КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

Кваліфікаційна робота
на здобуття освітнього ступеня «магістр»
на тему:

Векторизація растрових зображень з використанням технологій комп’ютерного зору

Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп’ютерні науки»
Освітньо-наукова програма «Технології штучного інтелекту»

Виконав:
студент 2 курсу магістратури групи ТШІ-21
Сітко Денис Олегович _____

Науковий керівник:
доктор технічних наук, професор
Самохвалов Юрій Якович _____

Засвідчую, що в цій кваліфікаційній роботі немає запозичень з праць інших авторів без відповідних посилань
Студент _____

Кваліфікаційна робота допущена до захисту
рішенням кафедри інтелектуальних технологій

Протокол № 8 від «5» травня 2022 р.
Зав. кафедри к. т. н., доцент Іларіонов Олег Євгенович _____

КИЇВ - 2022

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, 3 розділів, висновків, списку використаної літератури та додатків. Загальний обсяг роботи складає 46 сторінок. Робота містить 10 рисунків та 10 додатків.

Актуальність теми:

Векторне зображення відкриває широкий спектр можливостей в області його аналізу та взаємодії з мовою комп'ютера. Сучасне програмне забезпечення вміло інтерпретує лінгвістичну інформацію, оскільки природа мови передбачає символьну діалектику, яка є близькою до комп'ютерної. Векторизація растрового зображення дає змогу знайти певні закономірності в формі на наповненні звичайного растру. Векторне зображення оперує геометричними одиницями, на відміну від растру, описаного матрицею пікселів. Растрове зображення за формою та змістом є турбулентним з точки зору його структуризації.

Векторизація растрового зображення має на меті вирішення задачі ідентифікації форми зображуваних об'єктів, що дозволяє дешифрувати візуальну інформацію в структуровану математичну форму, яка є близькою до цифрової.

Мета роботи: дослідження та розробка системи векторизації растрових зображень.

Об'єкт дослідження: процес ідентифікації форми об'єктів, представлених растровими зображеннями.

Предмет дослідження: методи та алгоритми цифрової обробки зображення та комп'ютерного зору.

Ключові слова: бінаризація**,** векторизація растру, розпізнавання форм об'єктів, комп'ютерний зір, обробка зображення, виділення контурів.

ABSTRACT

The qualification work for obtaining the master degree consists of an introduction, 3 chapters, conclusions, literature sources, and annexes. The total volume is 46 pages, it contains 10 figures and 10 annexes.

Relevance:

Vector image introduces a wide range of possibilities in the field of its analysis and interaction with the computer language. Modern sowtware easily interprets linguistic information, because the nature of human language involves a symbolic dialect, which is rather close to digital. Vectorization of a raster aims to recognize patterns of the form and the content of a raster image. Vector image operates with geometric units, unlike a raster described by a pixel matrix. The raster image is turbulent in terms of structurization of its form and content.

Vectorization of a raster image enhances the shape identification problem solving. It enchances visual information decrypition into structured mathematical form that is close to digital.

Purpose: research and development of the raster image vectorization system.

The study object: the process of object shape detection depicted by a raster image.

The study subject: methods and algorithms of digital image processing and computer vision.

Keywords: binarization, raster vectorization, shape recognition, computer vision, image processing, edge detection.

ЗМІСТ

Вступ	7
Розділ 1. Аналіз задачі векторизаціїї растрового зображення	8
1.1. Актуальний стан задачі	8
1.2. Основні поняття інтелектуальної обробки зображення	8
1.2.1. Цифрова обробка зображення	8
1.2.2. Комп’ютерний зір	10
1.3. Характеристика типів зображення	11
1.3.1. Растрове зображення	11
1.3.2. Векторне зображення	11
1.4. Постановка задачі	12
1.5. Висновки до першого розділу	13
Розділ 2. Аналіз алгоритмів для векторизації растрового зображення	15
2.1. Алгоритм перетворення кольорового растру в чорно-білий	15
2.2. Алгоритми виділення контурів	15
2.2.1. Оператори Прюітта та Собеля	16
2.2.2. Алгоритм Кенні	17
2.2.3. Порівняння розглянутих алгоритмів виділення контурів	20
2.3. Алгоритм Хафа	22
2.4. Вдосконалений алгоритм Хафа	26
2.5. Висновки до другого розділу	27
Розділ 3. Розробка системи векторизації растрових зображень	28
3.1. Опис програмного забезпечення	28
3.2. Архітектура програмного забезпечення	29
3.3. Тестування програмного забезпечення	30
3.4. Висновки до третього розділу	33
Висновок	34
Список використаних джерел	35

Додаток 1 - Приклад векторизації растрового зображення	37
Додаток 2 - Приклад векторизації растрового зображення	37
Додаток 3 - Приклад векторизації растрового зображення	39
Додаток 4 - Приклад векторизації растрового зображення	40
Додаток 5 - Приклад векторизації растрового зображення	41
Додаток 6 - Приклад векторизації растрового зображення	42
Додаток 7 - Приклад векторизації растрового зображення	43
Додаток 8 - Приклад векторизації растрового зображення	44
Додаток 9 - Приклад векторизації растрового зображення	45
Додаток 10 - Приклад векторизації растрового зображення	46

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВК - виділення контурів

ІОЗ - інтелектуальна обробка зображень

КЗ - комп’ютерний зір

ООП - об’єктно орієнтоване програмування

ПЗ - програмне забезпечення

ЦОЗ - цифрова обробка зображень

ВСТУП

Людське життя невпинно зазнає глобальної діджиталізації. Спочатку комп’ютер навчили інтерпретувати строго формульовані команди, далі опрацьовувати вільні лінгвістичні запити. Сьогодні комп’ютер вміє в тій чи іншій мірі чути та бачити.

На відміну від мови, візуальна інформаціє не має чітких формалізованих рамок. Сучасне програмне забезпечення досягло колосального прогресу в області розуміння людської мови. Складність аналізу зображення є незрівнянно вищою у порівнянні зі символьною чи аудіальною інформацією. Той самий об’єкт у просторі під різними кутами зору може мати різні ознаки, об’єкт може рухатися, змінювати свої атрибути, мати складно розпізнаваний фон. Такі складнощі ідентифікації, аналізу та обробки візуальної інформації спричинили появу окремої галузі в області комп’ютерних наук та інформаційних технологій.

Комп'ютерний зір вже став невід’ємним інструментом багатьох людських сфер діяльності: медицини, освіти, правопорядку, інженерної справи тощо. Цифрові технології здатні швидко вирішувати поставлені задачі, але у купі з елементами штучного інтелекту спектр задач розширюється невпинно. Сьогодні комп’ютер вміє аналізувати інформацію візуального характеру, що відкриває низку нових задач, які можна виконувати в автоматизованому порядку з доволі високою ефективністю.

Векторне зображення є способом трансляції візуальної інформації в мову математики. Наявність формалізації в своїй структурі підносить аналіз та обробку зображення на принципово новий рівень в купі з відсутністю масштабних рамок. Наприклад, гравіювальний станок, що використовує лазерну технологію для розрахунку робочих поверхонь оперує векторними величинами. Так само і технологія автопілоту орієнтується в просторі згідно визначених просторових моделей, описаних геометричними одиницями.

РОЗДІЛ 1
АНАЛІЗ ЗАДАЧІ ВЕКТОРИЗАЦІЇ РАСТРОВОГО ЗОБРАЖЕННЯ

1.1. Актуальний стан задачі

Будь-яка інтелектуальна система аналізу та обробки візуальної інформації має на меті патернізацію зображення для отримання формалізованого опису його вмісту мовою математики, комп’ютера, а на основі перетвореного зображення можна виконувати спеціалізований аналіз. Маючи геометричний опис зображених об’єктів, потенціал аналізу візуальної інформації мігрує на принципово новий рівень глибини.

Аналіз зображення на основі кольорових даних значно обмежує глибину дослідження. Колір не є константним атрибутом об’єкту, в той час як форма об’єкту є доволі сталою.

Важливо відмітити, що векторизація зображення є нагальним кроком для наступних етапів більш високого порядку або вузької спеціалізації.

1.2. Основні поняття інтелектуальної обробки зображень

1.2.1. Цифрова обробка зображення

Цифрова обробка зображень (ЦОЗ) - це галузь використання цифрового комп’ютера з метою обробки цифрових зображень, застосовуючи до них певні алгоритми.

Будучи областю обробки цифрових сигналів, ЦОЗ має багато переваг над аналоговою обробкою зображень, оскільки дозволяє застосувати набагато ширший спектр алгоритмів до вхідних даних, надає гнучке налаштування параметрів, що дозволяє опрацювати зразок використовуючи повний арсенал галузі. При цьому замість оригіналу використовується його копія, тому завжди можна повернутись до початкової точки. Більше того, сучасні обчислювальні потужності часто дають змогу отримати перевагу в швидкості обробки [1-3].

Зародження галузі відбулося ще в середині 20-го століття, але тоді пропозиція перевищувала попит. Багато методів цифрової обробки зображень

були розроблені в 1960-их роках в лабораторії реактивного руху Белла в рамках Массачусетського технологічного університету. Основними задачами для вирішення були: аналіз супутникових зображень, перетворення стандартів дротових фотографій, обробка медичних зображень, розпізнавання символів та покращення фотографій. Рушійним фактором розвитку ЦОЗ стала популяризація телебачення [17].

Значна перепона реалізації попиту на галузь - складність обробки зображення через низьку потужність обчислювального обладнання епохи. Безупинне розширення телевізійної індустрії з часом почало вимагати можливість обробки зображення в режимі реального часу, що по суті заклало глобальний вектор як для розвитку науки, так і техніки з точки зору швидкості та ефективності роботи з графічним матеріалом. Для глобального поширення цього самого графічного контенту розвивалася глобальна мережа комунікацій - інтернет.

Сьогодні комп’ютери загального користування є достатньо потужними аби зникла потреба в спеціальному обладнанні за винятком масштабних або спеціалізованих задач. Сьогодні ЦОЗ - найпоширеніший інструмент обробки зображень через безкомпромісну універсальність в купі з низькими матеріальними витратами.

Таким чином основними факторами розвиток ЦОЗ є:

- розвиток комп'ютерів та графічних пристроїв;
- розвиток науки (здебільшого математики);
- попит на візуальний контент, його споживання, аналіз та обробку.

ЦОЗ вирішує низку повсякденних проблем у роботі з графікою:

- покращення якості;
- візуальна обробка;
- відновлення;
- шифрування;

- стиснення.

Типовими задачами інтелектуальної ЦОЗ є:

- класифікація;
- виділення ознак;
- обробки сигналів;
- розпізнавання образів.

1.2.2. Комп'ютерний зір

Комп'ютерний зір (КЗ) — це набір методів та технологій, що вирішує типові задачі ЦОЗ у поєднанні з елементами штучного інтелекту. КЗ дає змогу комп'ютерам "бачити" та "розуміти" інформацію візуального характеру [6-9, 13, 18, 23].

Основними прикладними задачами КЗ з точки зору аналізу візуальної інформації є:

- класифікація об'єктів;
- ідентифікація об'єкта;
- верифікація об'єкта;
- сегментація об'єкта;
- розпізнавання об'єктів.

Типові кроки обробки візуальної інформації для її подальшого аналізу є:

- попередня обробка (видалення шумів, гамма-корекція тощо);
- виділення атрибутів (меж об'єктів, певних кольорових зон тощо);
- сегментація (виокремлення істотних елементів для подальшого аналізу);
- спеціальна обробка згідно поставлених задач.

1.3. Характеристика типів зображення

1.3.1. Растрове зображення

Растрове зображення під “капотом” містить прямокутну матрицю пікселів, що представляє зображення.

