), June 2000. – N 9. – P. 647–658.

3. K. Chaudhury, S. DiVerdi and S. Ioffe, "Auto-rectification of user photos,"2014 IEEE International Conference on Image Processing (ICIP), Paris, France, 2014, pp. 3479-3483, doi: 10.1109/ICIP.2014.7025706.

4. Harris C. and Stephens M., “A combined corner and edge detector,” in 4th Alvey Vision Conference, 1988, pp. 147–151.

5. K. Javed and F. Shafait, "Real-Time Document Localization in Natural Images by Recursive Application of a CNN,"2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 2017, pp. 105-110, doi: 10.1109/ICDAR.2017.26.

6. https://github.com/khurramjaved96/Recursive-CNNs

7. A. Krizhevsky, I. Sutskever, et al. ”ImageNet classification with deep convolutional neural networks.” Advances Neural Information Processing Systems 25 : 1097 - 1105, 2012

8. Paul Clark, Majid Mirmehdi, Rectifying perspective views of text in 3D scenes using vanishing points, Pattern Recognition, Volume 36, Issue 11, 2003, Pages 2673-2686, ISSN 0031-3203, https://doi.org/10.1016/S0031-3203(03)00132-8.

9. Ying Xiong, Fast and Accurate Document Detection for Scanning – Dropbox.tech Publication, 2016.

10. Бондар, М. О. Вирівнювання зображень розлінованих аркушів : магістерська дис. : 113 Прикладна математика / Бондар Марія Олександрівна. – Київ, 2022. – 65 с.

11. Aksak, I. et al. (1997). Extraction of filled-in data from colour forms.

In: Sommer, G., Daniilidis, K., Pauli, J. (eds) Computer Analysis of Images and Patterns. CAIP 1997. Lecture Notes in Computer Science, vol 1296. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-63460-$6_1$05

12. Use of the Hough transformation to detect lines and curves in pictures, Duda, Richard O and Hart, Peter E, Communications of the ACM, pages 11–15, 1972, ACM New York, NY, USA

13. Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, (6), 679–698.

14. N. Otsu, "A Threshold Selection Method from Gray-Level Histograms,"in IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, Jan. 1979, doi: 10.1109/TSMC.1979.4310076.

15. Gonzalez, Rafael C. Woods, Richard E. (2002). Thresholding. In Digital Image Processing, pp. 595–611. Pearson Education.

16. Image Analysis and Mathematical Morphology, Serra, J., isbn 9780126372410, lccn 81066397, Image Analysis and Mathematical Morphology, https://books.google.co.uk/books?id=BpdTAAAAYAAJ, 1982, Academic Press

17. Huang, Thomas S.; Yang, George J.; Tang, Gregory Y. (February 1979). "A fast two-dimensional median filtering algorithm"(PDF). IEEE Transactions on Acoustics, Speech, and Signal Processing. 27 (1): 13–18. doi:10.1109/TASSP.1979.1163188

18. Cantzler, H. Random sample consensus. Institute for Perception, Action and Behaviour, Division of Informatics, University of Edinburgh 1981 doi 10.1.1.106.3035

19. Hartley, R. I. and Zisserman, A., Multiple View Geometry in Computer Vision, Second, 2004, Cambridge University Press, ISBN: 0521540518

20. http://robotics.stanford.edu/ birch/projective/node4.html

21. https://commons.wikimedia.org/wiki/File:Perspective1.jpg

22. https://en.wikipedia.org/wiki/Random$_s ample_c onsensus$: : $text$ = $Random$

23. https://docs.opencv.org/3.4/d9/db0/tutorial$_h ough_l ines.html$
24. https://scikit-image.org/
25. https://opencv.org/