МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ОДЕСЬКИЙ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ І. І. МЕЧНИКОВА
ФАКУЛЬТЕТ МАТЕМАТИКИ, ФІЗИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП’ЮТЕРНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

# КОМП'ЮТЕРНИЙ ЗІР ТА НАВІГАЦІЯ РОБОТОТЕХНІЧНИХ СИСТЕМ

## ЗМІСТОВНИЙ МОДУЛЬ 2
## СИСТЕМИ ТЕХНІЧНОГО ЗОРУ

Методичні вказівки
до лабораторних занять для студентів
першого (бакалаврського) рівня вищої освіти
галузі знань 12 Інформаційні технології

Одеса
Олді+
2023

**УДК 004.822(072)**
**К637**

**Укладач:**
***І. В. Шаріпова***, старший викладач кафедри комп'ютерних систем та технологій Одеського національного університету імені І. І. Мечникова;
**Рецензенти:**
***Ю. А. Ніцук***, доктор фіз.-мат.наук, професор, декан факультету математики, фізики та інформаційних технологій ОНУ імені І. І. Мечникова;
***Ю. А Максименко***, кандидат технічних наук, доцент, начальник кафедри організації розвідувальної-інформаційної роботи та технічних засобів розвідки ФПСВР та ССО Військова академія (м. Одеса).

*Рекомендовано вченою радою факультету математики, фізики та інформаційних технологій ОНУ імені І. І. Мечникова (Протокол № 5 від 12 травня 2023 р.)*

**Комп'ютерний** зір та навігація робототехнічних систем : Змістовний
К637 модуль 2 "Системи технічного зору" : метод. вказівки до лабораторних занять для студентів першого (бакалавр.) рівня вищої освіти галузі 12 Інформаційні технології / уклад.: І. В. Шаріпова. – Одеса : Олді+, 2023. – 44 с.

У методичних вказівках розкривається програма курсу, даються рекомендації для опанування лекційним матеріалом. Розглянуто зміст, завдання, наведено приклади розробки програм, порядок їх виконання. Рекомендовано вимоги до оформлення протоколів, правила захисту та оцінки. Наведено перелік завдань з темами лабораторних робіт, контрольні запитання та список рекомендованої літератури.

УДК 004.822(072)

Зміст

# ВСТУП

Комп'ютерний зір – вид діяльності, в якій для отримання даних застосовують статистичні методи та використовують моделі, які побудовані за допомогою геометрії, фізики і теорії навчання.

Комп'ютерний зір застосовують досить широко в управлінні мобільними роботами, військових додатках, промислових засобах спостереження, а також у сфері взаємодії людина/комп'ютер, пошуку зображення в бібліотеках, аналізі медичних зображень та реалістичному передаванні змодельованих сцен у комп'ютерній графіці.

Особливість комп'ютерного зору – це отримання описів зі зображень або послідовності зображень.

Системи технічного зору на сьогодні набули великого розповсюдження та знаходять все ширше застосування у різноманітних галузях людської діяльності. Важлива особливість сучасного підходу до створення систем технічного зору полягає у швидкому переході від ідеї до математичного алгоритму, перевірка якого не потребує значних затрат. Знадобиться лише ПК та найпростіша веб-камера. Але, не дивлячись, на простоту реалізації, застосування завдяки системи технічного зору більшість складних технологічних процесів стали автономними, а виконання поставлених завдань-більш швидким та зручним. Ціллю виконання лабораторних робіт є створення алгоритмів та програмного забезпечення, в якому по заданому кольору камера знаходить відповідний об'єкт, розпізнає його, передає сигнали до системи керування та рухається за об'єктом; все проходить в автономному режимі.

Таким чином **метою дисципліни** є сформувати систему знань студентів в області робототехнічних систем, знанні функціональних складових елементів, які лежать в основі інтелектуальних робототехнічних систем, на базі яких дипломований фахівець зможе приймати участь у розробці, застосуванні за призначенням і експлуатації таких систем різного функціонального призначення. Вироблення у студентів практичних навичок використання теоретичного матеріалу в галузі моделювання комп'ютерних систем, вивчення застосування та використання сучасних програмних засобів моделювання при розробці моделей для комп'ютерної інженерії.

Лабораторні роботи з дисципліни виконуються з метою закріплення та поглиблення теоретичних та практичних знань та вмінь, набутих у процесі засвоєння всього навчального матеріалу дисципліни.

Час, потрібний для виконання лабораторних робіт згідно навчального плану – 12 години у комп'ютерному класі та до 20 годин самостійної роботи.

Тематика лабораторних робіт присвячена оволодінню сучасних алгоритмів для системи технічного зору з технологією розпізнавання за кольором, тестовані алгоритми та програмне забезпечення.

## МЕТА, ЕТАПИ ПРОВЕДЕННЯ ТА ЗАХИСТ ЛАБОРАТОРНИХ РОБІТ

Лабораторні роботи з дисципліни виконуються з метою закріплення та поглиблення теоретичних та практичних знань та вмінь, набутих у процесі засвоєння всього навчального матеріалу дисципліни:

- ✓ закріплення, поглиблення та узагальнення теоретичних знань і розвиток навичок їх практичного застосування в галузі комп'ютерного зору та навігації робототехнічних систем
- ✓ самостійне розв’язання задач проектування та розробки алгоритмів оброблення зображень у системах технічного зору;
- ✓ уміння користуватися відповідною довідковою літературою, програмними засобами;

Проведення лабораторних робіт містить такі етапи:

- ✓ визначення теми, завдання і повторення теоретичного матеріалу;
- ✓ безпосереднє виконання роботи;
- ✓ оформлення пояснювальної записки;
- ✓ захист.

Після виконання лабораторної роботи і вирішення всіх поставлених у ній задач студент оформлює звіт з лабораторної роботи – протокол. Виконаний протокол студент підписує і після дозволу керівника він допускається до захисту. Якщо керівник не допускає студента до захисту, то це питання обговорюється на засіданні кафедри у його присутності.

Захист лабораторной роботи – це форма перевірки якості виконання програми та знань, отриманих під час виконання лабораторних робіт та на лекціях.

Під час захисту студент робить доповідь по суті програми та відповідає на запитання.

Якість протоколу та його захист оцінюється в балах (0- 5), за шкалою ECTS (A, B, C, D, E, FX, F) та за національною шкалою «відмінно», «добре», «задовільно», «незадовільно».

# ЛАБОРАТОРНА РОБОТА № 1

**Тема: «Формування зон селекції коефіцієнтів перетворення Хаара, їх властивості для відновлення зображень»**

Перетворення зображень – застосування різних послідовних технологій, які змінюють вигляд зображень та якість для кращого огляду, сприйняття, аналізу і розпізнавання, а також приведення до прийнятного вигляду й до подальшого застосування.

Вейвлети —узагальнена назва математичних функций певної форми, які є локальними і в часі, і по частоті, в яких усі функції отримуються із однієї базової, змінюючи її шляхом зсуву або розтягу.

Вейвлет-перетворення –перетворення, що розглядають функцію (взяту як функція від часу) у термінах коливань, локалізованих за часом (простором) і частотою. На відміну від звичайних спектральних перетворень, вейвлет-аналіз дозволяє з однаковою точністю апроксимувати як гладкі функції, так і функції з різкими випадами, що дає можливість визначати незначні об’єкти. В якості базисних функцій, що утворюють ортогональний базис, можна використовувати широкий набір вейвлетів.

Вейвлет-перетворення сигналів є узагальненням спектрального аналізу, типовим представником якого є класичне перетворення Фур'є. Термін "вейвлет" в перекладі з англійської означає "маленька (коротка) хвиля". Вейвлети – це узагальнена назва сімейств математичних функцій певної форми, які локальні в часі і по частоті, і в яких всі функції виходять з однієї базової та породжуються за допомогою її зрушень і розтягувань по осі часу. Вейвлет-перетворення розглядають аналізовані тимчасові функції в термінах коливань, локалізованих за часом і частотою. Як правило, вейвлет-перетворення (WT) класифікується на дискретне (DWT) і безперервне (CWT).

Кластерний аналіз — задача розбиття заданої вибірки об'єктів на підмножини, що називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Завдання кластеризації відноситься до статистичної обробки, а також до широкого класу завдань навчання без вчителя.

Гістограма зображення – дискретна функція H, визначена на множині значень [0;2bpp], де bpp — кількість біт, що відводиться для кодування яскравості одного піксела.

Гістограмні методи – методи поелементного перетворення зображень, метою яких є зміна закону розподілу ймовірностей яскравостей пікселів, що описує зображення.

**Постанова задачі та завдання:**

Розглянути методи формування зон селекції коефіцієнтів перетворення Хаара та обгрунтувати їх властивості для відновлення зображень

**Хід роботи**

ПЕРЕТВОРЕННЯ ХААРА

Наша мета - перетворити зображення так, щоб воно добре стискалося класичними алгоритмами. Подумаємо, як потрібно змінити його, щоб отримати довгі ланцюжки нулів.

У "реальних" зображень, таких як фотографії, є одна особливість — яскравість сусідніх пікселів зазвичай відрізняється на невелику величину. Справді, в світі рідко можна побачити різкі, контрастні перепади яскравості. А якщо вони і є, то займають лише малу частину зображення.