Піксель - найдрібніша одиниця растру, містить в собі дані кольору. Відтінок формується з інтенсивності основних кольорів: червоного, зеленого, синього. В байтовому представленні має діапазон [0...255], що на виході дає

255 * 255 * 255 = 16.6 мільйонів кольорів.

Найпоширеніші растрові формати: PNG, GIF, TIF, BMP, JPG.

До переваг використання растра можна віднести:

- поширеність: растрова графіка є головним стандартом для візуалізації графічного матеріалу;
- невисока складність декодування вхідних даних;
- формат представлення графічного зображення у формі RGB-матриці є близьким до природнього.

Недоліки:

- якість зображення залежить від розподільної здатності (розміру RGB-матриці);
- пряма пропорція між величиною розподільної здатності до розміру файлу;
- наявність такого параметру як розподільна здатність обмежує роботу з зображенням до певної площі.

1.3.2. Векторне зображення

Векторне зображення - спосіб представлення графічних об'єктів у комп'ютерній графіці, що використовує математичний опис зображуваних об’єктів як елементарних геометричних примітив: точок, ліній, фігур, кривих, кіл, багатокутників тощо.

З точки зору математики об’єкт векторної графіки описується згідно правил аналітичної геометрії, до якого застосовуються певні атрибути. Таким чином векторний об’єкт повинен містити:

- координати опорних точок;
- параметри: вихідні координати, коефіцієнти масштабування, повороти, коефіцієнти розтягування по осях;
- атрибути: колір, тип лінії, товщина тощо.

До переваг векторної графіки належать:

- незалежність розміру файлу від розмірів зображуваних об’єктів;
- відсутність такої характеристики як “розподільна здатність”, тому будь-який растровий об’єкт може бути як завгодно масштабований;
- апаратно-незалежні одиниці виміру для опису векторних об’єктів для зручного рендерингу на будь-якому пристрої.

Недоліки:

- складність перетворення деяких об’єктів складної форми у вектори, оскільки таке перетворення може потребувати створення багатьох геометричних примітив;
- растеризація векторної графіки є доволі простим процесом на відміну від зворотного;
- трасування растру вимагає значної кількості цифрових обчислень;
- зазвичай має місце складна специфікація векторних форматів.

1.4. Постановка задачі

З огляду на недоліки растрового зображенні, що описані в розділі 1.3.1, векторне має на меті їх усунути. Векторне зображення також має свої мінуси, але вони здебільшого стосуються складнощів перетворення, в той час як переваги мають значнішу вагу.

Прикладна задача наукового дослідження - розробка системи векторизації растрових зображень. Для аналізу етапів перетворення растру в вектор алгоритми, що використовуються в ході перетворення, будуть застосовуватись до креслення, представленого растровим зображенням (Рис.1.1). Додатково буде приведено приклади векторизації різноманітних растрів, які продемонструють роботу системи, що буде розроблена, за умови складних вхідних зображень у Додатках 1 - 10.

Декомпозиція задачі має вигляд:

- реалізація алгоритмів;
- проектування програмного забезпечення;
- розробка програмного забезпечення;
- тестування програмного забезпечення.

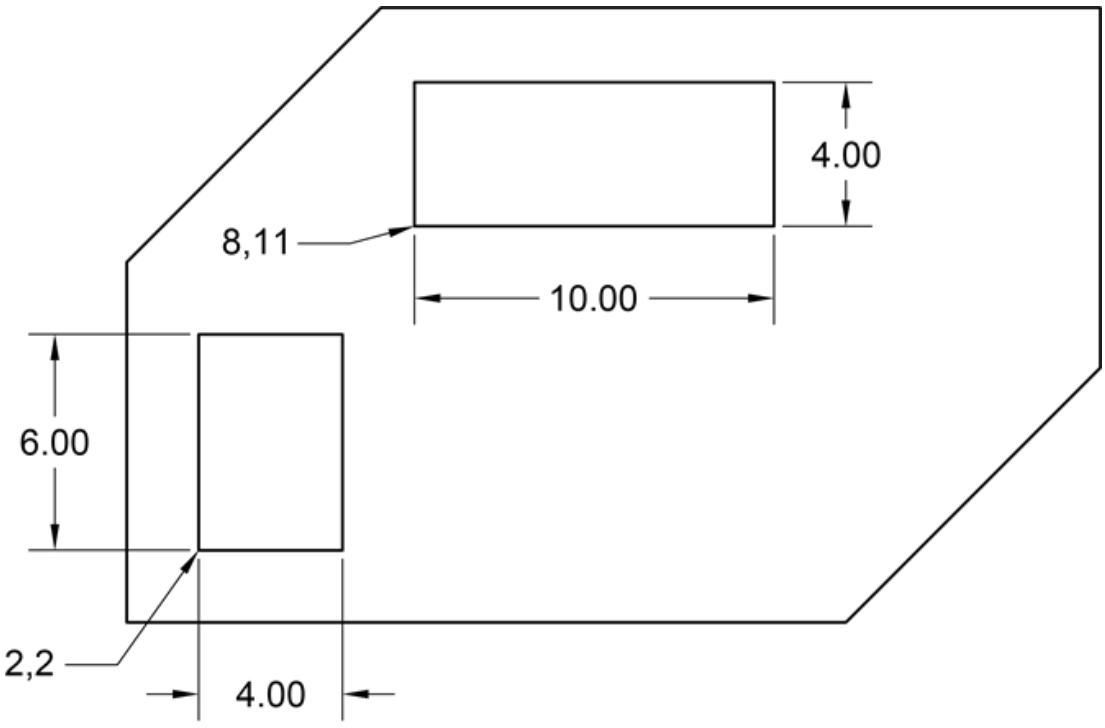


Рис. 1.1 – Креслення

1.5. Висновки до першого розділу

В першому розділі було проведено аналіз задачі векторизації растрового зображення, досліджено її актуальність, мету та цілі.

Розглянуто основні галузі комп’ютерних наук, що пропонують методи вирішення задачі. Було детально розглянуто основні поняття ЦОЗ, фактори, що дали поштовх розвитку галузі. Нерозривно з ЦОЗ нині існує більш сучасний та інтелектуальний напрям обробки зображення - комп’ютерний зір, основні поняття, ідеї та цілі якого також було досліджено.

Для розуміння об’єктів дослідження було приведено характеристику типів зображень, проаналізовано їх особливості.

В кінці розділу було сформульовано прикладну задачу, проведено її декомпозицію.

РОЗДІЛ 2
АНАЛІЗ АЛГОРИТМІВ ДЛЯ ВЕКТОРИЗАЦІЇ РАСТРОВОГО ЗОБРАЖЕННЯ

2.1. Перетворення кольорового растру в чорно-білий

Людське око сприймає кольори по різному через фізичні особливості світла. Дифракція світла різних за довжиною хвиль має свої особливості. Процес еволюції також вплинув на сприйняття світла оком, тому людина найкраще розрізняє зелений колір, оскільки первісний предок мав добре орієнтуватися в лісі. З огляду на це для коректного перетворення кольорового зображення на чорно-біле використовують коефіцієнти ваги піксельних компонент RGB. Таким чином формула отримання чорно-білого пікселя є

$$GRAYSCALE = RED * 0.3 + GREEN * 0.59 + BLUE * 0.11$$

2.2. Алгоритми виділення контурів

Застосування алгоритму виділення контурів (ВК) ставить за мету виявлення кривих, що окреслюють об’єкти в цифровому зображенні в зонах, де яскравість зображення різко змінюється. Виділення контурів - одна з основних технік ЦОЗ та комп'ютерного зору в задачах класифікації, виявлення ознак.

Одними з основних методів виділення контурів зображення є:

- оператор Собеля;
- оператор Прюітта;
- оператор Робертса;
- оператор Лапласа;
- алгоритм Кенні.

В результаті застосування алгоритму ВК отримується бінаризоване зображення. Бінаризація - процес перетворення кольорового зображення в чорно-біле, але таке, що містить лише два значення кольору: чорний, білий, представлені бітовими значеннями: 0 та 1.

2.2.1. Оператори Прюітта та Собеля

Оператори Прюітта та Собеля - дискретні диференційні оператори, що обчислюють наближені значення градієнта яскравості зображення. Результатом застосування операторів - отримання вектору градієнта яскравості в кожній точці зображення [10-16, 19, 20-22].

Дані оператори є абсолютно ідентичними з точки зору алгоритму реалізації. Відмінність полягає в матричних масках згортання, які ще називають зернами або ядрами, що застосовуються до вхідного зображення. Таким чином отримані результати застосування операторів є доволі близькими за своїм рівнем чіткості, контрастності [21].

Матриці згортання Прюітта G_x та G_y мають вигляд

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad Gy = \begin{bmatrix} -1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & 1 & 1 \end{bmatrix}$$

Матриці згортання Собеля G_x - по осі X, G_y – по Y мають вигляд

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad Gy = \begin{bmatrix} -1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & 1 \end{bmatrix}$$

Алгоритм реалізації операторів передбачає низку кроків:

1) Перетворення вхідного растру на чорно-білий.

2) Згортання вхідної матриці чорно-білих пікселів $g(x, y)$ по осях X та Y. Алгоритм згортання реалізується згідно формули

$$g(x, y) = w \cdot f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx, dy)\ f(x - dx, y - dy);$$

$$-a \le dx \le a \quad ma \quad -b \le dy \le b.$$

де $g(x,y)$ – відфільтроване зображення, $f(x,y)$ – вхідне зображення, w - матриця згортання.

Так для матриці $M_x(m,n)$ та зерна $w_y(m,n)$ згорнута вихідна матриця буде розрахована за формулою

$$M_x(m,n)\ \cdot\ w_y(m,n)\ = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} x_{(m-i)(n-j)}\, y_{(1+i)(1+j)}$$

Наприклад, для умовного зображення *M(5,5)* та зерна Собеля *w(3,3)* буде перемножено 9 зрізів матриці *M* на зерно *w* (Рис. 1.2).

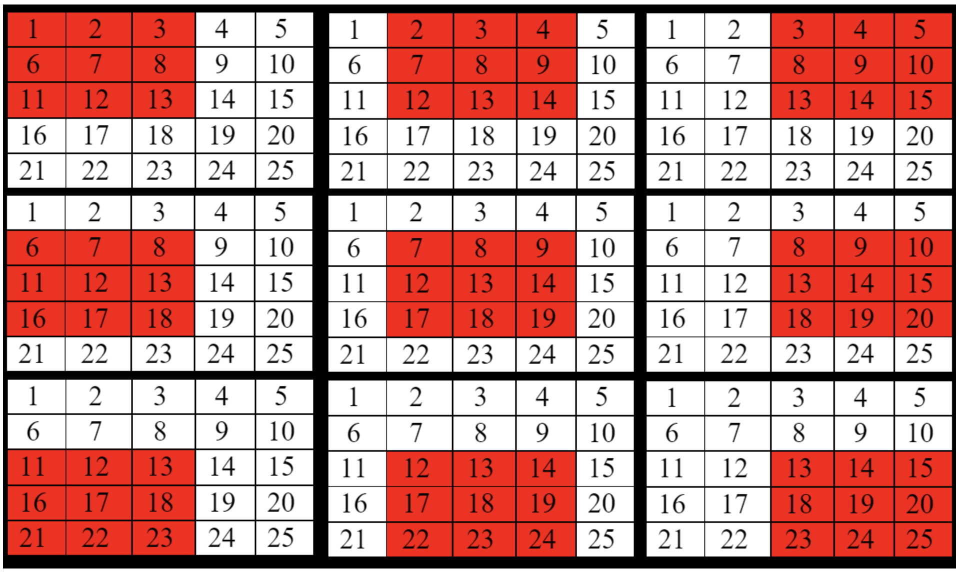


Рис. 2.1 - Зрізи матриці *M(5,5)* під час згортки зерном Собеля

3) Розрахунок матриці градієнтів *G*, що передбачає поелементний квадрат суми матриць G_x , G_y згорнутих по осях *X* та *Y* вхідних матриць зображання, за формулою

$$G\ =\ \sqrt{G_x^2\ +\ G_x^2}\ .$$

4) Згідно отриманої матриці градієнтів G можна отримати напрям градієнту

$$theta = atan2(G_x , Gy).$$

Створивши растр, піксельною матрицею якого є матриця градієнтів, буде отримано бінаризоване зображення контурів. Приклади застосування операторів до растрового креслення (Рис. 1.1) буде приведено в розділі 2.2.3.

