Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)
Сегментація зображень методами кластерного аналізу
(тема)

Виконав:
здобувач 4 року навчання,
групи ТРІМІ-21-2
Данііл Гавриш
(власне ім'я, прізвище)

Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Інформаційно-мережна інженерія
(повна назва освітньої програми)

Керівник ст. викл. Олексій Федоров
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ІМІ _____ Валерій Безрук
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інформаційно мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та Радіотехніка
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« 23 » травня 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Гавришу Даніілу Сергійовичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Сегментація зображень методами кластерного аналізу*

затверджена наказом по університету від « 23 » травня 2025 р. № 410 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 червня 2025 р.

3. Вхідні дані до роботи *Дослідити методи сегментації цифрових зображень. Основну увагу слід приділити застосуванню методів кластерного аналізу для сегментації зображень.* Створити програмну реалізацію алгоритма k-means і продемонструвати його роботу для різних тестових зображень.

4. Перелік питань, що потрібно опрацювати у роботі *Вступ*
1. Особливості зорової системи людини
2. Загальне поняття про сегментацію зображень
3. Застосування методів кластерного аналізу для сегментації зображень
4. Практичне застосування кластерної сегментації
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)
Слайди у форматі Power Point

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вступ	20.05-21.05	виконано
2	Особливості зорової системи людини	20.05-22.05	виконано
3	Загальне поняття про сегментацію зображень	23.05-30.05	виконано
5	Застосування методів кластерного аналізу для	02.06-11.06	виконано
	сегментації зображень		
7	Практичне застосування кластерної сегментації	12.06-17.06	виконано
8	Висновки	30.05-17.06	виконано
9	Оформлення пояснювальної записки	20.05 – 17.06	виконано

Дата видачі завдання 05 травня 2025 р.

Здобувач _____ *Гавриш Д.С.*
(підпис)

Керівник роботи _____ *ст. викл. Федоров О.В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 61 с. 23 рис. табл. джерел

Об’єкт дослідження – особливості зорової системи людини та сегментація зображень.

Мета роботи – дослідити механізми сприйняття кольору та зображень людиною, зокрема принцип трихроматичності, правила групування візуальних елементів за ґештальт-психологією та особливості сегментації зображень у контексті комп’ютерного зору.

У ході роботи було проаналізовано принципи людського сприйняття кольору, розглянуто роль трихроматичності та колірних рецепторів, вивчено основні ґештальт-принципи групування візуальних елементів, а також ознайомлено з підходами до сегментації зображень як ключового етапу комп’ютерного аналізу зорової інформації.

КОМП’ЮТЕРНИЙ ЗІР, ОБРОБКА ЗОБРАЖЕНЬ, СПРИЙНЯТТЯ КОЛЬОРУ, ТРИХРОМАТИЗМ, СЕГМЕНТАЦІЯ, ҐЕШТАЛЬТ-ПРИНЦИПИ

ABSTRACT

Explanatory note: 51 p. 23 fig. table of sources

The object of the study is the features of the human visual system and image segmentation.

The purpose of the work is to investigate the mechanisms of human perception of color and images, in particular the principle of trichromaticity, the rules for grouping visual elements according to Gestalt psychology and the features of image segmentation in the context of computer vision.

In the course of the work, the principles of human color perception were analyzed, the role of trichromaticity and color receptors was considered, the basic Gestalt principles of grouping visual elements were studied, and approaches to image segmentation as a key stage of computer analysis of visual information were introduced.

COMPUTER VISION, IMAGE PROCESSING, COLOR PERCEPTION, TRICHROMATISM, SEGMENTATION, GESTALT PRINCIPLES.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AI (Artificial Intelligence) — штучний інтелект.

RGB (Red, Green, Blue) — модель кольорів, яка базується на трьох основних кольорах: червоному, зеленому та синьому. Кольори комбінуються для створення широкого спектра кольорів у цифрових зображеннях.

HSV (Hue, Saturation, Value) — модель представлення кольору за тоном (Hue), насиченістю (Saturation) і яскравістю (Value). Часто використовується для зручнішого сприйняття кольорів людиною у графічному інтерфейсі та обробці зображень.

k-means (k-середніх) — один із найпоширеніших алгоритмів кластерного аналізу, який розділяє дані на k груп (кластерів) шляхом мінімізації відстані між точками та центроїдами кластерів.

MatLab (Matrix Laboratory) — середовище програмування та мова для чисельних розрахунків, візуалізації та обробки даних.

OpenCV (Open Source Computer Vision Library) — відкрита бібліотека для комп’ютерного зору та обробки зображень, що підтримує C++, Python, Java та інші мови.

IoU (Intersection over Union) — метрика якості сегментації, яка вимірює співвідношення площі перетину передбаченого та істинного сегментів до їх об'єднання.

ВСТУП

У сучасному світі цифрових технологій візуальна інформація відіграє важливу роль у багатьох сферах людської діяльності — від медицини до промисловості та розваг. Зображення стали одним із головних джерел даних, що використовуються для прийняття рішень, автоматизації процесів і створення інтелектуальних систем. З огляду на стрімке зростання обсягів візуального контенту, що генерується щодня, також за допомогою AI, постала нагальна потреба в ефективних методах його аналізу та обробки. У цьому контексті надзвичайно важливою є галузь комп'ютерного зору.

Комп'ютерний зір (англ. computer vision) — це область, що поєднує в собі знання з математики, штучного інтелекту, фізики та інженерії. Метою комп'ютерного зору є розробка алгоритмів і систем, здатних отримувати, інтерпретувати та аналізувати візуальну інформацію з навколишнього світу. Основними компонентами комп'ютерного зору є попередня обробка зображень, виявлення об'єктів, класифікація, розпізнавання образів, відстеження руху та, безумовно, сегментація. Сегментація зображень є одним з найважливіших етапів аналізу візуальної інформації. Вона полягає у поділі зображення на області (сегменти), які мають спільні характеристики — колір, яскравість, текстуру або просторове розташування. Залежно від мети, сегментація може бути грубою (виділення фону та переднього плану) або детальною (виділення окремих об'єктів, структур чи меж). Якісна сегментація є критичною для таких застосувань, як медична діагностика для виділення пухлин на знімках, автономне водіння для розпізнавання доріг, пішоходів, знаків, системи відеоспостереження, робототехніка та багато інших.

Серед численних підходів до сегментації особливу увагу привертають методи кластерного аналізу. Кластеризація — це метод машинного навчання, який дозволяє об'єднувати дані у групи, так названі кластери, на основі певної метрики схожості без попереднього навчання. У випадку зображень, кластерний аналіз дозволяє

розділяти пікселі на групи, що відповідають різним об’єктам чи областям, без необхідності мати марковані дані. Методи кластерної сегментації мають низку переваг: простоту реалізації, гнучкість у виборі ознак, адаптивність до різних типів зображень. Найпоширенішим підходом є алгоритм k-середніх (k-means). Цей метод може працювати в просторі кольору RGB, HSV або з урахуванням просторових координат, що дозволяє адаптувати сегментацію до різних завдань.

У даній роботі буде розглянуто поняття сегментації зображень, основні класи підходів до її реалізації, зосереджуючись на кластерному аналізі як основному методі сегментації. Будуть досліджені переваги, недоліки та особливості використання кластерних алгоритмів, а також продемонстровано приклади їх практичного застосування. Крім того, для порівняння будуть наведені інші поширені методи сегментації зображень, такі як порогова обробка, регіональні методи, алгоритми активних контурів та методи, засновані на глибокому навчанні.

1 ОСОБЛИВОСТІ ЗОРОВОЇ СИСТЕМИ ЛЮДИНИ

1.1 Сприйняття кольору людиною

Щоб ми могли точно описувати кольори, потрібно розуміти, як люди їх сприймають. Людське бачення кольору — це складний процес, на який впливають не лише сам колір, а й умови освітлення, пам’ять, знайомство з об’єктом, а також емоційний стан. Найпростішим завданням у вивченні кольору є визначення, які саме комбінації світла викликають однакову реакцію у людей у стандартних умовах спостереження (пункт 1.2). Це дозволяє побудувати просту лінійну теорію про те, як люди сприймають відповідність кольорів, вона добре працює і широко використовується для опису кольорових характеристик. У подальшому також розглядаються біологічні процеси, що лежать в основі того, як око перетворює світлові сигнали у нервові імпульси, тобто як відбувається трансдукція кольору (пункт 1.4) [4].

1.2 Підбір кольорів

Найпростіше вивчати сприйняття кольору в умовах, коли на темному фоні видно лише два кольорові джерела світла. У типовому експерименті людині показують певний кольоровий сигнал, так зване тестове світло, в одній частині поля зору. В іншій частині вона може змінювати налаштування суміші світла — тобто підбирати комбінацію з кількох базових, первинних, кольорів — так, щоб обидві половини виглядали однаково. Це налаштування відбувається шляхом зміни інтенсивності кожного з первинних світел у суміші. Для досягнення точного збігу між тестовим і змішаним кольорами іноді може знадобитися велика кількість джерел світла, однак існує багато різних варіантів комбінацій, які можуть дати однаковий візуальний ефект. Це демонструє, що людське сприйняття кольору не однозначне: один і той самий колір можна отримати різними шляхами [3].

Щоб дослідити, як люди сприймають кольори, можна провести експеримент, у якому людину просять підібрати комбінацію кольорових світел так, щоб вона виглядала так само, як задане тестове світло (рис. 1.1). Для цього використовують спеціальне поле зору, поділене на дві частини: в одній показують тестовий колір, а в іншій — суміш трьох базових кольорів, інтенсивність яких спостерігач може змінювати [1].