Розглянемо фрагмент першого рядка яркостей з відомого зображення "Lenna" (Рисунок 1)

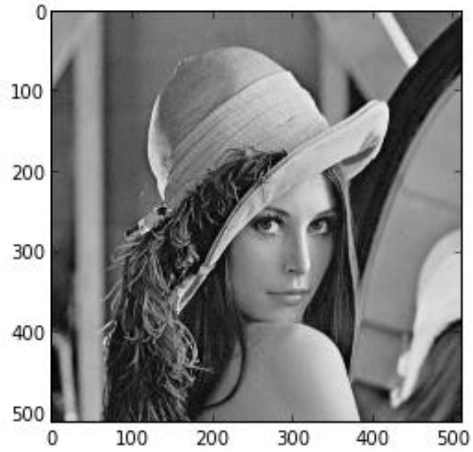



Рисунок 1 – Lenna

```
154, 155, 156, 157, 157, 157, 158, 156
```

Видно, що сусідні числа дуже близькі. Щоб отримати бажані нулі або

хоча б щось близьке до них, можна закодувати окремо перше число, а потім розглядати лише відмінності кожного числа від попереднього.

Отримуємо: `154, 1, 1, 1, 0, 0, -2.`

Такий метод справді використовується і називається дельта-кодуванням. Але у нього є серйозні недолік - він нелокальний. Тобто не можна взяти шматочок послідовності і дізнатися, які саме яскравості в ньому закодовані без декодування всіх значень перед цим шматочком.

Спробуємо вчинити інакше. Не будемо намагатися відразу отримати хорошу послідовність, спробуємо поліпшити її хоча б трохи.

Для цього розіб'ємо всі числа на пари і знайдемо напівсуми і напіврізності значень в кожній з них.

```
(154, 155), (156, 157), (157, 157), (158, 156)
(154.5, 0.5), (156.5, 0.5), (157, 0.0), (157, -1.0)
```

Чому саме напівсуми і напівразності? А все дуже просто! Напівсума — це середнє значення яскравості пари пікселів. А напіврізність несе в собі інформацію про відмінності між значеннями в парі. Очевидно, знаючи напівсумму a і напівразність d можна знайти і самі значення:

перше значення в парі = a-d,

друге значення в парі = a + d .

Це перетворення було запропоновано в 1909 році Альфредом Хааром і носить його ім'я.

Отримані числа можна перегрупувати, розділивши напівсуми і напіврізності:

```
154.5, 156.5, 157, 157, 0.5, 0.5, 0.0, -1.0
```

Числа в другій половині послідовності як правило будуть невеликими (те, що вони не цілі, нехай поки не бентежить). Чому так?

Як ми вже з'ясували раніше, в реальних зображеннях сусідні пікселі рідко відрізняються один від одного значно. Якщо значення одного велике, то й іншого велике. У таких випадках говорять, що сусідні пікселі корельовані.

Справді, розглянемо перші 2000 пар сусідніх пікселів і кожну пару представимо на графіку точкою.

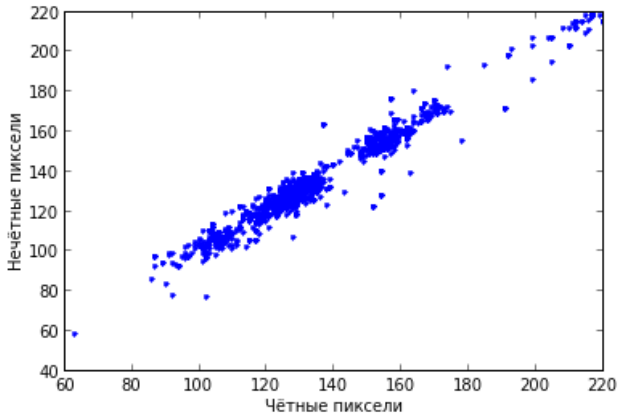



Рисунок 2

Всі точки шикуються уздовж однієї прямої лінії. І так практично у всіх реальних зображеннях. Верхній лівий і нижній правий кути зображення практично завжди порожні.

А тепер розглянемо графік, точками в якому будуть напівсуми і напіврізності.

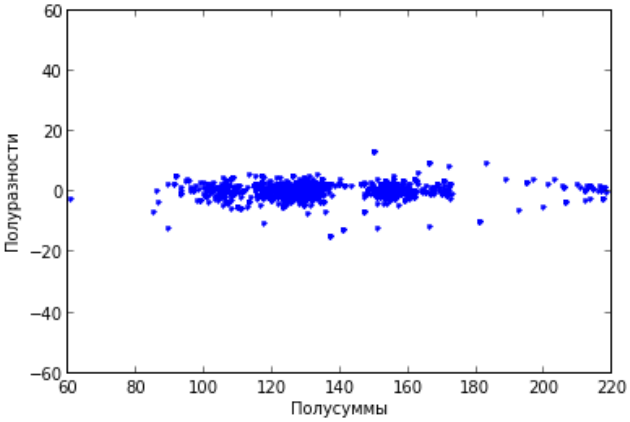



Рисунок 3

Видно, що напіврізності знаходяться в набагато більш вузькому діапазоні значень. А це означає, що на них можна витратити менше одного байта. Якеніяке, а стиснення.

Спробуємо записати математичні вирази, що описують перетворення Хаара.

Отже, у нас була пара пікселів (вектор) $\begin{pmatrix} x \\ y \end{pmatrix}$, а ми хочемо отримати пару $\begin{pmatrix} \frac{y+x}{2} \\ \frac{y-x}{2} \end{pmatrix}$.

Таке перетворення описується матрицею $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

Справді $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{y+x}{2} \\ \frac{y-x}{2} \end{pmatrix}$, що нам і потрібно.

Малюнки з точок на двох останніх графіках однакові. Різниця лише в повороті на кут в 45°.

В математиці повороти і розтягування називаються аффинными перетвореннями і описуються як раз за допомогою множення матриці на вектор. Що ми і отримали вище. Тобто, перетворення Хаара — це просто поворот точок таким чином, щоб їх можна було зручно і компактно закодувати.

Правда, тут є один нюанс. При афінних перетвореннях може змінюватися площа фігури. Не те, щоб це було погано, але якось неакуратненько. Як відомо, коефіцієнт зміни площі дорівнює засобу визначення матриці. Подивимося, який він для перетворення Хаара.

$$\det \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} = \frac{1}{2} \cdot \frac{1}{2} - (-\frac{1}{2}) \cdot \frac{1}{2} = \frac{1}{2}$$

Для того, щоб визначник дорівнював одиниці досить помножити кожен елемент матриці $\sqrt{2}$. На кут повороту (а значить, і на «стискає здатність» перетворення) це не вплине.

Отримуємо в результаті матрицю $H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$

Як відомо, якщо визначник матриці не дорівнює нулю, то для неї існує обернена матриця, «скасовуюча» її дію. Якщо ми знайдемо обернену матрицю для H, то декодування буде полягати просто в множенні векторів з полусуммами і полурізностями на неї.

$$\begin{pmatrix} x \\ y \end{pmatrix} = H^{-1} \begin{pmatrix} a \\ d \end{pmatrix}$$

Взагалі кажучи, пошук оберненої матриці — не така проста задача. Але, може, вдасться якось спростити цю задачу?

Розглянемо ближче нашу матрицю. Вона складається з двох вектор-рядків: $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$ і $\begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$. Назвемо їх $v_1$ і $v_2$. Вони володіють цікавими властивостями. По-перше, їх довжини рівні 1, тобто $v_1 v_1^T = v_2 v_2^T = 1$. Тут буква Т означає транспонування. Множення вектор-рядка на транспонований вектор-рядок — це скалярний добуток. По-друге, вони ортогональны, тобто $v_1 v_2^T = v_2 v_1^T = 0$. Матриця, рядки якої володіють вказаними властивостями називається ортогональною. Надзвичайно важливим властивістю таких матриць є те, що обернену матрицю для них можна отримати простим транспонуванням.

$$H^{-1} = H^T = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}^T = \begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

У справедливості цього вислову можна переконатися помноживши Н зворотну матрицю. На діагоналі ми отримаємо скалярні твори вектор-рядків на самих себе, тобто 1. А поза діагоналей — скалярні твори вектор-рядків один на одного, тобто 0. В результаті буде дорівнює одиничній матриці.

Все сказане добре працює для двох точок. Але що робити, коли точок більше?

У цьому випадку теж можна описати матрицею перетворення, але більшою за розміром. Діагональ цієї матриці буде складатися з матриць Н, таким чином вектор вихідних значень будуть вибиратися пари, до яких незалежно буде застосовуватися перетворення Хаара.

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \\ & & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \\ & & & & \ddots \end{pmatrix}$$

Тобто, вихідний вектор просто обробляється незалежно по парам.

Отже, коли ми знаємо, як виконувати перетворення Хаара, спробуємо розібратися з тим, що ж воно нам дає.

Отримані «полусуммы» (з-за того, що не ділимо на 2, доводиться використовувати лапки) — це, як ми вже з'ясували, середні значення в парах пікселів. Тобто, фактично, значення полусумм — це зменшена копія вихідного зображення! Зменшена тому, що полусумм в два рази менше, ніж вихідних пікселів.

Але що таке різниці? Полусумми усереднюють значення яскравостей, тобто «відфільтровують» випадкові сплески значень. Можна вважати, що це певний частотний фільтр.

Аналогічно, різниці виділяють серед значень межпиксельные «сплески» і усувають константну складову. Тобто, вони «відфільтровують» низькі частоти.

Таким чином, перетворення Хаара — це пара фільтрів, які поділяють сигнал на низькочастотну і високочастотну складові. Щоб отримати вихідний сигнал, потрібно просто знову об'єднати ці складові.

Що нам це дає? Нехай у нас є фотографія-портрет. Низькочастотна складова несе в собі інформацію про загальній формі особи, про плавних перепадах яскравості. Високочастотна — це шум і дрібні деталі.

