Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

Застосування перетворення Хафа для пошуку об’єктів на зображеннях

Текстова частина до курсової роботи
за спеціальністю «Програмна інженерія»

Керівник курсової роботи:

кандидат технічних наук, старший викладач

Бучко Олена Андріївна

Виконав студент:

Калінічев Гліб Олександрович

Київ - 2022

ЗМІСТ

РОЗДІЛ 4

СТВОРЕННЯ ТА АНАЛІЗ ВЛАСНИХ РЕАЛІЗАЦІЙ АЛГОРИТМІВ ПЕРЕТВОРЕННЯ ХАФА ДЛЯ ПОШУКУ ПРЯМИХ ТА КІЛ НА ЗОБРАЖЕННІ

ВСТУП

В сучасному світі спостерігається тенденція до автоматизації все більшої кількості процесів в різноманітних сферах життєдіяльності людини: виробництві, транспортній галузі, медицині, науці та ін. Задача автоматизації зазвичай передбачає створення певної комп'ютеризованої системи, яка б її вирішувала. Подібні системи повинні частково або повністю виконувати обов'язки кваліфікованого працівника. З огляду на це, вони мають задовольняти низку вимог, найважливішою з яких є можливість збирати та аналізувати візуальну інформацію як мінімум на рівні з досвідченим робітником. Задача аналізу зображення (візуальної інформації) часто декомпонується на кілька підзадач, що вирішуються поетапно. В багатьох випадках початковий етап аналізу передбачає виявлення на зображенні об'єктів, геометрична форма яких може бути описана певним аналітичним виразом. Для ефективного вирішення цієї задачі найчастіше використовується перетворення Хафа.

Метою цієї роботи є визначення можливих варіантів використання алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні в прикладних цифрових системах та аналіз реалізацій цих алгоритмів в сучасних бібліотеках комп'ютерного зору та обробки зображень на прикладі бібліотеки OpenCV.

Робота складається з чотирьох розділів.

В першому розділі надаються теоретичні описи алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні.

В другому розділі продемонстровані приклади конкретних цифрових систем, що використовують алгоритми перетворення Хафа для пошуку прямих або кіл на зображенні, та описано способи використання перетворень Хафа в цих системах.

Третій розділ присвячено аналізу реалізацій алгоритмів перетворення Хафа для пошуку прямих та кіл в бібліотеці OpenCV.

В четвертому розділі міститься опис власних реалізацій перетворень Хафа для пошуку прямих та кіл та порівняння цих реалізацій з аналогічними в бібліотеці OpenCV.

Постановка задачі:

1. Дослідити варіанти використання алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні в прикладних цифрових системах.
2. Провести аналіз реалізацій алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні в бібліотеці OpenCV.
3. Створити власні реалізації алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні.
4. Порівняти власні реалізації алгоритмів перетворення Хафа для пошуку прямих та кіл з відповідними реалізаціями в бібліотеці OpenCV.

РОЗДІЛ 1
АЛГОРИТМИ ПЕРЕТВОРЕННЯ ХАФА ДЛЯ ПОШУКУ ПРЯМИХ ТА КІЛ НА ЗОБРАЖЕННІ

1.1. Опис алгоритму та теоретичного підґрунтя перетворення Хафа для пошуку прямих на зображенні

Кожна пряма в декартовій системі координат може задаватися рівнянням:

$$y = kx + b,$$

де $k = \operatorname{tg}\alpha$,

α – кут між прямою та додатнім напрямком осі абсцис,

b – ордината точки перетину прямої з віссю ординат.

Вводиться поняття параметричного простору – система координат, де на горизонтальній осі знаходяться значення параметру b, а на вертикальній – значення кутового коефіцієнту k. Таким чином кожна точка в параметричному просторі з координатами (b_i, k_j) відповідає деякій прямій у декартовій системі координат, що задається рівнянням: $y = k_j x + b_i$. Однак даний підхід до визначення параметричного простору не дозволяє задавати в ньому вертикальні прямі, адже в такому разі кутовий коефіцієнт k прямої є невизначеним.

Більш універсальним підходом, що дозволяє задавати прямі будь-які прямі в параметричному просторі є використання нормального рівняння прямої:

$$\rho = x\cos\theta + y\sin\theta, \quad (1)$$

де ρ – довжина перпендикуляра, опущеного на пряму з початку координат $O(0,0)$,

θ – кут між цим перпендикуляром та додатнім напрямом осі абсцис.

Параметричний простір задається наступним чином: горизонтальна вісь використовується для значень кута θ, вертикальна – для значень довжини перпендикуляра ρ. Кожна точка в даному параметричному просторі відповідає певній прямій в декартовій системі координат за умови, що $\theta \in [0,2\pi]$ та $\rho \geq 0$. Площина (θ, ρ) називається «простір Хафа» (або параметричний простір). Через кожну точку звичайного координатного простору проходить безліч прямих.

Тому кожній точці з координатного простору (x_0, y_0) однозначно відповідає синусоїдна крива в просторі Хафа, що задається рівнянням:

$$\rho(\theta) = x_0 \cos\theta + y_0 \sin\theta.$$

Пряма, яка проходить через дві точки A(x_A, y_A), B(x_B, y_B) координатного простору є точкою P(θ_P, ρ_P) перетину двох синусоїд, що відповідають точкам A та B в просторі Хафа. Тобто координати точки P (θ_P, ρ_P) є розв'язком системи рівнянь:

$$\begin{cases} \rho_P = x_A \cos\theta_P + y_A \sin\theta_P \\ \rho_P = x_B \cos\theta_P + y_B \sin\theta_P \end{cases}.$$

Таким чином, в просторі координат визначається пряма, яка проходить через точки A, B та має рівняння виду: $y = \left(-\frac{\cos\theta_P}{\sin\theta_P}\right)x + \left(\frac{\rho_P}{\sin\theta_P}\right)$.

Перед застосуванням перетворення Хафа необхідно провести попередню обробку зображення, а саме виявити границі на зображенні (наприклад, застосувавши детектор границь Кенні). Алгоритм перетворення Хафа для пошуку прямих використовує двовимірний масив, що називається *акумулятором*. Фактично цей масив є матрицею розмірності M×N, де індекси рядків є квантованими значеннями довжини перпендикуляра ρ, а індекси стовпців є квантованими значеннями кута θ. У комірці матриці на перетині i-ого рядка та j-ого стовпця $(0 \le i \le M, 0 \le j \le N)$ зберігається значення лічильника, що вказує на кількість граничних пікселів зображення, що лежать на прямій, яка задається рівнянням:

$$y = \left(-\frac{\cos j}{\sin j}\right)x + \left(\frac{i}{\sin j}\right).$$

Таким чином, для кожного граничного пікселя зображення обраховуються всі можливі пари параметрів (ρ, θ) за рівнянням (1) та збільшується значення лічильника у відповідній комірці акумуляторного масиву. Після проходження всіх граничних пікселів зображення знаходяться локальні максимуми в акумуляторному масиві за допомогою порогової фільтрації. Отриманий набір пар параметрів (ρ, θ) задає множину шуканих прямих. [1]

1.2. Опис алгоритму та теоретичного підґрунтя перетворення Хафа для пошуку кіл на зображенні.

Коло в декартовій системі координат задається рівнянням виду:

$$(x - a)^2 + (y - b)^2 = R^2, \quad (2)$$

де R – радіус кола,

(a, b) – координати центру кола.

Простір Хафа в цьому випадку є тривимірним, де вимірами є параметри рівняння кола (a, b, R).

Кожній точці (x_0, y_0) з координатного простору відповідає у просторі Хафа перевернутий прямокутний конус з вершиною у точці $(x_0, y_0, 0)$. Значення параметрів a, b, R, що задовольняють рівняння (2) знаходяться на поверхні цього конуса. Координати точок в просторі Хафа, що лежать на поверхні певної кількості таких конусів є параметрами шуканих кіл в початковому координатному просторі. Кількість конусів, поверхням яких повинна належати точка в просторі Хафа для того, щоб її координати вважалися релевантними параметрами кола на координатній площині, повинна бути більша за значення певного порогового параметру.

Як і для перетворення Хафа для пошуку прямих, перед застосуванням перетворення Хафа для пошуку кіл також необхідно застосувати детектор Кенні для виявлення границь об’єктів на зображенні. Акумуляторний масив у випадку перетворення Хафа для пошуку кіл має бути тривимірним з розмірністю M×N×K, де перший вимір використовується для представлення квантованих значень довжини радіуса, два інші виміри використовується для представлення значень параметрів a та b. У комірці акумуляторного масиву на позиції (i, j, k) $(0 \leq i \leq M, 0 \leq j \leq N, 0 \leq k \leq K)$ зберігається лічильник, що вказує на кількість пікселів, що належать колу з радіусом довжиною в i пікселів та центральним пікселем на позиції (j, k).

