Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Гібридні методи кластеризації цифрових зображень в автономних системах комп'ютерного зору

(тема)

Виконав:
студент II курсу, групи СПзм-18-2
Вакулік Є.В.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: ст. викл. Носик А.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Вакуліку Євгену Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Гібридні методи кластеризації цифрових зображень в автономних системах комп'ютерного зору

затверджена наказом по університету від "30" березня 2020 р. № 43 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2020 р.

3. Вхідні дані до роботи

1. Алгоритми кластеризації зображень
2. Вимоги до передобробки та кластеризації
3. Вимоги до оцінювання ефективності

4. Перелік питань, що потрібно опрацювати в роботі

Аналіз предметної області

Розбір основних методів кластеризації зображень

Розробка гібридного алгоритму кластеризації

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Плакати – 14 арк. ф. А4

6. Консультанти розділів роботи (заповнюється 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	Аналіз сучасного стану проблеми	31.03.20 – 15.04.20	
2	Постановка задачі дослідження	16.04.20 – 18.04.20	
3	Аналіз літератури	19.04.20 – 23.04.20	
4	Пошук методів кластеризації	23.04.20 – 27.04.20	
5	Аналіз знайдених методів	28.04.20 – 1.05.20	
6	Розробка гібридного алгоритму	2.05.20 – 12.05.20	
7	Оформлення пояснювальної записки	12.05.20 – 17.05.20	
8	Підготовка до захисту	18.05.20	

Дата видачі завдання 30 березня 2020 р.

Студент _____
(підпис)

Керівник роботи _____ ст. викл. Носик А.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 73 с., 27 рис., 3 табл., 1 дод., 20 джерел.

ЦИФРОВЕ ЗОБРАЖЕННЯ, КЛАСТЕРИЗАЦІЯ, ПЕРЕДОБРОБКА, СИСТЕМИ РЕАЛЬНОГО ЧАСУ, СИСТЕМИ РОЗПІЗНАВАННЯ ОБРАЗІВ.

Метою атестаційної роботи є дослідити тему кластеризації цифрових зображень та розробити додаток, який реалізує гібридний алгоритм кластеризації цифрових зображень.

У ході виконання атестаційної роботи була розглянута тема кластерного аналізу, а саме актуальність, проблеми та перспективи розвитку цього напряму. Також розібрані різноманітні нюанси щодо формування та обробки цифрових зображень. Було описано теоретичний процес розробки гібридного алгоритму кластеризації цифрових зображень, від етапу передобробки зображення до етапу самої кластеризації.

У фінальній частині було розроблено прототип, який може проводити поділ вхідного цифрового зображення на кластери.

ABSTRACT

Master’s thesis:: 73 pages, 27 figures, 3 tables, 1 appendix, 20 sources.

DIGITAL IMAGE, CLUSTERING, PRE-PROCESSING, REAL-TIME SYSTEMS, IMAGE RECOGNITION SYSTEMS

The purpose of the certification work is to investigate the topic of digital image clustering and to develop an application that implements a hybrid algorithm for digital image clustering.

During the attestation work the topic of cluster analysis was considered, namely the relevance, problems and prospects of development in this area. Also various nuances of the formation and processing of digital images were analyzed. The theoretical process of developing a hybrid algorithm for clustering digital images, the type of the stage of image pre-processing to the stage of clustering itself were described.

In the final part, the prototype was developed that can divide the input digital image into clusters.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – інтерфейс прикладного програмування (англ., Application Programming Interface)

CMYK – компоненети колірної схеми cyan, magenta, yellow та black

RGB – компоненети колірної схеми red, green, blue

HSV – компоненети колірної схеми hue, saturation, value

HSB – компоненети колірної схеми hue, saturation, brightness

HSL – компоненети колірної схеми hue, saturation, lightness

LAB – компоненети колірної схеми lightness, green-red, blue-yellow

Осередок – коефіцієнт зниження розмірності сіткової моделі у колірному кубі RGB (в англомовній літературі – cell)

ВСТУП

У сучасному світі завдання обробки зображень щороку стає все більше і більше актуальним, оскільки дозволяє отримати від зображень різноманітну корисну інформацію. Наукова дисципліна ЕОМ, або технічного бачення, займається проблемою отримання інформації з цифрових зображень.

Комп'ютерний зір - це дисципліна, яка розробляє алгоритми і, як наступний етап, системи, які здатні ідентифікувати, відслідковувати і класифікувати об'єкти, розташовані в зображенні.

Системи, засновані на технології комп'ютерного зору, використовуються в різних галузях людської діяльності. Серед найбільш значних з них:

- медицина;
- робототехніка;
- безпека;
- промисловість;
- розваги;
- військові технології;
- наука.

Обробка великих масивів слабо споріднених даних логічно складна і дуже затратна в часі. Тому одним з найважливіших кроків в процесі отримання інформації з образу є стадія класифікації/кластеризації даних. На цьому етапі вихідні дані групуються в групи, вміст яких більше схожий в межах групи, ніж різниця між вмістом інших груп.

Кластеризація та класифікація процесів дуже схожі, основною відмінністю між ними є необхідність процесу класифікації, завдання чіткої кількості груп. У процесі кластеризації групи спочатку не встановлюються, але формуються під час роботи алгоритму.

Як і будь-яке інше завдання, яке відбувається під час обробки

зображення, кластеризація часто відбувається в режимі реального часу, і тому дуже чутлива до тривалості алгоритму.

За останні десятиліття створено велику кількість різних алгоритмів кластеризації. Деякі з них були спочатку розроблені для використання в задачі обробки зображень. Реалізація цих алгоритмів у вигляді коду може бути досить складною і трудомісткою. Для багатьох поширених мов програмування існують готові набори бібліотек з реалізованими алгоритмами кластеризації. Використання цих бібліотек дозволяє не витрачати час на початковому етапі дослідження і відразу ж встановити, який алгоритм найбільш підходить для вирішення конкретної проблеми.

Метою цієї роботи є дослідження існуючих реалізацій найбільш поширених алгоритмів кластеризації даних для процесу обробки зображень. В роботі обговорюється концепція кластеризації даних, проводиться аналіз найбільш поширених алгоритмів кластеризації, відомих до цього часу, і проводиться оцінка якості результатів. Завершенням роботи є розробка алгоритму гібридної кластеризації, який має поєднати найсильніші сторони існуючих алгоритмів та мінімізувати їх недоліки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Цифрове зображення

Під поняттям «цифрове зображення» розуміють створення кодованого цифрового представлення візуальних характеристик об’єкту.

1.1.1 Векторна графіка

Векторна графіка використовується в комп’ютерній графіці, це спосіб опису об’єктів на зображенні, ґрунтований на описі геометричних примітивів, таких як точки, лінії, криві Безье, кола, багатокутники, за допомогою математичних функцій.

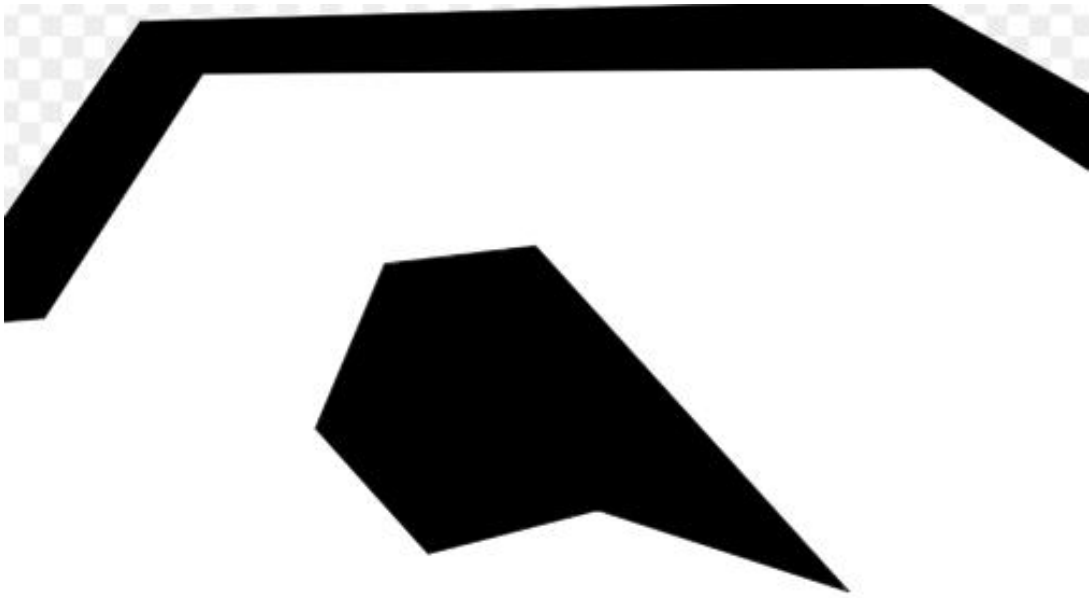

Рисунок 1.1 – Векторне зображення

Однією з головних переваг векторного зображення є відсутність проблеми масштабування. Завдяки способу опису об’єктів на зображенні, файли векторних зображень мають невеликий розмір і прекрасно масштабуються без втрати якості і зміни розміру.

Недоліки векторної графіки також є наслідком способу опису об’єктів. І головним з них є неможливість коректного опису "природних" зображень. У реальному світі практично відсутні ідеальні форми, і, як наслідок, при спробі описати фотографію доведеться використати величезну кількість формул, практично для кожної точки на зображенні, що веде до сильного збільшення займаного об’єму і необхідного часу для побудови зображення.

До ще одного недоліку векторної графіки можна віднести необхідність додаткового вивчення тонкощів процесів створення і редагування зображення унаслідок принципово іншого підходу в порівнянні з набагато поширеною растровою моделлю.

Найраціональніше - це використати векторну графіку для створення логотипів, іконок, технічних креслень і стилізованих зображень. Для обробки фото - реалістичних зображень - переважно обирають растрову графіку.

1.1.2 Растрова графіка

Растрова графіка - це спосіб опису зображення в комп’ютерній графіці за допомогою набору пікселів, найменших двовимірних логічних елементів, кожен з яких описується за допомогою визначення його розташування в двовимірному просторі і кольору у форматі певної колірної моделі.

До переваг растрової графіки можна віднести, в першу чергу, її універсальність і можливість створювати зображення будь-якої складності з великою кількістю різних дрібних деталей і тонкощів. Крім того, растрова графіка є простішою для розуміння, поширеніша і зручніша в обробці.

До головних недоліків растрової графіки варто віднести громіздкість опису і, як наслідок, великий займаний об’єм, а також складність в

масштабуванні.

Незважаючи на те, що завдання кластеризації зображень може бути успішно застосоване і до зображень у форматі векторної графіки, найбільш актуальною вона являється для зображень в растровому форматі, в першу чергу, унаслідок їх значно більшої поширеності в усіх сферах діяльності людини.

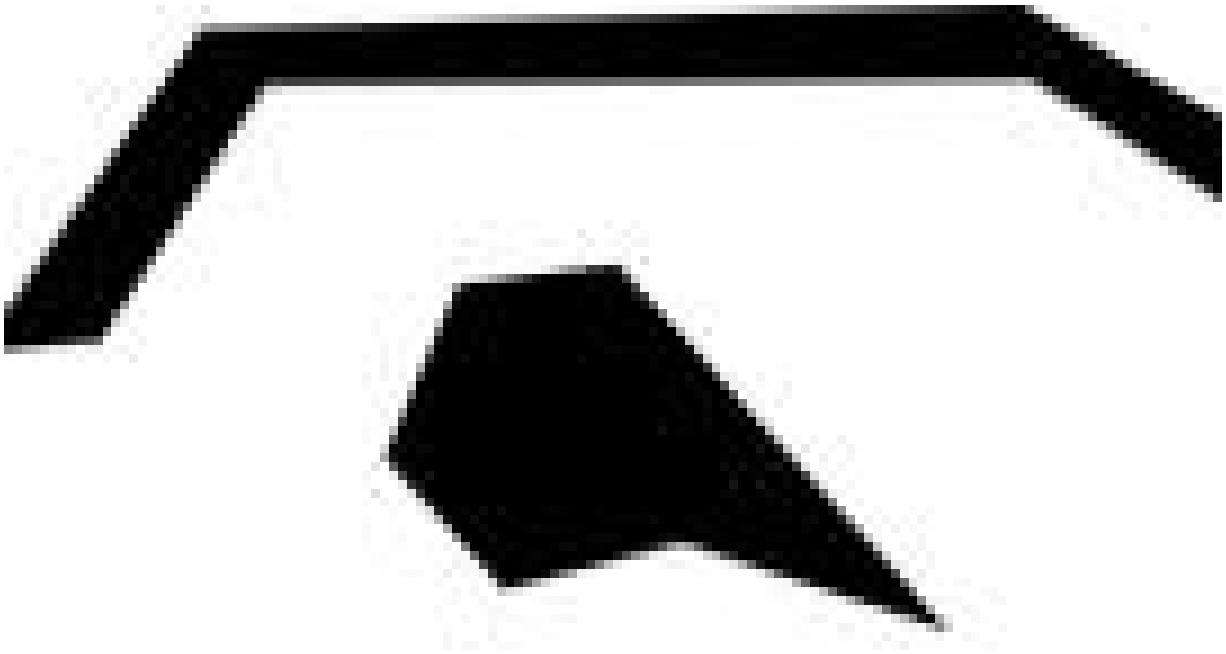

Рисунок 1.2 – Растрове зображення

Растрові зображення можуть зберігатися в різних графічних форматах, які відрізняються, в основному, в способі кодування інформації і, як наслідок, різними розмірами і якістю.

У форматі RAW файли RAW зберігаються без стиснення, зроблених на цифровій дзеркальній камері. З цієї причини ці файли, хоча вони містять максимальну кількість інформації про зображення, займають багато місця.

Це підходить для редагування фотографій, але не для їх зберігання, тому є стиснення зображень.

Формат BMP є одним з найстаріших форматів, які в даний час використовуються. Незважаючи на таку перевагу, як максимально можливий обсяг збереженої інформації про зображення, в даний момент цей формат майже не має значення через використання мало ефективних алгоритмів стиснення, так що зосередитися на цьому форматі при написанні алгоритму кластеризації є поганою ідеєю.

Формат PNG дуже схожий на BMP, але він використовує набагато більш ефективні алгоритми стиснення без втрати якості, що робить файли набагато меншими при аналогічній кількості інформації.

Формат JPG, мабуть, найвідоміший формат з втратою якості. Це найбільш ефективний формат з точки зору зберігання інформації на мінімальному томі. Це широко поширений формат, і необхідно зосередити увагу на розробці алгоритмів кластеризації при його використанні в першу чергу. Однією з основних особливостей формату JPG є використання необоротного стиснення. Як наслідок, відновити початкове зображення з цього формату неможливо.

Більшість сучасних пристроїв для запису фотографій і відео не тільки зберігають зображення JPEG, але й мають власні вбудовані алгоритми якості зображень, які часто ускладнюють процес кластеризації.

1.2 Колір в комп’ютерній графіці

Світло як фізичне явище являє собою потік електромагнітних хвиль різної довжини і амплітуди. Око людини, будучи складною оптичною системою, сприймає ці хвилі в діапазоні довжин приблизно від 350 до 780 нм. Світло сприймається або безпосередньо від джерела, наприклад, від освітлювальних приладів, або як відбитий від поверхонь об’єктів або переломлений при проходженні крізь прозорі і напівпрозорі об’єкти. Колір -

це характеристика сприйняття оком електромагнітних хвиль різної довжини, оскільки саме довжина хвилі визначає для ока видимий колір. Амплітуда, що визначає енергію хвилі (пропорційну квадрату амплітуди), відповідає за яскравість кольору. Таким чином, саме поняття кольору є особливістю людського ’бачення’ навколишнього середовища.