Зазвичай, коли ми дивимося на портрет, нас більше цікавить низькочастотна складова, а значить при стисненні частина високочастотних даних можна відкинути. Тим більше, що, як ми з'ясували, вона зазвичай має менші значення, а значить більш компактно кодується.

Ступінь стиснення можна збільшити, застосовуючи перетворення Хаара багаторазово. У самому справі, високочастотна складова — це лише половина від усього набору чисел. Але що заважає застосувати нашу процедуру ще раз до низькочастотних даними? Після повторного застосування, высокачастотная інформація буде займати вже 75%.

Хоч ми поки і говорили про одновимірних ланцюжках чисел, цей підхід добре підходить і для двовимірних даних. Щоб виконати двовимірне

перетворення Хаара (або аналогічне йому), потрібно лише виконати його для кожного рядка і кожного стовпця.

Після багаторазового застосування, наприклад, аерофотографії місцевості, або пейзаж отримаємо наступний малюнок.

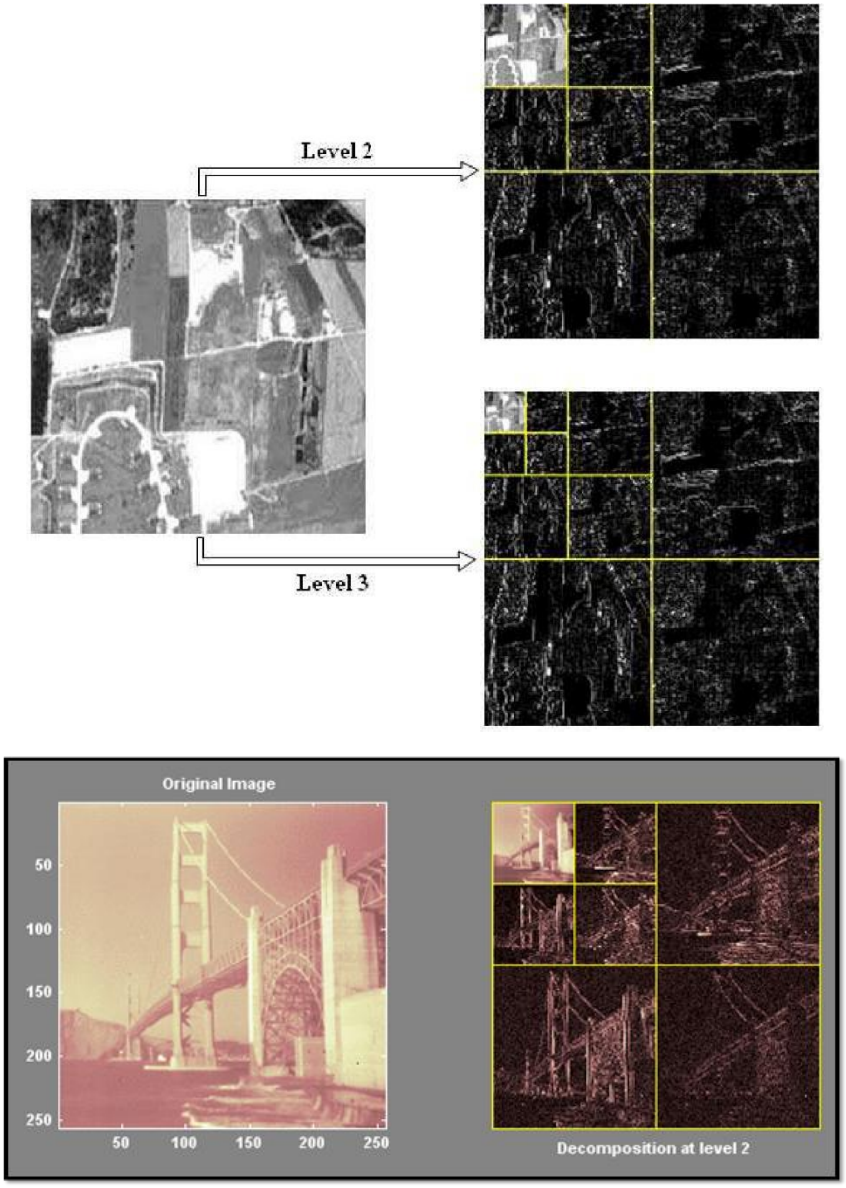



Рисунок 4

Чорні області відповідають низькій яскравості, тобто значенням, близьким до нуля. Як показує практика, якщо значення досить мало, то його можна округлити або взагалі обнулити без особливого збитку для декодованого зображення.

Цей процес називається квантуванням. І саме на цьому етапі відбувається

втрата частини інформації. (До речі, такий же підхід використовується в JPEG, тільки там замість перетворення Хаара використовується дискретне косинус-перетворення.) Змінюючи число обнуляемых коефіцієнтів, можна регулювати ступінь стиснення!

Звичайно, якщо обнулити занадто багато, то спотворення стануть видно на око.

Після всіх цих дій у нас залишиться матриця, що містить багато нулів. Її можна записати порядково в файл і стиснути якимось архіватором. Наприклад, тим же 7Z. Результат буде непоганий.

Декодування здійснюється в зворотному порядку: распаковывем архів, застосовуємо зворотне перетворення Хаара і записуємо декодовану картинку в файл.

**Перелік питань на захист лабораторної роботи**

1.Формування зон селекції коефіцієнтів перетворення Хаара.
2.Властивості коефіцієнтів перетворення Хаара істотні для стиску та для відновлення зображень

# ЛАБОРАТОРНА РОБОТА № 2

**Тема: «Алгоритм субквантизації ділянок зображення»**

Основною метою сегментації зображення є незалежне від домену розділення зображення на набір незалежних областей, які візуально відрізняються і мають різні значення щодо деяких характеристик або обчислюваних властивостей, таких як рівень сірого, текстура або колір, щоб забезпечити легкий аналіз зображення. Двома основними властивостями пікселів по відношенню до їх локального сусідства є розривність і подібність, які використовуються в багатьох методах сегментації.

Значення пікселів змінюється від 0 до 1, якщо воно належить об'єкту. Методи сегментації, які відносно пов'язані з властивістю розриву пікселів, приймаються як граничні або «методики на основі країв», а ті, що ґрунтуються на подібності або однорідності, є методами на основі регіонів. На жаль, технології на основі краю та регіону не завжди дають бажаних результатів. Використання сегментації зображень здійснюється на різних платформах або в різних програмах, але використання одного методу не може дати бажаного результату. Це все тому, що зображення, що містять різну властивість, а також деякі інші фактори, такі як звук, яскравість тощо, роблять акцент на зображеннях, і неможливо застосувати єдиний метод сегментації, а також єдину техніку оцінки для всіх типів зображень.

Сегментація зображень знизу вгору залишається складною проблемою, незважаючи на постійні дослідження, які тривають кілька десятиліть. За останні роки було запропоновано і також продемонстровано багато алгоритмів на кількох зображеннях, але справжнє питання полягає в тому, як це порівняння цих алгоритмів один з одним не було розглянуто належним чином. Здебільшого це пов'язано з недоступністю алгоритмів протягом цього часу в порівнянні з
останніми роками відповідного стандартного набору зображень і пов'язаної реальної інформації, а також через відсутність загальнодоступних реалізацій основних алгоритмів сегментації.

Порогова сегментація зображень – це проста форма сегментації зображень. Це спосіб створити двійкове або багатоколірне зображення на основі встановлення порогового значення інтенсивності пікселів вихідного зображення.У цьому процесі порогового значення розглянемо гістограму

інтенсивності всіх пікселів на зображенні. Потім встановимо поріг для поділу зображення на частини. Наприклад, враховуючи пікселі зображення в діапазоні від 0 до 255, встановлюємо поріг 60. Таким чином, усі пікселі зі значеннями, меншими або рівними 60, отримають значення 0 (чорний), а всі пікселі зі значенням більше ніж 60 буде надано значення 255 (білий).

Розглядаючи зображення з фоном і об’єктом, можемо розділити зображення на області на основі інтенсивності об’єкта та фону. Але цей поріг має бути ідеально встановлений, щоб розділити зображення на об’єкт і фон. Глобальне порогове значення: у цьому методі використовуємо бімодальне зображення. Бімодальне зображення – це зображення з 2 піками інтенсивності на графіку розподілу інтенсивності. Один для об'єкта, інший для фону. Потім виводимо порогове значення для всього зображення і використовуємо цей глобальний поріг для всього зображення (рисунок 2.2). Недоліком цього типу порогу є те, що він дуже погано працює при поганому освітленні зображення.

Встановлення порогу вручну: при виборі порогового значення в ручному режимі не завжди можна правильно підрахувати кількісні властивості того чи іншого об’єкту, в загальному випадку даний підхід оптимально підійде при умові, що параметри обєктів, які знаходяться на зображенні та параметри фону кардинально відрізняються один від одного. Наприклад, чорна деталь на білому фоні полотна транспортира. В інших випадках встановлення фону відчувається за рахунок багаторазового повторення експериментів з метою отримання оптимального, на погляд користувача, результату.

Адаптивне порогове значення: щоб подолати ефект освітлення, зображення поділяється на різні субрегіони, і всі ці області сегментуються за допомогою порогового значення, розрахованого для всіх цих регіонів. Потім ці субрегіони об’єднуються для відображення повного сегментованого зображення. Це допомагає до певної міри зменшити ефект освітлення.

Оптимальне порогове значення. Метод оптимального порогового значення можна використовувати, щоб мінімізувати неправильну класифікацію пікселів, що виконується сегментацією.