Фіксується певне квантоване значення r довжини радіуса з діапазону значень довжини радіуса. Для кожного граничного пікселя зображення на позиції (i, j)

знаходяться комірки акумуляторного масиву, координати (r, i_0, j_0) яких задовольняють рівність:

$$r^2 = (i_0 - i)^2 + (j_0 - j)^2.$$

При цьому лічильник в кожній з таких комірок збільшується на одиницю. Після проходження всіх значень довжини радіуса з діапазону значень довжини радіуса знаходяться локальні максимуми в акумуляторному масиві аналогічно до алгоритму перетворення Хафа для пошуку прямих. Отриманий набір трійок параметрів (R, a, b) задає множину шуканих кіл на зображенні. [2]

РОЗДІЛ 2
ВИКОРИСТАННЯ АЛГОРИТМІВ ПЕРЕТВОРЕННЯ ХАФА ДЛЯ ПОШУКУ ПРЯМИХ ТА КІЛ НА ЗОБРАЖЕННІ В ПРИКЛАДНИХ ЦИФРОВИХ СИСТЕМАХ

2.1. Використання алгоритму перетворення Хафа для пошуку кіл на зображенні в прикладних системах

Перетворення Хафа для пошуку кіл на зображенні може бути використане в процесі біометричної ідентифікації людини за райдужною оболонкою ока. [3] За візуальною ознакою, райдужна оболонка ока – це кольорова область навколо зіниці, яка є унікальною для кожної людини. Ця властивість райдужної оболонки дозволяє використовувати її зображення на рівні із зображеннями відбитків пальців в якості вхідних даних для систем автоматичної ідентифікації людини. З геометричної точки зору райдужна оболонка представляє собою область, обмежену двома концентричними колами: зовнішнім, що формує границю між склерою та райдужною оболонкою, та внутрішнім, що відділяє райдужну оболонку від зіниці. Задача локалізації райдужної оболонки на зображенні зводиться до знаходження параметрів (радіус та координати центру) зовнішнього та внутрішнього кіл. Для вирішення цієї задачі використовується алгоритм перетворення Хафа для знаходження кіл. Оскільки зображення, які опрацьовує система, здебільшого є однотипними (мають однаковий рівень освітлення, розмір, рівень шуму), можливо наперед визначити найбільш оптимальні параметри для детектора Кенні та алгоритму перетворення Хафа. Наприклад, знаючи середній розмір ока дорослої людини, можливо наперед визначити вузький діапазон значень довжини радіуса шуканих кіл на зображенні, що дозволяє значно зменшити кількість обчислень та мінімізувати час опрацювання зображення. Це робить перетворення Хафа для знаходження кіл найбільш ефективним інструментом для рішення задачі локалізації райдужної оболонки на зображенні.

Ще один приклад застосування перетворення Хафа для пошуку кіл має місце у системах архівування та розсилання медичних зображень (PACS) [4]. Одна з основних функцій таких систем – цифрова обробка та зберігання

радіографічних знімків. Велика частина радіографічних знімків створюється з використанням конусоподібної трубки для зменшення площі опромінення. Цифрові зображення таких знімків складаються з двох частин: темного фону та круглої області, що містить корисну інформацію (інформативна частина знімку) (рисунок 2.1).

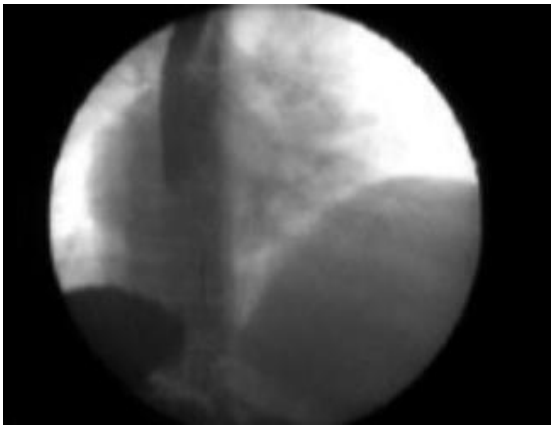

Рисунок 2.1 - Приклад радіографічного знімку

Залежно від типу радіографічного апарату розмір та розташування інформативної частини знімку може відрізнятись. Локалізація та розпізнавання інформативної частини необхідні для зменшення обсягу пам’яті, що використовується при зберіганні зображення знімку, та покращення якості самої інформативної частини для подальшого аналізу. Для цієї задачі використовується перетворення Хафа для пошуку кіл. Перед застосуванням перетворення Хафа виконується сегментація зображення за допомогою алгоритму росту регіону після чого застосовується оператор Собеля для отримання границі інформативної частини знімку. Після застосування алгоритму росту регіону визначається приблизне значення радіусу інформативної частини зображення за формулою площі кругу:

$$S = \pi R^2,$$

де S – кількість пікселів, що належать регіону інформативної частини.

Це дозволяє застосувати алгоритм перетворення Хафа для гранично вузького діапазону значень довжини радіусу кола, що значно зменшує кількість необхідних обчислень та мінімізує час обробки зображення.

Інше застосування перетворення Хафа для пошуку кіл в медицині – системи, що дозволяють частково автоматизувати процес клінічного аналізу крові людини. [5] Одним з видів аналізу крові є підрахунок лейкоцитів на мікроскопічному зображенні зразка крові. Ручний підрахунок лейкоцитів - доволі часозатратний процес, що великою мірою залежить від людського фактору. Натомість існують автоматизовані рішення, що дозволяють виконувати підрахунок лейкоцитів на цифровому зображенні зразка крові за мінімальний проміжок часу та з великою точністю. Мікроскопічне зображення зразка крові складається з трьох типів клітин: лейкоцитів, еритроцитів та тромбоцитів. Більшість з цих клітин візуально мають округлу форму. Саме тому існує потреба в попередній ідентифікації лейкоцитів на зображенні. Ця задача вирішується за допомогою алгоритмів машинного навчання. Лейкоцити можуть утворювати згустки клітин. Через це на етапі ідентифікації можливо підрахувати лише окремі клітини лейкоцитів, що не є частиною згустків. Кожен згусток лейкоцитів на зображенні сегментується та аналізується окремо. Для локалізації кожного лейкоциту в скупченні клітин застосовується алгоритм перетворення Хафа для пошуку кіл. Як і в попередніх прикладах застосування перетворення Хафа, велика часова складність алгоритму нівелюється наперед визначеним вузьким діапазоном значень довжини радіуса шуканих кіл, адже середній розмір лейкоциту є відомим.

2.2. Використання алгоритму перетворення Хафа для пошуку прямих на зображенні в прикладних системах.

Перетворення Хафа для пошуку прямих на зображенні може бути використане в системах неруйнівного контролю (Non-Destructive Testing). [6] Однією з функцій подібних систем є автоматизована перевірка наявності дефектів лінійної форми на радіографічних зображеннях зварювальних швів. Зображення такого типу складаються з шву, що представлений світлою областю продовгуватої форми, та темного фону (рисунок 2.2).

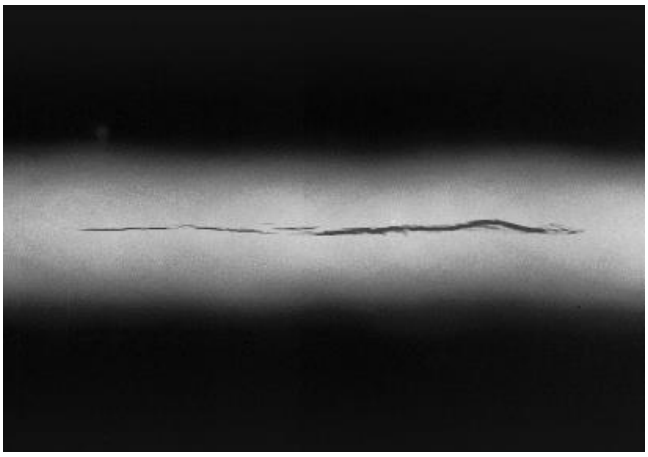

Рис 2.2 – Радіографічне зображення зварювального шву з наявним дефектом.

На зображенні дефект має вигляд паралельної до осі шву лінії, що може містити розриви (рисунок 2.2). Результат застосування перетворення Хафа на зображенні такого типу містить мінімум 2 паралельні прямі, що обмежують область шву. Ці дві лінії мають відповідно мінімальне та максимальне значення параметру ρ, що дозволяє виокремити їх з-поміж інших та не враховувати їх під час подальшого аналізу. Всі інші лінії є виявленими дефектами. Таким чином, перетворення Хафа в цьому випадку дозволяє не лише локалізувати, а й ідентифікувати прямі на зображенні.