Фоторецептори поділяються на два види: палички і колбочки. Палички є високочутливими елементами і працюють в умовах слабкого освітлення. Вони нечутливі до довжини хвилі і тому не «розрізняють» кольори. Колбочки ж, навпаки, мають вузьку ланку спектральної кривої і «розрізняють» кольори. Паличок існує тільки один тип, а колбочки поділяються на три види, кожен з яких чутливий до певного діапазону довжин хвиль (довгі, середні або короткі.) Чутливість їх також різна.

Якщо світло, що сприймається, містить всі видимі довжини хвиль в приблизно рівних кількостях, то віно називається ахроматичним і при максимальній інтенсивності сприймається як білий, а при більш низьких значеннях інтенсивності - як відтінки сірого кольору. Якщо ж світло містить довжини хвиль в нерівних пропорціях, то воно є хроматичним.

1.2.1 Колірні моделі

Кольорова модель - це сукупність абсолютних або відносних параметрів кольору, що описують даний колір в даному колірному просторі.

Колір має різну фізичну природу. На моніторі ми бачимо колір, який випромінюється екраном, на папері - колір, відбитий аркушем паперу. Кольорові моделі призначені для опису кольорів, утворених різними методами.

Кольорові моделі в графічних програмах підтримуються спеціальними графічними режимами. Всі використовувані в даний час колірні моделі можна умовно класифікувати наступним чином:

а) монохромні:

- двоградаційні (дуплексні);
- напівтонові (з відтінками сірих кольорів).

б) кольорові:

- індексні;
- повнокольорові:
- адитивні (RGB), засновані на додаванні кольорів;
- субтрактивні (CMY, CMYK), засновані на відніманні кольорів;
- перцепційні (HSV, HSB, HLS, LAB, і т.д.), засновані на сприйнятті (інтуїції).

1.2.2 RGB

Ця модель отримала свою назву за першими літерами англійських слів Red (Червоний), Green (Зелений), Blue (Синій).

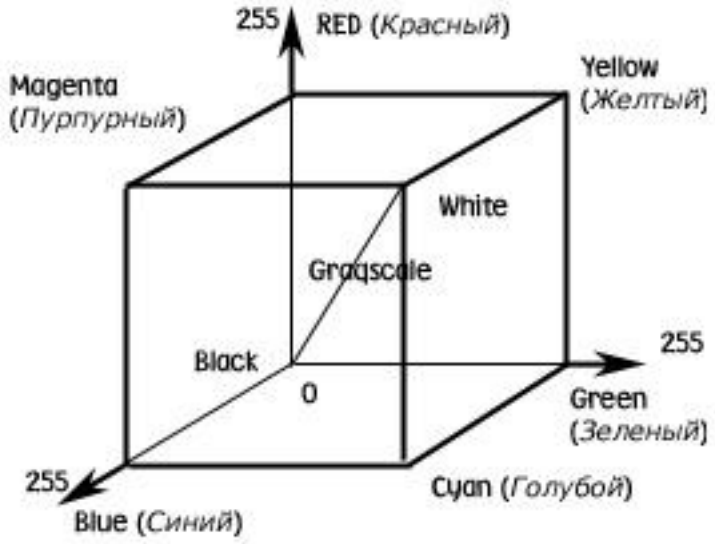

Рисунок 1.3 – Модель RGB

Модель RGB називають аддитивной (чинною), тому що будь-який колір в цій моделі утворюється шляхом змішування в різних пропорціях

трьох основних кольорів: червоного, зеленого і синього, які називаються первинними. При попарном змішуванні первинних кольорів утворюються вторинні кольори: блакитний, пурпурний і жовтий. Первинні і вторинні кольори називаються базовими кольорами.

Базовими кольорами називаються кольори, за допомогою яких можна отримати практично весь спектр видимих кольорів. Колір в даній колірній моделі описується трьома значеннями в діапазоні від 0 до 255. У тривимірній системі координат колірну модель RGB можна представити у вигляді куба.

Вершини куба, що розташовуються на осях, відповідають червоному, зеленому і синьому кольорам.

Подання колірної моделі RGB відбувається в наступному порядку: червона, зелена і синя складові. У цих точках відповідні складові мають максимальні значення: чистий червоний колір - 255, 0, 0 (рівень червоного максимальний, а зелена і синя складові відсутні):

- яскраво-червоний - 251, 61, 37;
- зелений колір - 0, 255, 0;
- темно-зелений - 29, 130, 104;
- синій - 0, 0, 255;
- точка початку координат відповідає чорному кольору (black) - 0, 0, 0 (жоден колір не випромінюється, всі складові рівні 0);
- в найближчій до глядача вершині куба рівні всіх трьох складових максимальні - це точка білого кольору (white) - 255, 255, 255;
- жовтий колір - 255, 255, 0;
- лимонний колір - 255, 255, 153.

Діагональ (R, G, B) = (0,0,0) і (R, G, B) = (255,255,255), що з’єднує точки чорного і білого кольорів - ахроматична вісь (шкала сірого Grayscale) - містить 256 відтінків сірого. На цій осі значення складові червоного, зеленого і синього кольорів однакові (R, G, B) = (50,50,50).

Таким чином, будь-який колір в цій моделі може бути представлений в колірному просторі за допомогою вектора, що описується рівнянням:

$$CC = rR + gG + bB$$

Обмеження RGB-моделі.

Апаратна залежність:

- колір, що виникає в результаті змішування колірних складових RGB елемента, залежить від типу люмінофора. У технології виробництва сучасних кінескопів знаходять застосування різні типи люмінофорів (мають різну спектральну характеристику) і установка одних і тих же інтенсивностей електронних променів в разі різних люмінофорів приведе до синтезу різного кольору;

- старіння люмінофора і зміна характеристик електронних прожекторів.

Для усунення (або принаймні мінімізації) залежності RGB-моделі від апаратних засобів використовуються різні пристрої і програми градуювання.

1.2.3 Субтрактівні колірні моделі CMY і CMYK

На відміну від екрану монітора, відтворення кольорів якого заснована на випромінюванні світла, друкована сторінка може тільки відображати колір. Тому, в даному випадку RGB - модель неприйнятна. Замість неї для опису друкованих кольорів використовується модель CMY, що базується на субтрактивних кольорах.

Модель CMY описує кольори, отримані в результаті відбиття світла об’єктами. Субтрактивні кольори, на відміну від адитивних кольор, виходять вирахуванням вторинних кольорів (блакитний, пурпурний і жовтий) із загального променя світла. Жовтий, пурпурний і блакитний є базовими для цієї колірної моделі.

Колірна модель CMY є зворотною моделі RGB, тому білий колір - це повна відсутність фарби - білий аркуш паперу (рівні всіх трьох складових дорівнюють 0), присутність всіх кольорів дає чорний колір (рівні всіх трьох

складових мають максимальні значення).

На практиці за допомогою трьох базових фарб (Cyan, Magenta, Yellow) не можна отримати весь колірний діапазон, а при змішуванні всіх трьох складових колір виходить не чисто чорним, а брудно-коричневим.

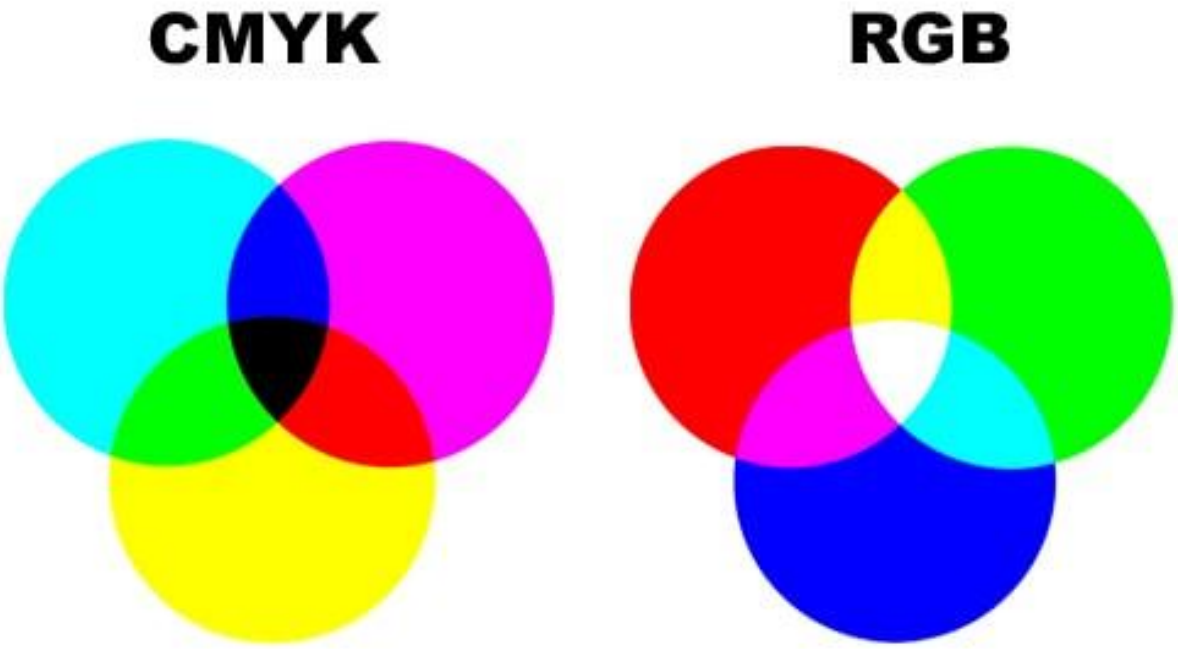

Рисунок 1.4 – Моделі CMYK та RGB

Для усунення даного недоліку до трьох фарб додали четверту - чорну (Black), і колірна модель отримала назву CMYK - Cyan, Magenta, Yellow, Black. У слові Black використовується не перша буква, а остання, щоб уникнути плутанини з кольором Blue моделі RGB.

Кольори в моделі CMYK утворюються шляхом вирахування з чорного кольорів жовтого (Yellow), пурпурного (Magenta) і блакитного (Cyan). Тому модель CMYK називається субтрактивной (віднімає).

На відміну від моделі RGB, в колірній моделі CMYK рівень складових задається значеннями в діапазоні від 0 до 100% (величина 100% в моделі CMYK відповідає 255 одиницям в моделі RGB).

Обмеження моделі CMYK.

CMYK-модель має ті ж два типу обмежень, що і RGB-модель:

1. Апаратна залежність. CMYK-модель є навіть більш апаратно-залежною моделлю, ніж RGB. Це пов’язано з тим, що в ній є більша кількість дестабілізуючих факторів, ніж в RGB-моделі: варіація складу кольорових барвників; тип вживаного паперу, спосіб друку; зовнішнє освітлення.

2. Обмежений колірний діапазон. Кольорові барвники мають гірші характеристики в порівнянні з люминофорами, тому колірна модель CMYK має вужчий колірний діапазон в порівнянні з RGB-моделлю. Про екранних кольорах, які неможливо відтворити при друку, кажуть, що вони лежать поза колірного охоплення моделі CMYK. Під такими кольрами розуміють кольори, які можуть бути представлені в форматі RGB, але при цьому не мають друкованих аналогів в колірному просторі CMYK.

Наприклад: розбіжність кольорів, що відображаються на екрані монітора, друкується на принтері. В результаті, отримана в результаті напруженої роботи прекрасна картинка на екрані монітора при роздруківці раптом перетворюється погану подобу оригіналу.

1.2.4 Колірна модель HSB

Колірна модель HSB розроблена з максимальним урахуванням особливостей сприйняття кольору людиною. Колір описується трьома компонентами: відтінком (Hue), насиченістю (Saturation) і яскравістю (Brigfitness).

Значення кольору вибирається як вектор, що виходить із центру кола. Точка в центрі відповідає білому кольору, а точки по периметру кола - чистим спектральним кольорам. Напрямок вектора задається в градусах і визначає колірний відтінок. Довжина вектора визначає насиченість кольору. На окремій осі, званої ахроматичною, задається яскравість, при цьому нульова точка відповідає чорному кольору.

Колірний обхват моделі HSB перекриває всі відомі значення реальних кольорів. Модель HSB прийнято використовувати при створенні зображень на комп'ютері з імітацією прийомів роботи і інструментарію художників.

Під колірним тоном розуміється світло з домінуючою довжиною хвилі. Параметр Hue (Тон) характеризується становищем на колірному колі і визначається вершиною кута (від 0 до 360).

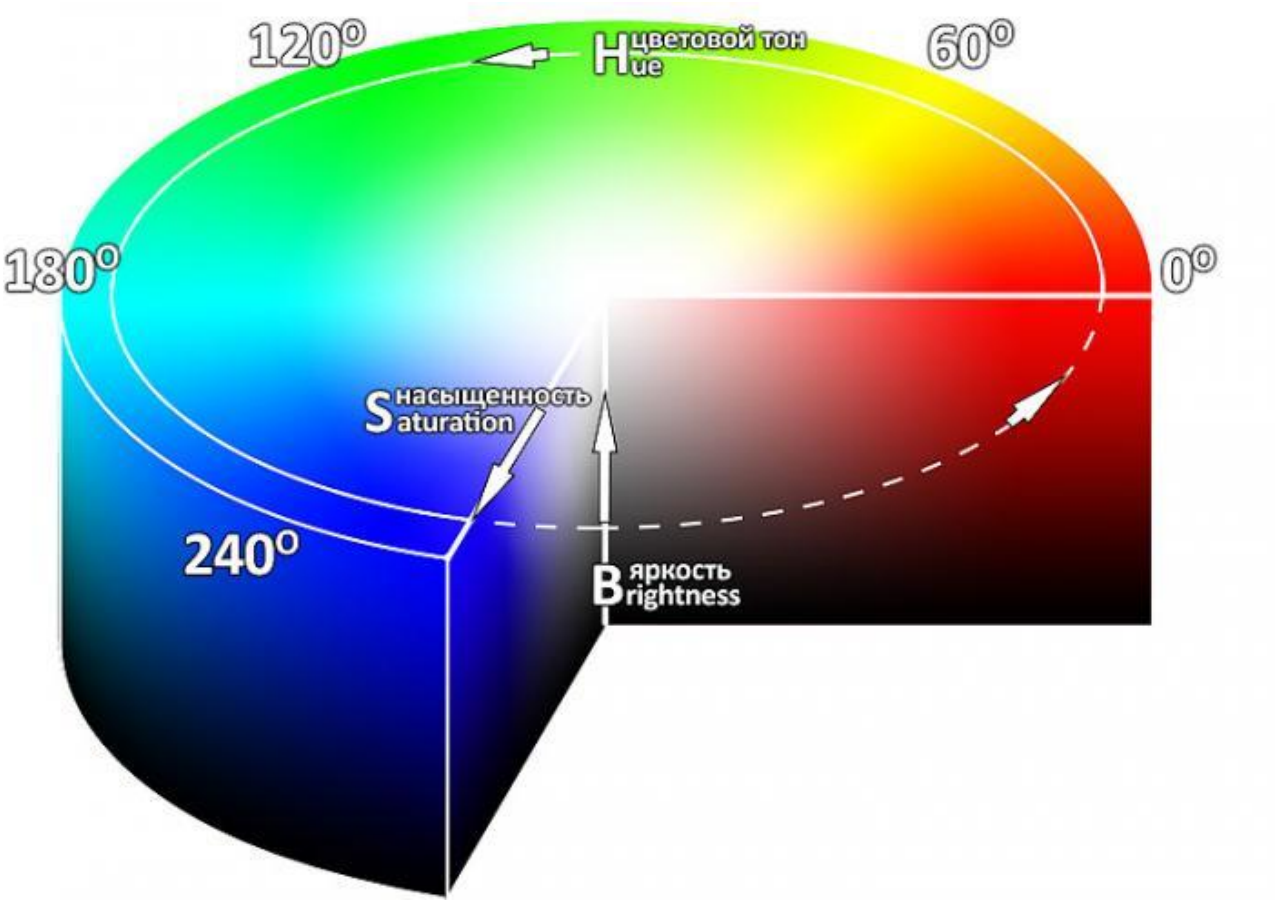

Рисунок 1.5 – Модель HSB

На колірному колі первинні кольори розташовані на рівній відстані один від одного. Вторинні кольори знаходяться між первинними. У свою чергу, кожен колі, розташований навпроти, доповнює його (комплиментарно) колір і знаходиться між двома кольорами, з яких отримано.