Локальне адаптивне порогове значення: через різницю в освітленні пікселів на зображенні глобальне порогове значення може мати труднощі з сегментацією зображення. Таким чином зображення розбивається на менші підгрупи, а потім виконується адаптивне порогове значення цих окремих груп. Після окремої сегментації цих підгруп усі вони об’єднуються, щоб утворити завершене

сегментоване зображення вихідного зображення. Отже, гістограма підгруп допомагає забезпечити кращу сегментацію зображення.

Сегментація на основі порогів. Для сегментації зображення використовуються пороги та зрізи. Вони застосовуються безпосередньо до зображення. Сегментація на основі порогів не вимагає попередньої інформації про зображення. Сегментація на основі порогів, яка використовується для реалізації, є порівняно швидкою та простою , а також її можна використовувати для реалізації в реальному часі. Але неправильний вибір порогу може призвести до надмірної або недостатньої сегментації.

Сегментація на основі країв – ця методика використовується для виявлення країв у зображенні, які, як передбачається, представляють межі об’єкта, і використовується для ідентифікації об’єктів.

Сегментація на основі регіонів – тут метод на основі країв може спробувати знайти межі об’єкта, а потім знайти сам об’єкт, заповнивши їх, а техніка на основі регіонів використовує протилежний підхід, починаючи з середини об’єкта, а потім «зростає» назовні, поки не буде досягнуто межі об’єкта. Дає найкращий результат у порівнянні з будь-якими іншими методами сегментації. Також дуже гнучкий вибір між інтерактивною та автоматичною технікою для сегментації зображення, яка перетікає від внутрішньої області до зовнішньої, чітко генеруючи межі об’єкта. Це також дає точний результат, ніж будь-який інший метод, коли все насіння відібрано належним чином. Нажаль,це порівняно дороге в пам'яті або також за часом обчислень. Важко визначити
критерії зупинки для сегментації.

**Виконання**

Зробити – Порівняльний аналіз алгоритмів сегментації з описом у вигляді презентації

*Параметр*. Пороговий метод .Метод на основі виділення регіонів.Метод на основі кластеризації.

*Швидкість* : швидкий , повільний.

*Складність обчислень*: низька, швидкий .

*Автоматичність*: напівавтоматичний, автоматичний.

*Шумостійкість*: низька,помірна.

*Точність* : помірний, точний .
*Виявлення кількох об'єктів:*поганий, допустимий .

Квантизація - зменшення кольорів зображення. Наприклад, старий добрий формат GIF використовує палітру, максимум на 256 кольорів. Якщо ви захочете зберегти серію своїх Селфі як GIF-анімацію (кому б це треба було), то перше, що вам, а точніше програмі, яку ви будете для цього використовувати, треба буде зробити – створити палітру. Можна використовувати статичну палітру, наприклад web-safe colors, алгоритм квантизації вийде дуже простим і швидким, але результат буде «не дуже». Можна створити оптимальну палітру на основі кольорів зображення, що дасть результат найбільш візуально схожий на оригінал.
**Виконання**: на підставі вибірки зображень за допомогою алгоритмів вихідного коду розглянути можливості субквантизації ділянок зображення

**Перелік питань на захист лабораторної роботи**

1. Поняття субквантизації коефіцієнтів перетворення.
2. Алгоритм субквантизації ділянок зображення

## ЛАБОРАТОРНА РОБОТА № 3

**Тема: «Реалізація алгоритму субквантизації ділянок зображення»**

Квантизація - зменшення кольорів зображення. Наприклад, старий добрий формат GIF використовує палітру, максимум на 256 кольорів. Якщо ви захочете зберегти серію своїх селфі як GIF-анімацію (кому б це треба було), то перше, що вам, а точніше програмі, яку ви будете для цього використовувати, треба буде зробити – створити палітру. Можна використовувати статичну палітру, наприклад web-safe colors, алгоритм квантизації вийде дуже простим і швидким, але результат буде «не дуже». Можна створити оптимальну палітру на основі кольорів зображення, що дасть результат найбільш візуально схожий на оригінал.

**Задача:** на підставі вибірки зображень за допомогою алгоритмів вихідного коду розглянути можливості субквантизації ділянок зображення

**Хід роботи**:

ВИБІРКА ЗОБРАЖЕННЯ

Вибірка - це можливість виконувати (оберати) операцію перетворення безперервних зображень (аналогових зображень) в просторової або тимчасової області в набір дискретних точок вибірки (пікселів) (цифрових зображень).

Чим дрібніше вибірка, тим менше пікселі і тим краще зображення, вплив різних інтервалів вибірки полягає в наступному:

Рисунок 1- Інтервал відбору проб: інтервал відбору проб 16

Рисунок 2 - Інтервал відбору проб: інтервал відбору проб 32

Рисунок 3 - Інтервал відбору проб: інтервал відбору проб 64

Алгоритм вихідного коду 1:

```
/**
 * Вибірка зображения
  * @param ріх зберегти пікселі зображения
```

```
 * @param iw 2D піксельна матрична ширина
 * @param ih висота 2D піксельної матриці
 * @парам сірий інтервал вибірки
 * @return
 */
private static int[] sample(int[] pix, int iw, int ih, int grey)
{
        // Вибірка зображения
        ColorModel cm = ColorModel.getRGBdefault();

        int d = (int) (256 / сірий); // інтервал виборки
        int dd = d*d;
        for(int i = 0; i < ih; i = i+d)
        {
        for(int j = 0; j < iw; j = j+d)
        {
              int r = 0, g = 0, b = 0;
           for(int k = 0; k < d; k++)
              for(int l = 0; l < d; l++)
                 r = r + cm.getRed(pix[(i+k)*iw+(j+l)]);
           for(int k = 0; k < d; k++)
              for(int l = 0; l < d; l++)
                 g = g + cm.getGreen(pix[(i+k)*iw+(j+l)]);
           for(int k = 0; k < d; k++)
              for(int l = 0; l < d; l++)
                 b = b + cm.getBlue(pix[(i+k)*iw+(j+l)]);
           r = (int)(r/dd);
           g = (int)(g/dd);
           b = (int)(b/dd);
           for(int k = 0; k < d; k++)
              for(int l = 0; l < d; l++)
                // pix[(i+k)*iw+(j+l)] = 255<<24|r<<16|g<<8|b;
                pix[(i+k)*iw+(j+l)] = new Color(r, g, b).getRGB();
        }
        }
        return pix;
```

```
}
/**
  * Вибірка зображення
  * @param srcPath шлях до вихідного файлу зображення
  * @param distPath шлях до  файлу зображення призначення
  * @парам сірий інтервал вибірки
 */
public static void sampleImage(String srcPath, String distPath, int grey) {
        OutputStream out = null;
        try {
        BufferedImage img = ImageIO.read(new File(srcPath));
        int imgType = img.getType();
        int w = img.getWidth();
        int h = img.getHeight();
        int[] pix = new int[w*h];
        pix = img.getRGB(0, 0, w, h, pix, 0, w);
        int[] newpix = sample(pix, w, h, grey);

        out = new FileOutputStream(distPath);
        BufferedImage imgOut = new BufferedImage( w, h, imgType);
        imgOut.setRGB(0, 0, w, h, newpix, 0, w);
        ImageIO.write(imgOut, "jpg", out);
        } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
        } finally {
        try {
                out.close();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
        }
}
```

## КВАНТУВАННЯ ЗОБРАЖЕННЯ

При цифровій обробці зображень безперервний динамічний діапазон значень яскравості ділиться на ряд дискретних рівнів. Ця процедура називається квантуванням. Її суть полягає в перетворенні безперервної змінної в дискретну змінну, яка приймає кінцеву множину значень . Ці значення називаються рівнями квантування. У загальному випадку перетворення виражається ступінчастою функцією (рис. 8.4). Якщо інтенсивність відліку зображення належить інтервалу (тобто, коли), то вихідний відлік замінюється на рівень квантування , де – пороги квантування. При цьому покладається, що динамічний діапазон значень яскравості обмежений.

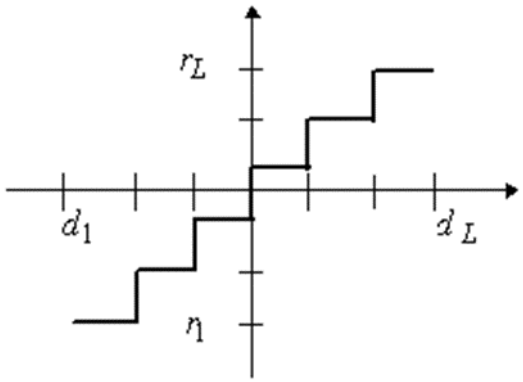



Рисунок 4 - Функція, що описує квантування

Основне завдання при цьому полягає у визначенні значень порогів і рівнів квантування.

Квантування - це операція, яка перетворює градації сірого (темрява) пікселя в дискретні цілочисельні значення. Найпростіше квантування представлено двома значеннями чорного (0) і білого (255) (тобто рівня 2), який стає двійковим зображенням.