Ще один приклад використання перетворення Хафа для пошуку прямих має місце в удосконалених системах допомоги водію (ADAS). [7] Підсистемою ADAS є адаптивний круїз-контроль (ACC), однією з ключових задач якого є розпізнавання смуги руху або виявлення меж проїжджої частини на основі зображень дороги з зовнішніх камер, встановлених в передній частині автомобіля. Перетворення Хафа в цьому випадку використовується для локалізації прямих, що умовно відмежовують проїжджу частину дороги від узбіччя. Крім того, застосування перетворення Хафа дозволяє швидко визначити розташування лінії горизонту на зображенні за точкою перетину прямих, що позначають границі проїжджої частини. Для зменшення кількості обчислень та однозначної ідентифікації граничних прямих на етапі застосування перетворення Хафа початкове зображення розділяється навпіл на праву та ліву частини, які аналізуються окремо, причому діапазон значень кута θ

встановлюється від -45° до 45°. Таким чином, права та ліва границі дороги матимуть найбільшу кількість «голосів» в акумуляторному масиві після обробки відповідно правої та лівої частин зображення, що дозволяє уникнути застосування подальшої обробки результатів перетворення Хафа для відсіювання не граничних ліній.

Інший варіант застосування перетворення Хафа для пошуку прямих реалізований в системах автоматичного сканування астрономічних зображень нічного неба. [8] Задача таких систем – локалізувати та ідентифікувати астрономічні об’єкти на знімку. У той же час, на зображеннях такого типу можуть міститися об’єкти лінійної форми не астрономічної природи, такі як треки від штучних супутників (рисунок 2.3), сліди від літаків, подряпини або волокна пилу на лінзах телескопа, тощо. (далі - дефекти зображення)

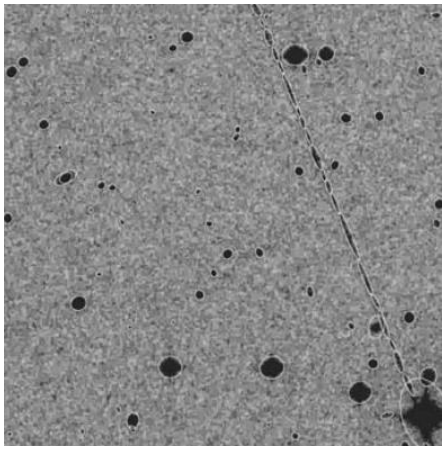

Рисунок 2.3 - Трек штучного супутника на зображенні нічного неба

Подібні об’єкти не становлять інтерес для подальшого аналізу, а тому мають бути завчасно локалізовані та проігноровані системою. Зважаючи на те, що певні астрономічні об’єкти можуть також мати лінійну форму, перетворення Хафа для пошуку прямих в цьому випадку може бути застосоване лише в якості попереднього етапу виявлення дефектів зображення. Виконується подальша обробка результату перетворення Хафа для запобігання хибної класифікації астрономічного об’єкту в якості дефекту зображення.

РОЗДІЛ 3
АНАЛІЗ РЕАЛІЗАЦІЙ АЛГОРИТМІВ ПЕРЕТВОРЕННЯ ХАФА ДЛЯ ПОШУКУ ПРЯМИХ ТА КІЛ НА ЗОБРАЖЕННІ В БІБЛІОТЕЦІ OPENCV

3.1. Тестування та аналіз реалізації алгоритму перетворення Хафа для пошуку прямих в бібліотеці OpenCV

В бібліотеці OpenCV реалізацією стандартного алгоритму перетворення Хафа для пошуку прямих є функція *cv::HoughLines(image, lines, rho, theta, threshold, srn = 0, stn = 0, min_theta = 0, max_theta = Math.PI)* [9], що приймає 9 параметрів, 4 з яких мають замовчувані значення. Для застосування даної функції користувач має надати значення принаймні перших 5 параметрів. Параметр *image* – одно канальне бінарне зображення, що є результатом застосування детектора Кенні та інших етапів попередньої обробки початкового багатоканального зображення. lines – це відсилка на екземпляр класу *_OutputArray*, який неявно створюється на основі *std::vector*, що параметризований типом *Vec2f* або *Vec3f*. В свою чергу *Vec2f* та *Vec3f* – це псевдоніми класу *Vec<float, 2>* та *Vec<float, 3>* відповідно, що представляють двовимірний та тривимірний вектори, які використовуються для збереження двох та відповідно трьох значень типу float в якості координат. Таким чином, параметр lines у випадку використання *std::vector<Vec2f>* використовується для збереження у зовнішню колекцію (вектор) пар значень параметрів (ρ, θ), а у випадку використання *std::vector<Vec3f>* в зовнішній колекції зберігаються трійки значень $(\rho, \theta, votes)$, де *votes* – значення лічильника у відповідній комірці акумуляторного масиву. Параметр *rho* позначає «ціну поділки» на осі ρ в просторі Хафа. Наприклад, зважаючи на те, що максимальним значенням ρ є довжина діагоналі зображення, то, взявши «ціну поділки» в 1 піксель, акумуляторний масив матиме кількість рядків рівну довжині діагоналі зображення в пікселях. *theta* є «ціною поділки» на осі θ в просторі Хафа. Якщо значенням theta взяти $\frac{\pi}{180}$ (радіанну міру 1°), то кількість колонок в акумуляторному масиві буде рівною 180. *threshold* – порогове значення

лічильника в акумуляторному масиві. В результат потрапляють лише ті пари (ρ, θ), для яких значення лічильника у відповідній комірці акумуляторного масиву строго більше за значення *threshold*. Параметр *srn* є значенням дільника «ціни поділки» *rho*. Якщо значення цього параметру більше за 0, то застосовується алгоритм Multi-Scale Hough Transform, [10] що використовує частку *rho / srn* в якості уточненої «ціни поділки» осі ρ в просторі Хафа. У випадку, коли *srn* має значення 0, використовується стандартний алгоритм перетворення Хафа. *stn* – дільник параметру *theta*. Використовується аналогічно до параметру *srn* в алгоритмі Multi-Scale Hough Transform. Параметр *min_theta* задає мінімальне значення кута нахилу перпендикуляра θ в радіанах. Значення цього параметру має бути більше або рівне 0 та менше за значення *max_theta*. Відповідно *max_theta* – максимальне значення кута θ в радіанах, $min\_theta < max\_theta \leq \pi$.

Для тестування функції *HoughLines* було використане зображення шахової дошки розміром 800×1200 пікселів (рисунок 3.1). В якості попередніх етапів обробки початкове зображення конвертується у відтінки сірого за допомогою функції *cvtColor*, після чого до одноканального зображення-результату застосовується детектор границь Кенні (функція Canny).

Рисунок 3.1 – Початкове зображення шахової дошки

Шляхом багаторазового застосування функції *HoughLines* було визначено оптимальне значення параметру *threshold = 190*, яке дозволяє з одного боку

розпізнати найбільшу кількість прямих, що візуально спостерігаються на початковому зображенні, а з іншого – відфільтрувати всі прямі, які не мають візуального відображення на початковому зображенні. Також варто зазначити, що найкраща точність результату перетворення Хафа досягається при значенні параметрів *rho* та *theta* в 1 піксель та 1° відповідно. Решті параметрів були надані замовчувані значення. Результат застосування функції *HoughLines* із зазначеним набором параметрів продемонстровано на рисунку. 3.2.

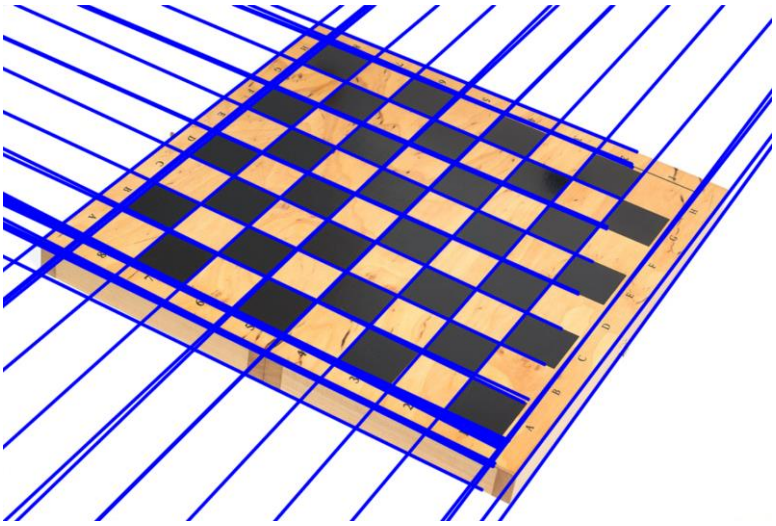

Рисунок 3.2 – Результат застосування функції *HoughLines* на початковому зображенні