2.2.2. Алгоритм Кенні

Алгоритм Кенні є найбільш просунутим серед перерахованих алгоритмів виділення контурів зображення, тому і найскладнішим у реалізації [10 - 12]. Перед застосуванням даного алгоритму зазвичай також перетворюють

зображення на чорно-біле аби зменшити обчислювальні витрати за рахунок зниження контрасту гамми. Реалізація алгоритму складається з наступних етапів:

1) Згладжування зображення - його розмиття з метою видалення шуму, використовуючи зерно Гауса, що параметризується значеннями розмірності та *sigma*.

Зерно Гауса для розмірності *(x, y)* розраховується за формулою

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

$$1 \leq i, j \leq (2k+1)$$

де σ - *sigma*.

Чим більші значення параметрів розміру та *sigma* зерна Гауса, тим значніше розмиття, тому підбір параметрів залежить від зображення. Для контрастного зображення не потрібні великі значення параметрів і навпаки, більший розмір матриці спричинює більше розмиття зображення для утворення рівномірно розподіленої гамми [12, 16, 22].

Для застосування зерна Гауса також використовується алгоритм згортання матриці

$$g(x,y) = w \cdot f(x,y) = \sum_{dx=-a}^{a} \sum_{dy=-b}^{b} w(dx,dy) f(x-dx, y-dy).$$

2) Отримання матриці градієнтів, застосувавши один із алгоритмів виділення контурів: Собеля, Прюітта, Кенні, Робертса, Лапласа тощо.

3) Розрахунок кута градієнта за формулою

$$theta = atan2(G_x, Gy).$$

4) Згортка немаксимальних точок контурів - лише локальні максимуми позначаються як межі згідно напрямів кута градієнта (Рис. 2.2).

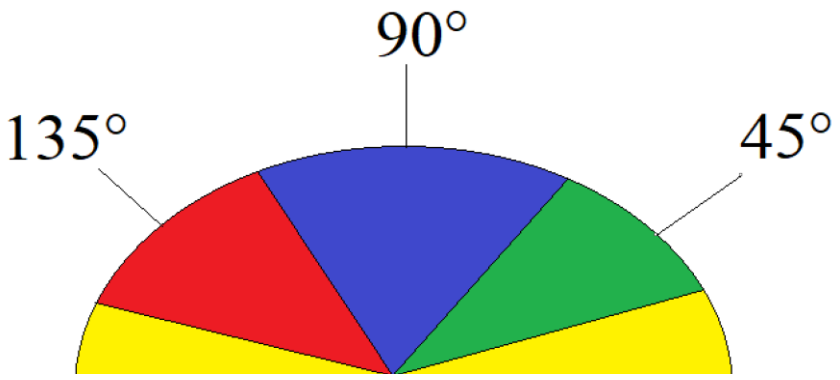


Рис. 2.2 - Кут напряму градієнта

Існує 4 умови згортки локальних немаксимумів:

- Якщо *theta = 0°* , контур знаходиться в північному чи південному напрямах, то точка є контурною, якщо її градієнт більший ніж у сусіда в східному та західному напрямах.
- Якщо *theta = 45°* , контур знаходиться в північно-західному та південно-східному напрямах, то точка є контурною, якщо її градієнт більший ніж у сусіда в східному та південно-західному напрямах.
- Якщо *theta = 90°* , контур знаходиться в східному чи західному напрямах, то точка є контурною, якщо її градієнт більший ніж у сусіда в північному та південному напрямах.
- Якщо *theta = 135°* , контур знаходиться в схід-південний захід, то точка є контурною, якщо її градієнт більший ніж у сусіда в північно-західному та південно-східному напрямах.

5) Подвійна контурна фільтрація для пошуку потенційних точок-контурів згідно встановлених для алгоритму порогових значень градієнтів: *MaxG*, *MinG* в діапазоні від 0 до 255. На вхід отримуються матриця *M*, що пройшла етап згортки локальних немаксимумів.

Умови фільтрації для кожного елементу *M*, що буде позначено як *p*:

- якщо $p \geq MaxG$, то $p = 1$, є сильним контуром;
- якщо $p \geq MinG$ та $p < MaxG$, то $p = 2$, є слабким контуром;

Порогові значення визначаються емпірично, їх величина залежить від гамми вхідного зображення. Для контрастних зображень має місце менша

амплітуда порогів, ніж для більш рівномірних зображень з точки зору гамми.

6) Трасування неоднозначних пікселів, визначення їх меж шляхом придушення всіх країв, які пов'язані з певними сильними контурами.

Задля отримання максимально чітких тонких контурів потрібно опрацювати слабкі контурні зони з метою мінімізації шумів. Зазвичай слабкий контурний піксель, що належить дійсному контуру, має зв’язок із сильним контурним пікселем, на відміну від слабких контурних пікселів, що на справді є кольоровим шумом.

Для реалізації даної ідеї пошуку сильного контурного пікселя потрібно:

- виконати перебір сусідства пікселів за кожним із 8 напрямів;
- якщо в сусідстві міститься сильний контурний піксель, то його слід зберегти, інакше присвоїти йому значення 0.

Приклад застосування алгоритму Кенні до растрового зображення буде приведено в розділі 2.2.3. Результат застосування алгоритму до креслення (Рис. 1.1) буде порівно зі зображеннями контурів, отриманих шляхом застосування більш простих в реалізації операторів Прюітта та Собеля.

2.2.3. Порівняння розглянутих алгоритмів виділення контурів

У даному розділі буде проведено порівняльний аналіз якості вихідних зображень контурів, отриманих шляхом застосування описаних алгоритмів виділення контурів в розділах 2.2.1 та 2.2.2 Вхідним зображенням є креслення з Рис.1.1.