Параметр Saturation визначає насиченість кольору. Збільшення насиченості призводить до збільшення концентрованості кольору, а

зменшення - до його розбілюванню.

Насиченість має максимальне значення на окружності - 100% і мінімальне в центрі кола - 0%.

Параметр Brightness (яскравість) визначає ступінь освітленості або затемненості кольору - це інтенсивність, з якою енергія світла впливає на рецептори нашого ока. Величина яскравості вимірюється в % в діапазоні від 0% (чорний колір) до 100% (білий колір).

Ахроматичні кольори, тобто білі, сірі та чорні, характеризуються тільки яскравістю.

Цю модель можна представити у вигляді циліндра, в якому:

- контур підстави (окружність) відповідає осі зміни параметра Hue;
- радіус підстави - осі зміни параметра Saturation;
- бічна сторона - осі зміни параметра Brightness.

Ця модель більше, ніж інші, відповідає традиційному сприйняттю кольору людиною і найбільш проста в розумінні: спочатку можна визначити колірний тон, а потім задати йому насиченості і яскравість. Крім того, модель HSB зручно використовувати при редагуванні малюнків. Наприклад, ви хочете замінити зелений лист на жовтий редагованої фотографії. Досить поміняти тільки колірну складову використовуваних кольорів, не змінюючи яскравість і насиченість. Малюнок при цьому не зміниться, але прийме інший відтінок.

Яскравість і колірний тон не є повністю незалежними параметрами. Зміни яскравості зображення впливають на зміну колірного тону, що створює небажане колірне зрушення в зображенні.

Наприклад, при значному зменшенні яскравості зелені кольори синіють, сині - наближаються до фіолетових, жовті - до помаранчевих, а помаранчеві - до червоних. Сильне збільшення яскравості випромінювання викликає інший ефект. Червоні кольори переходять в помаранчеві, потім в жовті і, нарешті, - в білі.

Недолік колірної моделі HSB: так само як і в попередніх колірних

моделях - обмежений колірний простір.

Переваги:

- апаратна незалежність;
- більш простий і інтуїтивно зрозумілий механізм управління кольором.

1.2.5 Колірна модель LAB

Колір в даній колірній моделі визначається трьома параметрами, два з яких - хроматичні компоненти:

- a - кольоровість в діапазоні від зеленого до червоного;
- b - кольоровість в діапазоні від синього до жовтого;
- L - світлота (Lightnesss), що представляє собою аналог яскравості.

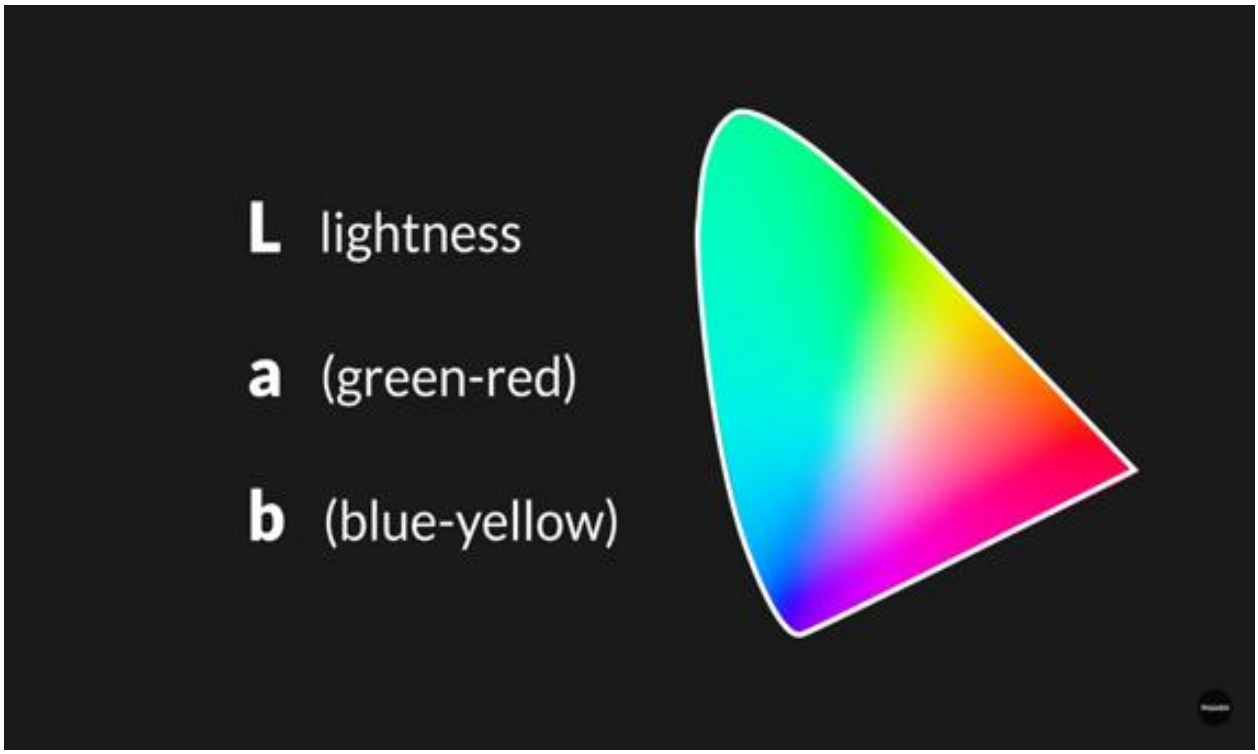

Рисунок 1.6 – Модель LAB

Параметри *a* і *b* задаються числами, що знаходяться в діапазоні від -120 до 120. Для параметра *a* значення -120 відповідає зеленому кольору, а +120 - червоному. Для параметра *b* значення -120 - це синій колір, а значення +120 - жовтий. Все за умови, що L дорівнює 100%. Светлота змінюється в діапазоні від 0 до 100%.

Використовувані в Lab - моделі колірні координати узгоджуються з біологічним механізмом сприйняття кольору, відкритим в 1981 році американськими вченими Давидом Хьюблом (David H. Hubel) і Торстеном Вайзелом (Torsten N. Wiesel), які отримали Нобелівську премію за дослідження зору. Вони довели, що очі надають в мозок зовсім не інформацію про червоний, зелений і синій. Замість цього мозок отримує:

- різницю між світлим і темним;
- різницю між зеленим і червоним.

Схема колірного зору.

На горизонтальному зрізі всі кольори мають однакову яскравість. Це означає, що кожен колір може бути точно описаний в колірних координатах *a* й *b*.

Переваги Lab - моделі:

- апаратна незалежність;
- більший колірний обхват в порівнянні з RGB- і CMYK- моделями;
- на базі параметрів цієї колірної системи можна визначити параметри інших колірних моделей.

1.2.6 Колірні моделі HSV и HLS

Як було сказано, психофізіологічне сприйняття світла визначається не інтенсивністю трьох первинних кольорів, а колірним тоном, насиченістю і світлиною. Тон дозволяє розрізняти кольори, насиченість задає ступінь ’розведення’ чистого тону білим кольором, а світлота - це інтенсивність світла в цілому. Тому для адекватного нашому сприйняттю підбору відтінків

зручнішими є моделі, в числі параметрів яких присутній тон (Hue). Цей параметр прийнято вимірювати кутом, відлічуваним навколо вертикальної осі. При цьому червоному кольору відповідає кут 0°, зеленому - 120°, синьому - 240°, а доповнюють один одного кольори, розташовані один навпроти іншого, тобто кут між ними становить 180°. Кольори CMY розташовані посередині між складовими їх компонентами RGB. Існує дві моделі, які використовують цей параметр.

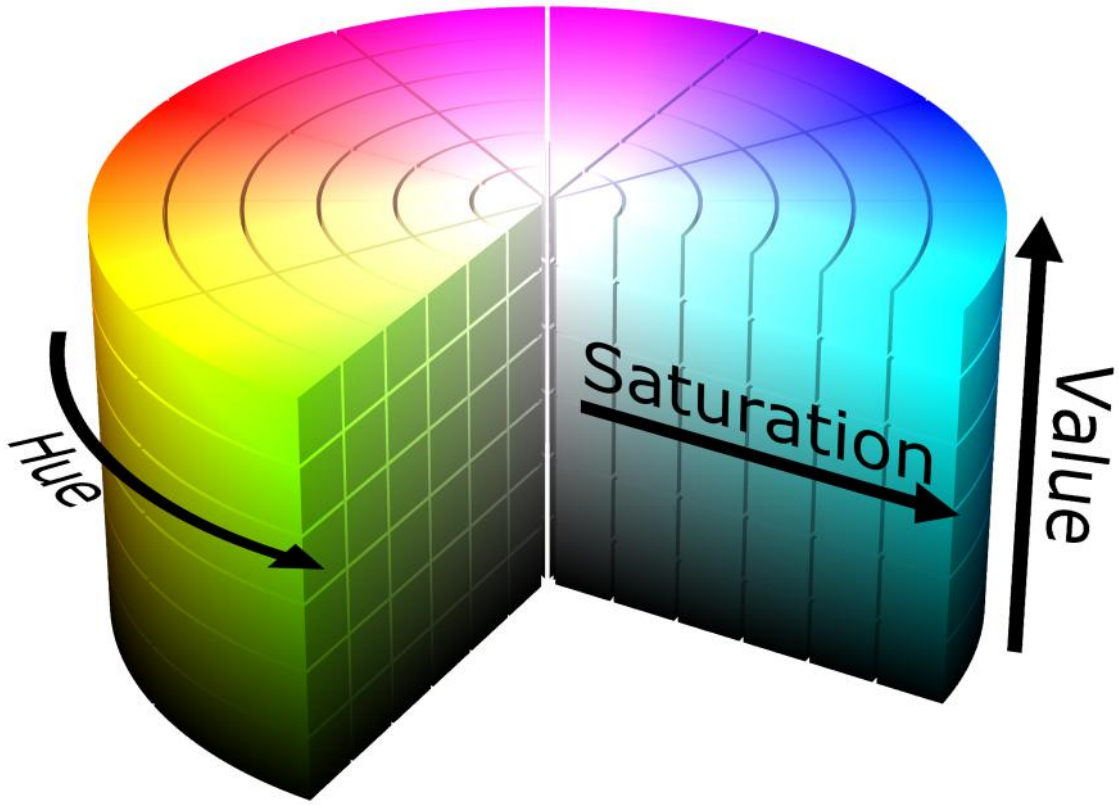

Рисунок 1.7 – Модель HSV

Модель HSV (Hue, Saturation, Value, або тон, насиченість, кількість світла) можна представити у вигляді світлової шестигранної піраміди, по осі якої відкладається значення *V*, а відстань від осі до бічної грані в горизонтальному перетині відповідає параметру *S* (за діапазон зміни цих величин приймається інтервал від нуля до одиниці). Значення *S* дорівнює

одиниці, якщо точка лежить на бічній грані піраміди. Шестикутник, що лежить в основі піраміди, є проекцією колірного куба в напрямку його головної діагоналі.

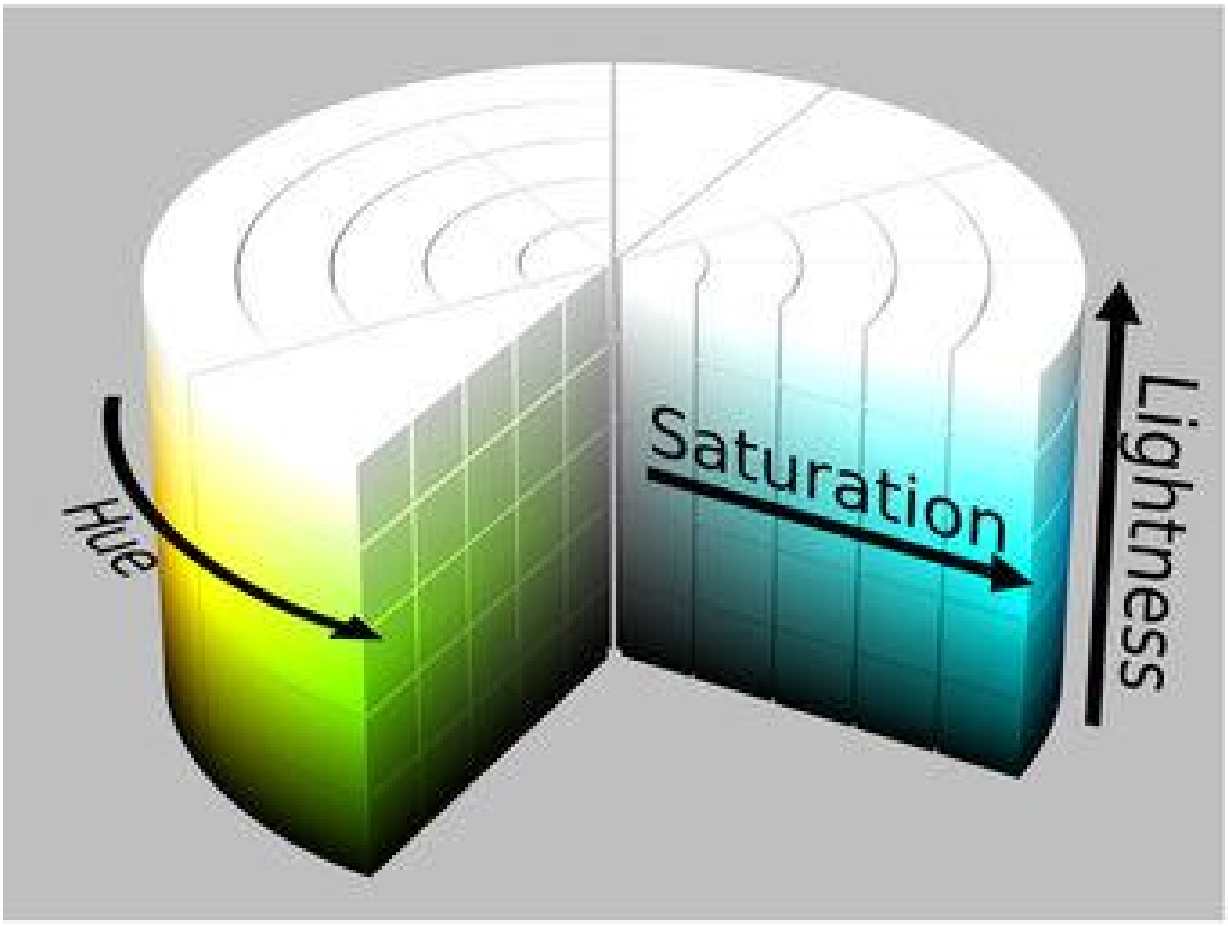

Рисунок 1.8 – Модель HSL

Колірна модель HLS (Hue, Lightness, Saturation, або тон, світлина, насиченість) є розширенням моделі HSV. Тут колірний простір уже представляється у вигляді подвійної піраміди, в якій на вертикальній осі відкладається L (світлина), а інші два параметри задаються так само, як і в попередній моделі. У літературі ці піраміди іноді називають шестигранним конусом.

Порівняння моделей HSL і HSV:

- насиченість в моделі HSL завжди змінюється від повністю

насиченого кольору до еквівалентного сірому кольору, в той час як в моделі HSV при $V = 1$ повністю насичений колір переходить до білого.

- освітлення в моделі HSL змінюється від чорного через вибране значення кольоровості - до білого, а в моделі HSV - проходить лише половину шляху - від чорного до вибраного кольоровому.

1.3 Поняття кластеризації

Кластеризація - це процес об’єднання об’єктів в групи, відповідно до їх схожості. Процес кластеризації є однією з базових задач в області аналізу даних і Data Mining. Список прикладних областей, де вона застосовується, широкий: сегментація зображень, аналіз текстів, маркетинг, прогнозування, боротьба з шахрайством і багато інших.