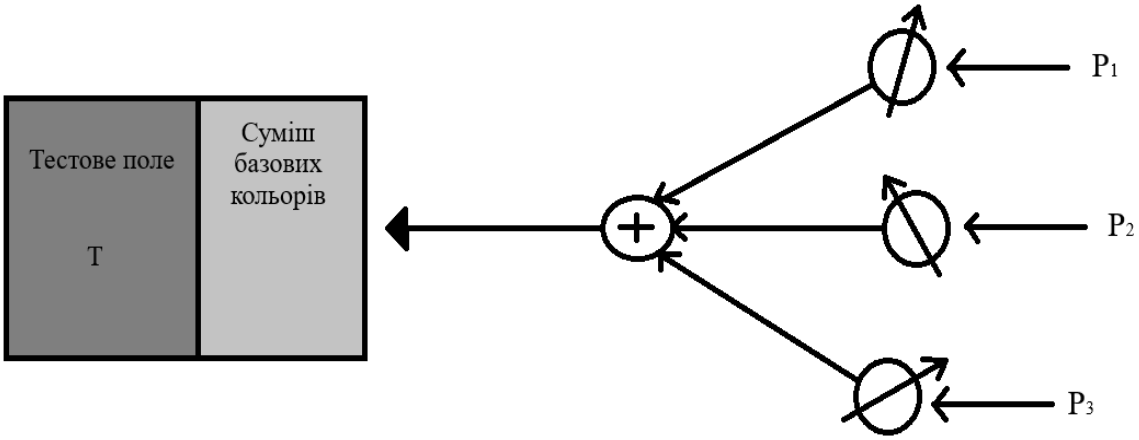

Рисунок 1.1 – Схема проведення експерименту з дослідження особливостей зорової системи людини

Мета спостерігувача — відрегулювати яскравість кожного з трьох базових світел так, щоб суміш виглядала візуально ідентично тестовому світлу. Якщо це вдається, таку відповідність можна описати математично як:

$$\mathrm{T} = w_1 P_1 + w_2 P_2 + w_3 P_3 \,, \tag{1.1}$$

де T — тестове світло, P_i — первинні кольори, а w_i — вагові коефіцієнти з якими ці кольори змішуються [1].

Вражаючим є те, що лише трьох базових кольорів зазвичай достатньо, щоб відтворити більшість кольорів, які бачить людина. А якщо дозволити додавати або віднімати світло, так звану субтрактивну відповідність, то можна досягти відповідності з абсолютно будь-яким кольором. Крім того, більшість людей обирає

дуже схожі комбінації ваг, щоб досягти однакового ефекту, що свідчить про стабільність людського сприйняття кольору [1].

Формула (1.1) означає, що тестовий колір T можна відтворити як суміш кількох базових (первинних) кольорів P_1, P_2, P_3, кожен з яких додається з певною інтенсивністю (вагою) — w_1, w_2, w_3. Однак у деяких ситуаціях потрібно вдаватися до субтрактивного зіставлення. Це означає, що замість того, щоб лише додавати кольори, ми можемо уявно «віднімати» певну кількість одного з базових світел від тестового сигналу. У цьому випадку одна або кілька ваг бути від'ємними. Це дозволяє створювати ще ширший спектр кольорів, навіть якщо деякі з них не можна відтворити простою адитивною сумішшю трьох базових кольорів (червоний, зелений, синій) [1].

1.3 Трихроматичність

Трихроматичність — це властивість людського зору, яка полягає в тому, що більшість людей можуть сприймати й відтворювати будь-який колір за допомогою всього трьох базових кольорів. Проте для цього мають виконуватись дві умови:

- Має бути дозволено не лише додавання, а й віднімання кольорів (субтрактивне змішування);

- Первинні кольори повинні бути незалежними — тобто жодні два з них не можуть у сумі давати третій.

Цей принцип називається принципом трихроматичності. Його пояснюють тим, що людське око має три типи світлочутливих клітин, кожна з яких реагує на певний діапазон довжин хвиль: червоне, зелене та синє світло. Вони перетворюють світлову інформацію на нейронні сигнали, які надходять до мозку для подальшої обробки. Цікавим є той факт, що більшість людей, які бачать одне й те саме тестове світло, обирають дуже схожі пропорції первинних кольорів, щоб досягти точного збігу. Це вказує на те, що три типи колірних рецепторів є однаковими або дуже подібними у більшості людей [7].

1.4 Кольорові рецептори ока

Принцип трихроматичності свідчить про те, що у зоровій системі людини діють певні глибинні обмеження на те, як ми сприймаємо кольори. Щоб це пояснити, вчені висунули припущення, що в оці людини є три типи спеціалізованих рецепторів, кожен з яких відповідає за сприйняття певного діапазону довжин хвиль — тобто певного кольору. Ці рецептори перетворюють світлові сигнали, які надходять в око, у нервові імпульси, що потім обробляються мозком. Завдяки експериментам з порівнянням кольорів можна робити висновки про чутливість кожного типу рецептора. Наприклад, якщо два різні джерела світла, які насправді мають різні спектри, виглядають для людини однаково — це означає, що вони однаково стимулюють усі три типи рецепторів. Тобто для зорової системи ці два спектри еквівалентні [4].

1.4.1 Принцип універсальності

Принцип універсальності пояснює важливу особливість роботи світлочутливих клітин (колбочок) у нашому оці: кожен рецептор лише вимірює, наскільки інтенсивно він був збуджений, але не знає, якою саме була довжина хвилі світла. Тобто він може дати сильну або слабку відповідь, але не містить інформації про колір як фізичну величину. Уявімо, що один і той самий рецептор реагує однаково як на слабке червоне світло, так і на сильне зелене — бо обидва викликають однаковий рівень збудження. Це означає, що за реакцією одного рецептора ми не можемо однозначно визначити колір. Саме тому нам потрібно три різні типи рецепторів — для порівняння [5].

Це явище також добре описується з математичної точки зору: загальна реакція рецептора — це сума впливу світла на всіх довжинах хвиль, помноженого на чутливість рецептора до кожної з цих довжин. Завдяки цьому ми отримуємо лінійну модель — тобто реакція рецептора є сумою незалежних внесків від кожної частини спектра [5].

1.4.2 Палички і колбочки

В людському оці є два основні типи світлочутливих клітин: палички та колбочки. Вони мають різну форму: палички виглядають як тонкі циліндри, а колбочки — як конуси. Ця форма не просто косметична відмінність — вона пов'язана з функціями клітин. Колбочки — це ті, що відповідають за бачення кольорів. Вони зосереджені в центральній частині сітківки, яка забезпечує найвищу гостроту зору. Саме завдяки колбочкам ми добре розрізняємо кольори та дрібні деталі при денному або яскравому освітленні. Палички, навпаки, мають високу світлочутливість, але не розрізняють кольори. Вони активні при слабкому освітленні, наприклад у сутінках або вночі. Саме тому, коли стає темно, кольори “зникають”, і ми бачимо світ у відтінках сірого. Також через те, що в центрі сітківки паличок немає, стає важко читати в темряві — око втрачає здатність чітко фокусуватися [6].

Щоб з'ясувати, як кожен тип колбочок реагує на світло різних довжин хвиль вчені порівнюють, як різні люди сприймають кольори. Три типи колбочок коректно називаються S-, M- і L-колбочками (від англійського short, medium, long — через їхню максимальну чутливість до коротко-, середньо- і довгохвильового світла відповідно). Наприклад, якщо одна група людей має всі три типи колбочок, а інша — не має одного з них, наприклад, L-колбочок, то, аналізуючи, як кожна з груп виконує завдання на підбір кольору, можна вирахувати, яку роль відіграє кожен тип рецептора. Таким чином отримують графіки спектральної чутливості (рис. 1.2) для кожного типу колбочок — S, M і L. Вони мають максимальну реакцію відповідно на короткохвильове (синє), середньохвильове (зеленувате) та довгохвильове (жовто-червонувате) світло [1].

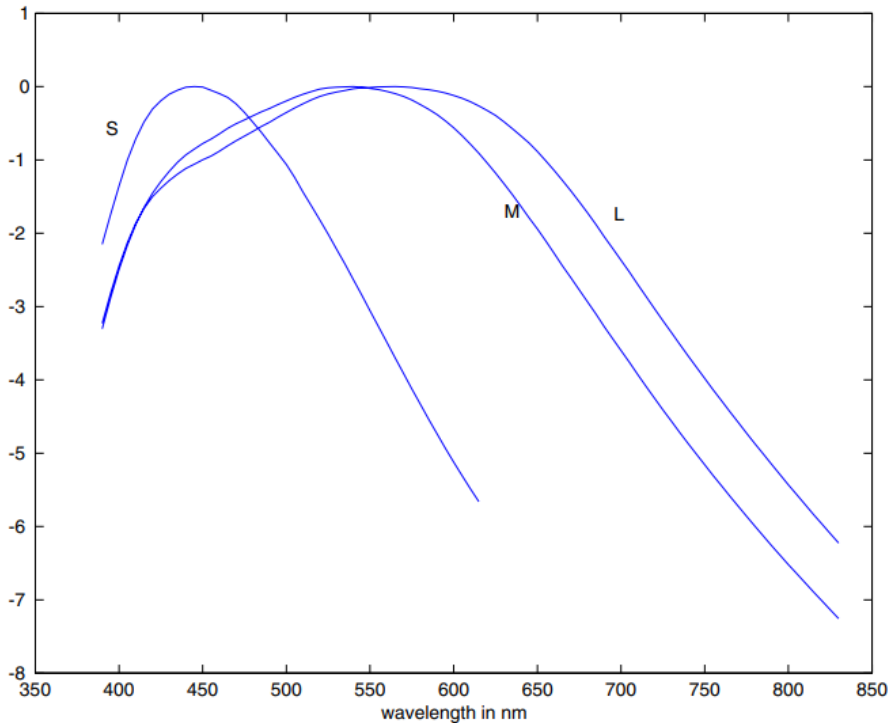

Рисунок 1.2 – Графіки спектральної чутливості

Попри те, що ці рецептори іноді називають «синіми», «зеленими» і «червоними», це не зовсім правильно. Наприклад, сприйняття червоного кольору виникає в результаті складної взаємодії сигналів з кількох колбочок, а не лише з однієї «червоної». Тому в науці використовують точніші назви: S-, M- і L-колбочки [1].