Збільшення значення параметру *threshold* призводить до зменшення кількості виявлених ліній в результаті. Зменшення значення *threshold* навпаки призводить до збільшення числа ліній в результаті, серед яких можуть бути хибні лінії, що не спостерігаються на початковому зображенні. Збільшення значення «ціни поділки» *rho* може призвести до втрати ліній, для яких значення ρ потрапляє в проміжок між сусідніми поділками на осі ρ в просторі Хафа. Водночас, при обрахунку за формулою (1) значення параметру ρ, відбувається округлення цього значення до найближчої поділки на осі ρ простору Хафа, що в свою чергу призводить до збільшення лічильника в акумуляторному масиві для прямих, які мають значення параметру ρ, що «потрапляє» на одну з поділок осі ρ в просторі Хафа. Таким чином, збільшуючи значення параметру *rho*, бажано збільшувати і

порогове значення *threshold* для запобігання потраплянню в результат «хибних» прямих. Збільшення «ціни поділки» *theta* просто призводить до втрати в результаті прямих, для яких значення кута θ потрапляє в проміжок між сусідніми поділками на осі θ в просторі Хафа. Тому при збільшенні значення параметру *theta* не обов'язково збільшувати порогове значення *threshold*. Збільшення значень параметрів *rho* та *theta* має сенс при обробці зображення великого розміру, адже це дозволяє зменшити кількість обчислень та обсяг використаної оперативної пам'яті. Параметри *min_theta* та *max_theta* обмежують діапазон значень кута нахилу перпендикуляра θ, тим самим дозволяючи виявляти прямі, які знаходяться в конкретному положенні на зображенні. Наприклад, при значеннях параметрів *min_theta* = $\frac{7\pi}{36}$ та *max_theta* = $\frac{\pi}{3}$ в результат проходять лише прямі, що паралельні рядкам шахової дошки (рисунок 3.3).

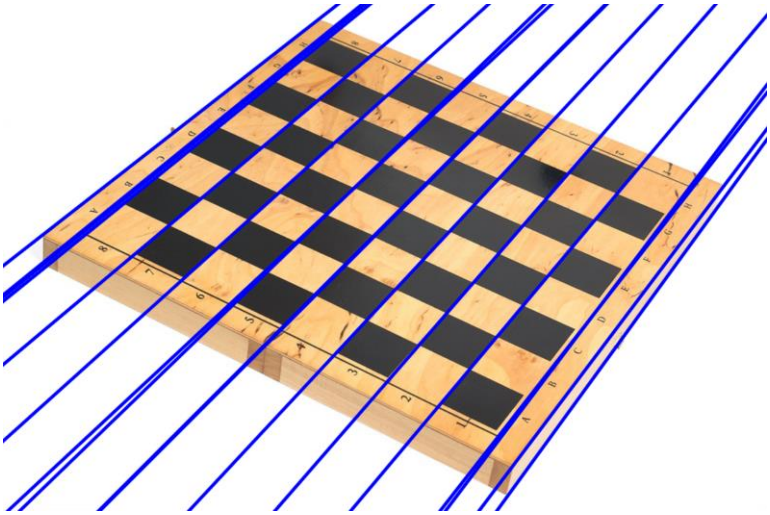

Рисунок 3.3 – Результат застосування функції *HoughLines* з обмеженим діапазоном значень кута θ.

Окрім стандартного перетворення Хафа для пошуку прямих в бібліотеці OpenCV наявна реалізація алгоритму ймовірнісного перетворення Хафа для пошуку прямих, що представлена функцією *HoughLinesP(image, lines, rho, theta, threshold, minLineLength, maxLineGap)*. [11] На відміну від функції *HoughLines* *HoughLinesP* дозволяє не лише виявити прямі на зображенні, а й локалізувати конкретні відрізки цих прямих. Призначення параметрів *image*, *lines*, *rho*, *theta*,

threshold функцій *HoughLines* та *HoughLinesP* є аналогічними. Відмінність полягає у тому, що в якості параметру *lines* у функцію *HoughLinesP* має бути передана колекція *std::vector*, що параметризована типом чотириелементних векторів *Vec4f*. Таким чином, результатом роботи функції є набір четвірок значень (x_1, y_1, x_2, y_2), де (x_1, y_1) та (x_2, y_2) – координати кінців відрізка прямої. Параметр *minLineLenght* задає значення мінімальної довжини виявленого відрізка, а параметр *maxLineGap* задає значення максимальної дозволеної відстані між двома послідовними пікселями в одному відрізку.

Для тестування функції *HoughLinesP* було використане зображення шахової дошки з попереднього прикладу. Значення параметрів *rho*, *theta*, *threshold* були взяті ті самі, що і під час тестування функції *HoughLines*. Значення параметрів *minLineLength* та *maxLineGap* були взяті в 1 та 100 пікселів відповідно для виявлення максимальної кількості відрізків. Результат застосування функції HoughLinesP з вказаним набором параметрів продемонстровано на рисунку 3.4.

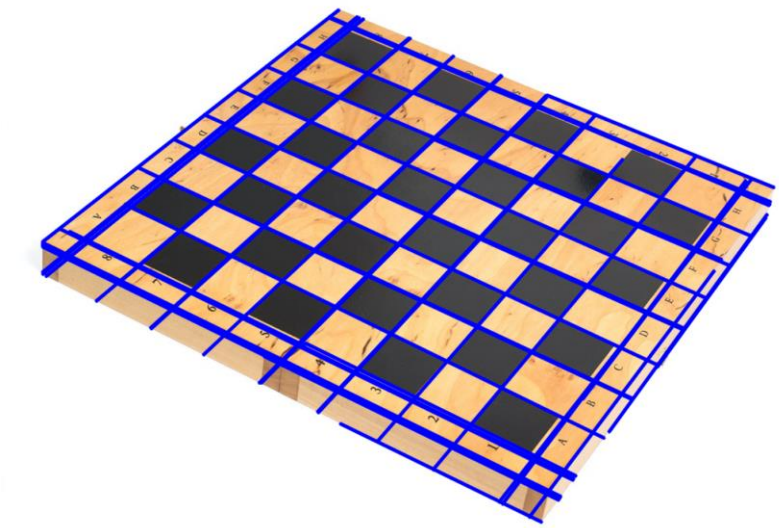

Рисунок 3.4 – Результат застосування функції *HoughLinesP* на початковому зображенні.

Результат функції *HoughLinesP* містить відрізки всіх тих прямих, що були виявлені функцією *HoughLines*. Збільшення значення мінімальної довжини відрізка призводить до зменшення кількості відрізків в результаті. В даному випадку значне зменшення кількості виявлених відрізків відбулося після встановлення значення *minLineLength* у 300 пікселів. Зменшення значення параметру *maxLineGap* призводить до збільшення кількості виявлених відрізків.

При цьому зменшується середня довжина відрізків. Надання параметру *maxLineGap* малого значення (наприклад в 1 піксель) може призвести до появи в результаті певних ізольованих відрізків малої довжини, що візуально не спостерігаються на початковому зображенні.

Функція *HoughLinesP* опрацьовує зображення за менший проміжок часу, ніж функція *HoughLines*. Так, для зображення шахової дошки розміром 800×1200 пікселів час роботи імовірнісного перетворення Хафа становить приблизно 70 мілісекунд, а час роботи стандартного перетворення Хафа – 108 мілісекунд. Для зображення розміром 1024×1024 пікселів час роботи функцій *HoughLinesP* та *HoughLines* становить відповідно 41 мілісекунда та 64 мілісекунд. Тому можна стверджувати, що в бібліотеці OpenCV імовірнісне перетворення Хафа працює швидше за стандартне перетворення Хафа в середньому на 36%.

Таким чином, функцію *HoughLinesP* варто застосовувати в тих випадках, коли необхідно виявити межі лінійних границь на зображенні. Крім того, функція імовірнісного перетворення Хафа дозволяє з більшою точністю локалізувати найдрібніші лінійні властивості об’єктів зображення за допомогою підбору відповідних значень мінімальної довжини відрізка та максимальної довжини проміжку між послідовними точками на прямій. Також функцію *HoughLinesP* доцільно використовувати при необхідності мінімізації часу обробки зображення або за потреби швидкої обробки зображення великого розміру без втрати точності результату. З іншого боку, якщо існує потреба у розпізнаванні лінійних границь, що знаходяться в певному кутовому положенні на зображенні, потрібно застосовувати функцію *HoughLines* з відповідним діапазоном значень кута θ. Також функцію стандартного перетворення Хафа доцільно застосовувати у ситуаціях, коли межі лінійних границь об’єкта неважливі.

3.2. Тестування та аналіз реалізації алгоритму перетворення Хафа для пошуку кіл на зображенні в бібліотеці OpenCV.