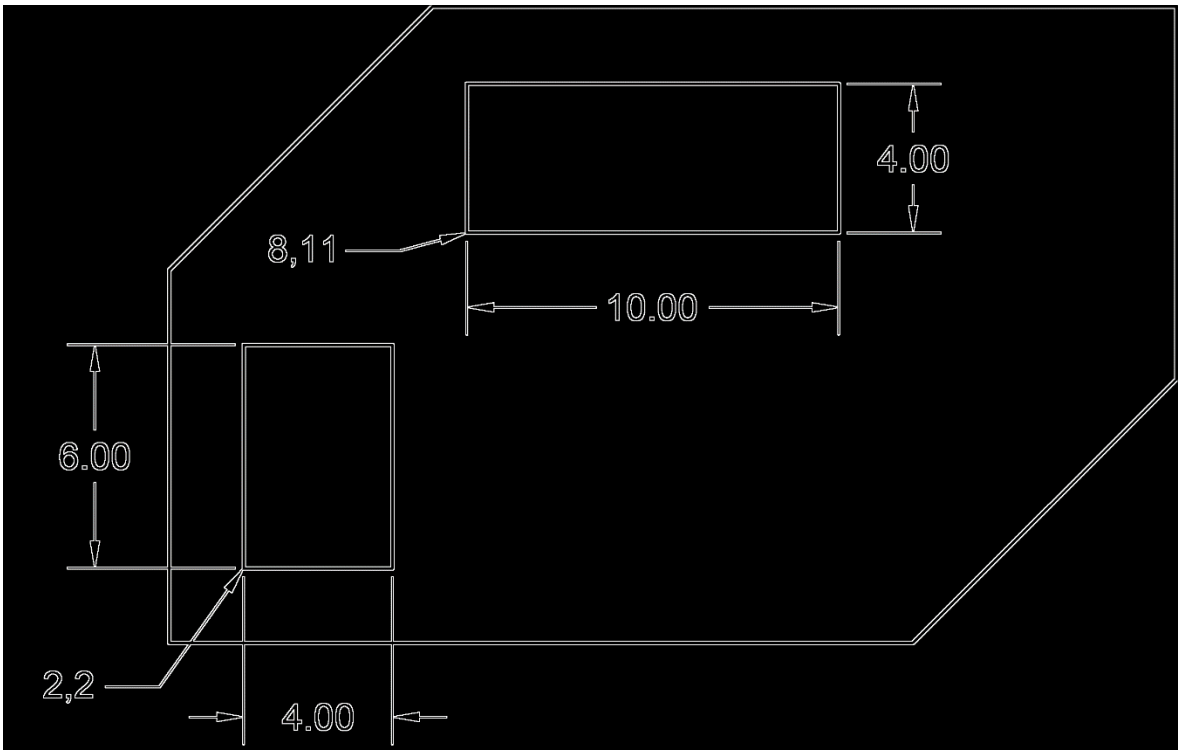


Рис. 2.3 - Застосування оператору Прюітта

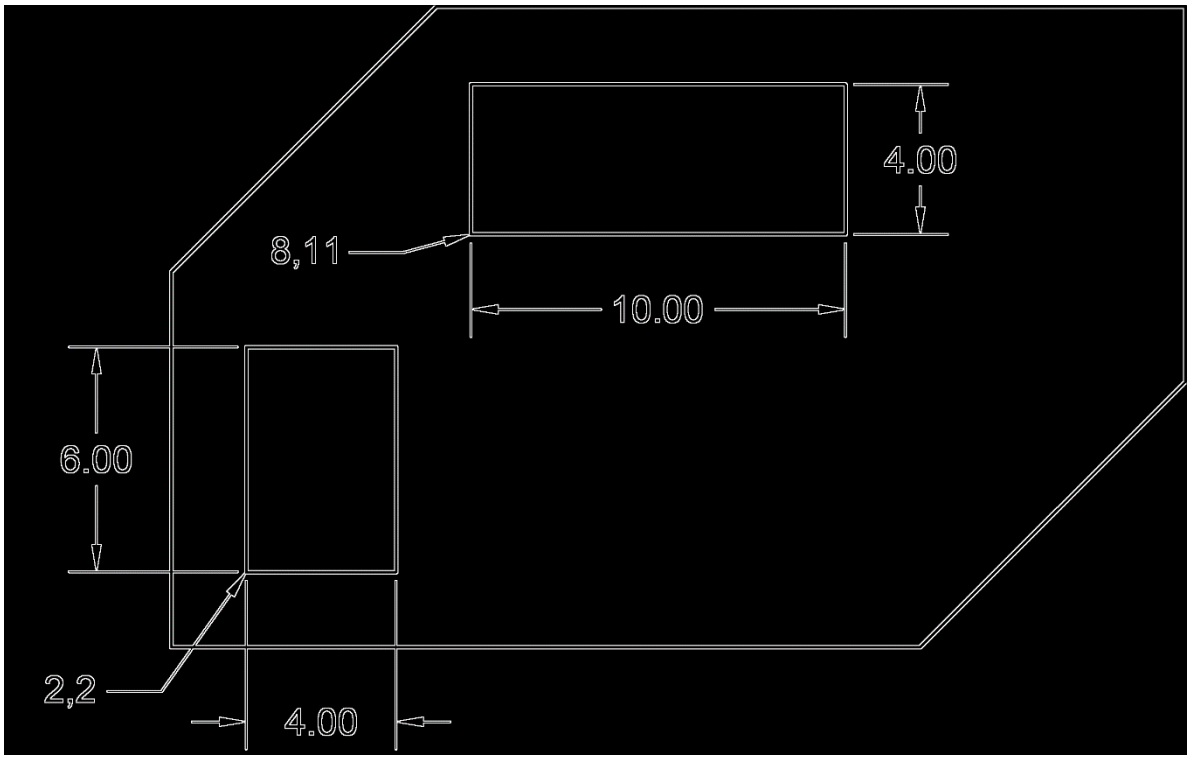


Рис. 2.4 - Застосування оператору Собеля

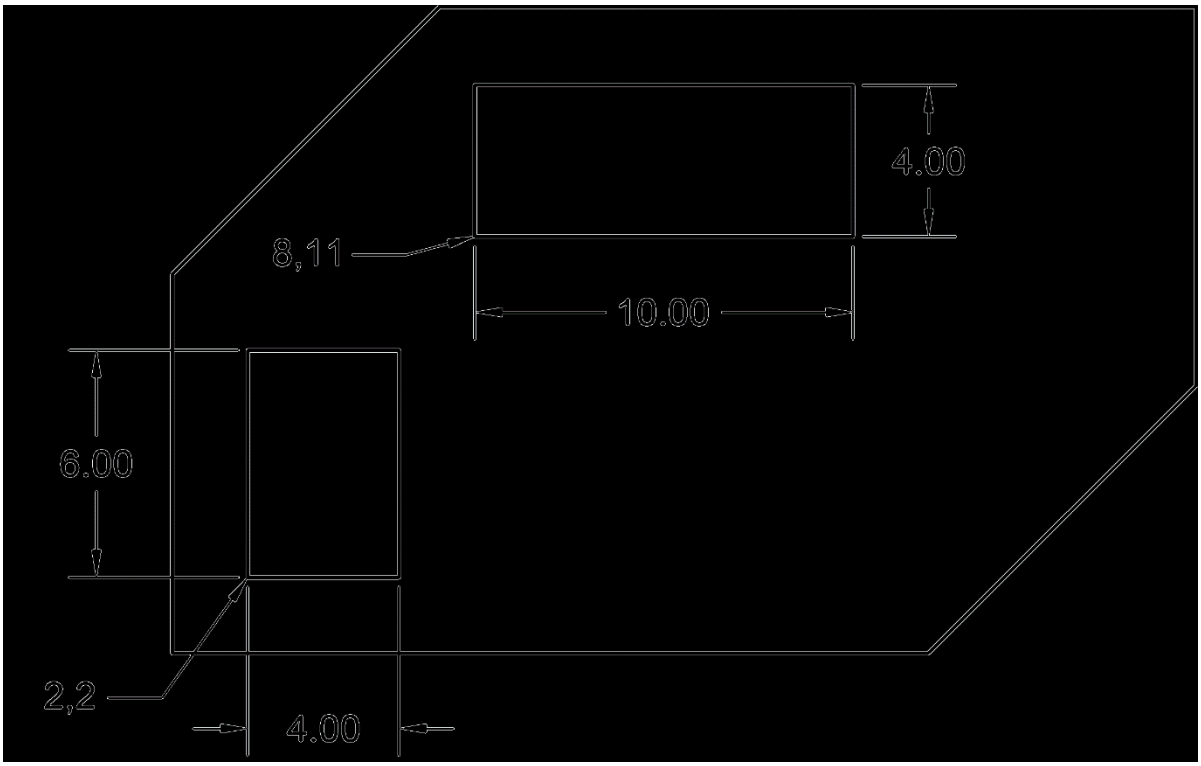


Рис. 2.5 - Застосування алгоритму Кенні

При детальному аналізі отриманих зображень контурів можна зробити висновок, що застосування алгоритму Кенні дає найвищу якість результату: вихідне зображення контурів містить найчіткіші та найтонші контури об’єктів. Така тенденція зберігається для будь-якого вхідного зображення. За умови влучного підбору параметрів алгоритму Кенні вихідне зображення контурів матиме ще більш вагомий якісний відрив.

Перевага в ефективності оператора Кенні досягається завдяки наявності низки просунутих етапів обробки вхідних даних. Більше того, даний алгоритм використовує застосування простих операторів ВК як проміжний етап. Отримані зображення контурів через застосування операторів Собеля та Прюітта навіть за умови накладання фільтра Гауса для розмиття гамми не дають близьких за якістю результатів.

2.3. Алгоритм Хафа

Алгоритм Хафа - обчислювальний алгоритм для параметричної ідентифікації геометричних елементів растрового зображення, запатентований у

1962 році Полем Хафом. Використовується для аналізу зображень методами цифрової обробки та комп'ютерного зору. Призначений для вирішення типової задачі ідентифікації форми: пошуку об'єктів, що належать певному класу фігур з використанням процедури голосування [24-27, 30].

Простір Хафа — це двовимірна площина, яка має горизонтальну вісь нахилу і вертикальну вісь, що представляє перетин лінії на зображенні контурів.

Пряма може бути задана рівнянням $y = ax + b$, обчислюється за будь-якою парою точок (x, y). Алгоритм Хафа оперує лініями, які описані в термінах її параметрів, тобто a - кутового коефіцієнта і b - точки перетину з віссю ординат. З цього випливає, що пряма, задана рівнянням $y = ax + b$, може бути представлена у вигляді точки з координатами (a, b) у просторі параметрів.

Лінійна трансформація Хафа передбачає створення двовимірного масиву акумулятора, оскільки є два невідомі параметри: a і b. Два виміри акумулятора відповідають значенням параметрів a та b. Для кожної точки та її сусідів алгоритм визначає, чи достатня вага контуру у цій точці. Якщо так, то алгоритм обчислює параметри прямої і збільшує значення в акумулятора для даної координати.

В термінах алгоритму Хафа використовують параметри *rho* і *theta* замість a і b. Означення параметрів:

- *rho* - довжина радіус-вектора найближчої до початку координат точки на прямій, нормалі до прямої, проведеної з початку координат;
- theta - кут між цим вектором та віссю X.

Рівняння прямої з полярними координатами або параметричної прямої має вигляд

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

$$rho = x * \cos(theta) + y * \sin(theta).$$

Алгоритм реалізації перетворення Хафа складається з кроків:

1) Застосування алгоритму виділення контурів до вхідного растрового зображення (оператори Собеля, Прюітта, Кенні).

2) Визначення діапазону *rho* і *theta* для вхідного зоброження. Зазвичай діапазон *theta* становить *[0, 180]* градусів, а rho - *[-d, d]*, де *d* — довжина діагоналі вхідного зображення контурів. Важливо квантувати діапазон *rho* і *theta*, тобто має бути скінченна кількість можливих значень.

3) Створення нульової двовимірної матриці "акумулятора" розміру (N_{rho}, N_{theta}), де N_{rho} - діапазон *rho*, N_{theta} - діапазон *theta*.

4) Для кожного пікселя зображення перевіряється, чи є він контурним. Якщо так, то потрібно:

- перебрати всі можливі значення *theta*;
- обчислити відповідні значення *rho* ;
- знайти індекси *theta* і *rho* в акумуляторі та збільште значення акумулятора на цих парах індексів.

5) Перебір всіх значень в акумуляторі. Для значень більших за певний поріг, отримується індекс *rho* і *theta*. Значення *rho* і *theta* з пари індексів можна перетворити назад у форму $y = ax + b$.

Даний алгоритм є дуже чутливим до якості вхідного зображення контурів, тому попередня обробка зображення, а саме якісне виділення контурів, є дуже важливим етапом. Серед розглянутих алгоритмів виділення контурів оператор Кенні - найбільш досконалий представник.

Важливо відмітити, що перетворення Хафа не завжди дає бажану точність ідентифікації прямих, оскільки пороговий підхід до виявлення сильних ліній не дає змогу оцінити якісні показники лінії, лише кількісні.

Сильною лінією вважається така, що перетинає багато контурних точок зображення, але це на завжди є вірно. Об'єкти зображення можуть знаходитись близько один до одного, на одному рівні чи взагалі одній площині, при цьому не бути цілим. Кривизна прямої також знижує точність її ідентифікації.

З точки зору обчислювальних потужностей, доволі вимогливим є величина крок *theta*, яка підвищує якість трасування растрів великого розміру. Окільки даний параметр впливає на розмір акумулятора, зростає споживання ресурсів заліза через масивні об’єми виділеної пам’яті та наявність великої кількості арифметичних операцій.

Рис. 2.6 ілюструє застосування алгоритму Хафа до креслення (Рис. 1.1). Отримані лінії описані параметрично, а не лінійно, тому простираються по всій площі растра. Точна ідентифікація окремих сутностей зображення є дещо ускладненою. Зважаючи на те, що ми бачили оригінальне зображення креслення, неозброєним оком видно, що результати є близькими до істини. Отримані лінії дійсно мають місце бути, оскільки облямовують існуючі елементи креслення. Прослідковуються відповідність форми та розміщення елементів креслення.

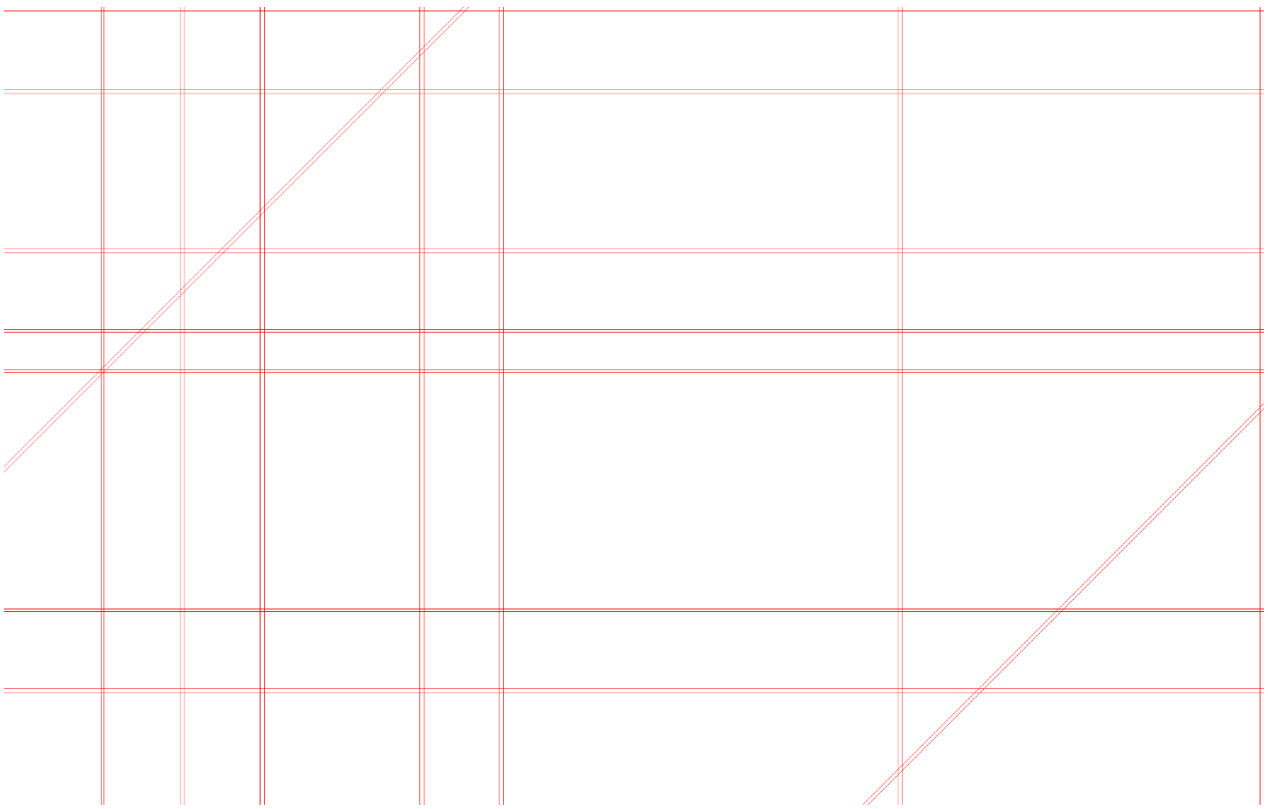


Рис. 2.6 - Застосування перетворення Хафа

2.4. Вдосконалений алгоритм Хафа

Даний розділ має на меті висвітлення деталей розробки власного підходу до перетворення нескінченних параметричних ліній в просторі Хафа у чіткі вектори, що відповідають лініям форм зображуваних об'єктів [25].