Чим детальніше квантування, тим багатше рівень сірого (градація), Комп'ютери зазвичай використовують 8 біт (256 рівнів) для кількісної оцінки, що означає, що градація сірого (темрява) пікселя становить значення від 0 до 255. Схема ефекту серії виглядає наступним чином:

Рисунок 5 - Рівень квантування:  рівень квантування 2

Рисунок 6 - Рівень квантування:  рівень квантування 8

Рисунок 7 - Рівень квантування:  рівень квантування 64

Вихідний код 2:

```
/**
         * Кількісна оцінка зображення
         * @param srcPath шлях до вихідного файлу зображення
         * @param distPath шлях до файлу зображення призначення
         * @param сірий квантований ряд
        */
       public static void quantize(String srcPath, String distPath, int grey) {
              OutputStream out = null;
              try {
                     BufferedImage img = ImageIO.read(new File(srcPath));
                     int imgType = img.getType();
                     int w = img.getWidth();
                     int h = img.getHeight();
                     int pix[] = new int[w*h];
                     img.getRGB(0, 0, w, h, pix, 0, w);
                     int greyScope = 256/grey;
                     int r,g,b,temp;
                     r=b=g=temp=0;
                     ColorModel cm=ColorModel.getRGBdefault();
                     for(int i=0; i<w*h; i++) {
```

```
                    r = cm.getRed(pix[i]);
                    temp = r/greyScope;
                    r = temp*greyScope;
                    g = cm.getGreen(pix[i]);
                    temp = g/greyScope;
                    g = temp*greyScope;
                    b = cm.getBlue(pix[i]);
                    temp = b/greyScope;
                    b = temp*greyScope;
                    pix[i] = new Color(r, g, b).getRGB();
                }
                out = new FileOutputStream(distPath);
                BufferedImage imgOut = new BufferedImage(w, h, imgType);
                imgOut.setRGB(0, 0, w, h, pix, 0, w);
                ImageIO.write(imgOut, "jpg", out);
                System.out.println("test");
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } finally {
                try {
                    out.close();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
```

**Перелік питань на захист лабораторної роботи**

1. Методика оцінки зображень за ступеню насиченості.
2. Особливості комп’ютерної реалізації систем комп'ютерного зору.

# ЛАБОРАТОРНА РОБОТА № 4

## Тема : «Реалізація оцінки зображень за ступеню насиченості»

При обробці зображень моделі загальні колірні включають HSB (відтінок, насиченість, яскравість), RGB (червоний, зелений, синій), CMYK (блакитний, пурпуровий, жовтий, чорний) і CIE і т.д. Таким чином, відповідними кольорами є RGB, CMYK, Lab і т.д. режими. У колірній моделі HSB відтінок, насиченість і контраст є основними описами атрибутів зображення.

Насиченість можна визначити як хроматику, поділену на яскравість, яка, як і хроматичність, являє собою ступінь відхилення кольору від сірого при однаковій яскравості. Зверніть увагу, що це не те ж поняття, що і колір. Але оскільки саме і хроматичність визначають той же ефект, який проявляється в очах людей, виникає ситуація, коли хроматичність і насиченість - одне і те ж поняття.

Насиченість відноситься до яскравості кольору, також відомого як чистота кольору. Насиченість залежить від співвідношення колірного компонента і знебарвлювального компонента (сірого) в кольорі. Чим більше колірна складова, тим більша насиченість; чим більше ахроматична складова, тим нижче насиченість. Чисті кольори дуже насичені, такі як яскраво-червоний і яскраво-зелений. Кольори, змішані з білими, сірими або іншими тонами, є ненасиченими кольорами, такими як фіолетовий, рожевий, жовтувато-коричневий і т.д. Повністю ненасичені кольори взагалі не мають відтінку, наприклад, різні відтінки сірого від чорного до білого.

*Перетворення RGB → HSV*

З математичної точки зору алгоритм зміни насиченості кольору Найпростіше рішення - в лоб: перетворити значення RGB HSV/HSB, змінити значення "Saturation", потім назад в RGB.

HSV (Hue, Saturation, Value - тон, насиченість, значення) або HSB (англ. Hue, Saturation, Brightness - тон, насиченість, яскравість)

$$H \in [0, 360)$$
$$S, V, R, G, B \in [0, 1]$$

Нехай $MAX$ — максимальне значення з $R$, $G$ і $B$, а $MIN$ — мінімальне з них.

$$H = \begin{cases} 0, \text{ якщо } MAX = MIN \\ 60 \times \frac{G - B}{MAX - MIN} + 0, \text{ якщо } MAX = R \text{ і } G \geq B \\ 60 \times \frac{G - B}{MAX - MIN} + 360, \text{ якщо } MAX = R \text{ і } G < B \\ 60 \times \frac{B - R}{MAX - MIN} + 120, \text{ якщо } MAX = G \\ 60 \times \frac{R - G}{MAX - MIN} + 240, \text{ якщо } MAX = B \end{cases}$$

$$S = \begin{cases} 0, & \text{if } MAX = 0; \\ 1 - \frac{MIN}{MAX}, & \text{otherwise} \end{cases}$$

$$V = MAX$$

Два кольори називаються додатковими, якщо при змішуванні їх в рівній пропорції виходить чистий сірий колір. Якщо заданий один колір (, , ), то обов'язково існує додатковий йому колір (, , ). Оскільки вислідний колір повинен бути сірим, його насиченість (S) повинна дорівнювати 0. Таким чином,

$$H' = \begin{cases} H - 180, & \text{if } H \geq 180 \\ H + 180, & \text{if } H < 180 \end{cases}$$
$$S' = \frac{VS}{V(S-1)+1}$$
$$V' = V(S-1)+1$$

*Тон*

Відтінок відноситься до загальної колірної тенденції зображення на відображенні, що є чудовим колірним ефектом. У природі ми часто бачимо таке явище: об'єкти різних кольорів або оповиті шматком золотого сонячного світла, або оповиті синім місячним світлом, як легкий марлевий туман; або оповитий чарівною золотисто-жовтою осінню; або об'єднані в сріблясто-білий світ взимку. Цей тип кольору огортає об'єкти різних кольорів, так що всі об'єкти різних кольорів мають однакову колірну тенденцію. Це колірне явище є відтінком.

Відтінок визначається домінуючою довжиною хвилі у світлі, відбитому об'єктом. Різні довжини хвиль дають різні кольори. Відтінок є важливою

характеристикою відтінку, і він визначає фундаментальну характеристику кольорової есенції.

*Насиченість*

Використовується для оцінки візуальних атрибутів чистих кольорів в загальному колірному баченні (включаючи ахроматичні кольори). Вона пов'язана (або приблизно пов'язана) у фізичних і психологічних аспектах чистоти кольору. Визначається кількістю білого світла, змішаного з кольоровим світлом. Чим чистіше спектральний колірний зміст, тим він вище

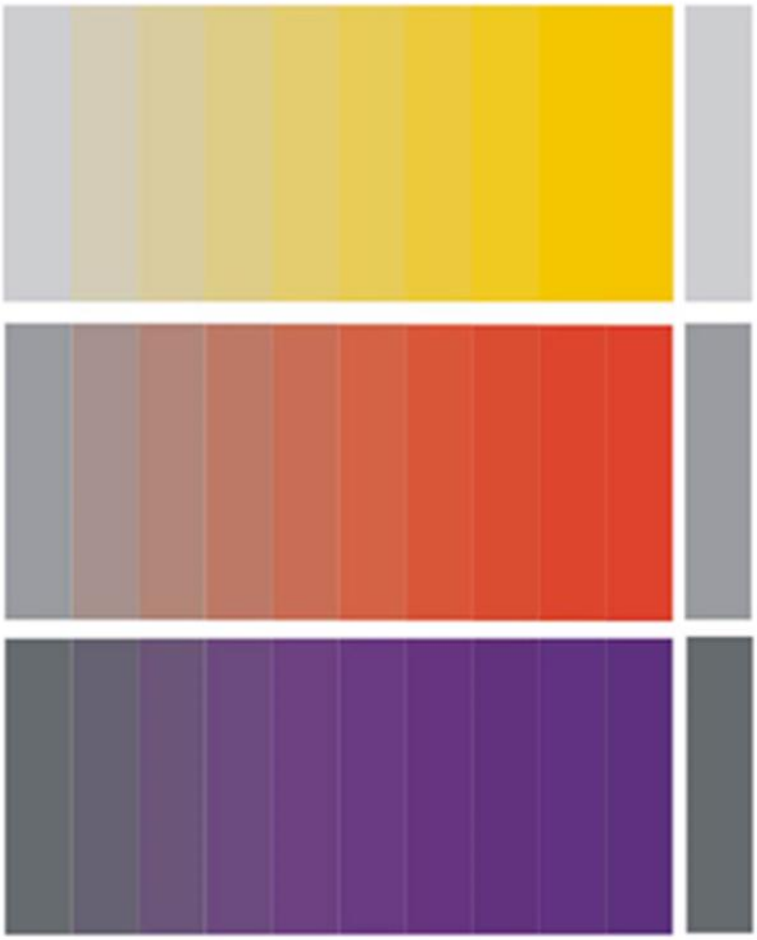


Рисунок 1 – насиченість кольорів

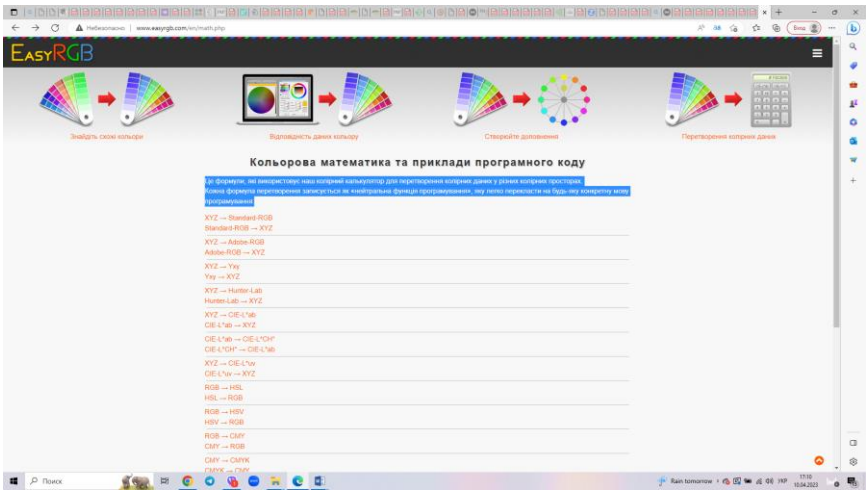
*Контраст*

Контраст - це вимірювання різних рівнів яскравості між найяскравішим білим і найтемнішим чорним на зображенні. Чим більше діапазон різниці, тим більший контраст і чим менше діапазон різниці, тим менше контраст. Гарне співвідношення контрастності 120:1 може легко відображати яскраві та насичені кольори.Коли контрастність досягає 300:1, воно може підтримувати всі рівні кольору. Однак контрастність страждає від тієї ж дилеми, що і