2 ЗАГАЛЬНЕ ПОНЯТТЯ ПРО СЕГМЕНТАЦІЮ ЗОБРАЖЕНЬ

2.1 Методи сегментації в обробці візуальних даних

Комп’ютерний зір можна розглядати як задачу з’ясування: «Що насправді відбувається на зображенні?». Ми маємо багато пікселів — це і є наші вимірювання — але часто не знаємо, які з них дійсно важливі для розуміння об’єктів, а які є лише шумом чи фоном. Тобто ми не можемо «на око» визначити, чи певний піксель є частиною об’єкта чи фону. Сегментація — це процес, який дозволяє відокремити важливі частини зображення, які несуть зміст, і зменшити обсяг даних, щоб комп’ютер працював із суттю, а не з усіма пікселями поспіль. Ідея в тому, щоб знайти структуру в даних: розділити зображення на області за подібністю — наприклад, за кольором, яскравістю, текстурою тощо. У різних задачах те, що вважається важливим або цікавим, може змінюватися. У медицині — це можуть бути пухлини, у системах відеонагляду — люди чи транспорт. Саме тому не існує єдиної універсальної теорії сегментації. Методи сегментації бувають різні: прості, що не враховують ймовірності, та складніші, засновані на статистиці та байєсівських моделях. Попри розмаїття технік, у більшості з них закладено спільний принцип: згрупувати пікселі або об’єкти за певною схожістю, іноді складно знайти цю схожість, але вона присутня [2].

Природне уявлення про сегментацію полягає в тому, що ми намагаємося визначити, які елементи даних належать один до одного природним чином. Ця задача відома як кластеризація. Загалом, існує два підходи до кластеризації:

1. Розділення (partitioning): маючи великий набір даних, ми «розрізаємо» його згідно з певним уявленням про зв’язки між його елементами. Ми хочемо розкласти набір на частини, які є хорошими відповідно до нашої моделі. Наприклад, ми можемо:

– розділити зображення на області зі схожим кольором і текстурою;
– розбити відеопослідовність на кадри або сцени — тобто сегменти відео,

які показують приблизно одне і те саме з подібного ракурсу;
– розкласти відео на «рухомі плями» — ділянки, які мають узгоджені колір, текстуру і рух [2].

2. Групування (grouping): маючи набір окремих елементів, ми хочемо об’єднати ті, які «логічно» поєднуються між собою згідно з нашою моделлю. Ефекти, як-от оклюзія (тобто коли об’єкти перекривають один одного), можуть спричинити те, що частини одного об’єкта виявляються розділеними. Приклади групування включають:

– об’єднання маркерів (токенів), які разом утворюють певний об’єкт;
– об’єднання маркерів, які, схоже, рухаються разом [2].

2.2 Людське бачення: групування та Ґештальт

У найперші часи вчені вивчали, як сильно потрібно змінити світло, звук чи інший стимул, щоб людина це помітила. Один із таких підходів — закон Вебера, він описував, наскільки потрібно підсилити, наприклад, світло, щоб воно здавалося яскравішим. Але пізніше психологи школи Ґештальт показали, що людське сприйняття складніше — ми не просто реагуємо на зміни, а бачимо об’єкти як цілісні форми. Один із найцікавіших моментів зору — це як наш мозок автоматично відділяє об’єкт від фону. Проте, як показано на рисунку 2.1, ми часто легко визначаємо, де фігура, а де тло, але іноді зображення може бути неоднозначним: один і той самий малюнок можна сприймати по-різному, залежно від того, на що звернути увагу. Це говорить про те, що наше бачення — не просто автоматична реакція на світло, а складний процес, який потребує глибшого пояснення [8].

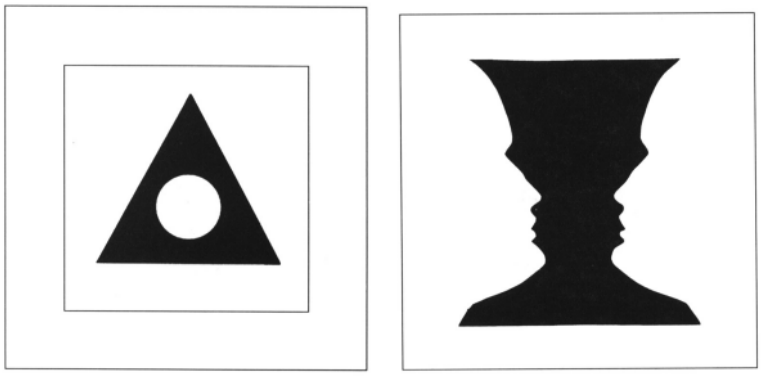

Рисунок 2.1 – Приклади об’єктів як цілісних форм

На рисунку 2.1 зображення зліва ілюструє один із видів неоднозначності, який виникає при такому підході: біле коло можна сприймати як фігуру на чорному трикутному тлі, або ж — як тло, тоді фігурою буде чорний трикутник з круглим отвором — а тлом буде білий квадрат. Справа — інший приклад неоднозначності: якщо фігура чорна, то ми бачимо вазу, а якщо фігура біла — то зображено пару облич [1].

Психологи Ґештальту виокремили низку чинників, які, на їхню думку, сприяють тому, що окремі елементи сприймаються як група. Існує багато таких факторів:

– близькість: об’єкти, розташовані поруч, мають тенденцію групуватися;

– подібність: схожі об’єкти сприймаються як частини одного цілого;

– спільна доля: об’єкти, які рухаються синхронно або в одному напрямку, сприймаються як група;

– спільна область: елементи, що знаходяться в межах однієї замкненої області, сприймаються як пов’язані;

– паралельність: паралельні лінії або об’єкти схильні до групування;

– замкненість: якщо контури елементів можна об’єднати у замкнену фігуру — вони сприймаються як група.

– симетрія: елементи, що утворюють симетричну фігуру, сприймаються як єдине ціле;

– неперервність: елементи, що плавно переходять один в одного (без різких змін), групуються разом [1].

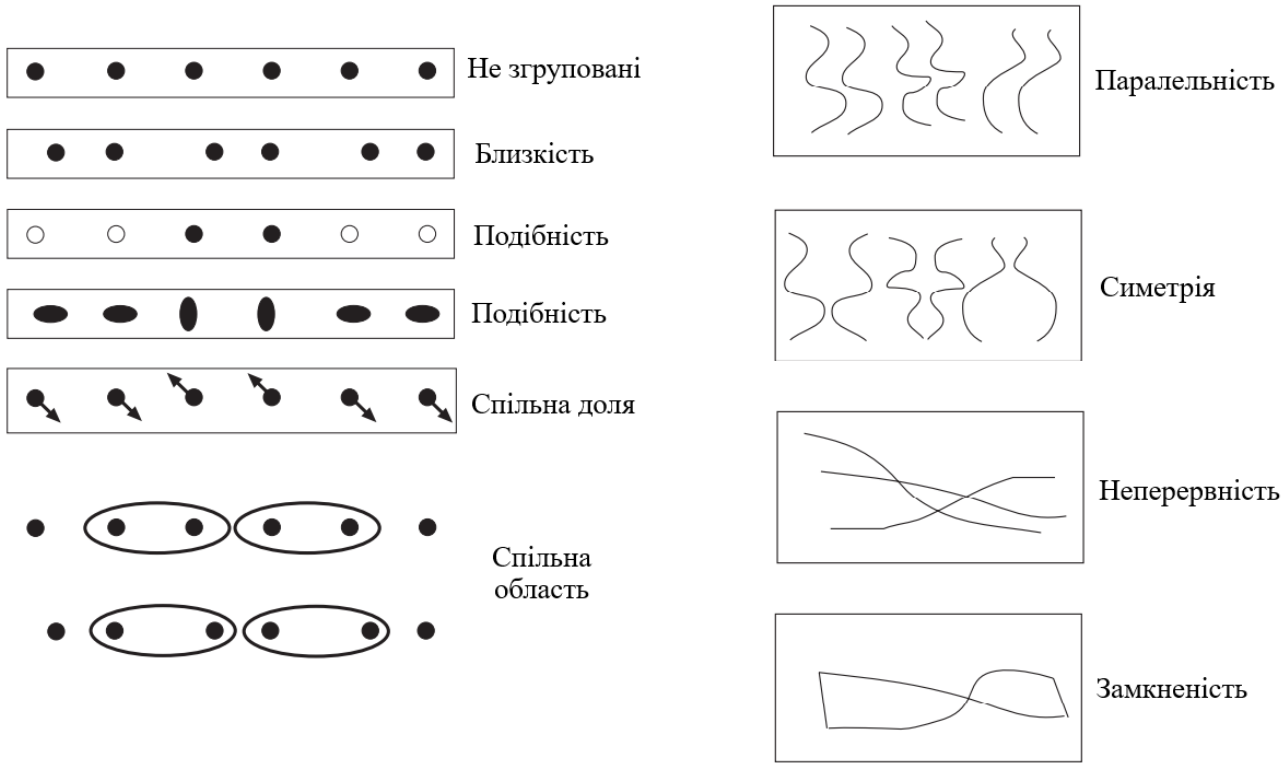

Рисунок 2.2 – Ілюстрація Ґештальт-факторів

Приклади Ґештальт-факторів, які призводять до групування зображено на рисунку 2.2. Ці правила можуть досить непогано працювати як пояснення, але вони недостатньо чіткі, щоб вважатися алгоритмом. Психологи Ґештальту стикалися з великими труднощами щодо деталей — наприклад, коли слід застосовувати одне правило, а коли інше. Дуже складно створити задовільний алгоритм, який би коректно застосовував ці правила. Особливо складною виявилася ідея «знайомої конфігурації» — вона каже, що ми об'єднуємо елементи, якщо вони утворюють щось знайоме, наприклад кішку. Але незрозуміло, як мозок дізнається, що ця конфігурація — це саме кішка. Це передбачає якийсь гігантський пошук серед усіх знайомих образів, але як він працює? Чи ми перебираємо всі варіанти кішок із різними положеннями плям? Це дуже неефективно, тому досі немає чіткого алгоритмічного пояснення, як саме відбувається впізнавання знайомих фігур у

зоровій сцені. Правила Ґештальту все ж дають певне розуміння, оскільки пропонують пояснення того, що відбувається в різних прикладах. Ці пояснення здаються логічними, оскільки припускається, що правила допомагають розв’язувати проблеми, які виникають через зорові ефекти, типові для реального світу — тобто, вони мають екологічну валідність. Наприклад, правило безперервності може бути розв’язанням проблем, які виникають через оклюзію частини контуру об’єкта, який частково перекритий, можуть з’єднуватись в єдину лінію за принципом безперервності (рис. 2.3) [1].