Перетворення Хафа для пошуку кіл на зображенні в бібліотеці OpenCV реалізоване функцією *HoughCircles(image, circles, method, dp, minDist, param1, param2, minRadius, maxRadius).*[12] Функція приймає значення 9 параметрів, 4 з яких мають замовчувані значення. Параметр *image* – початкове одноканальне зображення у відтінках сірого. Рекомендовано застосувати функції розмиття зображення (наприклад фільтр Гауса) до початкового зображення *image* в якості попередньої обробки перед застосуванням перетворення Хафа задля запобігання потраплянню в результат «хибних» кіл. *circles* – відсилка на контейнер трьохелементних або чотириелементних векторів: *std::vector<Vec3f>* або *std::vector<Vec4f>*. В першому випадку в результуючій колекції зберігаються трійки значень (*x, y, radius*), де *(x, y)* – координати центру кола, а в другому випадку – четвірки значень *(x, y, radius, votes)*, де *votes* – значення лічильника у відповідній комірці акумуляторного масиву. Параметр *method* вказує на те, який саме алгоритм перетворення Хафа для пошуку кіл має бути використаний. *method* приймає іменовані значення типу *HoughModes*. [13] Цьому параметру можуть бути надані 2 значення: *HOUGH_GRADIENT* або *HOUGH_GRADIENT_ALT*. При передачі значення *HOUGH_GRADIENT* буде використаний алгоритм 2-1 Hough Transform [14], що є модифікацією стандартного алгоритму перетворення Хафа для пошуку кіл, а при передачі значення *HOUGH_GRADIENT_ALT* використовується модифікація алгоритму 2-1 Hough Transform, що дозволяє з більшою точністю виявляти кола відносно малого радіусу на зображенні. Параметр *dp* задає значення відношення розмірів (висоти і ширини) вхідного зображення до розмірів акумуляторного масиву. Іншими словами даний параметр є коефіцієнтом, що дозволяє визначити розмір та роздільну здатність акумуляторного масиву відносно розмірів вхідного зображення. Чим менше значення *dp*, тим більший розмір та роздільна здатність акумуляторного масиву. Значення мінімальної відстані між центрами виявлених кіл задається через параметр *minDist.* Значення параметру *param1* задає верхнє порогове значення для детектора границь Кенні. Нижнє порогове значення для

детектора Кенні визначається як вдвічі менше за верхнє. Призначення параметру *param2* залежить від значення параметру *method*: для *HOUGH_GRADIENT param2* використовується для передачі порогового значення лічильника в акумуляторному масиві, а для *HOUGH_GRADIENT_ALT param2* задає міру «досконалості» шуканих кіл. В цьому випадку значення *param2* має знаходитись між 0 та 1, де значення 1 означає, що алгоритм виявлятиме лише досконалі кола. Через параметр *minRadius* встановлюється значення мінімальної довжини радіусу виявленого кола. Параметр *maxRadius* використовується для встановлення значення максимальної довжини радіусу виявленого кола. Якщо значення параметру *method = HOUGH_GRADIENT* і значення *maxRadius* < 0, то алгоритм виявляє лише центри кіл, повертаючи 0 в якості радіусу знайденого кола. У всіх інших випадках при наданні параметру *maxRadius* від'ємного або нульового значення в якості максимальної довжини радіусу береться довжина найбільшого виміру зображення (ширини або висоти).

Для тестування функції *HoughCircles* було використане зображення 600×900 восьми монет різного розміру (рисунок 3.5).

Рисунок 3.5 – початкове зображення для тестування функції *HoughCircles*.

В якості етапу попередньої обробки початкового зображення була застосована функція *GaussianBlur* до зображення у відтінках сірого для зменшення рівня деталізації об'єктів.

Спершу тестування функції *HoughCircles* виконувалось зі значенням параметру *method = HOUGH_GRADIENT*. Для цієї реалізації перетворення Хафа шляхом багаторазового тестування був визначений набір значень параметрів *dp = 1*, *minDist = 225*, *param1 = 100*, *param2 = 70*, *minRadius = 0*, *maxRadius = 100*, при якому в результаті міститься найбільша кількість «актуальних» кіл та повністю відсутні «хибні» кола (рисунок 3.6).

Рисунок 3.6 – результат застосування функції *HoughCircles* зі значенням параметру *method = HOUGH_GRADIENT* на початковому зображенні.

Попри в цілому непогану точність результату, деякі візуально спостережні кола на початковому зображенні все ж лишаються невиявленими (наприклад, зовнішній контур третьої монети у верхньому рядку). Це можна пояснити тим, що на перший погляд концентричні кола, які є контурами гербу на монеті та самої монети, насправді не є концентричними. Відстань між центрами таких кіл набагато менша за значення параметру *minDist*, через що одразу обидва такі «хибно концентричні» кола не можуть потрапити до результату. Значне зменшення мінімальної відстані між центрами кіл *minDist* призводить до появи в результаті великої кількості «хибних» кіл, що не спостерігаються на початковому зображенні, хоч і вирішує проблему «хибно концентричних» кіл. На даному зображенні зменшення порогового значення лічильника *param2* не призвело до появи «хибних» кіл в результаті, адже такі кола відсіювалися за

допомогою значення мінімальної дистанції між центрами кіл. Значне збільшення порогового значення лічильника призвело до зникнення з результату кіл, що є контурами монет малого радіусу (наприклад останньої монети нижнього ряду). Зміна значень максимальної та мінімальної довжини радіуса відповідно впливало на потрапляння чи не потрапляння до результату кола того чи іншого радіусу. Окремої уваги заслуговує параметр *dp*. Для більш наглядної демонстрації того, як зміна значення цього параметру впливає на точність виявлення та кількість кіл в результаті, було використане інше зображення, на якому представлені «недосконалі» кола різного розміру (рисунок 3.7).

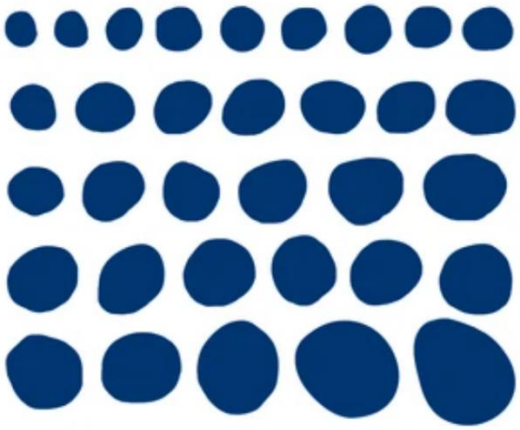

Рисунок 3.7 – початкове зображення «недосконалих» кіл.

Результат застосування функції *HoughCircles* на цьому зображенні зі значенням параметра *dp = 1* продемонстрований на рисунку 3.8.

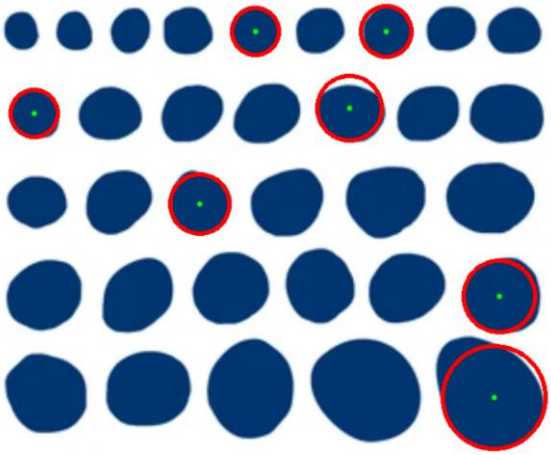

Рисунок 3.8 – результат застосування функції *HoughCircles* зі значенням параметрів *method = HOUGH_GRADIENT* та *dp = 1*.

В цьому випадку були виявлені лише кола, які базуються на контурах фігур, що мають найбільш круглу форму. Решта кругоподібних фігур була проігнорована алгоритмом. При збільшенні значення параметра *dp* функція виявляє кола навколо майже всіх кругоподібних об’єктів на зображенні (рисунок 3.9).

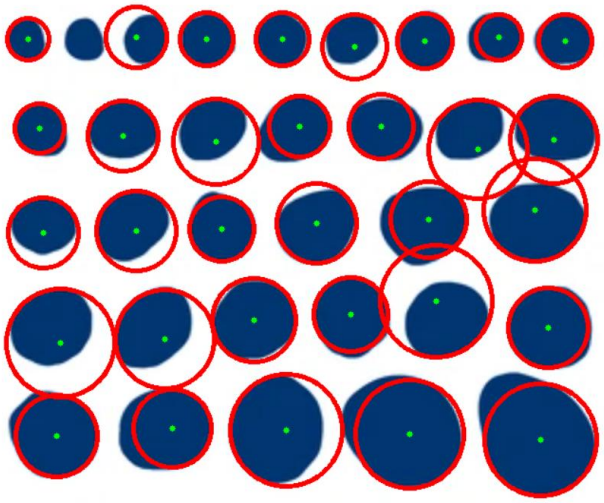

Рисунок 3.9 - результат застосування функції *HoughCircles* зі значенням параметрів *method = HOUGH_GRADIENT* та *dp = 2*.