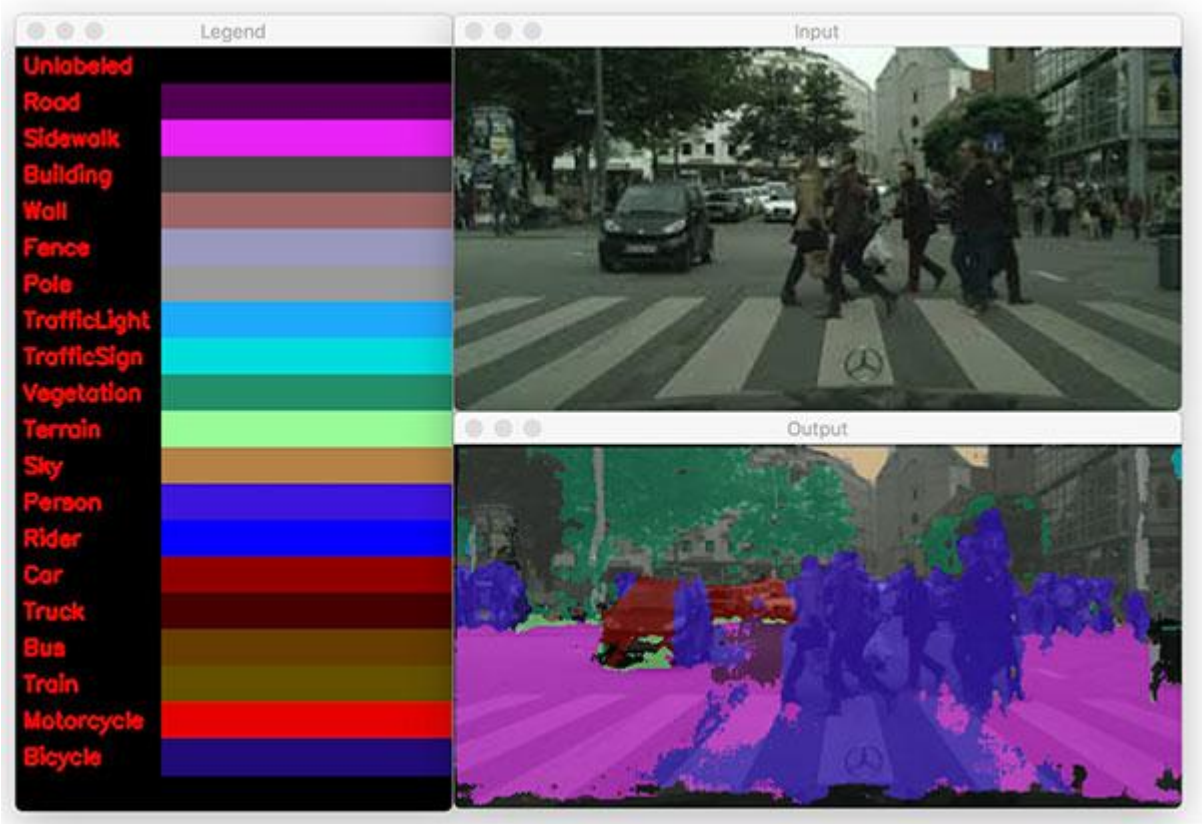

Рисунок 1.8 – Приклад кластеризації

Каутеризація набуває цінність головним чином тоді, коли вона виступає в ролі одного з базових етапів аналізу даних, перепідготовки, при побудові повного алгоритму аналітичного рішення. При дослідженні найчастіше простіше визначити групи схожих об’єктів, і на базі їх особливостей побудувати для кожної групи персональну модель, а не створювати єдину модель для всього масиву даних.

Далі в роботі будуть розглянуті найбільш поширені алгоритми кластеризації з метою визначення найбільш оптимального алгоритму для вирішення завдання кластеризації об’єктів на зображенні.

Кластеризація (або кластерний аналіз) - це задача розбиття множини об’єктів на групи, які називаються кластерами.

Усередині кожної групи повинні виявитися «схожі» об’єкти, а об’єкти, що входять в різні групи, повинні бути якомога більш відмінні.

Головна відмінність кластеризації від класифікації полягає в тому, що при кластеризації перелік груп чітко не заданий і визначається в процесі роботи алгоритму.

Застосування кластерного аналізу в загальному вигляді зводиться до наступних етапів:

- складання вибірки об’єктів для кластеризації;
- вибір набору змінних, за якими будуть порівнюватися об’єкти у вибірці, і, при необхідності - нормалізація значень;
- обчислення відстані між об’єктами;
- застосування методу кластерного аналізу для створення груп схожих об’єктів (кластерів);
- останнім етапом є представлення результатів аналізу, при цьому при необхідності здійснюється коригування обраної метрики і методу кластеризації з метою отримання оптимального результату.

1.3.1 Міри відстанні

Розподіл об’єктів під час кластеризації відбувається на основі їх «подібності». Для кожного об’єкту є виділення, ряд числових або менш поширених характеристик, при порівнянні яких розраховується відстань між об’єктами. Чим менше результат, тим більше "схожість" пари об’єктів в питанні. Нерідко при розрахунку відстані, для рівного внеску в результат, необхідно нормалізувати характеристики об’єкта до обраного асортименту провідника.

На даний момент існує багато алгоритмів або метрик, обчислюючих відстань. Як найбільш поширене, слід відзначити наступне.

Евклідова відстань.

Найпоширеніший алгоритм розрахунку. Геометрична відстань в багатовимірному просторі. Застосування її є актуальним у випадку з простором знаків в геометричному просторі:

$$\rho(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

Квадрат Евклідової відстані.

Він використовується при необхідності, для збільшення відстаней значень віддалених об’єктів один від одного. При необхідності конструкція може бути використана в значній мірі:

$$\rho(x, x') = \sum_{i}^{n} (x_i - x_i')^2$$

Манхеттенска відстань, або відстань міських кварталів.

Вона будується шляхом розрахунку середньої диференціації кожної з координат. Результат є схожий на стандартну відстань Евкліда. Але через

відсутність квадратної операції, ефект від різних характеристик зменшується.

$$\rho(x,x') = \sum_{i}^{n} | x_i - x_i^{'} |$$

Відстань Чебишева.

Застосовують при необхідності визначення міри подібності як найбільшої відстані між двома характеристиками.

$$\rho(x,x') = \max(| x_i - x_i^{'} |)$$

Вибір міри відстані і ваги характеристик об’єктів надзвичайно важливий при кластеризації, і саме від цих операцій залежить результат дослідження. Слід розуміти, що найчастіше оптимальним варіантом для розрахунку дистанції стають гібридні варіанти, використовуючи, як існуючі алгоритми, так і, можливо, деякі операції, запропоновані дослідником.

1.4 Класифікація алгоритмів кластеризації

Прийнято визначати дві основні класифікації алгоритмів кластеризації:

а) ієрархічні і плоскі алгоритми;

Ієрархічні алгоритми (інакше алгоритми таксономії) будують не одне розбиття вибірки на непересічні кластери, а систему вкладених кластерів. При цьому на виході ми отримуємо дерево кластерів, коренем якого є вся вибірка, а листям - найбільш дрібні кластери.

Плоскі алгоритми будують одне розбиття об’єктів на кластери.

б) чіткі і нечіткі алгоритми.

Чіткі (або непересічні) алгоритми привласнюють кожному об’єкту вибірки певний номер кластера, в результаті кожен об’єкт належить єдиному кластеру.

Рисунок 1.9 – Алгоритми кластеризіції

Нечіткі (або пересічні) алгоритми кожному об’єкту визначають набір речових значень, що показує ступінь належності об’єкта до кластерів. Таким чином, кожен об’єкт може належити до кількох кластерів з різною ймовірністю.

1.4.1 Алгоритми ієрархічної кластеризації

Серед алгоритмів ієрархічної кластеризації виділяються два основних типи: низхідні і висхідні алгоритми. Низхідні алгоритми працюють за принципом «зверху-вниз»: на початку всі об’єкти поміщаються в один кластер, який під час роботи алгоритму розбивається на все більш дрібніші.

У більшості випадків застосовуються висхідні алгоритми, в яких на старті кожен об’єкт поміщається в окремий кластер, а потім відбувається об’єднання кластерів в усі більші, до моменту, поки всі об’єкти вибірки не сконцентруються в одному кластері.

Таким чином відбувається побудова системи вкладених кластерів. Результат роботи подібного алгоритму зазвичай представляють у вигляді дерева.

Рисунок 1.10 – Приклад ієрархічного алгоритму

Для обчислення відстаней між кластерами зазвичай користуються двома типами відстаней: одиночним та повним зв’язком.

До основного недоліку алгоритмів даного типу відноситься система повної розбивки, яка при вирішенні певних завдань може виявитися надмірною.

1.4.2 Об’єднання кластерів

При використанні ієрархічних алгоритмів перед дослідником постає питання вибору метрики об’єднання кластерів.

До найбільш поширених метрик відносяться:

а) одиночний зв’язок (відстань до найближчого сусіда);

У цьому методі відстань між двома кластерами обчислюється як відстань між двома найближчими об’єктами (найбільш близькими сусідами) в

різних кластерах.

б) повний зв’язок (відстань між найбільш віддаленими сусідами);

У цьому методі відстанню між двома кластерами є найбільша відстань між об’єктами в різних кластерах (інакше - найбільш віддаленими сусідами). Цей метод добре працює, коли об’єкти походять з окремих груп.

в) незважене попарне середнє;

У цьому методі відстань між двома різними кластерами обчислюється як середня відстань між усіма парами об’єктів в них. Метод досить ефективний, але дуже витратний у плані обчислювальних ресурсів.

г) зважене попарне середнє;

Метод ідентичний попередньому, але при обчисленні вводиться ваговий коефіцієнт, заснований на розмірах оброблюваних кластерів. Таким чином цей метод доцільно використовувати в разі передбачуваних нерівних розмірів кластерів. Також досить ефективний, але ще більш вимогливий до часу виконання.

д) незважений центроїдний метод;

Відстань між центрами тяжіння двох кластерів, за цим методом, є власне відстанню між двома кластерами

е) зважений центроїдний метод (медіана).

Модифікація попереднього методу. За рахунок введення додаткового вагового коефіцієнта дозволяє ефективно оперувати кластерами різного розміру.

1.4.3 Алгоритми квадратичної похибки

Завдання кластеризації можна розглядати як побудову оптимального розбиття об’єктів на групи. При цьому оптимальність зазвичай визначається за допомогою мінімальної середньоквадратичної помилки розбиття:

$$e^2(X,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i^{(j)} - c_j$$

де c_j — «центр мас» кластера, j - крапка із середніми значеннями характеристик для даного кластера, K – кількість кластерів, n – кількість крапок.

Алгоритми квадратичної помилки відносяться до типу плоских алгоритмів. Базовим алгоритмом для цієї категорії є метод k-середніх. Даний алгоритм розбиває безліч об’єктів на заздалегідь задане число кластерів.

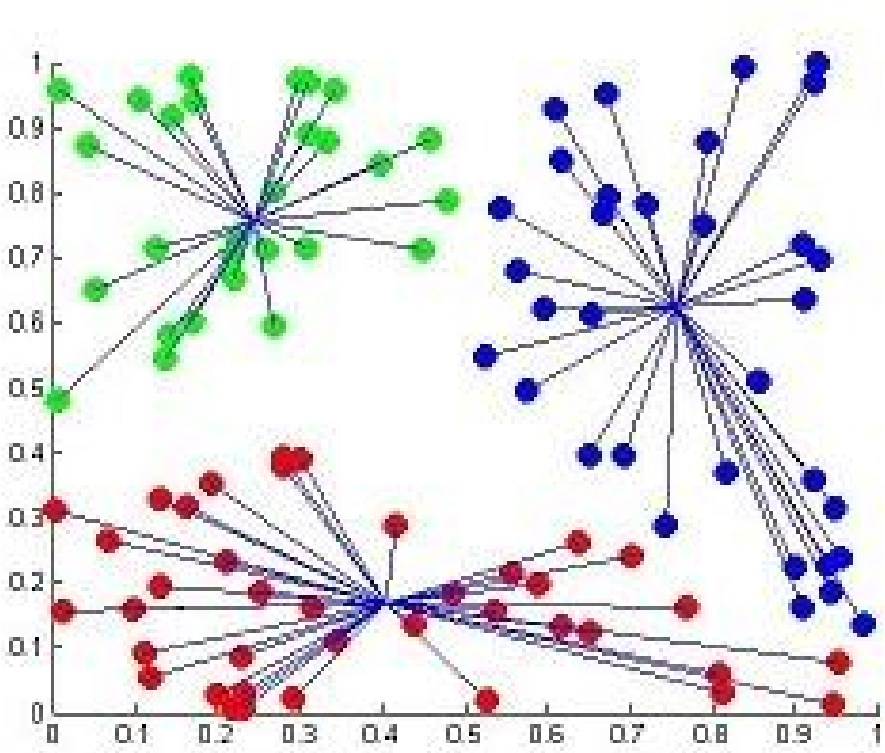

Рисунок 1.11 – Алгоритм k-means

Робота алгоритму складається з наступних етапів:

- визначення k точок, які призначаються початковими «центрами мас» кластерів.
- розподіл об’єктів по кластерам згідно найближчого «центру мас».
- обчислення нового «центру мас» кластерів щодо їх поточного складу.
- якщо критерій зупинки алгоритму не виконано, повернутися до п. 2.

Критерію зупинки роботи алгоритму зазвичай задається мінімальна зміна середньоквадратичної помилки. Іншим поширеним варіантом критерію зупинки роботи алгоритму є відсутність об’єктів, що перемістилися з кластера в кластер.

Головним недоліком алгоритму k-середніх є необхідність перед початком роботи чіткого визначення кількості кластерів для розбиття.

1.4.4 Нечіткі алгоритми

Найбільш популярним алгоритмом нечіткої кластеризації є алгоритм c-середніх (c-means). Він являє собою модифікацію методу k-середніх. Кроки роботи алгоритму:

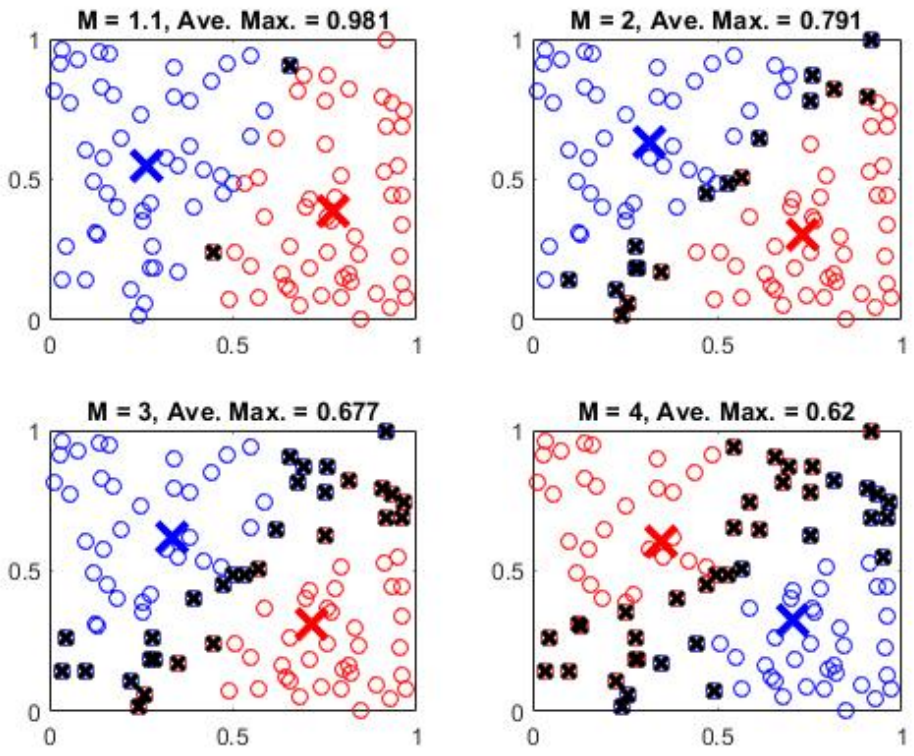

Рисунок 1.12 – Алгоритм c-means

- вибрати початкове нечітке розбиття n об’єктів на k кластерів шляхом

вибору матриці приналежності U розміру (n, k).