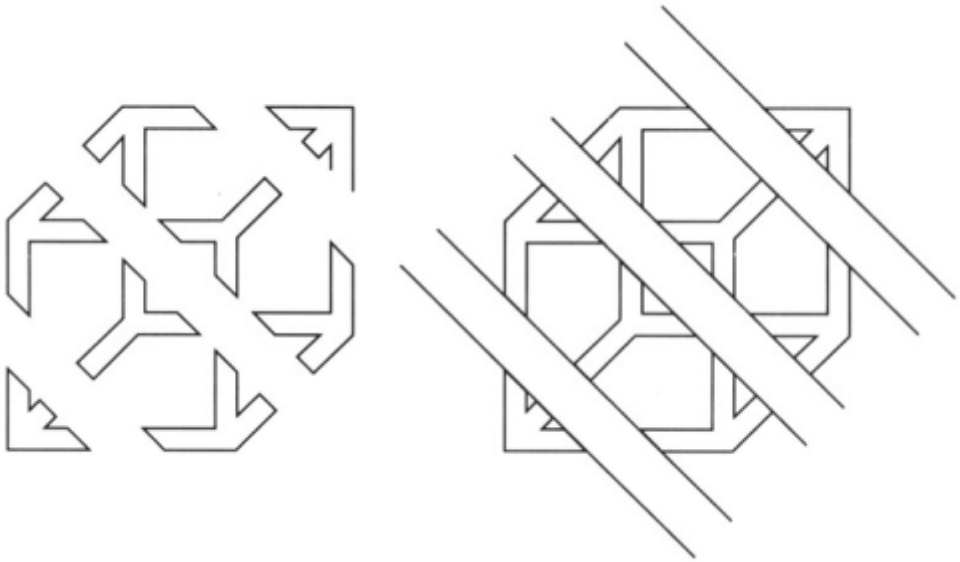

Рисунок 2.3 – Ілюстрація до правила безперервності

Перекриття об'єктів (оклюзія) допомагає зору зрозуміти, що деякі частини зображення пов’язані між собою, навіть якщо між ними є прогалини. Наприклад, якщо якісь частини зображення розташовані так, що схоже — один об’єкт перекриває інший, мозок сприймає їх як частини цілого. У прикладі з кубом: якщо зображення недостатньо чітке, щоб одразу побачити куб, ми можемо сприйняти окремі лінії як розрізнені. Але якщо є візуальні підказки на оклюзію, тобто що одна частина нібито закриває іншу, мозок швидко збирає ці лінії у знайому форму — куб. Тобто сприйняття покращується, коли є логіка в розташуванні частин, навіть якщо вони не з'єднані безперервно [1].

2.3 Порогова обробка зображень (Thresholding)

Порогова сегментація — це простий, але ефективний спосіб виділення об’єктів на зображенні. Якщо на зображенні є світлі об’єкти на темному фоні, а гістограма яскравості має два виражені піки (світлі та темні пікселі), то можна задати поріг T. Пікселі яскравіші за поріг вважаються об’єктами, а ті, що темніші або рівні — фоном. Є декілька видів порогової сегментації, але розглянемо тільки глобальний:

– Глобальний: один і той самий поріг використовується для всього зображення [2].

Іноді гістограма має більше ніж два піки, наприклад, коли є два типи об’єктів плюс фон. У такому випадку застосовується множинна порогова сегментація, де обирається кілька порогів і кожному діапазону яскравості призначається певний клас (об’єкт чи фон), також є важливі чинники для успівшої сегментації:

1. Відстань між піками на гістограмі: чим далі один від одного розташовані піки, тим легше знайти поріг, що чітко розділяє об’єкти та фон [2].

2. Рівень шуму: високий рівень шуму «розмиває» піки, ускладнюючи визначення чіткої межі між класами пікселів [2].

3. Співвідношення розмірів об’єктів і фону: якщо об’єкти занадто малі порівняно з фоном, вони можуть не формувати вираженого піку на гістограмі й бути «втрачені» при пороговій обробці [2].

4. Однорідність освітлення: нерівномірне освітлення призводить до варіацій яскравості одного й того самого об’єкта, що ускладнює вибір фіксованого порогу [2].

5. Однорідність відбивної здатності поверхонь: якщо різні ділянки об’єктів мають різну яскравість через матеріали або текстуру, це може призвести до некоректного класифікування частин об’єкта як фону [2].

2.3.1 Роль шуму в пороговій обробці зображень

Фрагмент на рисунку 2.4 ілюструє, як шум впливає на здатність автоматично розділити зображення на області за допомогою порогу. У «чистому» зображенні, де об'єкти мають різні рівні яскравості, гістограма буде мати два вузьких, чітких піки. Тому легко обрати поріг десь посередині. Коли додається шум — навіть помірний — піки стають ширшими, але їх ще можна розділити. Проте якщо шум сильний, піки зливаються, гістограма стає "розмитою", і в ній вже не видно, де закінчується один клас пікселів і починається інший. У такому випадку простий підхід з порогом вже не працює — потрібні складніші методи обробки [2].

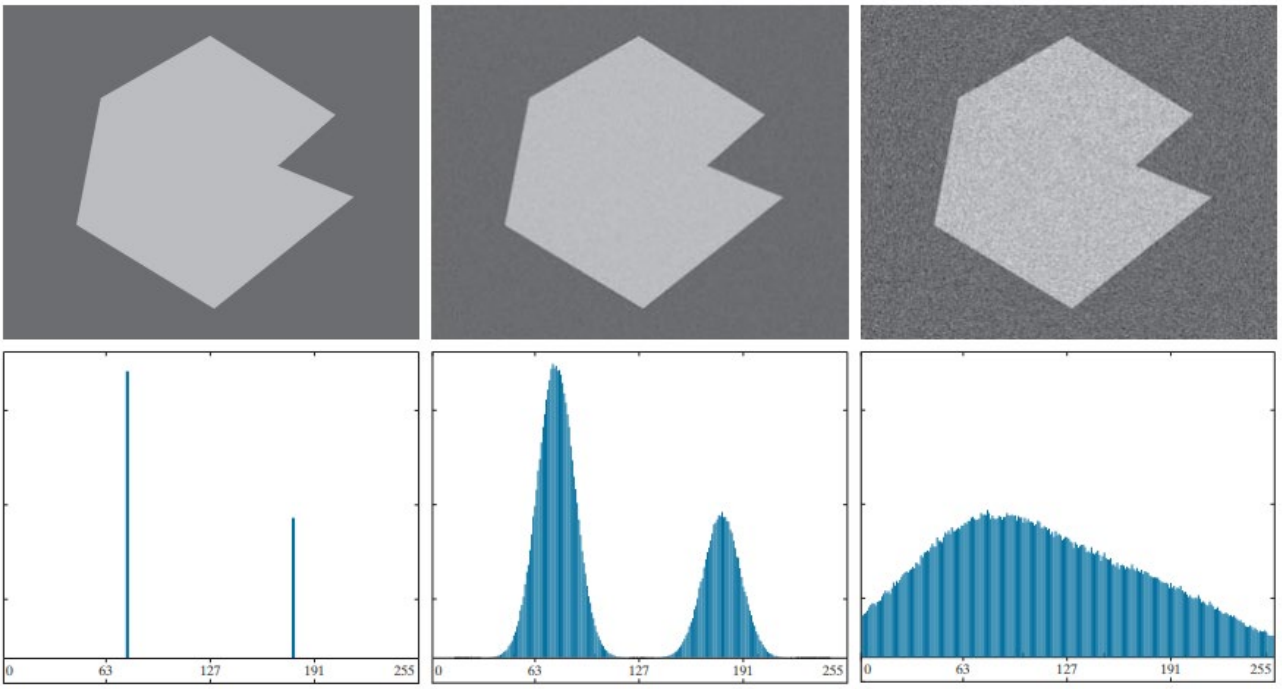

Рисунок 2.4 – Ілюстрація впливу шуму на показ гістограми

2.4 Базове глобальне порогове сегментування

Коли розподіли інтенсивностей пікселів об'єктів і фону достатньо відрізняються, можна використати один (глобальний) поріг, який застосовується до всього зображення. В більшості випадків між різними зображеннями спостерігається така варіативність, що, навіть якщо глобальне порогування є прийнятним методом, потрібен алгоритм, здатний оцінити значення порогу окремо

для кожного зображення. Для цього можна використати наступний ітеративний алгоритм:

1. Вибрати початкову оцінку глобального порогу T [9].

2. Сегментувати зображення за допомогою T (див. рівняння 10-46). Це призведе до формування двох груп пікселів:

G_1 — пікселі з інтенсивністю > T;

G_2 — пікселі з інтенсивністю ≤ T [9].

3. Обчислити середні значення яскравості m_1 і m_2 для груп G_1 і G_2 відповідно [9].

4. Обчислити нове значення порогу як:

$$T = \frac{1}{2}(m_1 + m_2) \text{ [9].} \tag{2.1}$$

5. Повторювати кроки 2–4, поки різниця між значеннями T в двох послідовних ітераціях не стане меншою за заздалегідь визначене значення ΔT [9].

Алгоритм подано в термінах повторного порогування зображення і обчислення середніх значень на кожному кроці, оскільки так його легше інтуїтивно зрозуміти. Однак існує еквівалентна (і більш ефективна) реалізація через гістограму зображення, яку потрібно обчислити лише один раз. Цей алгоритм добре працює, якщо між піками гістограми, що відповідають об'єктам і фону, є чітка «долина». Параметр ΔT використовується для зупинки ітерацій, коли зміни порогу стають незначними. Початкове значення T має бути більшим за мінімальну та меншим за максимальну інтенсивність у зображенні (середнє значення яскравості — хороший вибір для початку). Якщо ця умова виконується, алгоритм сходиться за скінченну кількість кроків, незалежно від того, чи піки гістограми добре розділені [9].