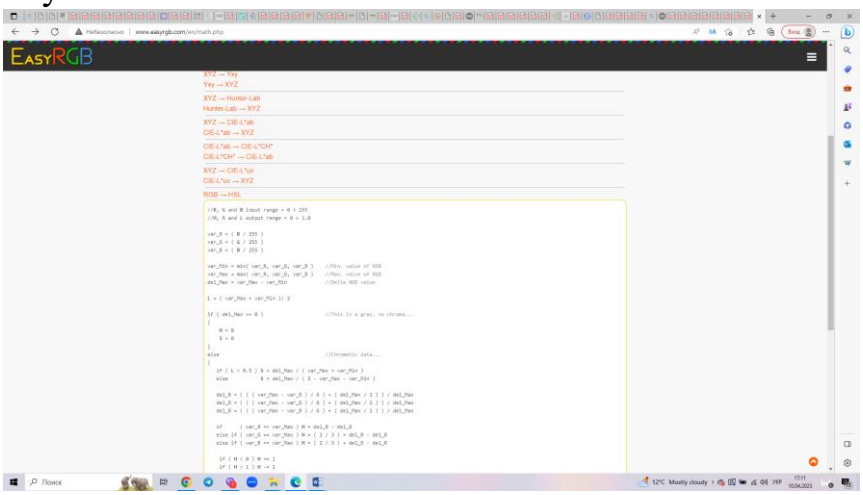
яскравість. Не існує ефективного і справедливого стандарту для вимірювання коефіцієнта контрастності, тому найкращий спосіб визначити його - покладатися на очі користувача.

**Постанова задачі**: метою роботи є розгляд та структурування за допомогою програмного продукту Функції програмування кольорів http://www.easyrgb.com/en/math.php

**Завдання** - потрібно за допомогою формул, які використовує колірний калькулятор перетворення колірних даних у різні колірні простори, розглянути моделі зображень за ступеню насиченості.




Кожна формула перетворення записується як «нейтральна функція програмування», яку легко перекласти на будь-яку конкретну мову програмування:RGB – HSL

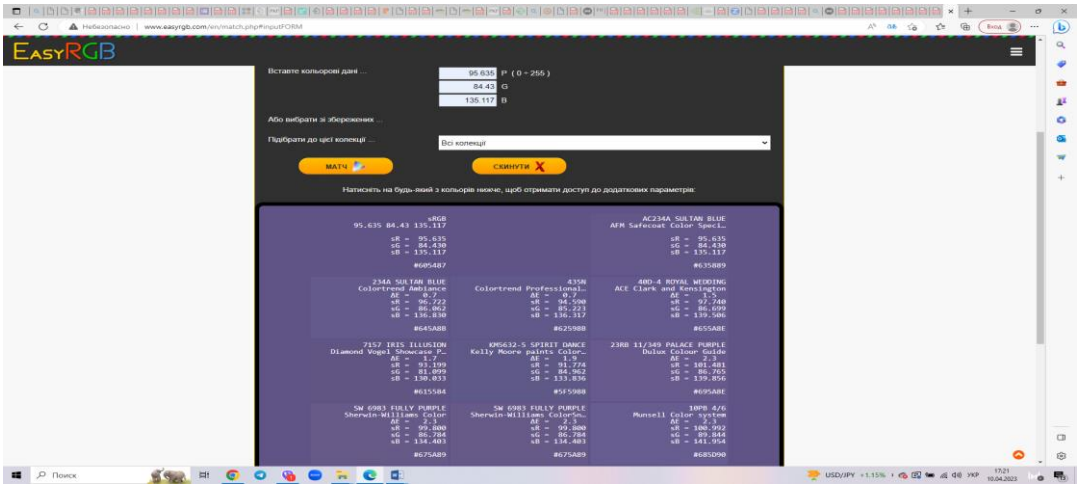



Хід роботи

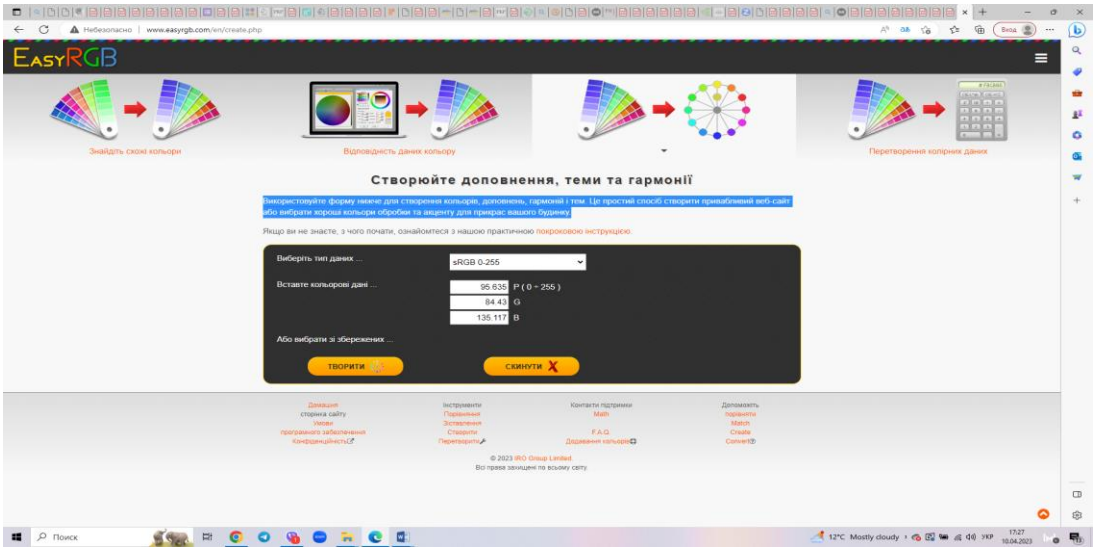
Зіставлення даних про кольори з комерційними кольорами

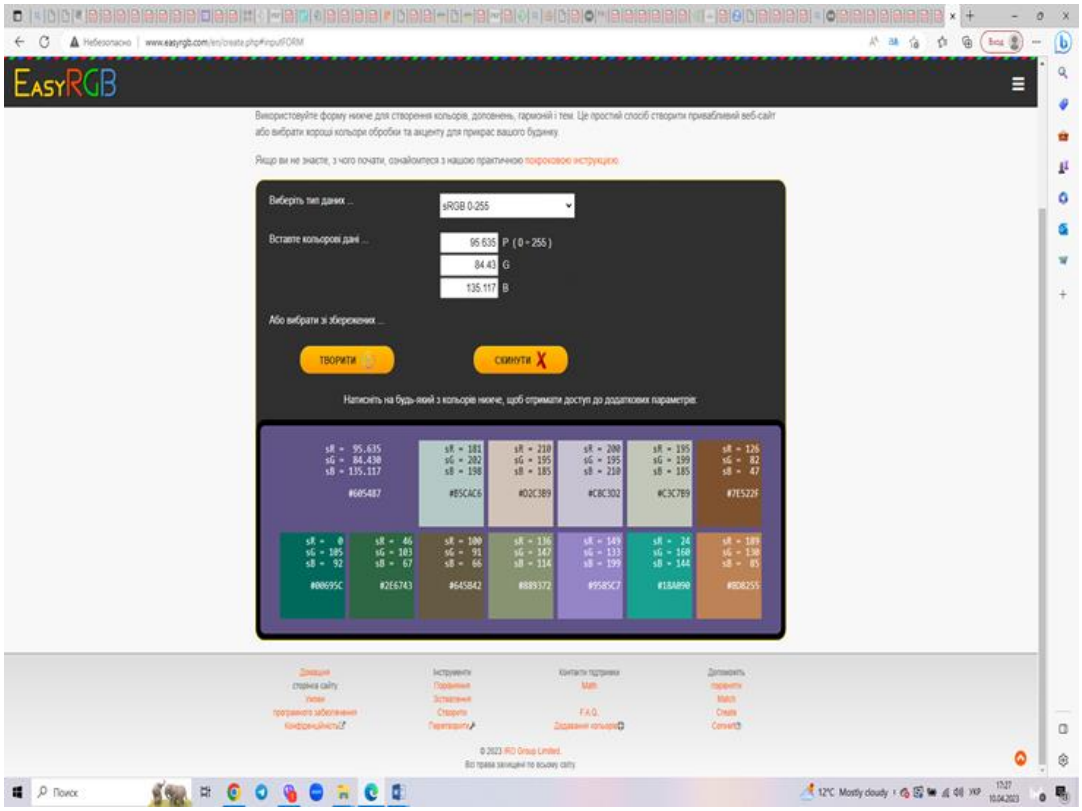
Використовуючи форму нижче, щоб зіставити свої колірні дані (RGB, XYZ тощо) для кольорових карток, ліній фарбування, фарб, фандек, стандартів тощо... Перетворите комп'ютерні кольори в продукти та посилання "реального світу".