- використовуючи матрицю U, знайти значення критерію нечіткої помилки.

- перегрупувати об’єкти з метою зменшення цього значення критерію нечіткої помилки.

- повертатися в п. 2 до тих пір, поки зміни матриці U не стануть незначними.

Цей алгоритм може не підійти, якщо заздалегідь невідомо число кластерів, або необхідно однозначно віднести кожен об’єкт до одного кластеру.

1.4.5 Алгоритми, засновані на теорії графів

Суть таких алгоритмів полягає в тому, що вибірка об’єктів представляється у вигляді графа G = (V, E), вершинам якого відповідають об’єкти, а ребра мають вагу, рівну «відстані» між об’єктами.

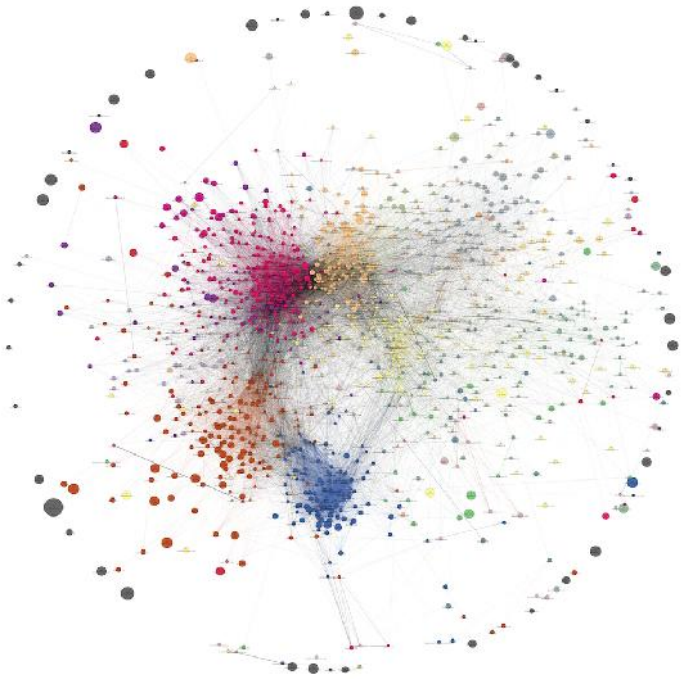

Рисунок 1.13 – Кластеризація за допомогою графів

Перевагою графових алгоритмів кластеризації є наочність, відносна простота реалізації і можливість внесення різних удосконалень, заснованих на геометричних міркуваннях.

Основними алгоритмами є алгоритм виділення зв’язкових компонент, алгоритм побудови мінімального покрива дерева і алгоритм пошарової кластеризації.

1.4.6 Алгоритм виділення зв’язкових компонент

В алгоритмі виділення зв’язкових компонент задається вхідний параметр R, і в графі видаляються всі ребра, для яких «відстані» більше R. Сполученими залишаються тільки найбільш близькі пари об’єктів. Сенс алгоритму полягає в тому, щоб підібрати таке значення R, що лежить в діапазоні всіх «відстаней», при якому граф «розвалиться» на кілька зв’язкових компонент. Отримані компоненти і є кластери.

Для підбору параметра R, зазвичай, будується гістограма розподілів попарних відстаней. У завданнях з добре вираженою кластерною структурою даних на гістограмі буде два піки - один відповідає внутрішньокластерним відстаням, другий - міжкластерній відстані. Параметр R підбирається із зони мінімуму між цими піками. При цьому управляти кількістю кластерів за допомогою порога відстані досить важко.

1.4.7 Алгоритм мінімального покрива дерева

Алгоритм мінімального покрива дерева спочатку будує на графі мінімального покрива дерево, а потім послідовно видаляє ребра з найбільшою вагою. Шляхом видалення зв’язку з позначкою CD, з довжиною рівною 6 одиницям (ребро з максимальною відстанню), отримуємо два кластери: {A, B, C} і {D, E, F, G, H, I}. Другий кластер в подальшому може

бути розділений ще на два кластери шляхом видалення ребра EF, яке має довжину, рівну 4,5 одиницям.

1.4.8 Пошарова кластеризація

Алгоритм пошарової кластеризації заснований на виділенні зв’язкових компонент графа на деякому рівні відстаней між об’єктами (вершинами).

Алгоритм пошарової кластеризації формує послідовність підграфів графа G, які відображають ієрархічні зв’язки між кластерами.

За допомогою зміни порогів відстані можливо контролювати глибину ієрархії одержуваних кластерів. Таким чином, алгоритм пошарової кластеризації здатний створювати як плоске розбиття даних, так і ієрархічне.

2 ПЕРЕДОБРОБКА ЗОБРАЖЕНЬ

2.1 Проблема швидкості обробки зображень в системах реального часу

Одним з найбільш актуальних завдань кластеризації цифрових зображень є операції розпізнавання образів. Завдання розпізнавання образів в сучасному світі найбільш часто виникає в системах, що працюють в режимах реального часу.

В останні роки системи розпізнавання образів, що працюють в режимі реального часу, знаходять все більше застосування. Якщо 10 років тому подібні системи використовувалися досить "нішево" і були доступні переважно в армії і поліції, то зараз вони широко використовуються в сфері розваг, охорони здоров’я, охоронної служби і в багатьох інших сферах діяльності людини.

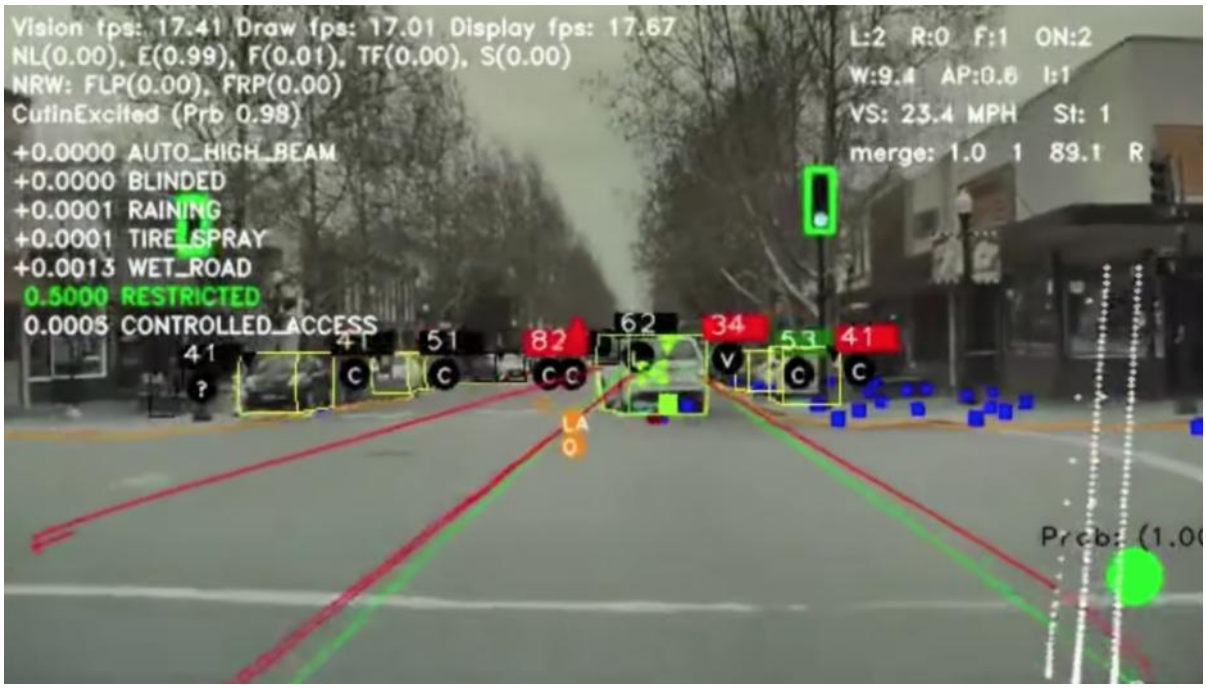

Рисунок 2.1 – Комп’ютерний зір автопілоту Tesla

Наприклад, за останні роки технології розпізнавання образів почали застосовуватися при розробці систем автоматичного пілотування автомобілів. Якщо в сфері розваг можливі помилки в роботі систем розпізнавання образів не є небезпечними, то у випадках з автопілотом найменші помилки розробників можуть призводити і, на жаль, призводять до трагічних наслідків.

У зв’язку з тим, що розпізнавання образів відбувається в режимі реального часу, вкрай гостро стоїть проблема швидкодії алгоритму. На даний момент існує ряд методів, таких як зменшення зображення, застосування сіткової моделі, які дозволяють на етапі попередньої обробки зменшити масив вхідних даних, по можливості уникнувши значних втрат в якості інформації.

2.2 Алгоритми масштабування зображень

Масштабування зображень є дуже важливим етапом ланцюжка обробки зображень. Трудомісткість будь-яких операцій, пов’язаних з обробкою зображень (отриманням будь-якої корисної інформації) безпосередньо залежить від розміру зображення і збільшується в геометричній прогресії. Саме тому настільки важливо використовувати максимально швидкий і якісний алгоритм масштабування зображення. Нижче будуть розглянуті найбільш поширені алгоритми масштабування цифрових зображень. У тому чи іншому варіанті вони представленні в багатьох програмах редагування зображень, та, у вигляді окремих бібліотек, у практично усіх більш-менш розповсюджених мовах програмування.

2.2.1 Алгоритм найближчого сусіда (nearest neighbor)

Це найпримітивніший і, як наслідок, один з самих швидких, метод. Для кожного пікселя кінцевого зображення вибирається один піксель вихідного,

найбільш близький до його положення з урахуванням масштабування. Такий метод дає пікселізовану картинку при збільшенні і сильну зернистість при зменшенні.

Взагалі, якість і продуктивність будь-якого методу зменшення можна оцінити по відношенню кількості пікселів, які брали участь у формуванні кінцевого зображення, до числа пікселів в оригінальному документі. Чим більше це відношення, тим більше вірогідність, що алгоритм якісніше, але і повільніше. Відношення, рівне один к одному, означає що кожен піксель вихідного зображення зробив свій внесок в кінцеве. Але для просунутих, найбільш ефективних, методів це відношення може бути і більше одного.

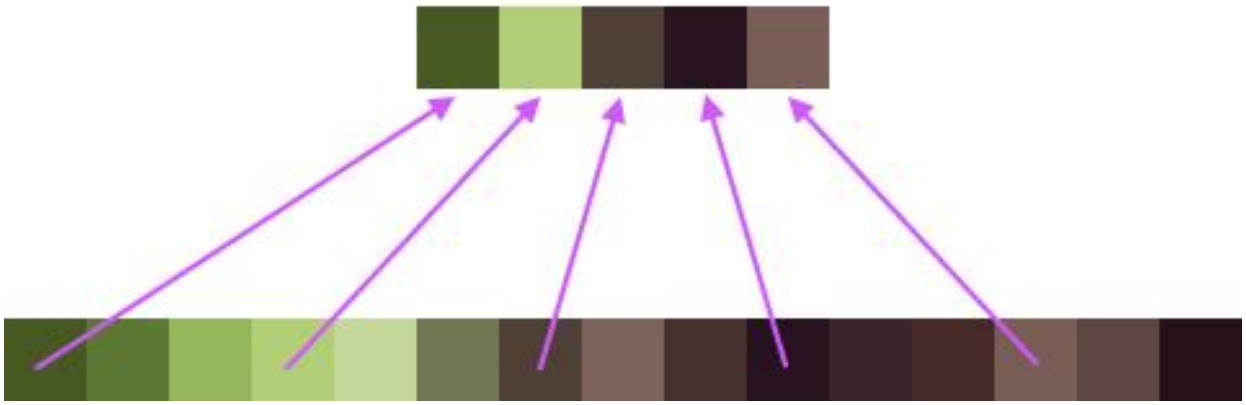

Рисунок 2.2 – Алгоритм найближчого сусіда

Наприклад, якщо проводиться операція зменшування зображення методом найближчого сусіда в 4 рази по кожній із сторін, то співвідношення кількості пікселів базового зображення, до кінцевого дорівнює 1/16. Тобто абсолютна більшість вхідних пікселів ніяк не враховується.

Теоретична швидкість роботи залежить тільки від розмірів базового та кінцевого зображень. Метод усвідомлено застосовується для зменшення дуже рідко, тому що дає дуже погану якість, але через швидкість і простоту реалізації він є у всіх бібліотеках та додатках, що працюють з графікою.

2.2.2. Афінні перетворення (affine transformations)

Афінні перетворення - загальний метод для зміни зображень. Вони дозволяють за одну операцію розтягнути, повернути і відобразити зображення. Тому в багатьох додатках і бібліотеках, що реалізують метод афінних перетворень, функція зміни зображень є просто обгорткою, яка розраховує коефіцієнти для перетворення.

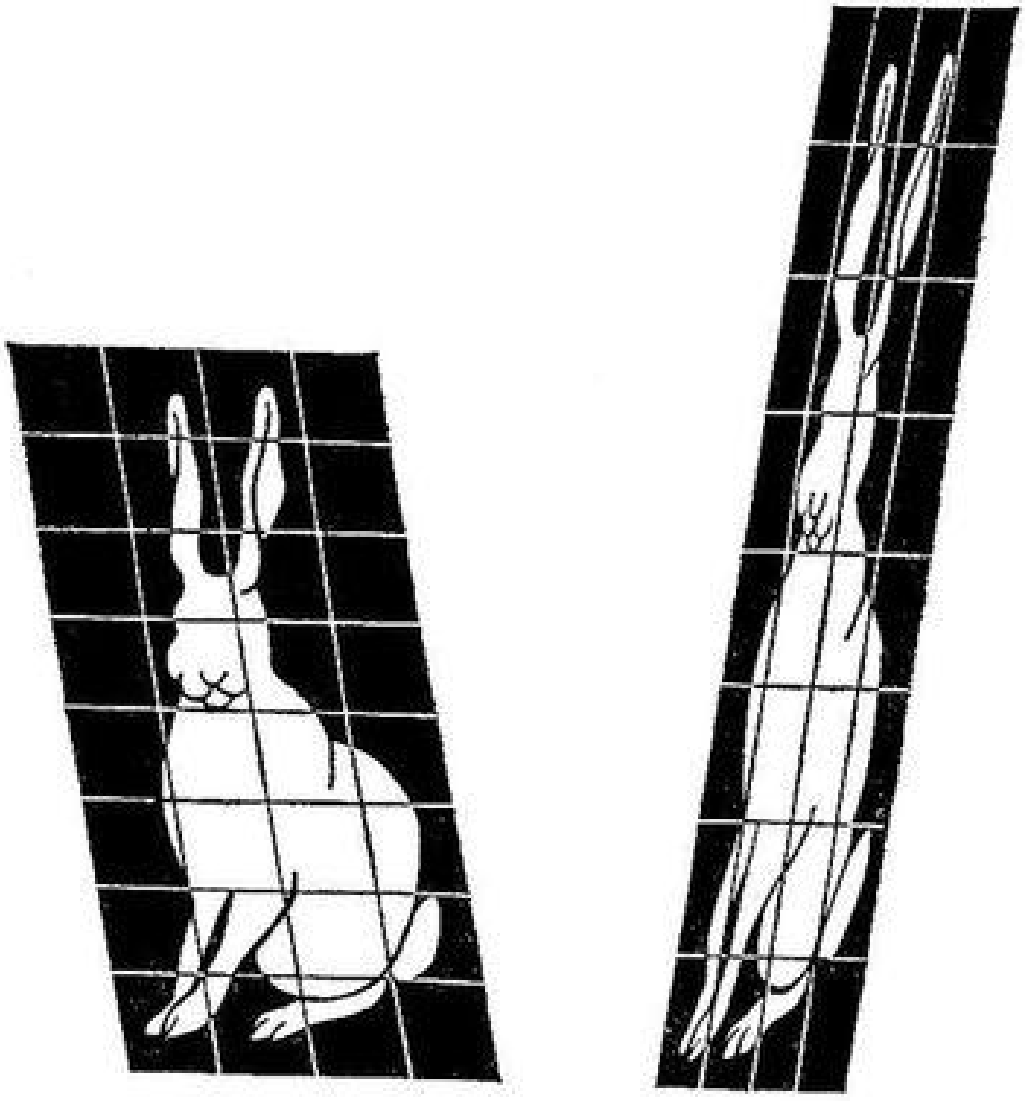

Рисунок 2.3 – Афінні перетворення

Принцип дії полягає в тому, що для кожної точки кінцевого зображення береться фіксований набір точок вхідного і інтерполюється за допомогою обраного фільтра, відповідно до їх взаємного становища. Кількість точок теж залежить від фільтра.