2.5 Сегментація за допомогою простих методів кластеризації

Побудувати сегментатор зображень на основі методу кластеризації досить просто. Вибір метрики відстані залежить від конкретного застосування, але зазвичай використовують міри відмінності кольору або текстури. Часто бажано, щоб кластери мали «плямисту» форму, цього можна досягти, додавши різницю в координатах пікселів у метрику кластеризації. Основна складність при використанні кластеризації (агломеративної чи дивізивної) напряму полягає в тому, що пікселів у зображенні дуже багато. Практично неможливо побудувати та проаналізувати дендрограму, бо вона буде надто великою. Крім того, механізм сумнівний: ми не хочемо вручну переглядати дендрограму для кожного зображення, а прагнемо, щоб сегментатор автоматично видавав корисні регіони для застосунків, наприклад, на довгих відеопослідовностях, без втручання людини [1].

На практиці сегментатори зупиняють процес поділу або злиття на основі певних порогових критеріїв. Наприклад, агломеративний алгоритм може зупинити злиття, коли відстань між кластерами стає занадто великою, або коли досягнуто певної кількості кластерів. Порогові значення зазвичай обираються емпірично — спостерігаючи роботу алгоритму на різних зображеннях і вибираючи найкращі параметри. Однак цей підхід в основному вийшов з ужитку, окрім спеціалізованих застосувань, бо важко передбачити ефективність сегментатора в майбутньому на інших даних. Ще одна проблема через велику кількість пікселів — це неможливість знаходити найкращий варіант поділу або злиття на кожному кроці. Щоб вирішити цю проблему, було розроблено безліч прийомів. Ось загальна схема основних стратегій:

– Розділяючі методи зазвичай модифікуються так, щоб спочатку узагальнювати кластер, а потім пропонувати хорошу стратегію розбиття. Природним узагальненням є гістограма кольору (або відтінків сірого) пікселів у кластері. В одному з перших алгоритмів сегментації, створеному Оландером, кластери розділяються шляхом виявлення піка в одній з дев’яти гістограм ознак (які включають координати кольору в трьох різних колірних просторах) та спроби виділити цей пік. Звісно, при цьому текстуровані області потрібно маскувати, щоб не розбити компоненти текстури між різними кластерами [2].

– Агломеративна кластеризація працює шляхом послідовного об'єднання схожих кластерів. Але на зображенні мільйони пікселів — порівнювати кожен з кожним занадто довго. Тому існує три основні проблеми:

1. Якщо кожен кластер містить багато пікселів, то обчислення середньої чи мінімальної відстані між усіма парами пікселів із двох кластерів стає дуже затратним. Тому замість цього часто використовують відстань між центрами ваг кластерів [1].

2. Зазвичай дозволяється зливати тільки ті кластери, які мають спільний кордон. Це досягається шляхом додавання до функції відстані штрафу: якщо кластери не сусідні, відстань між ними вважається нескінченною. Це запобігає ситуаціям, коли, наприклад, на зображенні прапора США в один кластер об’єднуються всі червоні пікселі, в інший — всі білі, а в третій — всі сині, навіть якщо вони просторово розділені [1].

3. Замість пошуку найближчої пари кластерів, можна просто пройтись по зображенню та об’єднати всі пари кластерів, відстань між якими менша за заданий поріг. Це швидше, але означає, що дендрограма більше не має сенсу. Проте дендрограми й так рідко використовуються на практиці, тому це майже не вважається проблемою [1].

3 ЗАСТОСУВАННЯ МЕТОДІВ КЛАСТЕРНОГО АНАЛІЗУ ДЛЯ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ

3.1 Кластеризація та сегментація за допомогою методу k-середніх

Метод k-середніх — це пошук центрів кластерів і розбиття зображення на області, де пікселі схожі між собою. Але він не враховує просторову зв’язність, тому може «рвати» об’єкти на шматки. Щоб цього не сталось, в ознаки додають координати пікселя. Прості методи кластеризації, такі як агломеративна кластеризація, працюють жадібно — на кожному кроці виконується найкраще доступне об'єднання. Але вони не завжди явно оптимізують якусь цільову функцію [10].

Метод k-середніх працює інакше: він починається з явної цільової функції, яку треба мінімізувати — суму квадратів відстаней між точками та центрами їхніх кластерів. Припустимо, ми знаємо, що є k кластерів. Кожен кластер має центр c_i. Кожен об'єкт (піксель, точка тощо) описується вектором ознак x_j – це можуть бути координати, інтенсивність яскравості, колір і т.д. [10].

Алгоритм k-means працює ітеративно у два кроки:

1. Фіксуємо центри кластерів і призначаємо кожну точку до найближчого центру (тобто, найменшої відстані) [10].

2. Фіксуємо призначення точок до кластерів, і для кожного кластера обчислюємо новий центр — як середнє всіх точок, що до нього належать [10].

Ці кроки повторюються, поки алгоритм не зійдеться до локального мінімуму (тобто, поки центри перестають змінюватись суттєво). Але, глобального мінімуму не гарантується. Результат залежить від початкового випадкового вибору центрів. Алгоритм може створити менше, ніж k кластерів, якщо деякі з них залишились без точок. Щоб уникнути цього, треба змінити алгоритм, примусово гарантуючи ненульову кількість точок у кожному кластері [10].

Отримані області (сегменти) можуть бути розкидані по всьому зображенню й не бути зв’язаними просторово (рис. 3.1, 3.2). Щоб цього уникнути, до ознак додають координати пікселя, але тоді великі області можуть розділитись на менші частини (рис. 3.3) [1].

Рисунок 3.1 – Ділення зображення на області за допомогою k-середніх

На рисунку 3.1 на лівому зображенні — фотографія змішаних овочів. Це зображення було сегментовано за допомогою алгоритму k-середніх і результат показано посередині та справа. У кожному випадку:

– кожен піксель замінено на середнє значення його кластера;

– результат нагадує адаптивну реквантизацію, тобто зменшення кількості унікальних кольорів/відтінків у зображенні з урахуванням вмісту.

Центр: сегментація, яка використовує лише інтенсивність (яскравість пікселів). Праворуч: сегментація, яка використовує колірну інформацію. Обидві сегментації проводились із припущенням, що в зображенні є п’ять кластерів (k=5) [1].

Якщо використовувати тільки яскравість, то кольори ігноруються — виходить «плоске» зображення. Якщо враховувати колір, то сегментація виходить більш природною, бо алгоритм бачить різницю між, наприклад, зеленим горошком і морквою [1].

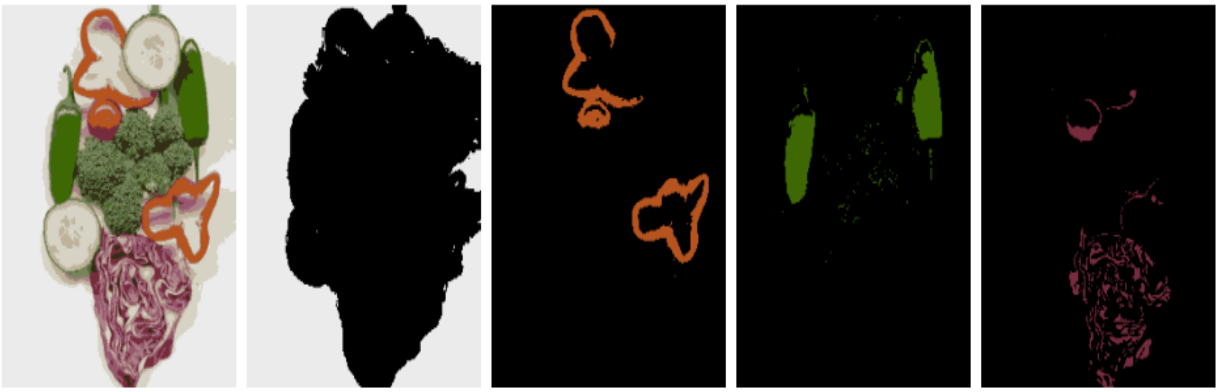

Рисунок 3.2 – Сегментація зображення з k = 11

Тут показано зображення овочів (рис. 3.2), сегментоване за допомогою алгоритму k-середніх, із припущенням про наявність 11 компонентів. Фігура у лівому куті показує всі сегменти разом, при цьому замість початкових значень зображення використано середнє значення кожного сегмента. Інші зображення демонструють чотири окремі сегменти. Треба звернути увагу, що цей підхід призводить до утворення набору сегментів, які не обов’язково є зв’язними (тобто суміжними). У цьому зображенні деякі сегменти дійсно досить добре співвідносяться з об'єктами, але один сегмент може відповідати багатьом об'єктам (наприклад, перцям); інші ж є здебільшого позбавленими сенсу. Відсутність урахування текстурних характеристик створює серйозні труднощі, про що свідчить велика кількість різних сегментів, що виникають унаслідок обробки зрізу червонокачанної капусти [1].

На рисунку 3.3 п’ять сегментів, отриманих шляхом сегментації зображення овочів за допомогою алгоритму k-середніх, який використовує координати положення як частину вектору ознак, що описує піксель. Тепер використано 20 сегментів замість 11. Тепер видно, що великі області фону, які мали б бути однорідними, були розділені, оскільки деякі пікселі виявилися занадто далекими від центра свого кластера [1].

У цьому експерименті алгоритм k-середніх був покращений: до ознак пікселя додано ще й позицію. Це дозволяє враховувати, наскільки пікселі поруч один з одним, що сприяє утворенню більш зв’язних (не розкиданих) сегментів.

Однак з’являються нові проблеми: великі однорідні області фону (які мали б залишитися одним цілим) розпадаються, бо їхні частини географічно далекі від центра свого кластера. З іншого боку, це дозволяє краще розділити окремі перці — тобто об’єкти стають більш локалізованими. Але проблема з червонокачанною капустою залишилася: її складна текстура не враховується, тому вона розбивається на багато непослідовних сегментів [1].