Згідно алгоритму Хафа, сильна лінія - лінія, що перетинає багато контурних точок. Будучі описаною параметрично, потрібно виявити її реальні межі або взагалі її сегментувати згідно контуру об'єкту, який вона облямовує. Наразі лінії Хафа обмежені розміром растру.

Таким чином для пошуку кінцевих ліній Хафа, перетворення їх у вектори, було винайдено наступний підхід, що базується на поняттях топологічного простору та лінійної алгебри:

1) Проведення перетворення Хафа для вхідного зображення, деталі реалізації якого було розглянуто в розділі 2.3.

2) Побудова асоціативного масиву, де множина ключів - отримані лінії Хафа, множина значень - масиви приналежних контурних точок.

Умова приналежності точки до лінії - пороговий параметр максимальної взаємної відстані *deviation*. Відстань від точки до лінії представляє собою перпендикуляр, проведений з точки до лінії. Для лінії *l(rho, theta)* в полярних координатах відстань до точки *p(x, y)* можна знайти за формулою

$$|d| = |x \cdot \cos(theta) + y \cdot \sin(theta) - rho|.$$

Вважається, що $p \in l$, якщо $d = 0$, але тонкощі трасування растру вимагають введення параметру відхилення *deviation* для того, щоб наявність мінімальної кривизни лінії або недостатність кроку theta в ході перетворення Хафа не спотворили результати, таким чином $p \in l$, якщо $d \leq deviation$.

Оптимальним значенням *deviation* є $\sqrt{2}$, оскільки являє собою гіпотенузу прямокутного трикутника з катетами 1 та 1, є доволі гармонічним числом для вирішення задачі приналежності точки до лінії, що має деяку кривизну, в рамках растрової матриці. У растрі відстань між пікселями становить 1, а величина гіпотенузи - відстань до сусіда по діагоналі. Величина *deviation* є змінною, тому

згідно отриманих результатів цей поріг можна коригувати.

3) Побудова асоціативного масиву, де множина ключів - отримані лінії Хафа, множина значень - масиви масивів (кластерів) приналежних контурних точок. Для кожного ключа потрібно провести кластеризацію масиву значень.

Для забезпечення кластеризації можна використати метод симплиціальних комплеків, для чого необхідне введення параметру *gap* - максимальної взаємної відстані для точок, що належать лінії. Параметр *gap* є змінним, тому підлягає коригуванню в залежності від зображуваних растром об'єктів, їх композиції.

Перед проведенням кластеризації масив точок рекомендується відсортувати за збільшенням відстані до початку координат. Для відсортованого масиву точок перевірити їх взаємне топологічне розташування задля отримання кластерів згідно умови: якщо відстань між точками менша за величину *gap*, то вони належать одному кластеру, інакше різним.

Отримані кластери фактично представляють собою множини точок кінцевих ліній, векторів. Оскільки масив точок був раніше відсортований, то в рамках кластерів, опорними точками векторів є дві найбільш взаємно віддалені, тобто перша та остання точки, що увійшли в кластер.

2.5. Висновки до першого розділу

В ході ознайомлення з розділом читач має змогу ознайомитись з аналізом поширених методів попередньої обробки зображень - алгоритмами виділення контурів растрового зображення. Для розглянутих алгоритмів наявний детальний опис, приведено переваги та недоліки.

В заключній частині розділу розглянуто алгоритм обробки більш високого порядку - алгоритми ідентифікації форми об'єкта. Перетворення Хафа є потужним інструментом для пошуку ліній в зображенні. Розглянуто його тонкощі та обмеження, розроблено власний підхід до його вдосконалення.

РОЗДІЛ 3
РОЗРОБКА СИСТЕМИ ВЕКТОРИЗАЦІЇ РАСТРОВИХ ЗОБРАЖЕНЬ

Даний розділ містить детальний опис процесу розробки системи векторизації растрових зображень. В кінці розділу представлені результати тестування розробленого програмного забезпечення, проведено аналіз результатів.

3.1. Опис програмного забезпечення

Для розробки програмного забезпечення були використані наступні інструменти та технології:

- Операційна система: Windows 11;
- Мова програмування: C#;
- Середовище розробки: Visual Studio 2022;
- Цільова платформа: crossplatform;
- Зовнішні бібліотеки: відсутні;
- Вхідні дані: растрове зображення формату BMP, PNG, GIF, TIF, JPEG;
- Вихідні дані: векторне зображення формату EMF та його растрове представлення.

Розроблене ПЗ за належить до типу “бібліотека класів” або “клієнтський API”. Це легкий та потужний інструмент, який може бути використаний як зовнішнє програмне забезпечення в будь-якому проекті.

Оскільки створена бібліотека класів буде використовуватись як “чорний ящик”, тобто користувач не матиме доступу до вмісту доступних функцій, було також розроблено клієнтську документації до наявного функціоналу. Це надає змогу швидко орієнтуватися в наявному функціоналі користувачам, дослідникам, що не мають глибоких знань в областях цифрової обробки зображення, комп’ютерного зору.

Використана архітектурна концепція - об’єктно орієнтоване програмування. До переваг використання архітектури ООП належить:

- зручне масштабування проекту через наявність функціональних інтерфейсів поведінки;
- незалежність компонентів системи.

Розроблене ПЗ дозволяє застосовувати до вхідних даних як лише алгоритми ВК, так і алгоритм пошуку ліній, що виконує ВК в ході роботи. Загалом усі алгоритми ВК та пошуку ліній реалізують спеціальні абстрактні класи, тому деталі реалізації не впливають на поведінку конкретного екземпляру. Для масштабування каталогу алгоритмів, новий екземпляр має наслідувати відповідний абстрактний клас для входу в архітектуру бібліотеки класів. Таким чином існує два головні абстрактних класи:

- для виділення контурів;
- для пошуку ліній на зображенні контурів.

Реалізовані алгоритми обробки зображення:

- алгоритми виділення контурів Собеля, Прюітта, Кенні;
- алгоритм генерації зерна Гауса згідно параметрів розміру та *sigma*.
- алгоритми пошуку ліній: перетворення Хафа, вдосконалений алгоритм пошуку кінцевих ліній на основі алгоритму Хафа.

3.2. Архітектура програмного забезпечення

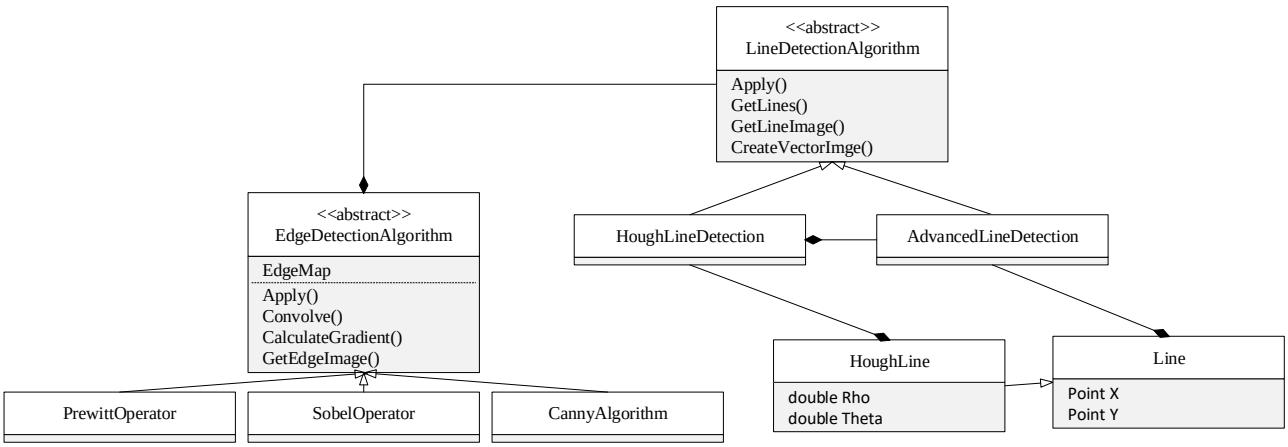


Рис. 3.1 – Архітектура програмного забезпечення

Для застосування алгоритму виділення контурів було розроблено абстрактний клас EdgeDetectionAlgoritm, що містить низку загальних методів для всіх алгоритмів ВК, як наприклад метод, що реалізує алгоритм згортання вхідного растру певним зерном та метод розрахунку градієнтів. Також містить публічні віртуальні методи та члени для нащадків:

- EdgeMap – поле типу двовимірна матриця, що зберігає контурні точки;
- Apply(path/stream) – метод для застосування алгоритму ВК до вказаного файлу;
- GetEdgeImage() – метод для отримання зображення контурів.

Для застосування алгоритму виділення контурів було розроблено абстрактний клас LineDetectionAlgoritm. Містить публічні віртуальні методи для нащадків:

- Apply(path/stream) – метод для застосування алгоритму пошуку ліній до вказаного файлу;
- GetLines(threshold) – отримання масиву ліній згідно порогового значення;
- GetLineImage(threshold) – отримання зображення ліній об’єкта згідно порогового значення;
- CreateVectorImage(threshold) – отримання векторного зображення ліній об’єкта згідно порогового значення.

3.3. Тестування програмного забезпечення

В даному розділі буде проведено тестування розробленого програмного забезпечення з метою вирішення прикладної задачі: векторизації растрового зображення креслення (Рис.1.1). До креслення буде застосовано вдосконалене перетворення Хафа, розглянуте в розділі 2.4. Використані алгоритми будуть оптимально параметризовані, встановлені значення було знайдено емпіричним шляхом.

В Додатках 1 - 10 буде приведено приклади векторизації різноманітних

растрів, що демонструють якість роботи створеної системи для різноманітних примірників вхідних растрів, що містять складніші зображення, ніж креслення з Рис. 1.1, яке є доцільним екземпляром для покрокового аналізу роботи алгоритмів комп'ютерного зору. Більше того, векторизація креслень є доволі широко розповсюдженою задачею.

В ході застосування перетворення Хафа буде використовуватись оператор Кенні, параметризація якого має вигляд *Canny(size, sigma, min, max)*:

- *size, sigma* - розмір зерна Гауса та величина *sigma*;
- *min, max* - порогові значення градієнту.

Для отримання ліній Хафа зазвичай використовують пороговий параметр фільтрації *threshold*, який набуває значень від 0 до 100. Даний діапазон представляє собою відносну інтенсивність ліній:

- *threshold = 0* - найслабша лінія, така, що перетинає мінімум контурних точок, тобто отримуються всі наявні лінії Хафа
- *threshold = 100* - найінтенсивніша серед усіх знайдених, максимальна фільтрація.

Параметризація вдосконаленого алгоритму пошуку ліній має вигляд *AdvancedHough(deviation, gap)*.

Тест №1

Вхідні дані: креслення (Рис. 1.1).