Таким чином, можна стверджувати, що при збільшенні значення параметра *dp* алгоритм став «менш вибагливим» до «досконалості» форми об’єкта. Це можна пояснити тим, що під час збільшення значення цього коефіцієнту зменшуються розміри та роздільна здатність акумуляторного масиву. Наприклад, при відношенні розмірів зображення до розмірів акумуляторного масиву як один до одного, кожному пікселю (*x*, *y*) зображення ставиться у відповідність пара значень (*a*, *b*) в акумуляторному масиві (просторі Хафа). При зменшенні роздільної здатності акумуляторного масиву кожній парі значень (*a*, *b*) з простору Хафа ставиться у відповідність декілька пікселів зображення. Через це одразу кілька граничних пікселів зображення можуть інкрементувати значення лічильника у спільній для них комірці акумуляторного масиву на певній позиції (a_i, b_j). Зменшення роздільної здатності акумуляторного масиву має свої переваги та недоліки: з одного боку зменшується кількість обчислень та обсяг оперативної пам’яті для зберігання акумуляторного масиву, спрощуються

вимоги до «правильності» форм об’єктів, що розпізнається, а з іншого – втрачається точність визначення центру кола на зображенні.

Під час другого етапу тестування функції *HoughCircles* використовувалось значення параметру *method = HOUGH_CIRCLES_ALT*. Функція застосовувалась для зображень з попередніх прикладів. Початковий набір значень параметрів був визначений наступним чином: *dp = 1, minDist = 100, param1 = 100, param2 = 0.99, minRadius = 0, maxRadius = 0.* Результат роботи функції з даним набором значень параметрів продемонстрований на рисунку 3.10.

Рисунок 3.10 - результат застосування функції *HoughCircles* зі значенням параметру *method = HOUGH_GRADIENT_ALT* на початковому зображенні.

Алгоритм коректно виявив всі кола-контури монет, але менші «хибно концентричні» кола (наприклад кола-контури гербів на монетах) залишились не виявленими. Значне зменшення значення мінімальної відстані між центрами кіл допомогло виявити кола-контури гербів лише на частині монет. Проте, на відміну від алгоритму *HOUGH_GRADIENT*, встановлення значення параметру *minDist = 1* в цьому випадку не призвело до появи в результаті «хибних» кіл. Причиною цього стало близьке до одиниці значення параметру *param2*, яке «змушувало» алгоритм «шукати» практично ідеальні кола на зображенні. Зменшення значення міри «досконалості» кола при низькому значенні параметру *minDist* спричинило потрапляння в результат великої кількості «хибних» кіл.

Алгоритм правильно виявляє всі візуально спостережні кола на зображенні при значенні мінімальної відстані між центрами кіл у 100 пікселів та значенні параметру *param2 = 0.8* (рисунок 3.11).

Рисунок 3.11 - результат застосування функції *HoughCircles* зі значеннями параметрів *method = HOUGH_GRADIENT_ALT* та *param2 = 0.8* на початковому зображенні.

Функція *HoughCircles* зі значенням параметру *method = HOUGH_GRADIENT* працює швидше, ніж зі значенням *HOUGH_GRADIENT_ALT*. Так, для зображення монет розміром 600×900 час роботи функції із цими значеннями становив 660 мілісекунд та 1390 мілісекунд відповідно. Для зображення кругоподібних фігур розміром 563×679 час роботи становив 80 та 430 мілісекунд.

Таким чином, якщо зображення містить велику кількість об’єктів недосконалої кругоподібної форми, варто застосувати функцію зі значенням *HOUGH_GRADIENT_ALT* та дещо зниженим значенням міри «досконалості» для виявлення кіл. В інших випадках, коли важлива мінімізація часу обробки або зображення містить в основному об’єкти досконало круглої форми, доцільно використати застосувати функцію зі значенням *HOUGH_GRADIENT*.

РОЗДІЛ 4
СТВОРЕННЯ ТА АНАЛІЗ ВЛАСНИХ РЕАЛІЗАЦІЙ АЛГОРИТМІВ ПЕРЕТВОРЕННЯ ХАФА ДЛЯ ПОШУКУ ПРЯМИХ ТА КІЛ НА ЗОБРАЖЕННІ

4.1. Опис власної реалізації алгоритму перетворення Хафа для пошуку прямих на зображенні.

Власна реалізація стандартного алгоритму перетворення Хафа для пошуку прямих на зображенні представлена функцією *myHoughLineTransform(edges, output, thrshld, intensity_image_p).* Для застосування цієї функції користувачу необхідно надати значення принаймні перших 3 параметрів. Останній параметр *intensity_image_p* має значення за замовчуванням. Параметр *edges* є сталою відсилкою на бінарне зображення, що є результатом застосування детектора Кенні до початкового зображення. Параметр *output* є відсилкою на зображення-результат, яке використовується для демонстрації виявлених прямих. Бажано, щоб це зображення було копією початкового для наочності результату. Порогове значення лічильника задається через параметр *thrshld*. Останній параметр *intensity_img_p* є указником на зображення, яке буде використане для візуалізації простору Хафа. В сигнатуру функції не виносились параметри, що дозволяють встановлювати «ціну поділки» на осях простору Хафа та обмежувати діапазон значень кута θ. Це дозволило спростити сигнатуру функції за рахунок обмеження гнучкості її використання.

Максимальне значення довжини перпендикуляру до прямої ρ визначене в коді функції та дорівнює подвійній довжині діагоналі зображення. Значення кута θ належить діапазону: $0° \leq \theta \leq 180°$. Акумуляторний масив (простір Хафа) представлений одновимірним динамічним масивом значень типу *int* під назвою *hspace*. Фактично, цей масив використовується в якості двовимірної матриці розміру $\rho_{max} \times \theta_{max}$, адже доступ до його комірок відбувається за формулою:

$$index = row * NCOLS + col,$$

де *index* – порядковий номер комірки,

NCOLS – кількість стовпців в матриці.

Перед використанням масиву *hspace* значення його комірок ініціалізуються нулем. За допомогою подвійного циклу відбувається ітерація по рядках та стовпцях зображення. У внутрішньому циклі відбувається перевірка належності пікселя на певній позиції границі об’єкта на зображенні через порівняння значення яскравості цього пікселя з нулем. Якщо піксель належить певній границі, то за допомогою третього циклу відбувається ітерація по всім цілим значенням градусної міри кута θ з величиною кроку ітерації в 1°. Після фіксації в третьому за вкладеністю циклі деякого значення градусної міри кута θ, відбувається обрахунок довжини перпендикуляра до прямої ρ за формулою рівняння прямої (1). При цьому до обрахованого за формулою значення довжини перпендикуляра додається значення довжини діагоналі зображення. Ця дія виконується для того, щоб кінцеве значення параметру ρ не було від’ємним та належало проміжку $[0, \rho_{max}]$. Після здійснення обрахунків, значення у відповідній комірці масиву *hspace* збільшується на одиницю. Таким чином опрацьовуються всі граничні пікселі зображення та заповнюється акумуляторний масив. На наступній стадії перевіряється, чи надане значення указника *intensity_img_p* не рівне *nullptr*. Якщо ця умова виконується, то інформація, яка міститься в акумуляторному масиві переноситься на вказане зображення. Інакше ця дія не виконуються. Кінцевий етап роботи функції – відображення виявлених прямих на зображенні *output*. За допомогою подвійного циклу відбувається ітерація по всім значенням параметрів ρ та θ з величиною кроку ітерації в 1 піксель та 1° відповідно. У внутрішньому циклі відбувається перевірка того, чи значення лічильника в певній комірці масиву більше за порогове значення *thrshld*. Якщо ця умова справджується, то лінія з координатами кінцевих точок, обчислених на основі відповідних значень довжини перпендикуляра та градусної міри кута нахилу, наноситься на результуюче зображення.

4.2. Порівняння власної реалізації алгоритму перетворення Хафа для пошуку прямих з відповідною реалізацією в бібліотеці OpenCV.

Для порівняння результатів роботи функцій *myHoughLineTransform* та *HoughLines* було використане зображення, продемонстроване на рисунку 4.1.

Рисунок 4.1 – початкове зображення для порівняння функцій *myHoughLineTransform* та *HoughLines.*

Функція *HoughLines* тестувалася з зі значеннями параметрів *rho = 1 theta = CV_PI / 180* для відповідності функції *myHoughLineTransform*, яка використовує значення «ціни поділки» параметрів ρ та θ в 1 піксель та 1°. Обидві функції використовували однакове порогове значення лічильника, що змінювалось протягом тестування. Результати застосування функцій *HoughLines* та *myHoughLineTransform* з різними значеннями порогового параметру *threshold* продемонстровані на рисунках 4.2, 4.3.