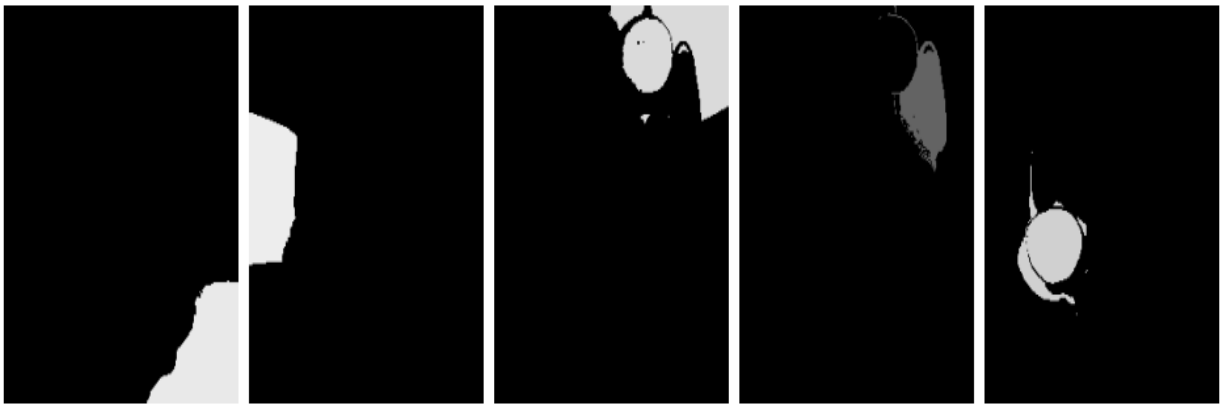

Рисунок 3.3 – Сегментація зображення з k = 20 та додано позицію

Як висновок: додавання позиції покращує просторову згуртованість сегментів, але не вирішує проблему складних текстур. Для повноцінної сегментації необхідно включати текстурні ознаки у вектор ознак пікселя — лише колір і позиція не дають повної картини [1].

3.1.1 Загальні недоліки та особливості сегментації методом k-середніх

Метод k-середніх – це простий і популярний підхід до сегментації зображень, однак він має низку суттєвих обмежень. По-перше, результати сильно залежать від початкового вибору центрів кластерів, що може призвести до потрапляння в локальний мінімум; для часткового вирішення цієї проблеми використовують багаторазовий запуск. По-друге, метод передбачає, що всі кластери мають сферичну (ізотропну) форму й однакову дисперсію, що не відповідає складним формам реальних об’єктів. Також потрібно заздалегідь вказати кількість кластерів k, яка не завжди очевидна, і часто підбирається на основі досвіду. K-

середніх не враховує просторову згладженість — через це сегменти можуть бути розірвані, особливо на великих однорідних областях. Додаткові труднощі виникають при роботі з великими зображеннями через високу обчислювальну складність. Щоб подолати ці обмеження, часто застосовують попередню обробку зображення або комбінують k-середніх з іншими методами (виділення меж, морфологічні фільтри, графові підходи) [10].

3.2 Сегментація за допомогою кластеризації на основі теорії графів

Кластеризацію можна уявити як задачу розрізання графа на «хороші» частини. Уявімо, що кожен елемент даних відповідає вершині у зваженому графі, де ваги на ребрах показують ступінь схожості між елементами:

– велика вага ребра означає, що елементи дуже схожі між собою;

– мала вага ребра — що елементи мало пов’язані або різні [2].

Мета — розбити граф на підграфи (компоненти зв’язності) так, щоб усередині кожного підграфа ребра мали великі ваги (тобто елементи всередині кластеру були схожі), а ребра, які розрізають граф, мали низькі ваги (тобто мінімальна схожість між різними кластерами). Цей підхід призводить до розробки різних алгоритмів сегментації, які часто працюють дуже успішно [2].

Це дає інтуїтивне уявлення про те, як можна застосувати структуру графа для задач кластеризації та сегментації. Важливо, що замість простого порівняння пікселів чи точок, ми дивимося на всю мережу зв’язків між ними і шукаємо «природні» розбивки [2].

3.2.1 Базові графи

Граф – це структура, що складається з множини вершин V та ребер E, які з’єднують пари цих вершин. Граф записується як:

$$G = \{V, E\}\,, \tag{3.1}$$

де $E \subset V \times V$ – множина пар вершин, кожна з яких є ребром [1].

Основні поняття графів:

1. Напрямлений граф — це граф, у якому ребра мають напрямок. Ребра (a, b) і (b, a) вважаються різними. Зображується стрілками, що вказують напрямок ребра [1].

2. Ненапрямлений граф — граф, де ребра не мають напрямку, тобто ребра (a, b) і (b, a) вважаються однаковими [1].

3. Зважений граф — граф, у якому кожне ребро має вагу, що відображає, наприклад, ступінь зв’язку чи схожості між вершинами [1].

4. Самопетля — ребро, яке з’єднує вершину само з собою. У практичних застосуваннях часто відсутні [1].

5. Дві вершини вважаються зв’язаними, якщо існує послідовність ребер, що з’єднує одну вершину з іншою (для направлених графів стрілки в послідовності мають бути спрямовані відповідно) [1].

6. Граф є зв’язним, якщо кожна пара вершин зв’язана [1].

3.2.2 Загальний підхід

Корисно розуміти, що зважений граф можна представити у вигляді квадратної матриці (рисунок 3.4). У ній є рядок і стовпець для кожної вершини. Елемент матриці в позиції i, j представляє вагу на ребрі від вершини i до вершини j. Для неорієнтованого графа використовують симетричну матрицю, розміщуючи половину ваги у кожному з елементів i, j та j, i [1].

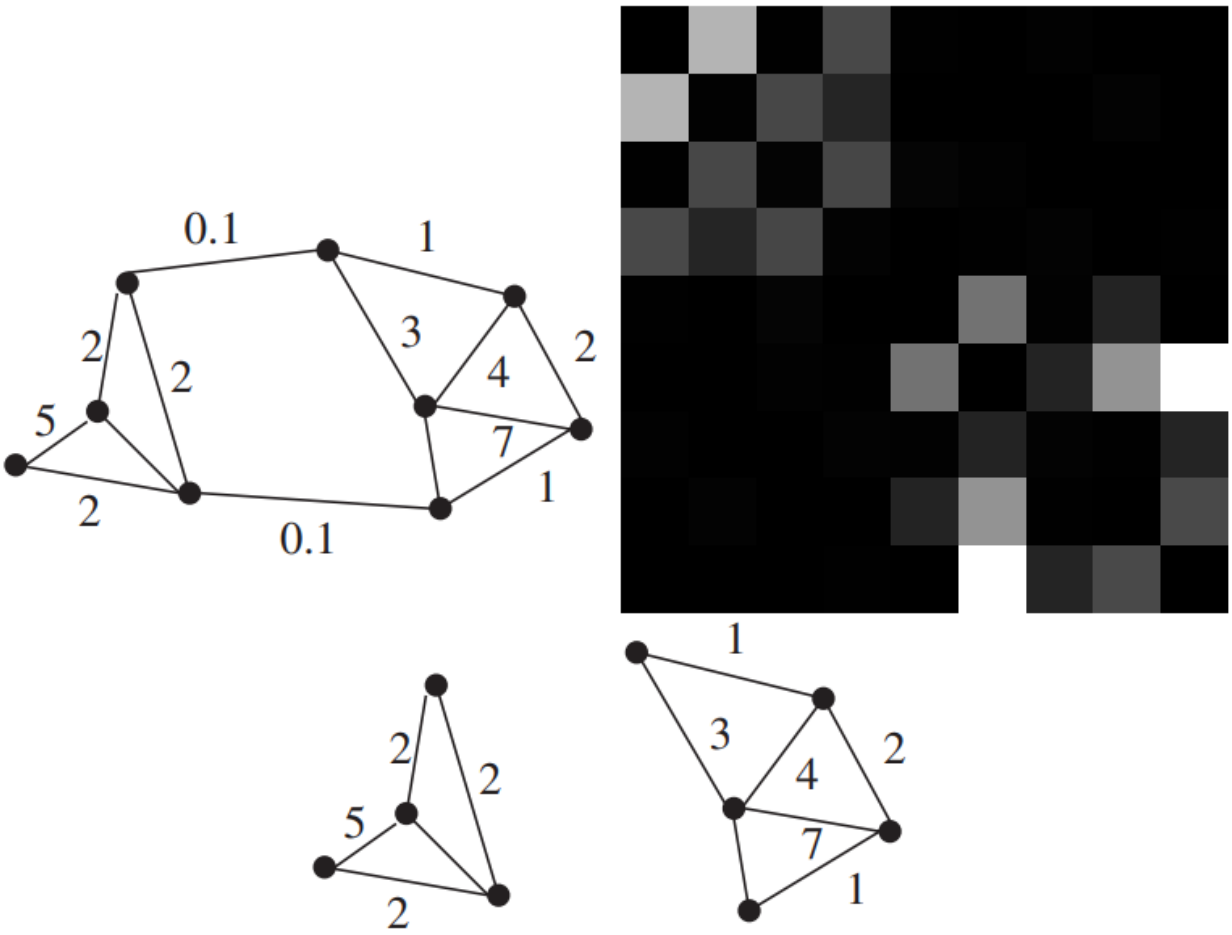

Рисунок 3.4 – Візуалізація кластерної структури зваженого графа

У верхньому лівому куті — зображення неорієнтованого зваженого графа; у верхньому правому — матриця ваг, що відповідає цьому графу. Більші значення позначені світлішими відтінками. Якщо переставити вершини, пов’язані з рядками (і стовпцями) у іншому порядку, матрицю можна перетасувати. Обрано такий порядок, щоб показати матрицю у вигляді, що підкреслює її майже блочно-діагональну структуру. На нижньому рисунку показано розріз цього графа, який розділяє його на дві тісно зв’язані компоненти. Цей розріз перетворює матрицю графа на дві основні діагональні блоки [1].