Такий метод дає гладке зображення при збільшенні, але при зменшенні результат дуже схожий на найближчого сусіда. Наприклад: теоретично, при використанні бікубічного фільтра і зменшенні в 3 рази відношення оброблених пікселів до вихідних рівняється $4^2 / 3^2 = 1,78$. На практиці результат значно гірше, тому що в існуючих реалізаціях вікно фільтра і функція інтерполяції не масштабується відповідно до масштабу зображення, і пікселі, які знаходяться ближче до краю вікна, беруться з негативними коефіцієнтами (відповідно до функції), тобто ці пікселі не вносять повний внесок в формування кінцевого зображення. В результаті зображення, зменшене з бікубічим фільтром, відрізняється від зображення, зменшеного з білінійним фільтром, тільки тим, що воно ще більш чітке. Ну а для білінійного фільтра при зменшенні зображення в три рази, відношення оброблених пікселів до вихідних рівняється $2^2 / 3^2 = 0.44$, що принципово не відрізняється від найближчого сусіда. Фактично, аффінні перетворення не можна використовувати для зменшення більш, ніж в 2 рази. І навіть при зменшенні до двох разів вони дають помітні дефекти на зображені, у виді спотворювання ліній.

Час роботи помітно більше, ніж у найближчого сусіда, і залежить не тільки від розміру базового та кінцевого зображень, але і від розміру вікна обраного фільтра.

У цілому, використання цього методу для масштабування зображень не е дуже гарною ідеєю, тому що результат виходить досить поганий і схожий на результат роботи алгоритму найближчого сусіда, а ресурсів на цей метод потрібно значно більше. Проте, цей метод знайшов широке застосування в програмах і бібліотеках.

2.2.3 Суперсемплінг (Supersampling)

За допомогою цього методу створюються mip-рівні, за допомогою нього (якщо сильно спростити) працює повноекранне згладжування в іграх. Його суть - у разі необхідності розділення вихідного зображення по сітці пікселів кінцевого і складанні всіх вихідних пікселів, що припадають на кожен піксель кінцевого відповідно до площі, що потрапила під кінцевий піксель. При використанні цього методу для збільшення, на кожен піксель кінцевого зображення доводиться рівно один піксель вихідного. Тому результат для збільшення дорівнює найближчому сусіду.

Можна виділити два підвиди цього методу: з округленням кордонів пікселів до найближчого цілого числа пікселів і без. У першому випадку алгоритм стає малопридатним для масштабування менше, ніж в 3 рази, тому що на який-небудь один кінцевий піксель може припадати один вихідний, а на сусідній - чотири (2x2), що призводить до диспропорції на локальному рівні. У той же час алгоритм з округленням, очевидно, можна використовувати у випадках, коли розмір початкового зображення кратний розміру кінцевого, або масштаб зменшення досить малий. Ставлення оброблених пікселів до вихідних при округленні кордонів завжди дорівнює одиниці.

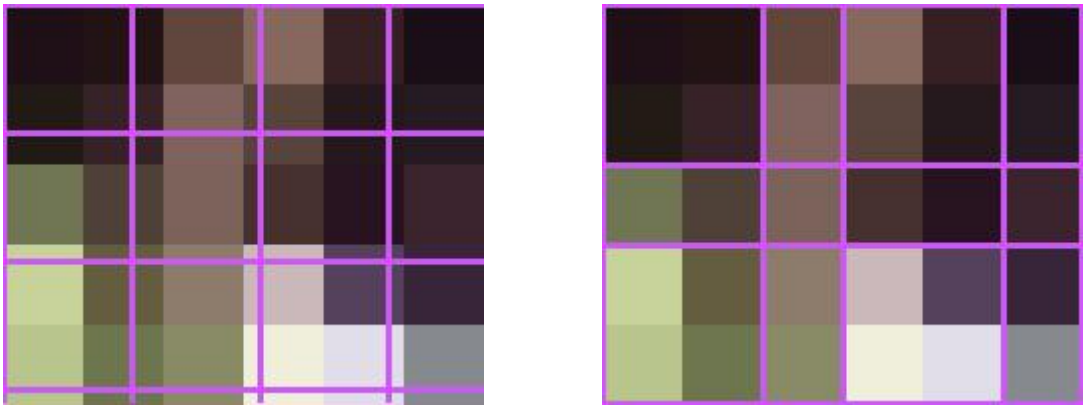

Рисунок 2.4 – Суперсемплінг

Підвид без округлення дає відмінну якість при зменшенні на будь-якому масштабі, а при збільшенні дає дивний ефект, коли велика частина вихідного пікселя на кінцевому зображенні виглядає однорідною, але на краях видно перехід. Ставлення оброблених пікселів до вихідних без округлення кордонів може бути використано для формування від одного об’єкту до чотирьох, тому що кожен вихідний піксель вносить вклад або в один кінцевий, або в два сусідніх, або в чотири сусідніх пікселів.

Продуктивність цього методу для зменшення нижче, ніж у афінних перетворень, тому що в розрахунку кінцевого зображення беруть участь всі пікселі вихідного. Версія з округленням до найближчих кордонів зазвичай швидше в кілька разів. Також можливо створити окремі версії для масштабування в фіксовану кількість разів (наприклад, зменшення в 2 рази), які будуть ще швидшими.

Метод суперсемплінга є досить ефективним, часто вживаним як в різноманітних програмах обробки зображень, так і в спеціалізованих графічних бібліотеках різноманітних мов програмування.

2.2.4 Згортки (Convolution)

Цей метод схожий на афінні перетворення тим, що використовуються фільтри, але має не фіксоване вікно, а вікно, пропорційне масштабу. Наприклад, якщо розмір вікна фільтру дорівнює 6, а розмір зображення зменшується в 2,5 рази, то у формуванні кожного пікселя кінцевого зображення бере участь $(2,5 * 6)^2 = 225$ пікселів, що набагато більше, ніж в разі суперсемплінга (від 9 до 16). На щастя, згортки можна вживати в 2 проходи, спочатку в одну сторону, потім в іншу, тому алгоритмічна складність розрахунку кожного пікселя рівняється не 225, а всього $(2,5 * 6) * 2 = 30$. Внесок кожного вихідного пікселя в кінцевий як раз визначається фільтром. Ставлення оброблених пікселів до вихідних цілком визначається розміром вікна фільтра і так само його квадрату. Тобто для білінійного

фільтра це відношення буде 4, для бікубічного 16 и т.д.. Алгоритм прекрасно працює як для зменшення, так і для збільшення.

Швидкість роботи цього методу залежить від усіх параметрів: розмірів вихідного зображення, розміру кінцевого зображення, розміру вікна фільтра . Одна з переваг цього методу в тому, що фільтри можуть задаватися окремою функцією, ніяк не прив’язаною до реалізації методу. При цьому функція самого фільтра може бути досить складною без особливої втрати продуктивності, тому що коефіцієнти для всіх пікселів в одному стовпці і для всіх пікселів в одному рядку вживаються тільки один раз. Тобто сама функція фільтра викликається тільки (m + n) * w разів, де m і n - розміри кінцевого зображення, а w - розмір вікна фільтру.

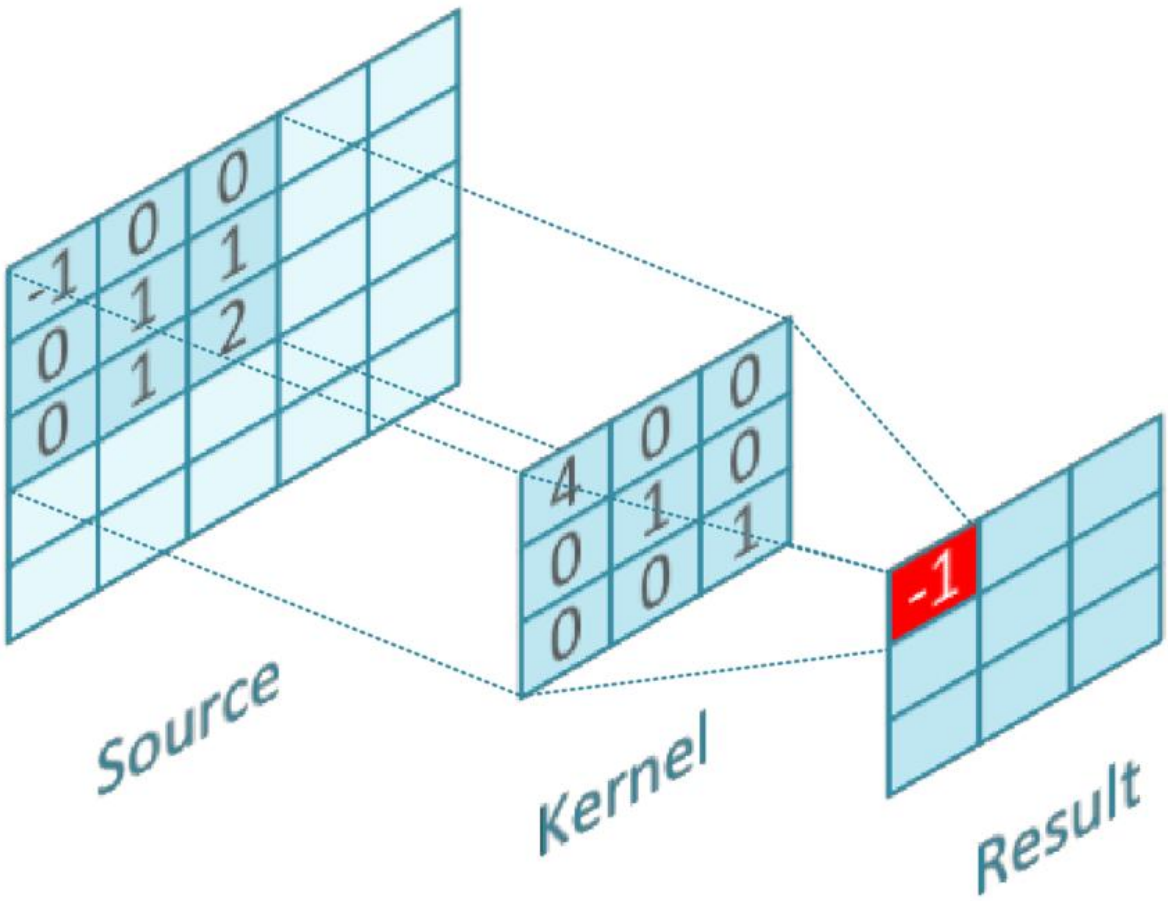

Рисунок 2.5 – Згортка

Примітно, що деякі фільтри мають зони негативних коефіцієнтів (як наприклад бікубічний фільтр або фільтр Ланцоша). Це потрібно для додання переходам на кінцевому зображенні різкості, яка була на вихідному.

Метод Згортки також є ефективним і часто вживаним як в різноманітних програмах обробки зображень, так і в спеціалізованих графічних бібліотеках різноманітних мов програмування.

Оскільки завдання обробки зображень часто актуальні при роботі з системами реального часу, поряд з критерієм якості вкрай важливим є критерій швидкодії. Сучасні пристрої надають можливість паралельних обчислень, які дозволяють в рази прискорити час роботи алгоритмів масштабування.

2.3 Сітковий підхід

Ще одним методом удосконалення алгоритму в боротьбі за швидкість є застосування сіткового підходу, який, крім того, дозволяє проводити об’єднання вхідних даних в кластери складної форми.

Основою сіткового методу є створення сіткової структури, розбиття N-мірного простору ознак за допомогою площин на осередки. Ідеєю методу являетса накладення сітки на простір ознак для формування набору осередків, визначення приналежності обє’ктів вхідної вибірки кожного з осередків, їх заміна на новостворені об’єкти (осередки). При цьому кожен об’єкт вхідної множини може належати тільки одному осередку, а осередок може включати в себе будь-яку кількість об’єктів.

Щільність осередку *w* визначається як кількість елементів, что попали в даний осередок.

Алгоритм побудови вибірки об’ктів на основі сіткового підходу можна розділити на наступні етапи:

- формування кроку осередку. здійснюється, виходячи з логічних вимог поставленого завдання з урахуванням компромісу швидкодії і якості.

бажано здійснити коригування експериментальним шляхом;

- формування значень ознак множини об’єктів. при цьому необхідно відбирати ознаки таким чином, щоб представник вхідної множини об’єктів не міг претендувати на потрапляння в декілька осередків;

- розподіл безлічі вхідних об’ктов на осередки. об’єкт належить деякому осередку тільки тоді, якщо кожне з значень його ознак належить діапазону відповідних ознак осередку;

- обчислення ваг осередків. являє собою суму ваг об’єктів вхідної множини, що потрапили в даний осередок. вага об’єктів може коригуватися за допомогою введення додаткових коефіцієнтів.

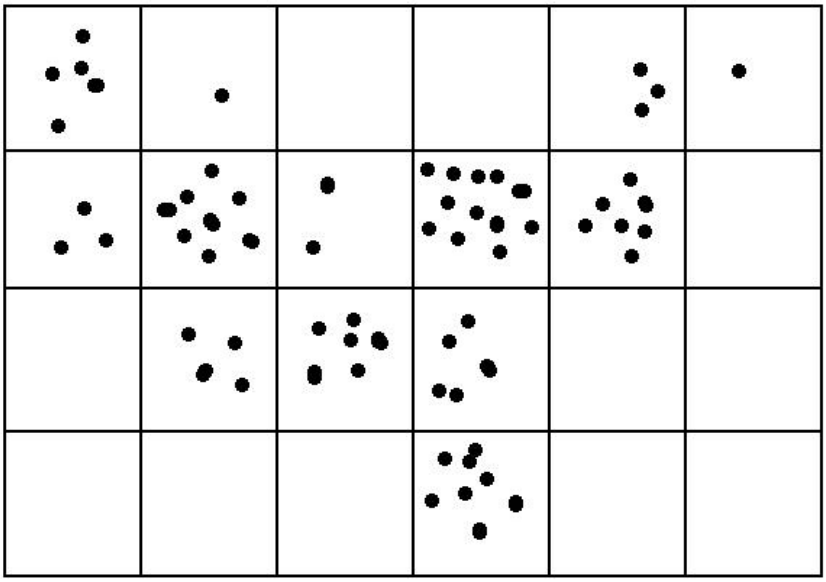

Рисунок 2.5 – Демонстрація сіткового підходу для двомірного простору признаків

Аналіз методу сіткового підходу показаав низьку тимчасову трудомісткість і коректну обробку безлічі об’ктів вхідної вибірки.

Застосування описаних в даному розділі методів дозволяє досягти багаторазового приросту швідкодії під час виконання процедури кластерізації, что дуже важливо при работі в системах реального часу.