Параметри:

- *Canny(3, 1.5, 10, 25)*
- *Hough(15)*
- *AdvancedHough(√2 , 2)*

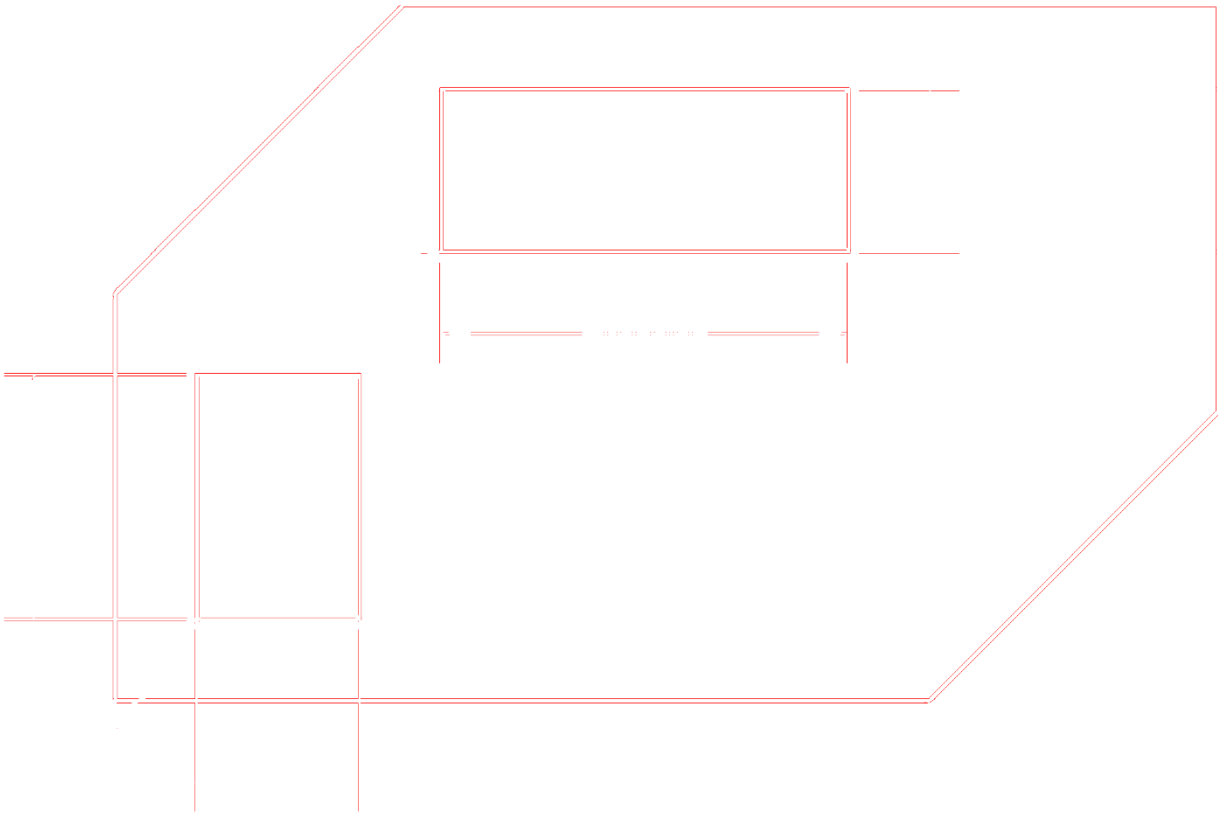


Рис. 3.2 - Векторизоване креслення

Аналіз результатів:

Рис. 3.2 на відміну від Рис. 2.6 ілюструє чітко-виділені основні елементи зображення креслення, прослідковується наявність геометричних примітив, що облямовують форми. Таким чином можна побудувати векторне зображення, яке складається з чітко-описаних геометричних примітив.

Оскільки *threshold = 15*, не всі необхідні лінії було отримано та трансформовано, що виливається в нестачу низки наявних елементів креслення.

Тест №2

Вхідні дані: креслення (Рис. 1.1).

Параметри:

- *Canny(3, 1.5, 10, 25)*
- *Hough(0)*
- *AdvancedHough(√2 , 2)*

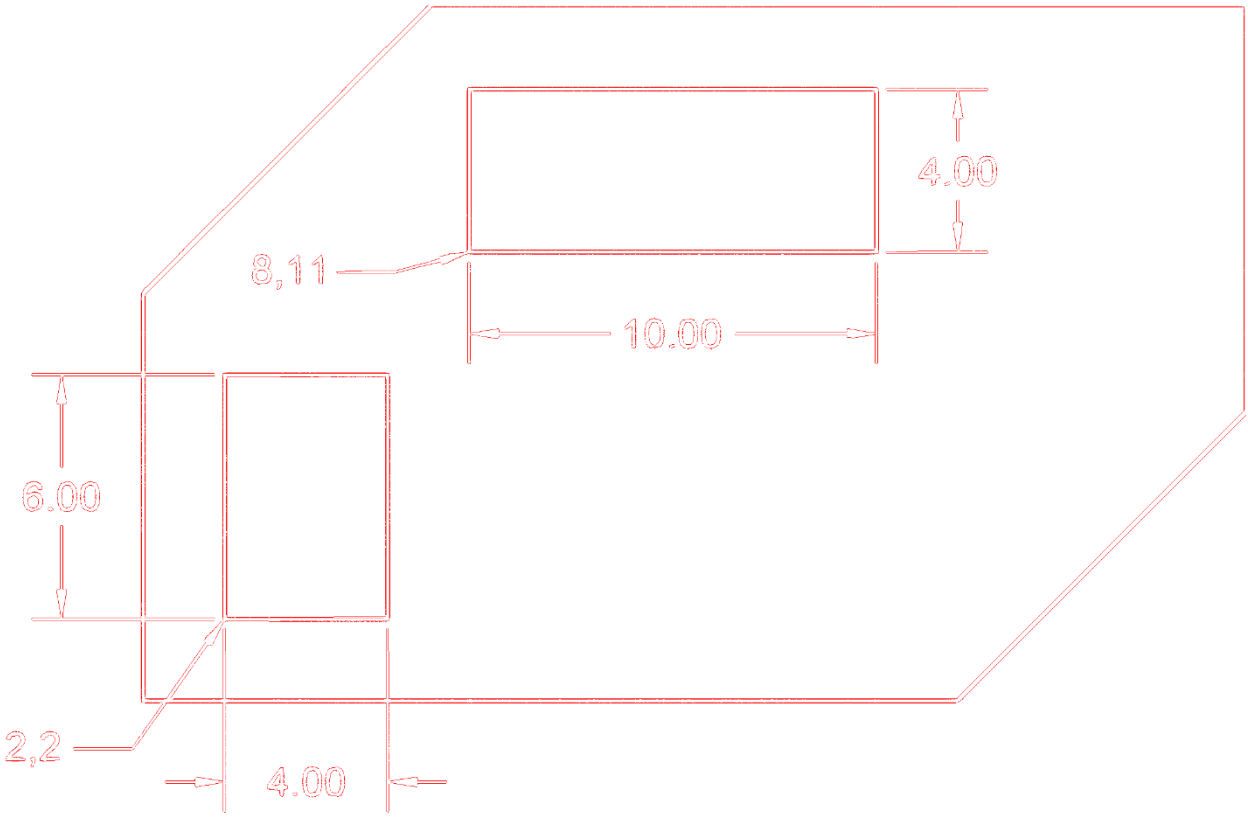


Рис. 3.3 - Векторизоване креслення

Аналіз результатів:

Як було описані в розділі 1.3.2, іноді векторизація складних за формою об’єктів вимагає створення багатьох геометричних примітив. Рис. 3.2 не містить низки елементів, зображених кресленням. В той же час Рис. 3.3 ілюструє всі елементи зображення креслення, оскільки було встановлено мінімальне значення параметру *threshold*, що дає змогу отримати всі лінії Хафа, які було успішно сегментовано в кінцеві лінії – вектори, але кількість складових ліній значно збільшилась.

3.4. Висновки до третього розділу

В даному розділі було розглянуто процес розробки системи векторизації растрових зображень, описано її архітектуру, особливості та функціонал. В кінці розділу наведено результати тестування системи, проведено їх аналіз.

ВИСНОВОК

В ході проведення науково-дослідної роботи було досліджено методи цифрової обробки зображень та комп’ютерного зору з метою розробки системи векторизації растрових зображень.

Для виконання поставленої задачі було досліджено та реалізовано алгоритми виділення контурів зображення та пошуку ліній. В ході реалізації алгоритму Хафа для пошуку ліній було виділено його недоліки, розроблено власний підхід його вдосконалення.

Слідом за виконанням теоретичної частини роботи, було розпочато проектування та розробку програмного забезпечення для векторизації растрових зображень, проведено його тестування, проаналізовано результати.

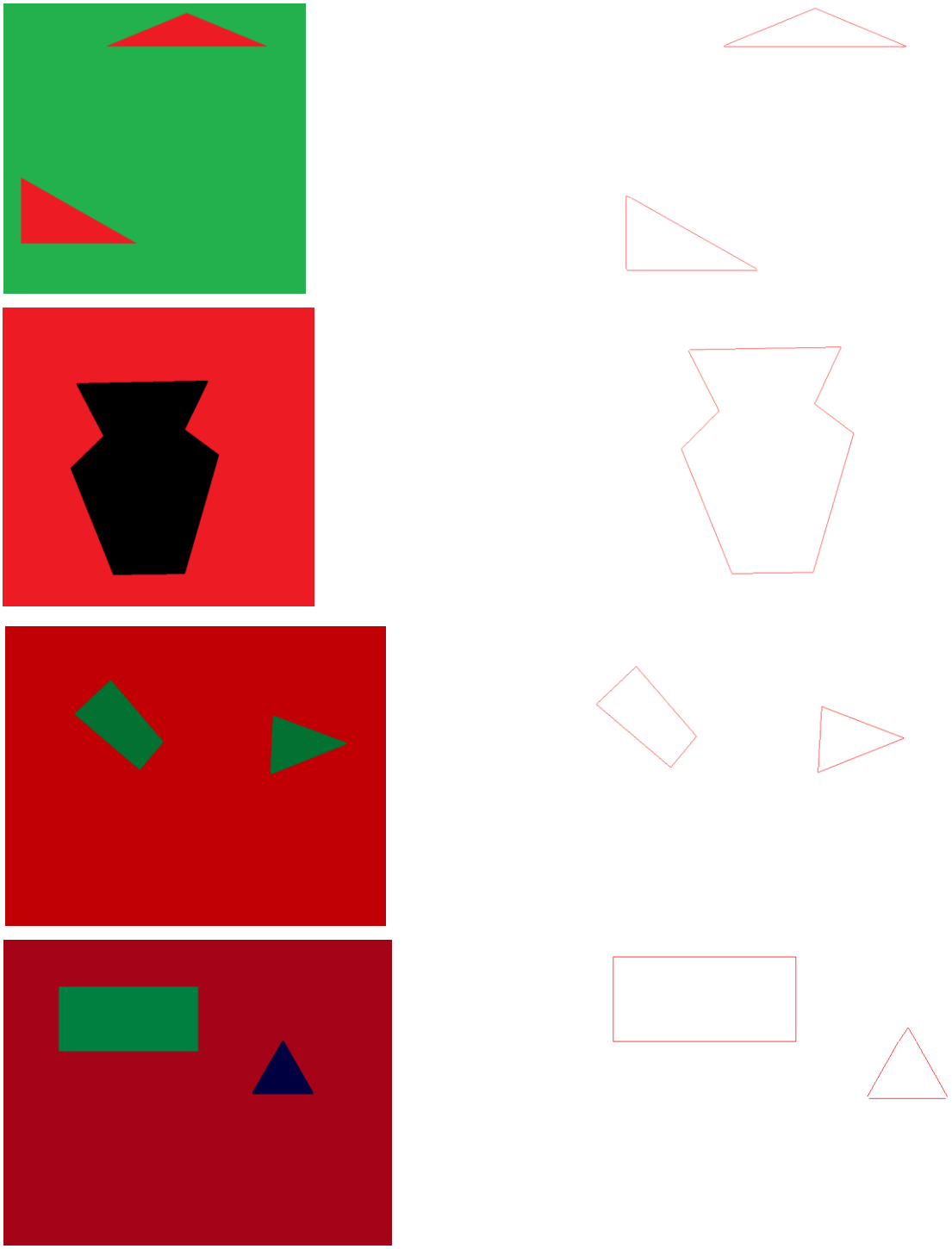
Розроблене програмне забезпечення має потенціал до масштабування через наявність гнучкої об’єктно орієнтованої архітектури. Наразі є можливість векторизації растру через пошук ліній, хоча існують варіації того ж алгоритму Хафа для пошуку більш складних геометричних примітив: прямокутників, еліпсів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