Застосування графів до кластеризації таке: беремо кожен елемент колекції, яку потрібно кластеризувати, і асоціюємо його з вершиною графа. Потім будуємо ребро між кожною парою елементів і прив'язуємо до цього ребра вагу, що представляє ступінь схожості між цими елементами. Далі «розрізаємо» ребра графа, щоб утворити «хороший» набір зв’язних компонент. Кожна з них і буде

кластером. Наприклад, на рисунку 3.5 є набір добре розділених точок і відповідну матрицю ваг (тобто неорієнтований зважений граф, просто зображений інакше), що виникає в результаті використання певної міри схожості. Бажаний алгоритм помітить, що ця матриця схожа на блочно-діагональну — оскільки внутрішньокластерні схожості сильні, а міжкластерні — слабкі — і розділить її на дві матриці, кожна з яких є блоком [1].

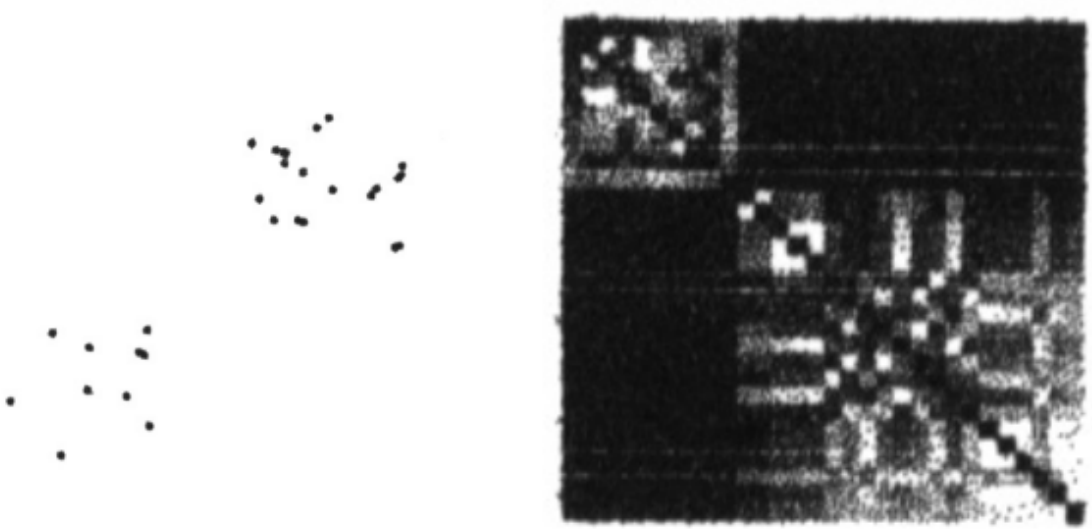

Рисунок 3.5 – Візуалізація кластерної структури через матрицю спорідненості

Зліва — набір точок на площині. Справа — матриця спорідненості для цих точок, обчислена з використанням експоненціального згасання залежно від відстані, де великі значення позначені світлими кольорами, а малі — темними. Зверніть увагу на майже блочно-діагональну структуру цієї матриці: два позадіагональні блоки містять значення, що майже дорівнюють нулю. Діагональні блоки відповідають зв’язкам всередині двох очевидних кластерів, а позадіагональні блоки — зв’язкам між цими кластерами [1].

3.3 Міри спорідненості

Коли ми розглядали сегментацію як просте кластерування, нам потрібно було визначити міру схожості між кластерами. У поточній моделі сегментації достатньо просто призначити вагу кожному ребру графа — ці ваги зазвичай називаються мірами спорідненості. Очевидно, що міра спорідненості залежить від конкретної задачі. Вага дуги, що з'єднує схожі вузли, повинна бути великою, а вага дуги між дуже відмінними вузлами — малою. Досить легко побудувати міри спорідненості з такими властивостями для різноманітних важливих випадків. Крім того, можна комбінувати кілька ознак, формуючи спільну функцію спорідненості як добуток степенів окремих функцій спорідненості. Слід розуміти, що можливі й інші способи визначення функцій спорідненості; немає жодної обґрунтованої причини вважати, що існує єдино правильний або канонічний варіант [1].

3.3.1 Спорідненість по інтенсивності

Спорідненість за інтенсивністю означає, що вага зв’язку між двома пікселями має бути великою, якщо їхні інтенсивності схожі, і зменшуватись зі збільшенням різниці між ними. Найчастіше для цього використовують експоненційну функцію, наприклад таку:

$$aff(x,y) = \exp\left\{-\left(\frac{(I(x)-I(y))^t(I(x)-I(y))}{2\sigma_I^2}\right)\right\} \text{[1]}. \qquad (3.2)$$

3.3.2 Спорідненість за кольором

Щоб побудувати осмислену функцію спорідненості за кольором (colour affinity function), потрібно використати правильну метрику кольору. Це означає:

– використовувати рівномірний колірний простір, наприклад CIE Lab або CIE Luv. У таких просторах відстані між кольорами відповідають тому, як людина їх сприймає (тобто суб’єктивне відчуття «подібності» кольорів є приблизно пропорційне евклідовій відстані);

– не використовувати RGB-простір, оскільки він не є рівномірним, однакова різниця в числах не означає однакову візуальну різницю та залежить від освітлення і пристрою [1].

Формула для функції спорідненості за кольором має вигляд експоненціального згасання:

$$aff(x,y) = \exp\left\{-\left(dist\big(c(x)\big), \frac{c(y))^2}{2\sigma_C^2}\right)\right\}, \tag{3.3}$$

де c_i це колір у пікселі i [1].

3.3.3 Спорідненість за текстурою

Щоб оцінити спорідненість за текстурою, потрібно зробити так, щоб вона була великою для схожих текстур, меншою для відмінних текстур. Для цього використовують набір фільтрів $f_1 \ldots f_n$, які охоплюють різні масштаби та орієнтації.

Однак для більшості текстур значення фільтрів не будуть однаковими в кожній точці текстури — наприклад, у випадку шахової дошки — але гістограма значень фільтрів, побудована на основі певного локального оточення, буде стабільною. Наприклад, якщо ми візьмемо нескінченну шахову дошку і побудуємо гістограму виходів фільтрів для області, що охоплює декілька клітин, ця гістограма залишатиметься приблизно однаковою, незалежно від того, де саме розташована ця область [1].

Це підказує такий підхід: спочатку на кожній точці визначається локальний масштаб — наприклад, шляхом оцінки енергії у фільтрах великого масштабу або іншим методом. Потім будується гістограма виходів фільтрів у регіоні, визначеному цим масштабом — наприклад, у круговій області з центром у відповідній точці. Тоді ми пишемо h для цієї гістограми та використовуємо експоненціальну форму:

$$aff(x,y) = \exp\left\{-\left(\frac{(f(x)-f(y))^t(f(x)-f(y))}{2\sigma_I^2}\right)\right\} \text{ [1].} \tag{3.4}$$

4 ПРАКТИЧНЕ ЗАСТОСУВАННЯ КЛАСТЕРНОЇ СЕГМЕНТАЦІЇ

4.1 Тестові зображення

Для практичного застосування кластерної сегментації методом k-середніх будуть використані так звані «тестові зображення». Це загальноприйняті зображення такі як:

– Lena — класичне зображення обличчя;

– boats, peppers — різноманітні зображення з чіткими структурами, текстурами, об’єктами;

– Barbara — особливої уваги заслуговує її одяг і шарф — містять багато дрібних регулярних візерунків (сітка, смужки);

– baboon — зображення мавпи з дуже насиченою і складною текстурою хутра, є багато дрібних варіацій яскравості, кольору і форми.

Це відкриті, легкі для завантаження зображення і вже інтегровані в багато бібліотек і систем, наприклад MatLab та OpenCV. Ці зображення стали загальноприйятими тому що, завдяки використанню однакових зображень, різні алгоритми можна безпосередньо порівнювати між собою для об’єктивної оцінки ефективності різних методів між собою. Оскільки для цих зображень вже є очікувані результати, то це що дозволяє точно оцінити якість сегментації за метриками Rand Index, IoU. Також тестові зображення охоплюють багато різних типів сцен: портрети, об’єкти, пейзажі, текстури, що дозволяє тестувати алгоритми на широкому спектрі ситуацій.

4.2 Вибір мови програмування для написання скрипту/коду

На мій погляд варто обрати мову R для написання скрипту, коли мова йде про статистичну обробку даних, аналіз, візуалізацію або прототипування алгоритмів,

пов’язаних із даними. Вона спеціально створена для статистики, тому має величезну кількість вбудованих функцій і бібліотек для кластеризації, обробки зображень, тощо. Завдяки цьому можна реалізовувати складні математичні ідеї з мінімумом коду. Крім того, у R зручно візуалізувати як вихідні дані, так і результати кластерної сегментації, що особливо корисно при дослідженнях у реальному часі. Мова підтримує інтерактивну роботу через RStudio або Jupyter, також існує багато наукових публікацій і прикладів саме на R, тому простіше знайти підтримку чи перевірені реалізації.

Окрім мови R для кластерної сегментації або обробки зображень можна використовувати інші мови програмування, наприклад Python, MatLab, C++. Python — найбільш популярна мова для аналізу даних, машинного навчання та обробки зображень, зручний та має велику активну спільноту. MatLab — популярний у наукових колах, особливо серед інженерів. Має інструменти для обробки сигналів, зображень, побудови графіків, а також простий синтаксис. C++ — підходить, якщо потрібна висока швидкість виконання, наприклад, для обробки великої кількості зображень у реальному часі. Можна використовувати разом із бібліотекою OpenCV, яка підтримує сегментацію, кластеризацію та обробку зображень.

4.3 Алгоритм виконання кластеризації на практиці

Для першого тесту було обрано полутонове зображення під назвою boats (рис. 4.1). Спочатку підключається бібліотека та зчитування зображення (рис. 4.2). Ці рядки підключають кастомну бібліотеку libpbm, яка містить функції для читання та запису зображень у форматі pgm (рис. 4.3).