3 РОЗРОБКА ГІБРИДНОГО АЛГОРИТМУ КЛАСТЕРИЗАЦІЇ

Рішення при виборі алгоритму для виконання завдання кластеризації без урахування її специфіки часто закінчується далеко не найкращим вибором у плані ефективності. У даній роботі метою було створення системи кластеризації зображень, що працює в режимі реального часу в автономних системах комп’ютерного зору. Головними вимогами до системи є швидкість виконання і, в зв’язку з обмеженими обчислювальними ресурсами мобільних пристроїв, низька трудомісткість.

Виходячи з вище зазначених вимог, було вирішено не використовувати цілком уже існуючі алгоритми, а спробувати розробити гібридний алгоритм, який включав би в себе найбільш цікаві ідеї вже розроблених алгоритмів і нівелював їх недоліки.

При створенні алгоритму базовим етапом є підбір комплексу змінних - коефіцієнтів, з використанням яких і буде працювати алгоритм. В алгоритмі, який буде описаний далі в цьому розділі, були використані наступні змінні:

- dc (decreasing) - коефіцієнт, який відповідає за лінійне зменшення зображення на етапі попередньої обробки; в зв’язку з особливостями алгоритму може мати значення від 1 до 2^n, що безпосередньо залежить від розміру вихідного зображення.

- cs (cell size) - коефіцієнт зниження розмірності сіткової моделі у колірному кубі RGB. У зв’язку з тим, що в використовуваному алгоритмі в якості базового колірного простору використовується rgb, в якому кожна з трьох компонент може приймати значення від 0 до 256, цей коефіцієнт може мати значення від 1 до 2^n. Дослідницьким шляхом було встановлено, що найбільш ефективними є значення 4, 8 і 16.

- lr (layer) - коефіцієнт, який відповідає за кількість шарів, на яких буде проводитися кластеризація. Чим більше значення коефіцієнта, тим теоретично вище точність, але при високих значеннях можливі складнощі з

подальшою обробкою і використанням інформації.

- st (step) - коефіцієнт зміни кроку. Дозволяє задавати як лінійне (при St = 1), так і нелінійне правило зміни кроку (при St <1).

3.1 Алгоритм зменшення

Оскільки великий розмір зображення, що обробляється, означає дуже високу трудомісткість, на найпершому етапі роботи алгоритму проводиться зменшення зображення до прийнятного в плані продуктивності.

Потім створюється піраміда зображень з лінійним коефіцієнтом зменшення 2, тобто, якщо базове зображення має розмір 4 Мрх, то наступні за ним в піраміді будуть мати розміри 1 Мрх, 0.256 Мрх, 0.064 Мрх і т.д.

В процесі розробки алгоритму були протестовані кілька методів масштабування зображення. Найбільш перспективними алгоритмами виявилися такі:

- проріджуванням (з кроком *h*);
- усередненням (середньоарифметичний фільтр).

Таблиця 3.1 – Порівняння часу виконання алгоритму при обробці зображення різного розміру

Розмір зображення (Мрх)	Час (мілісекунди)
48	1647
12	496
4	250
1	105
0,256	57

Алгоритм проріджування, незважаючи на відмінні показники швидкодії, показав себе вкрай незадовільним в плані якості, тому що відбуваються дуже великі втрати інформації.

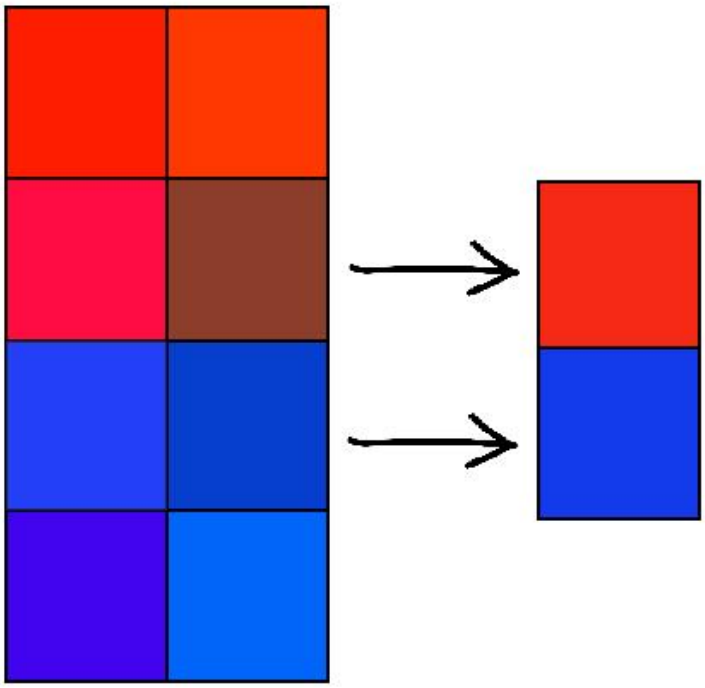

Рисунок 3.1 – Алгоритм зменшення зображення методом усереднення (середньоарифметичний фільтр)

На рисунку 3.1 проведене зменшення пікселів з наступними кольорами:

({235, 36, 0} + {241, 59, 0} + {242, 22, 67} + {124, 65, 47})/4 = {211, 46, 29}

({48, 66, 243} + {21, 67, 200} + {59, 13, 232} + {7, 101, 2473)/4 = {34, 62, 230}

```
     public static byte[,,] ReduceX2Avg(byte[,,] arr)
     { byte[,,] res = new byte[3, arr.GetLength(1) / 2, arr.GetLength(2) / 2];
     for (int i = 0; i < res.GetLength(1); i++)
           for (int j = 0; j < res.GetLength(2); j++)      {
     res[0, i, j] = (byte)((arr[0, i * 2, j * 2] + arr[0, i * 2, j * 2 + 1] +
arr[0, i * 2 + 1, j * 2] + arr[0, i * 2 + 1, j * 2 + 1]) / 4);
     res[1, i, j] = (byte)((arr[1, i * 2, j * 2] + arr[1, i * 2, j * 2 + 1] +
arr[1, i * 2 + 1, j * 2] + arr[1, i * 2 + 1, j * 2 + 1]) / 4);
     res[2, i, j] = (byte)((arr[2, i * 2, j * 2] + arr[2, i * 2, j * 2 + 1] +
arr[2, i * 2 + 1, j * 2] + arr[2, i * 2 + 1, j * 2 + 1]) / 4);
     }
     return res;}
```

Приклад 2.1 – Алгоритм зменшення зображення методом усереднення (середньоарифметичний фільтр), написаний на мові програмування C#

В результаті було прийнято рішення в якості оптимального з точки зору швидкості і якості використовувати метод обчислення середнього значення, де кожним чотирьом пікселям (2x2) вхідного зображення відповідає один піксель вихідного зображення.

3.2 Відображення даних в колірному кубі: сіткова модель

На поточному етапі розробки алгоритму кластеризації цифрових зображень в якості базового колірного простору використовується простір RGB.

Перш ніж відображати дані в куб RGB потрібно подумати над трудомісткістю кластеризації. Адже в кубі $256^3 = 2^{24}$= 16 777 216 позицій (кольорів). Не всі вони будуть використані (стандартне повнокольорове зображення розміром понад 3 Мрх містить близько півмільйона унікальних кольорів), але, тим не менше, для балансу співвідношення трудомісткість / якість пропонується розглядати куб, як розбитий на осередки кубічної форми.

Таблиця 3.2 – Порівняння часу виконання алгоритму при обробці зображення різного розміру

Розмір осередка	Час роботи алгоритму, мілісекунди
1x1x1	1989
2x2x2	495
4x4x4	121
8x8x8	45
16x16x16	37
32x32x32	32

Для зручності роботи доцільно розглядати осередки розміром

2^nx2^nx2^n. На першому етапі для експерименту досить розглянути n = 0, 1, 2 або 3. У цій версії алгоритму вироблено розбиття простору кубиками без перекриття, але в подальшому передбачається провести ряд експериментів з використанням часткового перекриття кубиків.

Як видно за результатами експерименту, представленими в таблиці, збільшення розміру осередку від 1 до 2 і далі до 4 супроводжується більш чотириразовим на кожному кроці зменшенням часу роботи алгоритму. Збільшення розміру осередку з 8 до 16 і далі фактично позбавлене сенсу, оскільки поряд з падінням якості час роботи алгоритму зменшується незначно.

В підсумку було прийнято рішення для подальшого дослідження використовувати розмір осередку 4 і 8, оскільки вони представляються компромісними щодо часу роботи алгоритму і якості.

3.3 Відображення даних в колірній куб: функція щільності

Цифрове зображення - це не просто набір пікселів. Це сукупність об’єктів, пікселі яких однорідні в сенсі кольору в деякій околиці. Цією властивістю потрібно скористатися для формування правильного уявлення колірних даних про об’єкти в кубі RGB. В тому сенсі, щоб значно підвищувати вагу і щільність кубиків з кольорами, що відповідають однорідним ділянкам об’єктів на зображенні.

Отже, кожному кубику в просторі RGB відповідає число - щільність кольору/групи кольорів, яке є функцією числа яскравості і міри однорідності їх околиці на зображенні. Другий параметр вважаємо пріоритетним.

При оцінці щільності допустимо, що на знімку - фіксований піксель з кольором *c* = (r*, g*, b*).* Знаходимо в його околиці *k* найближчих кольорів. Обмежимося розглядом квадратних околиць 3x3 або 5x5. Параметр *k* може змінюватися від 1 до 8 пікселів для околиці 3x3 (від 1 до 24 пікселів для околиці 5x5). Очевидно, що задавати граничні значення не має сенсу. Були

проведені експерименти зі значеннями 2 і 4 для околиці 3x3 (і від 4 до 12 пікселів для околиці 5x5). Відстань між двома кольорами обчислювалася по одній з трьох формул:

$$\rho_0(c^*, c^i) = \max\{|r^* - r^i|, |g^* - g^i|, |b^* - b^i|\};$$

$$\rho_M(c^*, c^i) = |r^* - r^i| + |g^* - g^i| + |b^* - b^i|;$$

$$\rho_E(c^*, c^i) = \sqrt{(r^* - r^i)^2 + (g^* - g^i)^2 + (b^* - b^i)^2}.$$

При цьому крайні ряди пікселів на знімку не розглядалися з причини неповного положення маски.

В кінцевому рахунку було вирішено використовувати $k = 4$ в околиці 3x3.

Коефіцієнт щільності кольору $c^* = (r^*, g^*, b^*)$ пікселя в околиці оцінювався таким чином:

$$w = 1/s, s = \sum_{i=1}^{k} \rho_i, if\ s = 0 \rightarrow s = 1.$$

Для однорідних ділянок зображення (коли кольори дуже близькі), коефіцієнт w буде прагнути в межі до 1 (абсолютна однорідність), щільність угруповання в кубику буде рости.

Також була випробувана в якості експерименту зміна оцінки вагового коефіцієнту - зведення відстані в ступінь p, так:

$$w = 1/s, s = \sum_{i=1}^{k} \rho_i^p, if\ s = 0 \rightarrow s = 1.$$

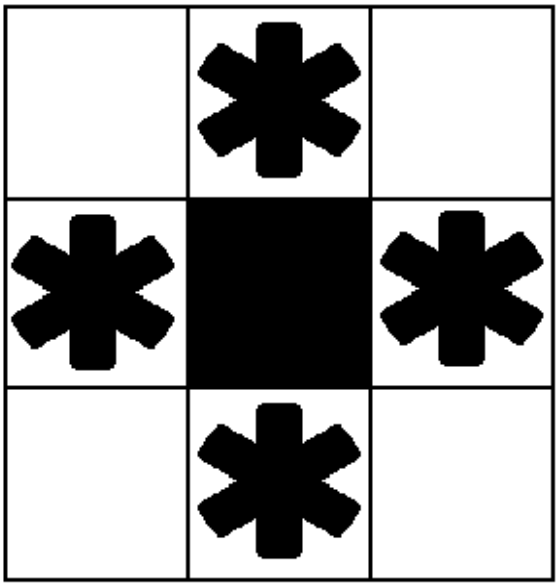

Рисунок 3.2 – Обчислення щільності

Такий підхід (при p>1) дозволяє більш значно підвищувати вагу кольорів в однорідних околицях (пригнічуючи кольори в неоднорідних околицях). Як наслідок, підвищується значимість кольорів однорідної околиці. Міра однорідності починає превалювати перед кількістю пікселів одного кольору. Такий підхід може себе виправдати, оскільки об’єкти, навіть порівняно невеликі - це, в першу чергу, області з однорідною яскравістю.

В рамках цього підходу було випробувана ще одна операція: після знаходження в околиці k найближчих кольорів $c1 \le c2 \le \cdots \le ck$, виключали з цієї групи $q < k$ найменші значення. Очікувалося, що для пікселів дрібних деталей і тіні це призведе до значного пригнічення щільності w. Однак ефект був малопомітний, а трудомісткість алгоритму підвищувалася.

В результаті реалізації даного етапу кожному кубику $2^n x 2^n x 2^n$ в просторі RGB буде підтверджена його вага w, яка буде характеристикою щільності кубика.

Для зручності подальшої роботи було проведено нормування щільності так, щоб найбільша щільність становила 1. Щільність інших кубиків можна обчислити, розділивши 1 на абсолютне значення максимальної щільності і

помноживши на абсолютне значення щільності поточного кубика.

```
      private List<Cell> CreateCellList(int[,,] arr, int dimension)
      {
            dimension = 1;
            List<Cell> result = new List<Cell>();
            for (int x = 0; x < arr.GetLength(0); x++)
                for (int y = 0; y < arr.GetLength(1); y++)
      for (int z = 0; z < arr.GetLength(2); z++)
      {
         if (arr[x, y, z] > 0)
             result.Add(new Cell (arr[x, y, z], x * dimension, y * dimension,
z * dimension));
       }
       result = result.Where(x => x.Count >= 1).ToList();
       result = result.OrderByDescending(x => x.Count).ToList();
       int max = result.Max(x => x.Count);
       for (int i = 0; i < result.Count; i++)
           result[i].CalcAbsCount(max);
        return result;
      }
```

Приклад 2.1 – Алгоритм зменшення розмірності у колірному кубі RGB, написаний на мові програмування C#

3.4 Попередній аналіз даних в колірному кубі і розбиття на шари

Імовірно, кубики з високою щільністю відповідають об’єктам, а кубики з низькою щільністю - це пікселі тіні і різноманітний шум. Для того, щоб кластери чітко відділялися один від одного, такі малозначущі кубики доцільно відфільтрувати.

Тому на першому етапі аналізу даних в колірному кубі проводиться фільтрація (обнулення) щільності кубиків зі значенням щільності менше порога *w**.

В результаті попередніх етапів роботи алгоритму вдалося отримати колірний куб, розбитий на ряд осередків із щільністю пікселів кожного з них в діапазоні від 0 до 1. В подальшому проводиться формування шарів для кластеризації в тому же в діапазоні. Принцип формування шарів задається за допомогою двох вхідних коефіцієнтів: кількість шарів і коефіцієнта *st*, що застосовується для визначення розміру кроку. У таблиці 3.2 наведено

приклад формування шарів за лінійним і нелінійним принципами.