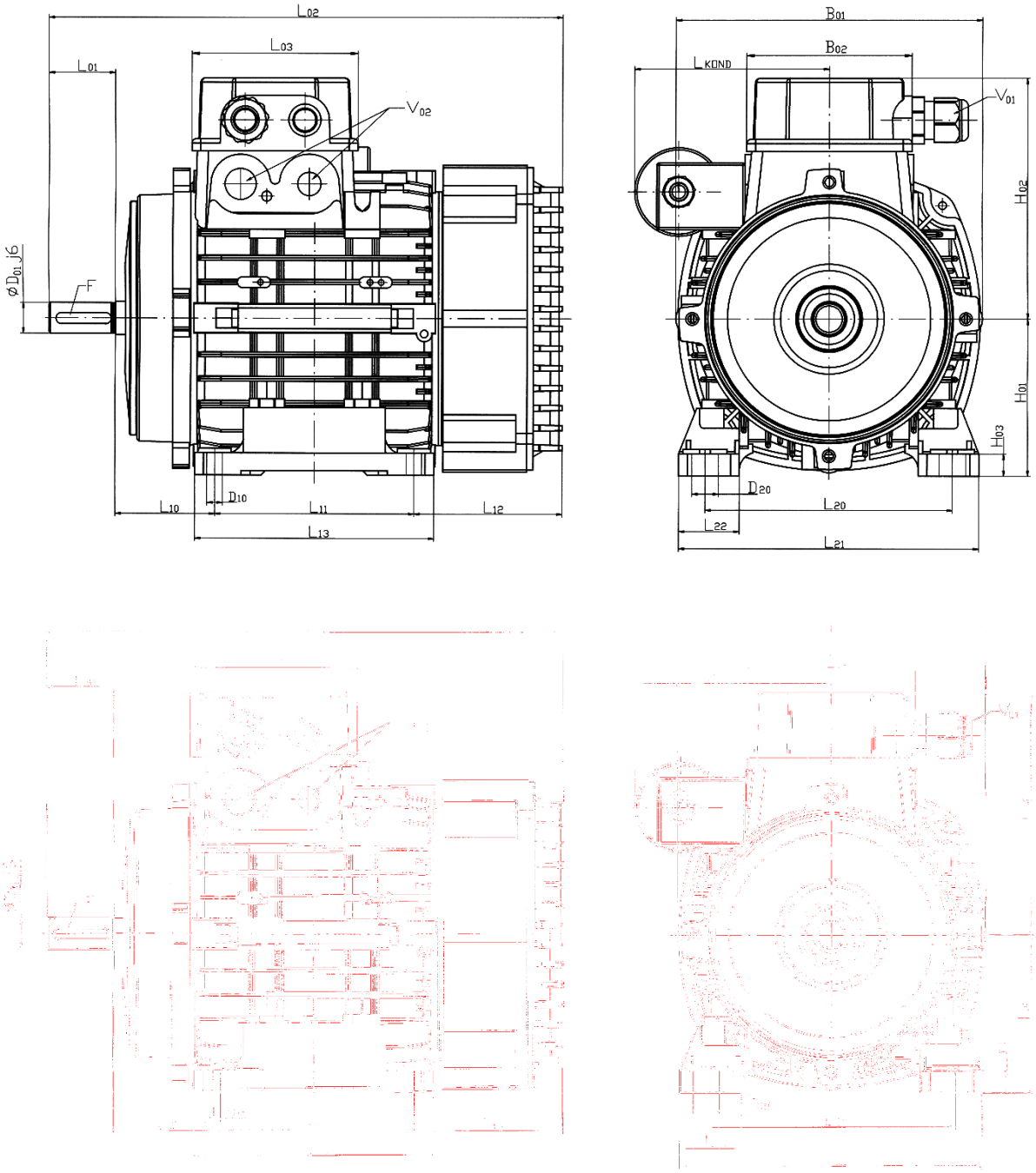
1. «Цифрова обробка зображень», Р. Гонсалес, Р. Вудс.
2. «Комп'ютерний зір. Сучасний підхід», Д. Форсайт, Ж. Понс.
3. «Цифрова обробка зображень», Б. Яне.
4. «Основи теорії розпізнавання образів», Е. А. Патрик.
5. «Regularized Laplacian zero crossings as optimal edge integrators», R. Kimmel and A. M. Bruckstein.
6. «Computer Vision: Algorithms and Applications», R. Szeliski.
7. «Dictionary of Computer Vision and Image Processing», R. Fisher, K. Dawson-Howe, A. Fitzgibbon, C. Robertson.
8. «Feature Extraction and Image Processing for Computer Vision», M. Nixon, A. Aguado.
9. «Handbook of Mathematical Models in Computer Vision», N. Paragios, Y. Chen, O. Faugeras.
10. «A computational approach to edge detection», J. Canny.
11. «A survey of edge detection techniques», L. Davis.
12. «Kernel Smoothing», T. Duong.
13. «Digital image smoothing and the sigma filter. Computer Vision, Graphics, and Information Processing», J. S. Lee.
14. «Edge detection techniques: An overview», D. Ziou, S. Tabbone.
15. «Multi-scale blur estimation and edge type classification for scene analysis», W. Zhang, F. Bergholm.
16. «Edge detection in grayscale, color, and range images», J. M. Park and Y. Lu.
17. «Picture Processing by Computer», A. Rosenfeld.
18. «Algorithms for Image Processing and Computer Vision», J. R. Parker.
19. «Partial Differential Equations, American Mathematical Society», L. Evans.
20. «Alternative approach for satellite cloud classification: edge gradient application», J. R. Dim and T. Takamura.
21. «History and Definition of the Sobel Operator», Irwin Sobel.

22. «Numerical optimization of kernel based image derivatives», Dirk-Jan Kroon.
23. «Computer vision», L. Shapiro and G. Stockman.
24. «Hough transform for straight lines», J. Jensen.
25. «Progressive probabilistic hough transform for line detection», C. Galambos, J. Kittler, and J. Matas.
26. «Generalizing the hough transform to detect arbitrary shapes», D. H. Ballard.
27. «Object Enhancement and Extraction», J.M.S. Prewitt.
28. «The Radon Transform and Some of Its Applications», S. R. Deans.
29. «An Algorithm for Automatically Fitting Digitized Curves», P. J. Schneider.
30. «Hierarchical Generalized Hough Transforms and Line Segment Based Generalized Hough Transforms», L. Davies.

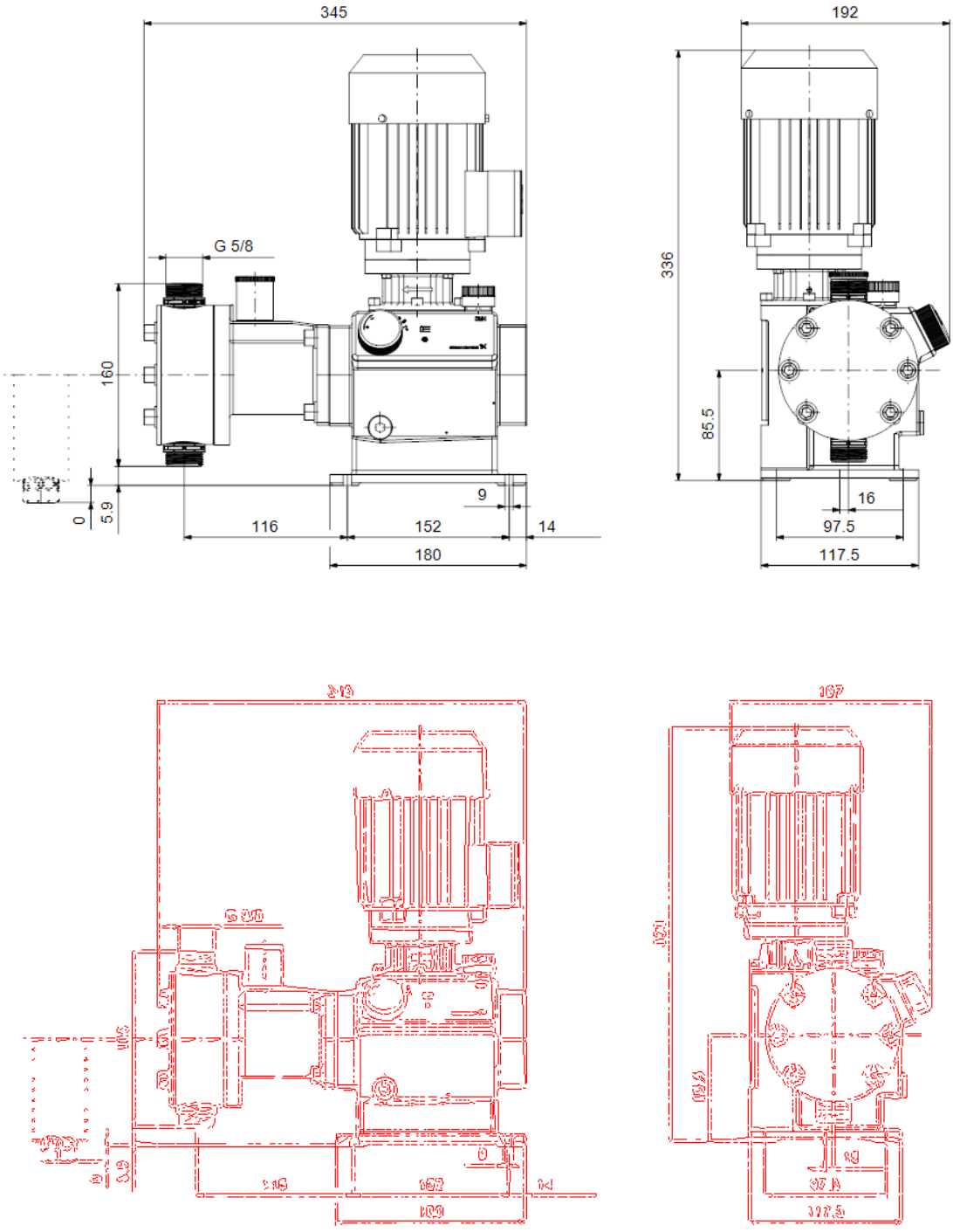
ДОДАТОК 1 - Приклад векторизації растрового зображення