Рисунок 4.1 – Тестове зображення boats

```
6  libPath <- "G:/Downloads/libpbm/libpbm";
7  source("G:/Downloads/libpbm/libpbm/libpbm.R");
```

Рисунок 4.2 – Підключення бібліотеки

```
 9  sGrayImgName <- paste0( "G:/Downloads/pgm+ppm_classic/",
10                          "pgm+ppm_classic/pgm/boats.pgm" );
11  sGrayClustImgName <- paste0( "G:/Downloads/pgm+ppm_classic/",
12                               "pgm+ppm_classic/","boats_cluster.pgm" );
13
14  mGrayPix <- readPGM( sGrayImgName );
```

Рисунок 4.3 – Запис зображення з директорії

У рядку 9 формуються шляхи до зображення boats.pgm та формується шлях для бререження кластеризованого зображення boats_cluster. У рядку 14 читається полутонове зображення у вигляді матриці піксельних значень від 0 до 255. На рисунку 4.4 будується матриця. Це таблиця, де кожен рядок – це піксель, представлений трьома числами:

– row: координата по горизонталі;

– col: координата по вертикалі;

– pix: якравість пікселя.

```
14  mGrayPix <- readPGM( sGrayImgName );
15
16  mGrayToClust <- cbind( as.matrix( expand.grid( seq( nrow( mGrayPix ) ),
17                                                 seq( ncol( mGrayPix ) ) ) ),
18                         as.vector( mGrayPix ) );
19  colnames( mGrayToClust ) <- c( "row","col","pix" );
```

Рисунок 4.4 – Побудова матриці

Наступним кроком обирається кількість кластерів, нехай k = 3. Тобто ми хочемо розбити зображення на 3 рівні яскравості. На рисунку 4.5 у рядку 25 виконується функція обчислення евклідової відстані між двома векторами, це тиаова метрика для кластеризації. Функція у рядку 27 (рис. 4.5) це ручна реалізація алгоритму, де:

– mData – це матриця даних;

– nClast – кількість кластерів;

– nIter – еількість ітерацій.

```
25  fdist <- function( v1,v2 )  sqrt( sum( (v1-v2 )^2 ) );
26
27  fkmeans <- function( mData,nClast,nIter=10 )
28
29  {
30    mCenters <- mData[ sample( seq( nrow( mData ) ),nClast,replace=FALSE ), ];
31    vClusters <- apply( mData,1,function( vData ) which.min( apply( mCenters,1,fdist,vData ) ) );
32
33    for( i in 1:nIter )
34    {
35      mCenters <- t( sapply( 1:nClast,function( i ) colMeans( mData[ vClusters==i, ] ) ) );
36      vClusters <- apply( mData,1,function( vData ) which.min( apply( mCenters,1,fdist,vData ) ) );
37    }
38
39    return( vClusters );
40  }
```

Рисунок 4.5 – Реалізація алгоритму k-середніх із нуля

У рядку 30 виконується дія, при якій випадково вибирається nClast точок з mData як початкові центри кластерів. Далі у рядку 31 кожна точка призначається до найближчого центру. З рядка 33 по 37 задане ітераційне уточнення. Кожну ітерацію обчислюються нові центри клстерів як середні по поточним точкам; точки переназначаються у найближчі нові кластери. Рядок 39 повертає результат у вигляді вектору, який для кожної точки вказує номер її кластера.

На рисунку 4.6 йде побудова зображення з кластеризованими значеннями. Середні значення округлюються до цілих, кожному пікселю присвоюється нове значення – відповідний центроїд кластера, до якого він належить. Результат записується у новий .pgm-файл.

```
54  lGrayClust <- kmeans( mGrayToClust[ ,c( -1,-2 ) ],K );
55  ( mGrayCenters <- floor( lGrayClust$centers ) );
56
57  mGrayClust <- as.vector( mGrayPix );
58  for( nClust in seq( K ) )
59    mGrayClust[ lGrayClust$cluster==nClust ] <- mGrayCenters[ nClust ];
60  dim( mGrayClust ) <- dim( mGrayPix );
61
62  writePGM( mGrayClust,sGrayClustImgName );
```

Рисунок 4.6 – Побудова зображення з кластеризованими значеннями

Отримав результат, файл під назвою boats_cluster (рис. 4.7). На оригінальному зображені багато варіацій сірого кольору, що передають дрібні деталі, текстуру й об’єм. Після кластеризації видно лише декілька рівнів яскравості, оскільки було обрано три кластери. Попри втрату дрібних елементів, загальні контури об’єктів залишаються пізнаваними. Це означає, що кластеризація ефективно зберігає суть сцени, навіть при сильній редукції. Також видно що кластероване зображення стало більш однорідним.

Рисунок 4.7 – Тестове зображення після кластеризації k = 3

Тепер можна спробувати змінити кількість кластерів, скажімо k = 10. Буде використовуватись той самий алгоритм, з тим самим тестовим зображенням. На рисунку 4.8 можна побачити, що при десяти кластерах зображення майже не змінюється. Це виходить тому, що рисунок 4.7 був поділений на 3 домінуючі області відтінків сірого. Оскільки зображення полутонове, воно не має великої варіативності в яскравості, тому нові кластери не додають нових деталей, як наслідок дуже схожий на оригінал.

Рисунок 4.8 - Тестове зображення після кластеризації k = 10

Для другого експерименту було обране кольорове тестове зображення pepper (рис. 4.9). Також для першої сегментації обрав три кластери, а потім сім, щоб порівняти результат та додатково порівняти з результатами тестового зображення boats.

Рисунок 4.9 – Тестове зображення pepper

Для кольорового зображення було виконано алгоритм на мові програмування Phyton. Кластеризація буде виконуватися тільки по кольоровим компонентам. На рисунку 4.10 зображено код програми. Якщо стисло, то він зчитує зображення, претворює його в список пікселів за кольором (RGB), далі групує пікселі у k кластерів за подібністю кольору. Потім замінює всі пікселі коьором центру свого кластера і зберігає нове зображення з k основними кольорами.

```
import numpy as np
from sklearn.cluster import KMeans
import imageio.v2 as imageio

input_path = "G:/Downloads/pgm+ppm_classic/pgm+ppm_classic/ppm/pepper.ppm"
output_path = "G:/Downloads/pgm+ppm_classic/pgm+ppm_classic/ppm/pepper_cluster3.ppm"

img = imageio.imread(input_path)
h, w, c = img.shape
pixels = img.reshape(-1, 3)

kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(pixels)
pixels_clustered = kmeans.cluster_centers_[labels].astype(np.uint8)

img_clustered = pixels_clustered.reshape(h, w, c)
imageio.imwrite(output_path, img_clustered)
```

Рисунок 4.10 – Код програми для кластеризації на мові Phyton

Призначив для kmeans три кластери та отримав результат (рис. 4.11). Можна спостерігати, що зображення знову, як і на рисунку 4.7, спрощене до використання трьох основних кольорів, які відповідають центроїдам кластерів. Деталі та ніті спрощено, або взагалі втрачені. Це приклад кольорової редукції, тобто зменшення кількості кольорів на зображені. Це може знадобитися для стискання зображень або візуалізації основних кольорів на сцені.

Рисунок 4.11 – Тестове зображення після сегментації при k = 3

Тепер спробую вказати більше кластерів і подивитися що відбудеться. Зробив зміни в коді програми і отримав результат на рисунку 4.12. Зображення з семи кластерів демонструє значно детальнішу сегментацію, ніж з трьома. Отримано більш різноманітні відтінки, значно краща видимість тіней, відблисків. Наприклад зелений перець тепер має кілька відтінків, а не один суцільний колір, червоні перці поділились на зони з різною насиченістю червоного. Краще розділення схожих, але різних за тоном об’єктів, зберігається більше структури оригіналу.

Як висновок можна зазначити, що чим більше кластерів, тим краще дозволяє зберегти візуальну структуру і кольорову подібність до оригіналу.

Рисунок 4.12 - Тестове зображення після сегментації при k = 7

ВИСНОВКИ

У ході роботи було розглянуто фізіологічні основи кольорового зору людини, зокрема трихроматичну модель сприйняття кольору, принцип уніваріантності та роль колірних рецепторів (S-, M-, L-конусів). Аналіз психофізичних експериментів дозволив краще зрозуміти механізми кольоросприйняття. Також детально розглянуто сегментацію як ключовий етап комп’ютерного зору, що дозволяє виділяти змістовні об’єкти на зображеннях. Були досліджені фактори, що впливають на візуальне об'єднання елементів, згідно з принципами Ґештальт-психології. Отримані знання мають практичне значення для розробки систем машинного зору, розпізнавання образів та аналізу візуальної інформації.

У ході практичного застосування кластерної сегментації методом k-середніх було продемонстровано ефективність цього підходу для виділення візуально значущих об'єктів на зображеннях різної складності. Алгоритм дозволяє автоматизувати попередню обробку зображень, зменшити кількість кольорів та спростити структуру сцени, що особливо важливо для подальшого аналізу в задачах комп’ютерного зору. Попри певні обмеження (залежність від кількості кластерів, чутливість до початкових умов, ігнорування текстур), k-середніх залишається надійним методом для базової сегментації. Практичні результати підтверджують придатність цього підходу для прототипування алгоритмів, візуалізації та автоматичного виділення об’єктів у зображеннях без маркованих даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Forsyth D. A., Ponce J. Computer Vision. Prentice Hall PTR, 2002. 58–61 401–406 с.

2. Gonzalez R. C., Woods R. E. Digital Image Processing, 4e. New York : Pearson, 2019. 742–747 с.

3. Bishop M. Pattern Recognition and Machine Learning. New York : Springer, 2006. 428–430 с.

4. Everitt B. S. Cluster Analysis. Chichester, West Sussex, U.K : Wiley, 2011.

5. Aggarwal C. C., Reddy C. K. Data Clustering: Algorithms and Applications. Chapman & Hall/CRC Press, Boca Raton : Chapman & Hall/CRC, 2014.

6. Shapiro L. G., Stockman G. C. Computer Vision. New Jersey, USA : Prentice Hall, 2001.

7. Segmentation of Natural Images by Texture and Boundary Compression / M. Hossein та ін. *International Journal of Computer Vision*. 2011. Т. 95. С. 86–98.

8. Solomon C. J., Toby P. B. Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab. Chichester, West Sussex, U.K. : Wiley-Blackwell, 2010.

9. Tyagi V. Understanding Digital Image Processing. Boca Raton, FL, USA : CRC Press (Taylor & Francis Group), 2018.

10. Sonka M., Hlavac V., Boyle R. Image Processing, Analysis, and Machine Vision. Boston, USA : PWS Publishing, 1999.