1. Виберіть тип даних
2. Вставте кольорові дані
3. Підібрати до запропонованої колекції
4. Натиснути МАТЧ та натисніть на будь-який з кольорів нижче, щоб отримати доступ до додаткових параметрів



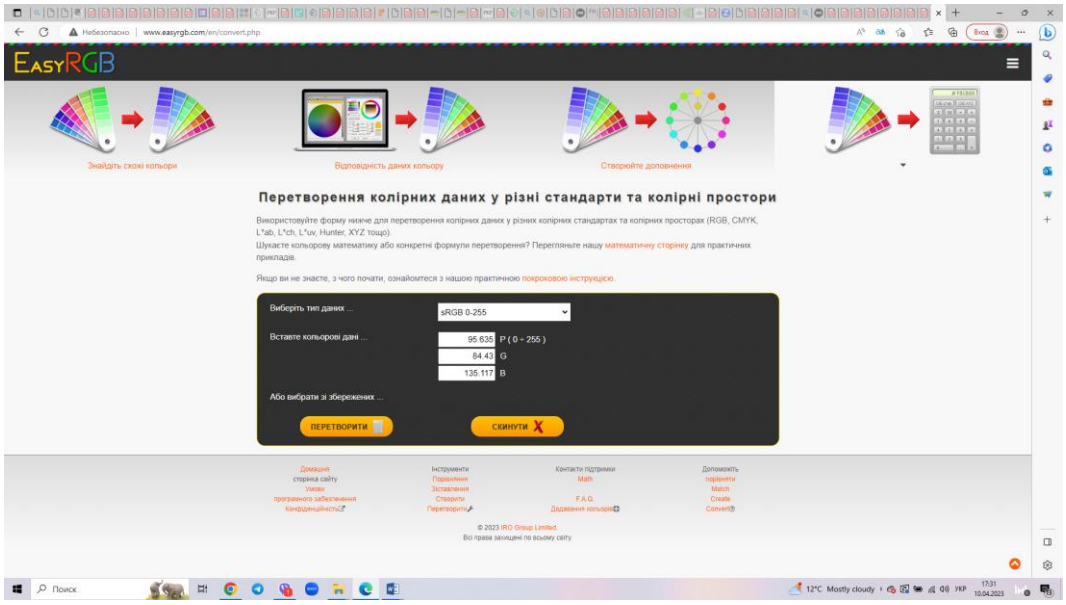

Використовуючи форму нижче для створення кольорів, доповнень, гармоній і тем. Це простий спосіб створити привабливий веб-сайт або вибрати хороші кольори обробки та акценту для прикрас вашого будинку.

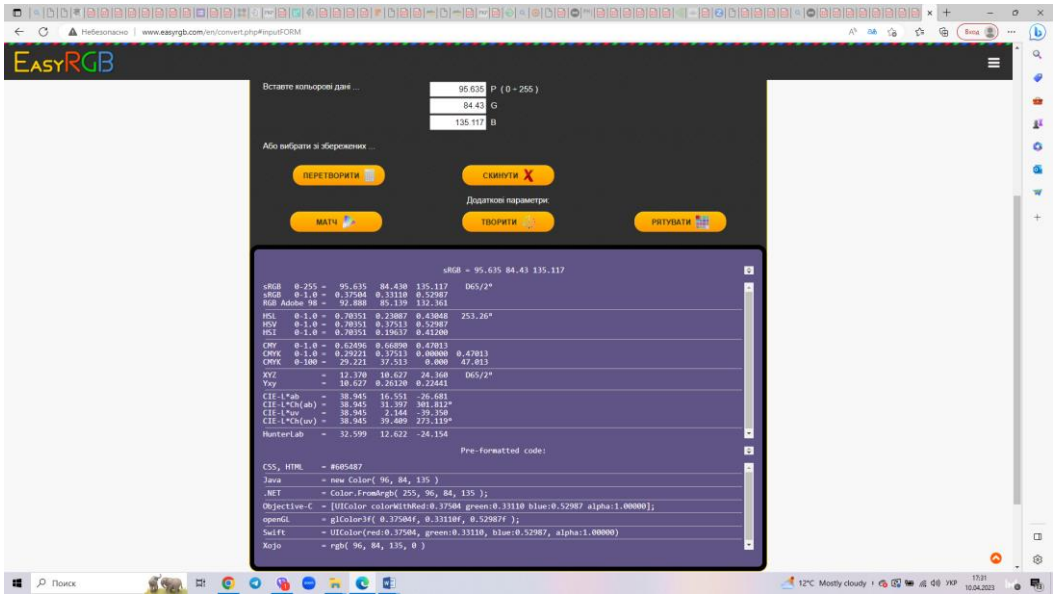





*Перетворення колірних даних у різні стандарти та колірні простори*

Використовуйте форму нижче для перетворення колірних даних у різних колірних стандартах та колірних просторах (RGB, CMYK, L*ab, L*ch, L*uv, Hunter, XYZ тощо).






## Перелік питань на захист лабораторної роботи

1.Методика оцінки зображень за ступеню насиченості.
2.Особливості комп’ютерної реалізації систем комп'ютерного зору.
3.Реалізація оцінки зображень за ступеню насиченості.

# ЛАБОРАТОРНА РОБОТА № 5

**Тема: «Реалізація алгоритму виявлення країв Собеля в системах комп'ютерного зору робота»**

Оператор Собеля заснований на згортку зображення невеликими сепарабельними цілими фільтрами у вертикальному і горизонтальному напрямках, тому його відносно легко обчислювати. З іншого боку, апроксимація градієнта, що використовується, досить груба, особливо це позначається на високочастотних коливаннях зображення.
Оператор обчислює градієнт яскравості зображення у кожній точці. Таким чином знаходиться напрям найбільшого збільшення яскравості та величина її зміни у цьому напрямку. Результат показує, наскільки «різко» або «плавно» змінюється яскравість зображення у кожній точці, отже, ймовірність знаходження точки межі, і навіть орієнтацію кордону.
**Задача**: на підставі вибірки зображень за допомогою алгоритмів вихідного коду розглянути можливості виявлення країв Собеля в системах комп'ютерного зору робота

**Хід роботи**:

Насправді обчислення величини зміни яскравості (ймовірності приналежності до грані) надійніше і простіше в інтерпретації, ніж розрахунок напрями.
Оператор використовує ядра 3×3, з якими згортає зображення для обчислення наближених значень часткових похідних по горизонталі та по вертикалі. Якщо A вихідне зображення, а Gx та Gy — два зображення, де кожна точка містить часткові похідні по x та по y відповідно. Вони обчислюються наступним чином:

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A}$$

Координата x зростає «направо», а y — «вниз». Для кожної точки зображення наближене значення градієнта обчислюється через наближенні значення часткових похідних:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Оператор Sobel не тільки дає хороший ефект виявлення, але і має ефект плавного придушення шуму, але отримані краї товщі і можуть з'явитися помилкові краї.
Оператор Sobel є важливим методом обробки даних в області комп'ютерного зору. В основному він використовується для створення градієнта цифрових зображень.
Візерунок оператора Sobel.

Gx=f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)-(f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)) Gy=f(x-1,y+1)+2f(x,y+1)+f(x+1,y+1)-(f(x-1,y-1)+2f(x,y-1)+f(x+1,y+1)-(f(x-1,y-1)

Gx=[-1 0 1] , Gy=[-1 -2 -1]
[-2 0 2] [ 0 0 0]
[-1 0 1] [ 1 2 1]

Результатом використання оператора Собеля є напрямок найбільшого збільшення яскравості та величина її зміни у цьому напрямку в кожній точці зображення. Це може показати, наскільки «різко» або «плавно» змінюється яскравість зображення у кожній точці, що може бути корисним для виявлення точок межі і навіть орієнтації кордону.

Сперш за все встановимо бібліотеку OpenCV для того аби застосовувати її функції та методи. Після успішного завантаження бібліотеки, напишемо реалізацію алгоритма Собеля на мові програмування C++. Наведемо відповідний код нижче (лістинг 1):

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include <string.h>

using namespace cv;

/**
  @function main
**/
int main(int argc, char** argv)
{

   Mat src, src_gray;
   Mat grad;
   std::string window_name = "Sobel Demo - Simple Edge Detector";
   int scale = 1;
   int delta = 0;
```

```
  int ddepth = CV_16S;

  int c;

  /// Load an image
  src = imread(argv[1]);

  if (!src.data)
  {
     return -1;
  }

  GaussianBlur(src, src, Size(3, 3), 0, 0, BORDER_DEFAULT);

  /// Convert it to gray
  cvtColor(src, src_gray, 0);

  /// Create window
  namedWindow(window_name, 1);

  /// Generate grad_x and grad_y
  Mat grad_x, grad_y;
  Mat abs_grad_x, abs_grad_y;

  /// Gradient X
  //Scharr( src_gray, grad_x, ddepth, 1, 0, scale, delta, BORDER_DEFAULT );
  Sobel(src_gray, grad_x, ddepth, 1, 0, 3, scale, delta, BORDER_DEFAULT);
  convertScaleAbs(grad_x, abs_grad_x);

  /// Gradient Y
  //Scharr( src_gray, grad_y, ddepth, 0, 1, scale, delta, BORDER_DEFAULT );
  Sobel(src_gray, grad_y, ddepth, 0, 1, 3, scale, delta, BORDER_DEFAULT);
  convertScaleAbs(grad_y, abs_grad_y);