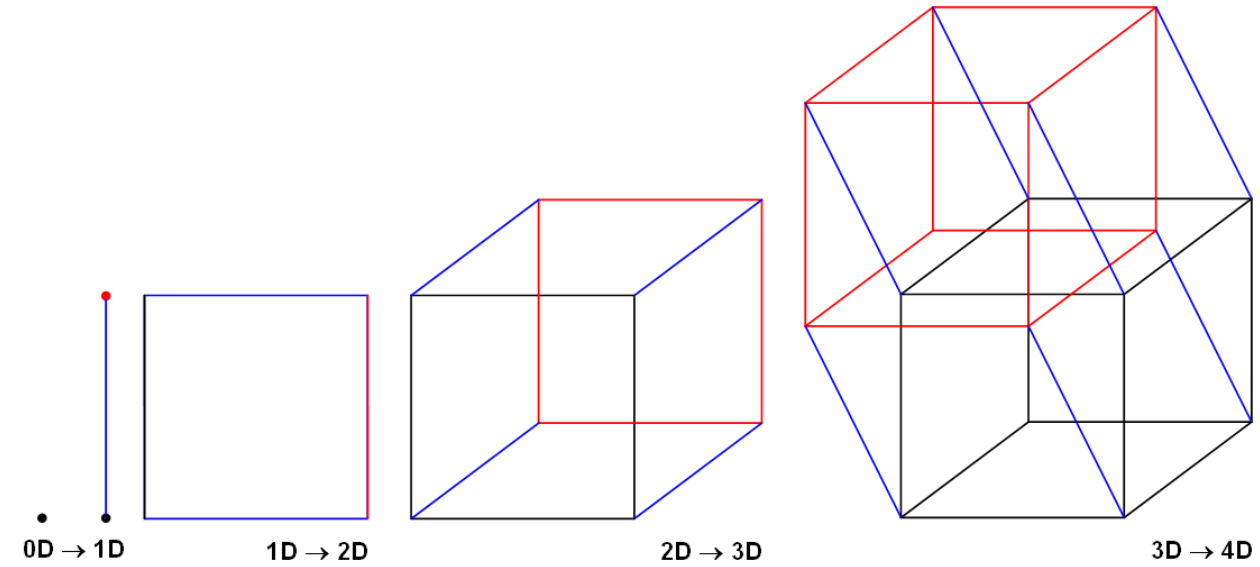
ДОДАТОК 2 - Приклад векторизації растрового зображення

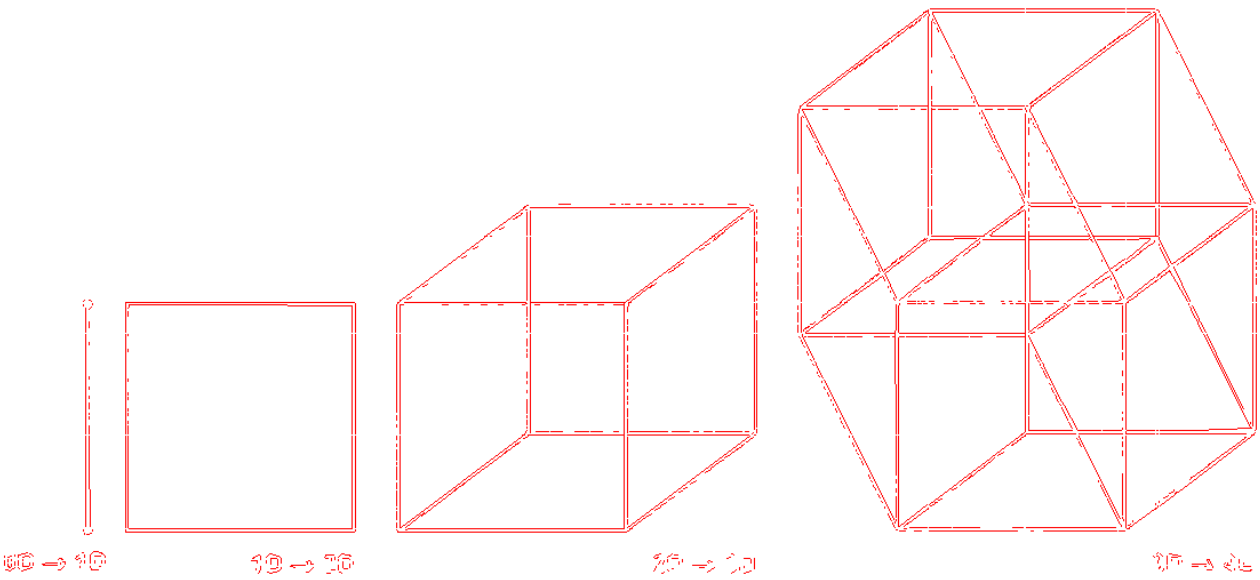


ДОДАТОК 3 - Приклад векторизації растрового зображення



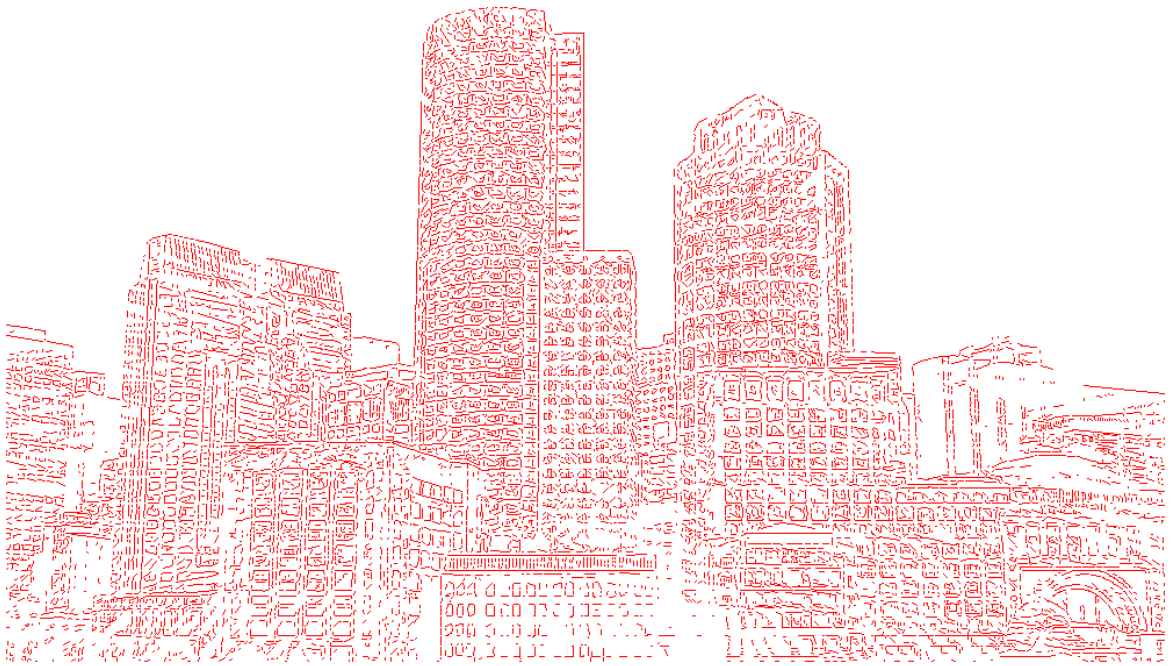
ДОДАТОК 4 - Приклад векторизації растрового зображення



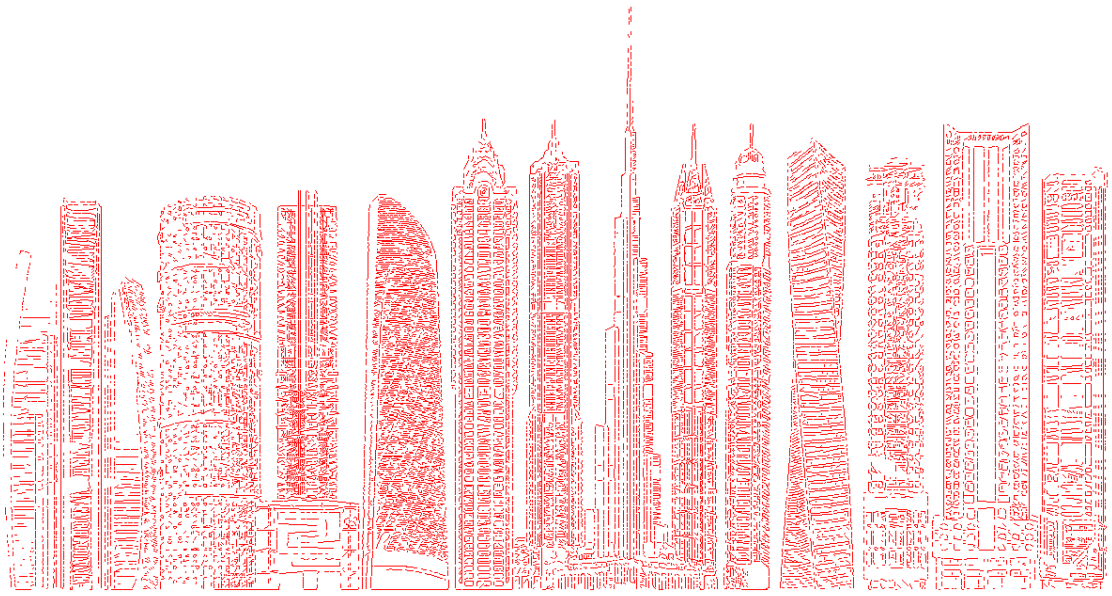


ДОДАТОК 5 - Приклад векторизації растрового зображення

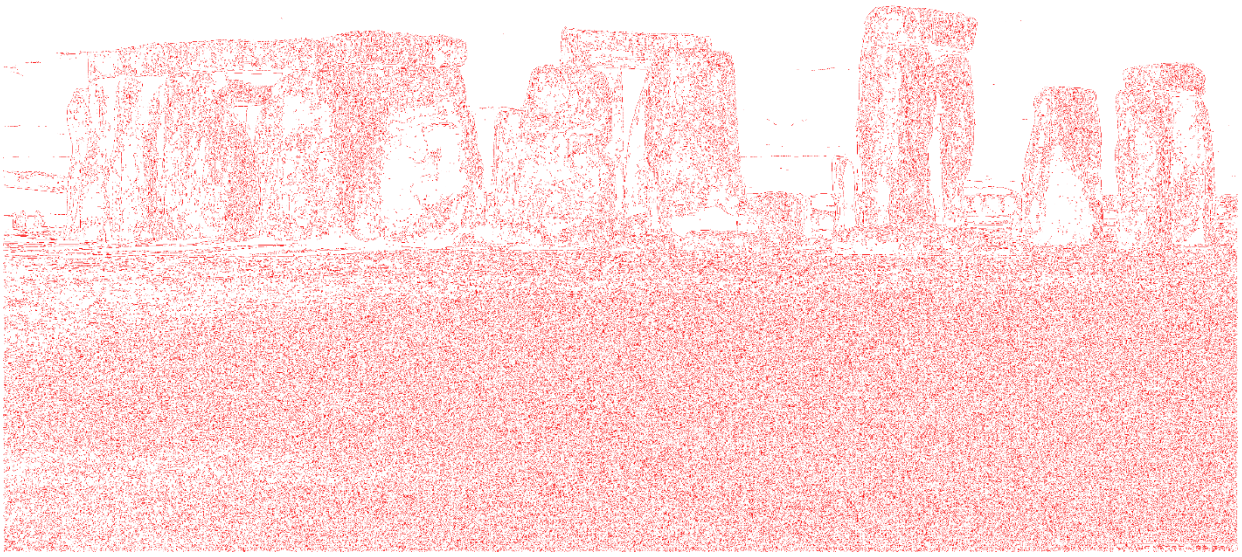
ДОДАТОК 6 - Приклад векторизації растрового зображення



ДОДАТОК 7 - Приклад векторизації растрового зображення



ДОДАТОК 8 - Приклад векторизації растрового зображення



ДОДАТОК 9 - Приклад векторизації растрового зображення

ДОДАТОК 10 - Приклад векторизації растрового зображення