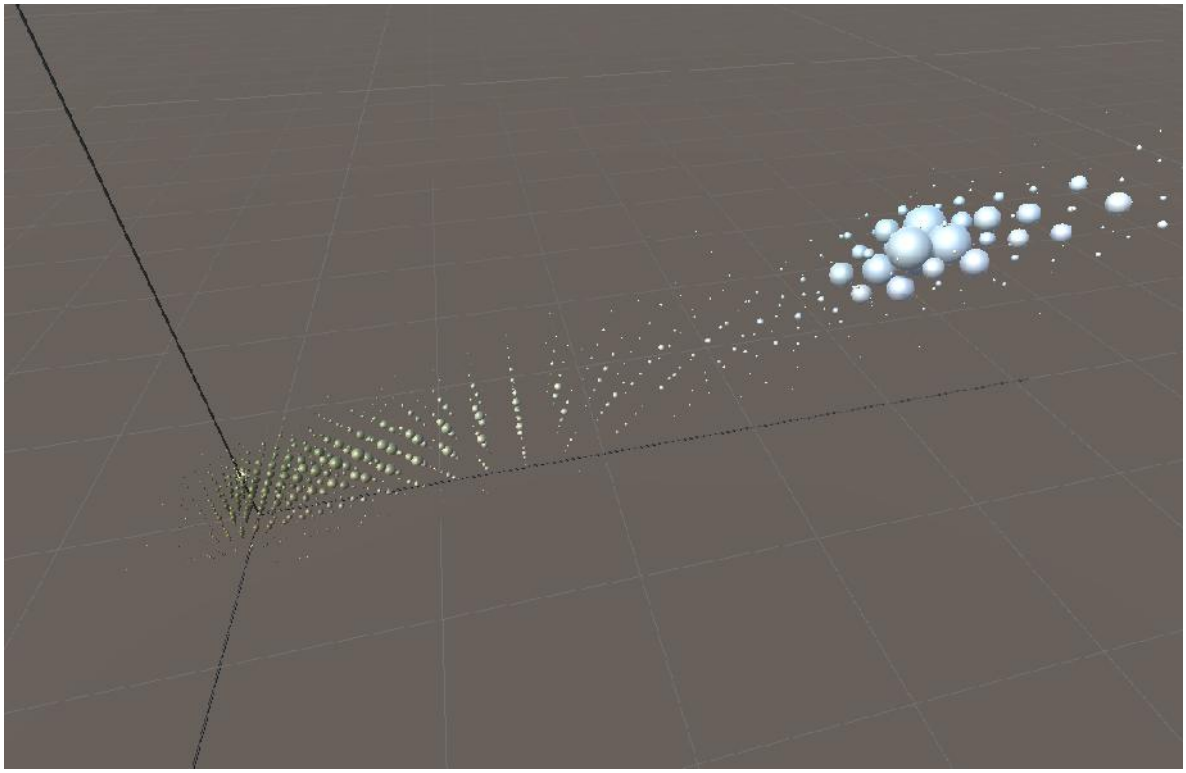

Рисунок 3.3 – Данні, представлені в колірному кубі за допомогою середи розробки Unity

Таблиця 3.3 – Діапазон значень шарів збудованих за лінійним та нелінійним принципами (загальна кількість шарів - 10)

Номер шару	Лінійний (1)	Нелінійний(0.9)
1	1 <= ly < 0.9	1 <= ly < 0.81
2	0.9 <= ly < 0.8	0.81 <= ly < 0.65
3	0.8 <= ly < 0.7	0.65 <= ly < 0.51
4	0.7 <= ly < 0.6	0.51 <= ly < 0.39
…	…	…
10	0.1 <= ly < 0	0.04 <= ly < 0

3.5 Процес кластеризації хвильовим методом

Остаточний етап виконання даної роботи - це проведення процесу кластеризації хвильовим методом. Основою алгоритму кластеризації є хвильовий метод в тривимірному просторі.

При переході на новий рівень відбувається перебір всіх осередків колірного куба, починаючи з такого з найбільшою щільностю, пересуваючись до такого з найменшою щільностю. Формування нового кластера відбувається при наявності наступних умов:

- цей осередок не є членом вже існуючого кластера;
- у поточний момент відсутній кластер, що формується.

При виконанні вище зазначених умов відбувається формування нового кластера, а цей осередок стає для нього головним і отримує позначку, що входить до складу кластера. Потім проводиться аналіз сусідніх осередків в просторі 3x3x3 (всього 26 одиниць). Всі сусідні осередки, що знаходяться в одному шарі з розглянутим осередком, приєднуються до кластеру і отримують позначку про це.

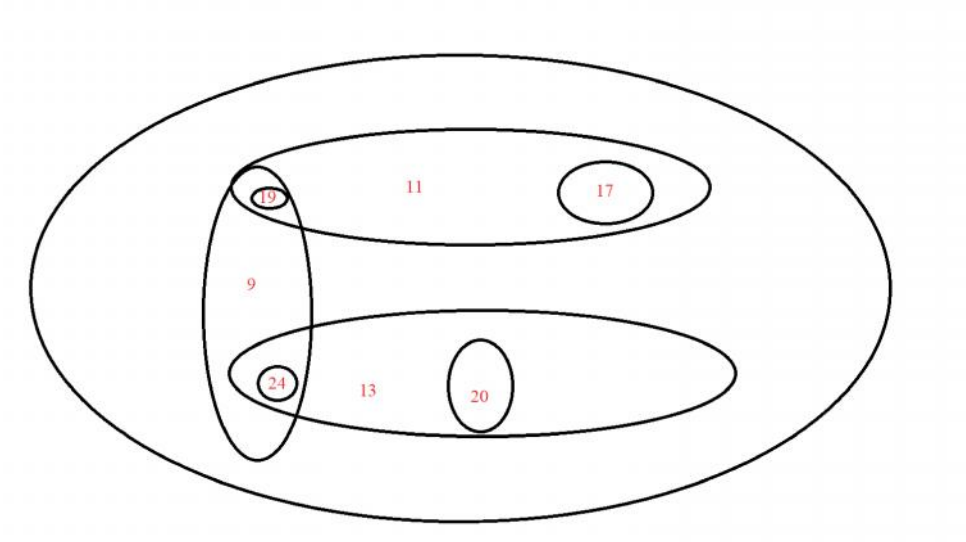

Рисунок 3.4 – Принцип об’єднання кластерів, розташованих, у тому числі і на різних шарах

Формування кластера триває до того моменту, поки жодний з вхідних в нього осередків не матиме жодного вільного осередку-сусіда, що знаходиться на тому ж шарі.

Після перебору всіх осередків на даному шарі проводиться перевірка зіткнення новосформованих кластерів з кластерами, що знаходяться на шарі щаблем вище. У разі, якщо будь-якої кластер даного шару стикається з кластером, що знаходяться сходинкою вище, він приєднує його. У разі, якщо два кластери мають загального сусіда на попередньому шарі, вони об’єднуються в один. Даний ефект може бути названо кумулятивним.

Наприклад (Рисунок 3.4), ми маємо на одному з шарів (№3) кластери з номерами 17, 19, 20, 24. Кластери з номерами 17 і 19 мають загального сусіда зверху з номером 11, кластери 20 і 24 - загального сусіда з номером 13, а кластери з номерами 19 і 24 - загального сусіда з номером 9. у результаті, всі зазначені кластери будуть об’єднані в єдиний кластер, розташований на шарі №3.

Результатом роботи алгоритму є набір кластерів, розташованих на різних шарах. Причому, кластери, розташовані на нижніх шарах, зазвичай включають в себе також і кластери, розташовані вище.

Таким чином, використовуючи розроблений гібридний алгоритм кластеризації цифрових зображень, вдалось досягти значної економії часу при достатній якості результату.

3.6 Приклад роботи алгоритма

В якості вхідного зображення було обрано фото розміром 4 Мрх (Рисунок 3.5).

Рисунок 3.5 – Вхідне зображення

Рисунок 3.5 – Приклад розвитку кластеру на різних рівнях

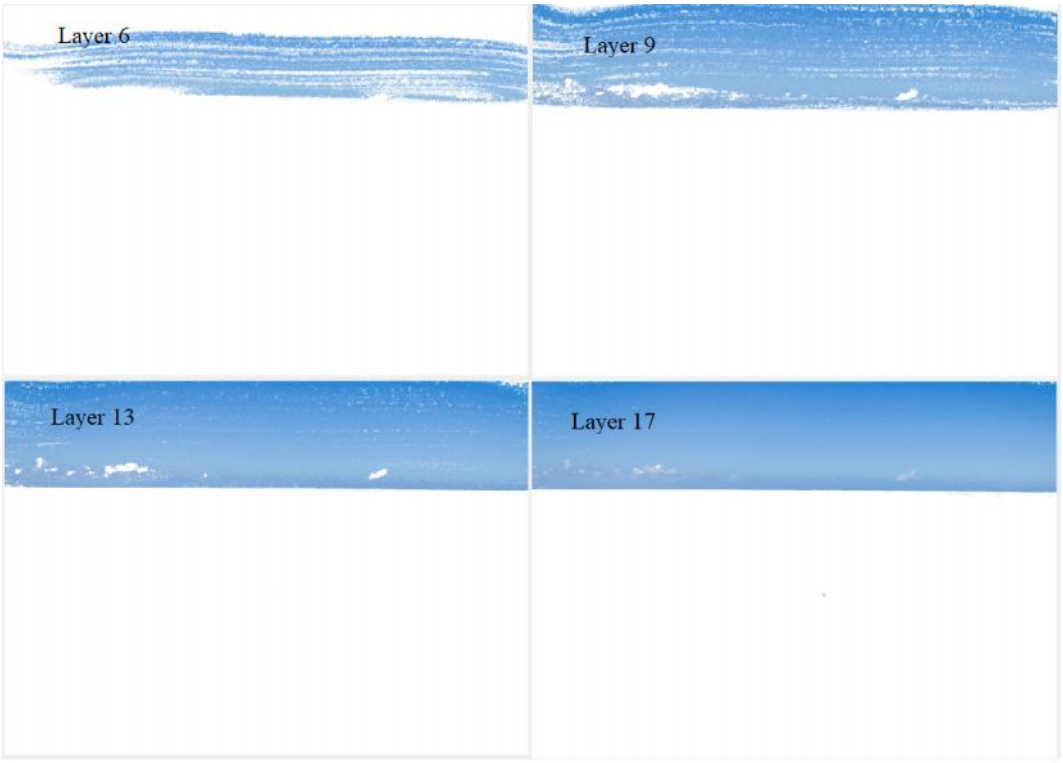

Рисунок 3.6 – Приклад розвитку кластеру на різних рівнях

Коефіцієнти були задані наступним чином:

- dc (decreasing) – 1(вхідне зображення не зменшувалось);
- cs (cell size) – 8;
- lr (layer) – 20;
- st (step) – 0,75.

Час роботи алгоритму – 939 міллісекунд.

ВИСНОВКИ

В рамках роботи була розглянута основна базова інформація щодо обробки цифрового зображення. Формати цифрових зображень, колірні простори, алгоритми перед обробки та найбільш поширені алгоритми кластеризації. З урахуванням вивченої інформації був розроблений власний, гібридній алгоритм кластеризації, орієнтований на використання у системах, працюючих в режимі реального часу.

Протягом останніх десятиліть була розроблена велика кількість різноманітних алгоритмів кластеризації, найбільш поширені з них реалізовані як бібліотеки у різних мовах програмування. Але використання цих бібліотек у великих проектах не є дуже ефективним варіантом через багатофункціональну направленість (конкретний продукт розроблений під конкретну задачу зазвичай буде кращим), низьку швидкість (найбільша кількість готових бібліотек написана для мови інтерпретації Python), обмежену кількість коефіцієнтів, що доступно для настроювання, та неможливо вносити зміни до алгоритму.

Кращим виходом з цієї ситуації є розробка нового гібридного алгоритму під конкретне завдання, з урахуванням особливостей вхідних даних та майбутнього використання.

У зв’язку з тим, що метою роботи є створення алгоритму кластеризації, який має працювати у системах реального часу, головною ціллю було добитися максимально можливої швидкості, навіть, при необхідності, жертвуючи якістю.

Під час розробки було прийняте рішення використовувати сіткову модель ознак у колірному кубі RGB та хвильовий метод при формуванні та об’єднанні кластерів. Перехід до сіткової моделі дозволяє значною мірою скоротити кількість оброблених значень, а використання хвильового методу значно прискорити процес прийняття рішення належності сутності до того чи

іншого кластеру.

Результатом даної магістерської роботи, є гібридний алгоритм кластеризації цифрових зображень, який має гарні показники щодо швидкості, при достатній якості, що дозволяє розглядати його, як можливий варіант для використання в системах, працюючих в режимі реального часу. Проте існують ще нереалізовані можливості для пришвидшення роботи алгоритму. Перш за все за допомогою використання паралельного обчислення. Щодо поліпшення якості, одним із можливих перспективних рішень може стати використовування, в додаток до RGB, іншого колірного простору, наприклад HSV.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. M. Castrillón, . O. Déniz, . D. Hernández и J. Lorenzo, A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework, International Journal of Computer Vision, № 22, pp. 481-494, 2011.

2. Arsenov A., Ruban I., Smelyakov K., Chupryna A. Evolution of Convolutional Neural Network Architecture in Image Classification Problems. Selected Papers of the XVIII International Scientific and Practical Conference on IT and Security (ITS 2018). – CEUR Workshop Processing. – Kyiv, Ukraine, November 27, 2018. – P. 35-45.

3. Залогова, Л. А. Комп’ютерна графіка [Текст]. Практикум. – 2005. – Т. 20, №2 – С. 254.

4. Khan, H. Abdullah и M. Shamian Bin Zainal, Efficient eyes and mouth detection algorithm using combination of viola jones and skin color pixel detection, International Journal of Engineering and Applied Sciences, № Vol. 3 № 4, 2013

5. Залогова, Л.А. Комп’ютерна графіка: Практикум [Текст]. Л.А. Залогова. – Т. – 2005. – 67 С.

6. Розпізнавання зображень людських облич за допомогою нейронної мережі [Текст]. Г.Ю. Костецька, О.І. Федяєв. – Д. – 2010. – 218 С.

7. Макаренко, А. А. Методика локалізації зображення обличчя для систем відеоконтролю на основі нейронної мережі [Текст]. А.А. Макаренко – Д. – 2006. – 118 С.

8. Пестунов И.А., Синявский Ю.Н., Алгоритмы сегментации в задачах кластеризации спутниковых изображений. Вестник Кемеровского государственного университета. – 2012. – №4(52) Т.2. – с. 110-125.

9. Рылов С.А., Пестунов И.А., Использование графических процессоров NVIDIA при кластеризации мультиспектральных данных сеточным алгоритмом ССА. Интерэкспо Гео-Сибирь. – 2015. – №1 – с. 75-79.

10. Куликова Е.А., Пестунов И.А., Синявскию Ю.Н., Непараметрический алгоритм кластеризации для обработки больших массивов данных Конф. Математические методы распознавания образов: сб. тр. – Москва: Изд-во MAKS Press, 2009 – С. 149.

11. Сікорський, О.С. Огляд згортальних нейронних мереж для задачі класифікації зображень. [Текст] О.С. Сікорський. – М. – 2011. 134 С.

12. Varga B., Karas K., High-resolution image segmentation using fully parallel mean shift EURASIP Journal of Advances in Signal Processing. – 2011.- 111.

13. Martin D.R., Fowlkes C.C., Tal D., Malik J., A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Proceedings of the 8th International Conference Computer Vision. – 2001. – 2. – p. 416-423

14. Гонсалес Р. Цифровая обработка изображений Р. Гонсалес, Р. Вудс. – М.: Техносфера, 2005. – 1072 с.

15. Журавлев Ю. И. Распознавание. Математические методы. Программная система. Практические применения. Ю. И. Журавлев, В. В. Рязанов, О. В. Сенько. – М. : Фазис, 2005. – 159 с.

16. Друки А.А., Алгоритм масштабирования и кластеризации в системе поиска лиц на изображении Приволжский научный вестник. – 2011. - №. – с. 3-11.

17. David A. Forsyth, Jean Ponce Computer Vision: A Modern Approach (2nd ed.). – Pearson Education Limited, 2015. 792p.

18. Peter Corke Robotics, Vision and Control (2nd ed.). – Springer, 2017. – 693p.

19. Шапиро Л. Компьютерное зрение. Л. Шапиро, Дж. Стокман. – М.: БИНОМ, 2006. – 752 с.

20. Методы компьютерной обработки изображений / под ред. В. А. Сойфера. – М. : Физматлит, 2003. – 784 с.