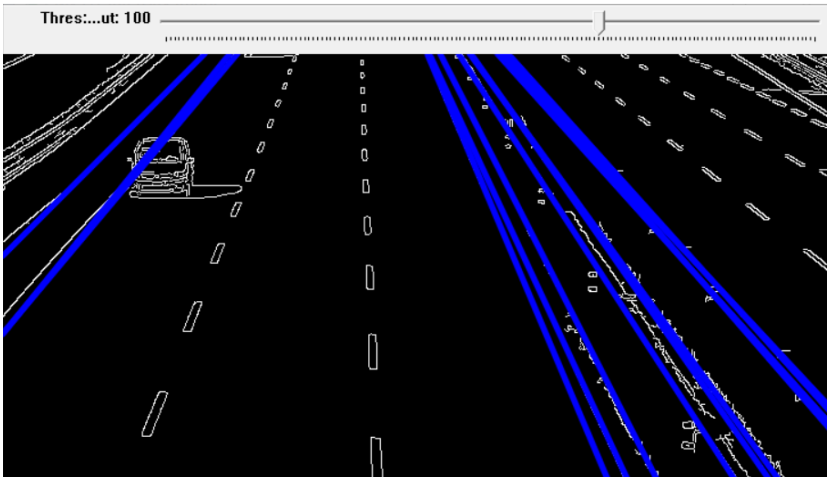

Рисунок 4.2 – результат застосування функції *HoughLines* зі значенням параметру *threshold = 100*.

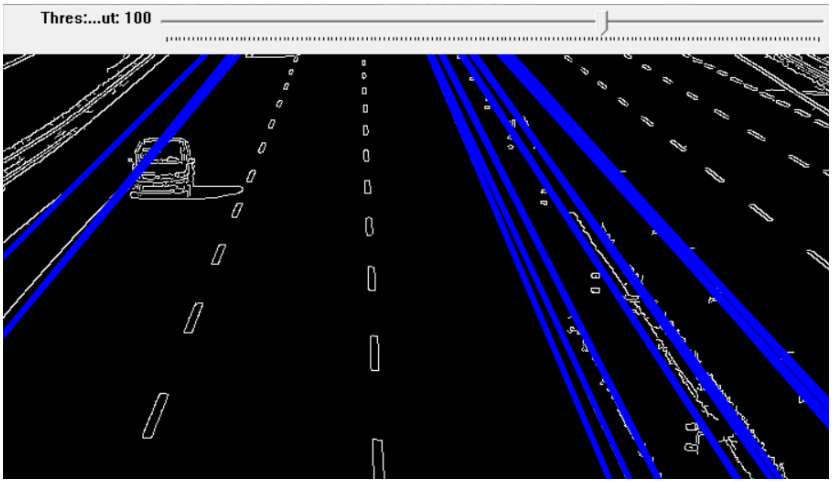

Рисунок 4.3 – результат застосування функції *myHoughLineTransform* зі значенням параметру *thrshld = 100*.

Функції *myHoughLineTransform* та *HoughLines* на однаковому наборі значень параметрів та вхідному зображенні щоразу повертали в результаті фактично ідентичний набір виявлених прямих. Попри це, функція *HoughLines* працює швидше за *myHoughLineTransform*: для зображення дороги (рис. 4.1) розміром 373×727 пікселів час роботи функцій становив відповідно 30 та 80 мілісекунд, а для зображення шахової дошки (800×1200) – 80 та 380 мілісекунд.

4.3. Опис власної реалізації алгоритму стандартного перетворення Хафа для пошуку кіл на зображенні.

Власна реалізація стандартного алгоритму перетворення Хафа для пошуку кіл на зображенні представлена методом *detect(input, output, min_radius, max_radius, thrshld, lo_canny_thrshld, hi_canny_thrshld)* класу *HoughCircleDetector*. Параметри цього методу не мають замовчуваних значень, тому користувач має надати значення для всіх 7 параметрів. Параметр *input* – стала відсилка на вхідне багатоканальне зображення без попередньої обробки, *output* – відсилка на зображення-результат, яке використовується для демонстрації виявлених кіл. Параметри *min_radius* та *max_radius* обмежують діапазон значень довжини радіусу шуканих кіл. Через параметр *thrshld* задається порогове значення лічильника в акумуляторному масиві. *lo_canny_thrshld* та *hi_canny_thrshld* задають відповідно нижнє та верхнє порогове значення для детектора границь Кенні.

В процесі реалізації перетворення Хафа для пошуку кіл виникла потреба у декомпозиції алгоритму на декілька частин, які б реалізовувались окремо кількома функціями. Такий підхід потребує створення певної функції-інтерфейсу для безпосередньої взаємодії з користувачем. Решта допоміжних функцій мають бути прихованими від користувача. Рішенням цих задач стало створення окремого класу *HoughCircleDetector*, що інкапсулює деталі реалізації перетворення Хафа та надає користувачу єдиний метод-інтерфейс *detect* для передачі значень параметрів в алгоритм. Через велику часову складність стандартного перетворення Хафа для пошуку кіл виникла необхідність застосування багатопоточності для прискорення процесу знаходження кіл у великому діапазоні значень довжини радіусу. Метод *detect* виконує попередню обробку зображення шляхом його переведення у відтінки сірого функцією *cvtColor*, застосування фільтру Гауса та детектора границь Кенні з визначеними користувачем верхнім та нижнім пороговими значеннями. Після цього відбувається ініціалізація акумуляторного масиву під назвою *hspace*, що представлений двовимірним масивом значень типу *int*. Наступним етапом відбувається перевірка розміру діапазону значень довжини радіуса кіл: якщо цей

діапазон містить більше 30 значень, то створюються 8 паралельних потоків, кожен з яких виконує метод *single_thread_HCT* з відповідним піддіапазоном значень довжини радіуса. Інакше метод *single_thread_HCT* виконується в основному потоці. В свою чергу метод *single_thread_HCT* реалізує основну логіку алгоритму перетворення Хафа. Зовнішній цикл в цьому методі використовується для ітерації по всім значенням довжини радіуса з наданого піддіапазону значень. Решта дій виконуються в тілі цього циклу. Відбувається ініціалізація одновимірного підмасиву розміру, рівного кількості пікселів початкового зображення, на певній позиції в масиві *hspace*. Після цього за допомогою подвійного циклу перевіряється належність кожного пікселя певній границі на зображенні. Якщо піксель на деякій позиції є граничним, то за допомогою методу *denoteCircle,* що використовує алгоритм растеризації кола [15] для визначення множини точок в просторі Хафа, які умовно формують коло з центром в точці, що має координати граничного пікселя, збільшується на одиницю значення лічильника у відповідних комірках одновимірного підмасиву. Останній подвійний цикл ітерується по комірках одновимірного підмасиву та перевіряє значення лічильника: якщо воно більше за порогове значення *thrshld* – на зображенні-результаті *output* відображається коло з відповідними координатами центру та радіусом.

4.4. Порівняння власної реалізації алгоритму перетворення Хафа для пошуку кіл з відповідною реалізацією в бібліотеці OpenCV.

Для тестування методу *detect* класу *HoughCircleDetector* та функції *HoughCircles* було використане зображення монет розміру 600×900 пікселів. Під час застосування функції HoughCircles був визначений набір значень параметрів *method = HOUGH_GRADIENT, dp = 1* для більш точної відповідності власній реалізації перетворення Хафа. Порогове значення лічильника в акумуляторному масиві та діапазон значень довжини радіусу шуканих кіл є спільними для обох функцій. Результати застосування функції *HoughCircles* та методу *detect* продемонстровані на рисунках 4.4, 4.5.

Рисунок 4.4 – Результат застосування функції *HoughCircles* на початковому зображенні.

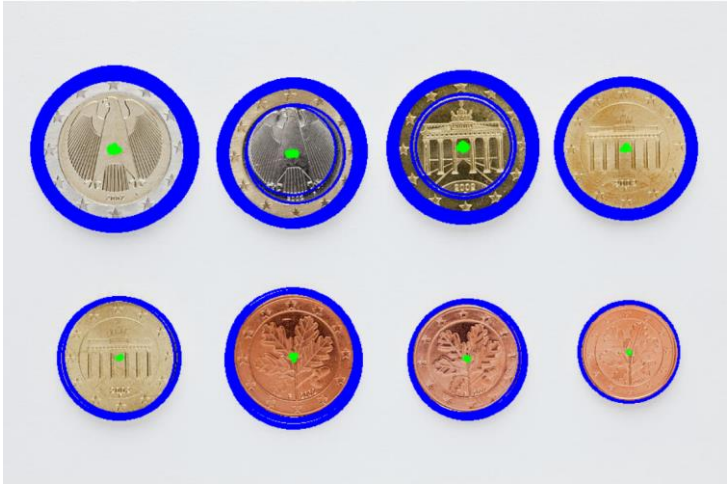

Рисунок 4.5 – Результат застосування методу *detect* на початковому зображенні.