  /// Total Gradient (approximate)
  addWeighted(abs_grad_x, 0.5, abs_grad_y, 0.5, 0, grad);

  imshow(window_name, grad);

  waitKey(0);

  return 0;
}
```

Лістинг 1 – Код програми

Цей код використовує функції OpenCV для обробки зображень, зокрема для виявлення ребер на зображенні. Саме у цьому випадку використовуються

функції Sobel та convertScaleAbs для обчислення градієнту зображення за допомогою диференціальних фільтрів Sobel. Результат обробки відображається у вікні з назвою "Sobel Demo - Simple Edge Detector".

Код починається з підключення необхідних бібліотек та оголошення змінних. Наступним кроком є завантаження зображення за допомогою функції imread та перевірка, чи вдалося завантажити зображення. Після завантаження зображення виконується операція розмиття зображення за допомогою фільтра Гаусса (GaussianBlur) з метою зменшення шуму на зображенні та поліпшення якості виявлення ребер.

Далі, зображення перетворюється у відтінки сірого за допомогою функції cvtColor. Потім створюється вікно з назвою "Sobel Demo - Simple Edge Detector".

Далі, використовуються функції Sobel та convertScaleAbs для обчислення градієнту зображення. Конкретно, обчислюється градієнт у напрямку x (горизонтальний градієнт) та y (вертикальний градієнт) за допомогою функції Sobel. Після цього використовується функція convertScaleAbs для перетворення результатів у беззнакове ціле число.

Нарешті, обчислені градієнти поєднуються за допомогою функції addWeighted, а результат відображається у вікні за допомогою функції imshow. Функція waitKey очікує на натискання клавіші та повертає код натисканої клавіші. Після цього програма повертається з кодом 0, що означає успішне виконання програми.

Результати обробки зображень

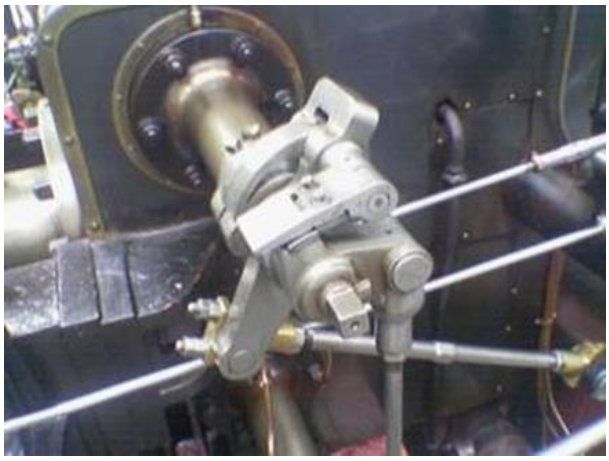


Рисунок 1- Вихідне зображення

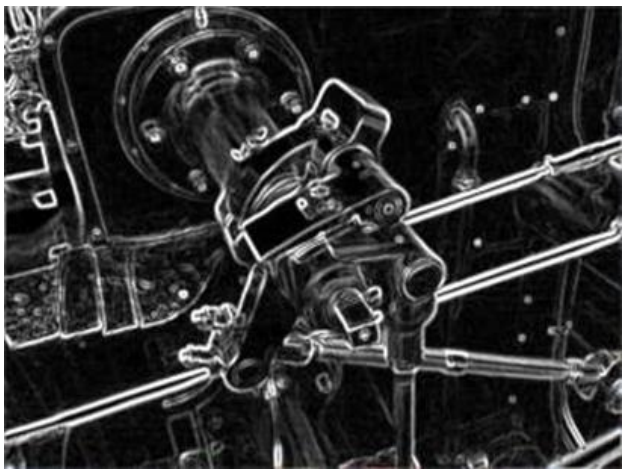


Рисунок 2 - Результат застосування оператору Собеля

**Перелік питань на захист лабораторної роботи**

1. Застосування в області комп'ютерного зору методів виділення кордонів
2. Огляд найвідоміших і часто використовуваних алгоритмів виділення меж об'єктів на зображеннях: методи Робертса, Превітта і Собел

# ПРАВИЛА ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

1. Текстовий та графічний матеріали записки друкують комп'ютерним способом на одному боці односортних білих аркушів формату А4 (розмір 210 Н 297 мм) через 1,5 міжрядковий інтервал, текст вирівнюють по ширині аркуша. Текстовий редактор – Word з пакета Microsoft Office, Open Office Writer, Star Office Writer та ін. Шрифт – Times New Roman Cyr, 14.

2. Оформлення ілюстрацій

Усі ілюстрації в пояснювальній записці (креслення, схеми, фотографії, діаграми, графіки) називають рисунками.

Кількість ілюстрацій має бути достатньою для пояснення тексту, який викладається. Ілюстрації потрібно розміщувати як по тексту записки (якомога ближче до відповідних частин тексту), так і в кінці його або наводити в додатках. Ілюстрації належить виконувати у відповідності до вимог стандартів ЄСКД і ЕСПД за допомогою різних графічних редакторів та систем автоматизованого проектування.

Усі ілюстрації послідовно нумерують у межах розділу арабськими цифрами. Номер ілюстрації складається з номера розділу і порядкового номера ілюстрації, наприклад, «Рис 2.5 Граф алгоритму». Посилання на ілюстрації подають так: «... на рис. 2.5 ...». Повторне посилання на ілюстрацію наводять із скороченням слова «дивись», наприклад, «... див. рис. 2.5 ...». Допускається нумерація ілюстрацій у межах лабораторної роботи.

Ілюстрації повинні мати назву, яку розміщують під ілюстрацією в одному рядку з її номером, наприклад, «Рис. 3.2. Схема». За потреби під назвою ілюстрації записують пояснювальні дані

Розмір шрифту всіх без винятку надписів у рисунках має бути таким самим, як і в тексті пояснювальної записки.

Ілюстрації розміщують так, щоб їх можна було розглядати, не повертаючи аркуш або повертаючи його за ходом стрілки годинника.

# СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ


Основна література

1. Вовк С.М., Гнатушенко В.В., Бондаренко М.В. Методи обробки зображень та компютерний зір // Навчальний посібник. Д.: ЛІРА. 2016. – 148 с

2. Дорощенков Г.Д. Системи телебачення та технічного зору: навчальний посібник. – Вінниця: ВНТУ, 2015. -209 с.

3. Ю.А.Ніцук, О.М.Семчак, Шаріпова І.В. Визначення шляхів зменшення похибок розрахунків координат бортовими ЕОМ автономного рухомого об'єкту для реалізації алгоритмів SLAM навігації // Збірник наукових праць Житомирський військовий інститут імені С. П. Корольова. – Житомир : ЖВІ, 2020.- № 27 (4). – С. 38 – 49.

Додаткова література

1.Глухов О. В. Вивчення властивостей мікроконтролерів і електронних систем на базі платформи Ардуіно : навч. посіб. / О. В. Глухов, О. О. Кравчук, Є. В. Левченко ; М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. – Харків : ХНУРЕ, 2019. – 192 с.

2. [Електронний ресурс]. – Режим доступу https://docs.nvidia.com/isaac/apps/carter/gmapping/doc/index.html

3.Артеменко В. В. Інформаційні технології в галузі туризму. Аналіз застосувань та результатів досліджень / О. І. Артеменко, В. В. Пасічник, В. В. Єгорова // Вісник Національного університету "Львівська політехніка". – 2015. – № 814: Інформаційні системи та мережі. – С. 3–22.

4. [Електронний ресурс]. – Режим доступу: https://opencv.org/releases

5. Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart. 2000. Sobel Edge Detector. Retrieved September 8, 2020. [Електронний ресурс]. – Режим доступу: https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm

6. Система технічного зору: особливості, завдання, принципи роботи, основні компоненти [Електронний ресурс]- Режим доступу: http://bigbro.com.ua/sistematehnichnogo-zoru-osoblivosti-zavdannya-printsipi-roboti-osnovni-komponenti/ //(дата звернення 13.05.2023)

# Додаток А

Одеський національний університет імені І. І. Мечникова

(повне найменування вищого навчального закладу)

Факультет математики, фізики та інформаційних технологій

(повне найменування інституту/факультету)

Кафедра комп’ютерних систем та технологій

(повна назва кафедри)

Звіт з лабораторної роботи № __

з дисципліни «Комп'ютерний зір та навігація робототехнічних систем»

Тема: _ «_____»

Виконав (-ла): студент (ка) денної форми навчання
спеціальності 123 «Комп’ютерна інженерія»
курс___гр. ___

Прізвище Ім’я _____

Керівник (доц.) кафедри КСТ
Прізвище Ім’я _____

Одеса – 2023

ДЛЯ НОТАТОК

*Навчальне видання*

# КОМП'ЮТЕРНИЙ ЗІР ТА НАВІГАЦІЯ РОБОТОТЕХНІЧНИХ СИСТЕМ

## ЗМІСТОВНИЙ МОДУЛЬ 2
## СИСТЕМИ ТЕХНІЧНОГО ЗОРУ

Методичні вказівки
до лабораторних занять для студентів
першого (бакалаврського) рівня вищої освіти
галузі знань 12 Інформаційні технології

***Укладач***
**Шаріпова** Ільнара Вільївна

*В авторській редакції*

Затвердж. авт. 2,55 умов. друк.арк. Шрифт Times New Roman.
Системні вимоги: операційна система сумісна з програмним забезпеченням для читання файлів формату PDF. Зам. № 1123-2701.

Видавництво та друк: ОЛДІ+
65101, Україна, м. Одеса, вул. Інглезі, 6/1
Свідоцтво ДК № 7642 від 29.07.2022 р.

Тел.: +38 (098) 559-45-45,
+38 (095) 559-45-45, +38 (093) 559-45-45
Для листування: 65101, Україна, м. Одеса, вул. Інглезі, 6/1
E-mail: office@oldiplus.ua