Попри спільні значення параметрів, набір кіл, виявлених функцією *HoughCircles* та методом *detect*, відрізняється. Це можна пояснити тим, що дані функції фактично реалізують різні алгоритми перетворення Хафа для пошуку кіл. Метод *detect* є прямолінійною реалізацією класичного алгоритму перетворення Хафа для пошуку кіл, в той час як функція *HoughCircles* зі значенням параметру *method = HOUGH_GRADIENT* є реалізацією алгоритму 2-1 Hough Transform, що є модифікацією стандартного перетворення Хафа. Можна помітити, що результат методу *detect* містить велику кількість «псевдо концентричних» кіл, довжина радіусу яких відрізняється менше, ніж на 10 пікселів. Причиною цього є те, що у методі *single_thread_HCT* відбувається ітерація по діапазону значень

довжини радіусу з кроком ітерації в 1 піксель. Можливим рішенням цієї проблеми є збільшення кроку ітерації по діапазону значень довжини радіусу або зменшення роздільної здатності одновимірних підмасивів масиву *hspace*. Можна зробити висновок, що власна реалізація перетворення Хафа для пошуку кіл не є повністю досконалою та потребує певної модифікації для використання на практиці.

ВИСНОВКИ

В результаті дослідження прикладних областей можливого застосування алгоритмів перетворення Хафа для пошуку прямих та кіл на зображенні було розглянуто шість варіантів їхнього використання. В роботі продемонстровано приклади цифрових систем, що так чи інакше використовують ці алгоритми для автоматизації певних процесів в галузях біометричної ідентифікації людини, медицини, неруйнівного контролю, транспорту та астрономії. Так, алгоритм перетворення Хафа для пошуку кіл використовується для локалізації райдужної оболонки на зображенні ока людини в системі автоматичної біометричної ідентифікації. В системах архівування та розсилання медичних зображень перетворення Хафа для пошуку кіл застосовується для локалізації інформативної частини радіографічного знімку. Третій варіант використання алгоритму перетворення Хафа для пошуку кіл має місце в автоматизованих системах аналізу крові, де він застосовується для розпізнавання лейкоцитів на мікроскопічному зображенні зразка крові. У свою чергу алгоритм перетворення Хафа для пошуку прямих використовується системами неруйнівного контролю для перевірки наявності дефектів зварювального шву на радіографічному зображенні. Системи адаптивного круїз-контролю використовують перетворення Хафа для пошуку прямих задля розпізнавання смуг руху, лінії горизонту та меж дороги. Крім того, перетворення Хафа для пошуку прямих застосовується в системах автоматичного сканування астрономічних знімків нічного неба для виявлення об’єктів не астрономічної природи. Через обмеженість обсягу курсової роботи було розглянуто лише малу частину від всіх можливих варіантів практичного використання алгоритмів перетворення Хафа.

В результаті аналізу та тестування функцій *HoughLines* та *HoughLinesP*, що є реалізаціями перетворення Хафа для пошуку прямих в бібліотеці OpenCV, було досліджено призначення параметрів в їхніх сигнатурах та продемонстровано залежність результатів роботи функцій від значень цих параметрів. За підсумками порівняння цих реалізацій були визначені їхні переваги та недоліки. Так, основною перевагою функції *HoughLinesP* є можливість визначення меж лінійних границь на зображенні. Крім того, серед переваг цієї функції можна

зазначити можливість встановлення значень мінімальної довжини відрізка та максимальної довжини проміжку між послідовними точками прямої, що дає змогу збільшити точність виявлення найбільш дрібних лінійних властивостей об'єктів зображення. Ще однією перевагою функції *HoughLinesP* є її висока часова ефективність. Недоліком даної реалізації перетворення Хафа для пошуку прямих є неможливість обмежити діапазон значень градусної міри кута θ. Перевагою функції *HoughLines* є можливість визначення діапазону значень градусної міри кута θ, що дає змогу виявляти прямі в конкретному кутовому положенні на зображенні.

Аналізуючи функцію *HoughCircles*, що реалізує певні модифікації алгоритму перетворення Хафа для пошуку кіл на зображенні в бібліотеці OpenCV, визначено призначення параметрів в її сигнатурі та залежність результату роботи функції від значень цих параметрів. За результатами тестування *HoughCircles* з різними значеннями параметру *method* були описані переваги та недоліки використання значень *HOUGH_GRADIENT* та *HOUGH_GRADIENT_ALT*. Серед переваг використання цієї функції зі значенням параметру *method = HOUGH_GRADIENT_ALT* є можливість визначення значення міри «досконалості» кіл, що дозволяє більш точно виявляти об'єкти недосконалої кругоподібної форми на зображенні. Іншою перевагою використання цього значення параметру *method* є відсутність необхідності визначення порогового значення лічильника в акумуляторному масиві, що дозволяє зменшити залежність набору значень параметрів функції від конкретного зображення. Перевагою використання значення *HOUGH_GRADIENT* є зменшення часу роботи функції. Недоліком застосування функції з цим значенням параметру *method* є збільшення «чутливості» алгоритму до досконалості форми об'єкту.

Створено власну реалізацію алгоритму перетворення Хафа для пошуку прямих на зображенні. Було продемонстровано ідентичність результатів роботи власної функції-реалізації *myHoughLineTransform* та відповідної функції *HoughLines* з бібліотеки OpenCV на одному зображенні. За підсумками

порівняння часу роботи цих функцій було з’ясовано, що власна реалізація є менш ефективною за відповідну реалізацію в бібліотеці OpenCV.

Створено власну реалізацію алгоритму стандартного перетворення Хафа для пошуку кіл на зображенні. Порівнюючи результати роботи функції *HoughCircles* з бібліотеки OpenCV та методу *detect*, що представляє власну реалізацію перетворення Хафа для пошуку кіл, була продемонстрована відмінність наборів виявлених кіл на зображенні. Це обумовлене використанням даними функціями різних алгоритмів перетворення Хафа для пошуку кіл. В ході тестування методу *detect* був виявлений недолік реалізації, що полягає у потраплянні в результат великої кількості «хибно концентричних» кіл. Було запропоноване можливе рішення цієї проблеми та зазначено, що продемонстрована реалізація перетворення Хафа для пошуку кіл на зображенні вимагає певного доопрацювання для повноцінного використання на практиці.

СПИСОК ДЖЕРЕЛ

1. Hough Transform – Wikipedia. URL: https://en.wikipedia.org/wiki/Hough_transform#Detecting_lines
2. Circle Hough Transform – Wikipedia. URL: https://en.wikipedia.org/wiki/Circle_Hough_Transform
3. Recognition of Human Iris Patterns – Animesh Das, Ст. 12, URL: http://ethesis.nitrkl.ac.in/3635/1/108CS056.pdf
4. Hough transform applications in Computer Graphics (with focus on medical visualization) - Michael Wohlfart, Ст. 4, URL: https://www.cg.tuwien.ac.at/courses/Seminar/SS2003/Ergebnisse/WohlfartMichael_HoughTransform.pdf
5. A leucocytes count system from blood smear images: Segmentation and counting of white blood cells based on learning by sampling - Cecilia Di Ruberto, Andrea Loddo, Lorenzo Putzu, Ст. 1, URL: https://www.researchgate.net/publication/309466585_A_leucocytes_count_system_from_blood_smear_images_Segmentation_and_counting_of_white_blood_cells_based_on_learning_by_sampling
6. Automatic Detection of Weld Defects Based on Hough Transform - Chiraz AJMI, Sabra El FERCHICHI, Abderrahmen ZAAFOURI, Kaouther LAABIDI, Ст. 1, URL: https://ieeexplore.ieee.org/document/9116162
7. Real Time Lane Detection for Autonomous Vehicles - Abdulhakam.AM.Assidiq, Othman O. Khalifa, Md. Rafiqul Islam, Sheroz Khan, Ст. 85, URL: https://www.researchgate.net/publication/4356226_Real_time_lane_detection_for_autonomous_vehicles
8. Cleaning sky survey data bases using Hough transform and renewal string approaches - A. J. Storkey, N. C. Hambly, C. K. I. Williams, R. G. Mann, Ст. 38, URL: https://academic.oup.com/mnras/article-pdf/347/1/36/4099759/347-1-36.pdf

9. HoughLines – OpenCV Documentation, URL: https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a
10. A multi-scale plane-detection method based on the Hough transform and region growing - Xiaoxu Leng, Jun Xiao, Ying Wang, Ст. 1, URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/phor.12145
11. HoughLinesP – OpenCV Documentation, URL: https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga8618180a5948286384e3b7ca02f6feeb
12. HoughCircles – OpenCV Documentation, URL: https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga47849c3be0d0406ad3ca45db65a25d2d
13. HoughModes – OpenCV Documentation, URL: https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga073687a5b96ac7a3ab5802eb5510fe65
14. A comparative study of Hough Transform methods for circle finding - H.K. Yuen, J. Princen, J. Illingworth, J. Kittler, Ст. 170, URL: http://www.bmva.org/bmvc/1989/avc-89-029.pdf