Лабораторна робота №3

Бібліотека OpenCV в Python

Мета: ознайомитись з основними можливостями та функціями бібліотеки OpenCV по обробці зображень

3.1 Теоретичні відомості

Гістограма інтенсивності зображення є фундаментальним інструментом цифрової обробки зображень, який описує, як значення яскравості пікселів розподілені у зображенні. Вона дозволяє оцінити контраст, освітлення, експозицію, а також ступінь деталізації сцени. У найпростішому вигляді гістограма — це статистичне представлення частот появи певних рівнів інтенсивності у межах зображення. Для побудови гістограми необхідно мати дискретне зображення I(x,y), де кожен піксель має значення яскравості III у діапазоні [0, $L-1$], де L — кількість можливих рівнів інтенсивності (для 8-бітного зображення L=256).

Формально гістограма визначається як функція *h(i)*, що підраховує кількість пікселів, значення яких дорівнює i:

$$h(i) = \#\{(x,y) \mid I(x,y) = i\}, \quad i = 0, 1, \ldots, L-1$$

Якщо N — загальна кількість пікселів у зображенні (N=M×N), то нормалізована гістограма визначається як:

$$p(i) = \frac{h(i)}{N}, \quad \sum_{i=0}^{L-1} p(i) = 1.$$

Ця нормалізована форма дозволяє розглядати гістограму як оцінку щільності ймовірності розподілу інтенсивностей у зображенні. Іншими словами, *p(i)* показує ймовірність того, що випадково вибраний піксель має яскравість *i*.

Будування гістограми реалізується через ітерацію по всіх пікселях зображення та інкрементування відповідного лічильника:

$$h(i) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \delta(I(x,y) - i),$$

де δ — дельта-функція, що дорівнює 1, якщо *I(x,y)=i*, і 0 в іншому випадку. Цей вираз показує, що гістограма — це просто сума всіх входжень певного рівня яскравості.

Якщо аналізується кольорове зображення, гістограми можуть бути побудовані для кожного каналу (R, G, B) окремо або у перетвореному просторі (наприклад, HSV). Гістограма інтенсивності у такому випадку може будуватися для компонента яскравості, наприклад, Y у моделі YCrCb або V у HSV.

Форма гістограми несе у собі велику кількість інформації про характеристики зображення. Якщо розподіл *p(i)* концентрується у вузькому діапазоні значень, то зображення має низький контраст — його динамічний діапазон обмежений. Якщо розподіл рівномірний, зображення містить як темні, так і світлі ділянки — воно контрастне. Якщо гістограма зсунута вліво (до i=0), це свідчить про недоекспозицію — більшість пікселів темні. Якщо ж вона зсунута вправо (до i=255), це ознака пересвітлення. Багатомодальна гістограма з кількома піками може вказувати на наявність різних об'єктів або зон у сцені — наприклад, фон, об'єкт і тінь.

З гістограми можна вивести різні статистичні параметри. Середня яскравість (математичне сподівання) обчислюється як:

$$\mu = \sum_{i=0}^{L-1} i \cdot p(i)$$

Це середнє значення яскравості всіх пікселів у зображенні. Дисперсія, що характеризує контраст, визначається як:

$$\sigma^2 = \sum_{i=0}^{L-1} (i - \mu)^2 \cdot p(i),$$

а стандартне відхилення σ показує ступінь розкиду значень інтенсивності відносно середнього. Якщо σ мале, то зображення має низький контраст; якщо велике — контраст високий. Медіанна яскравість im визначається як значення, при якому кумулятивна сума нормалізованої гістограми дорівнює 0.5:

$$\sum_{i=0}^{i_m} p(i) = 0.5.$$

З математичної точки зору, гістограма — це наближена оцінка дискретної щільності розподілу P(I), яка зводить складне зображення до одновимірного статистичного опису. Вона не зберігає просторової інформації, тобто два різні зображення можуть мати однакову гістограму, але зовсім іншу структуру. Тому для більш глибокого аналізу часто використовують локальні гістограми, які будуються для обмежених ділянок зображення. У поєднанні з іншими методами гістограма є потужним інструментом для аналізу, обробки та оптимізації зображень. Таким чином, побудова гістограми — це перший і найпростіший крок у математичному описі візуальної інформації. Вона дозволяє отримати кількісну характеристику інтенсивності, на основі якої будуються більш складні методи корекції, сегментації та аналізу. Хоча сама гістограма є лише частотним розподілом, її аналітична потужність полягає у здатності відображати ключові властивості зображення через просту статистичну форму.

У цифровій обробці зображень фільтри розмиття (або згладжувальні фільтри) є фундаментальним інструментом для зменшення шуму, усунення дрібних артефактів і підготовки даних до подальших етапів обробки, таких як детекція контурів, сегментація чи морфологічні операції. Найпоширенішими фільтрами, що застосовуються для згладжування,

є усереднювальний (blur)**,** гаусовий (Gaussian Blur)**,** медіанний (Median Blur) **та** білатеральний (Bilateral Filter)**.**

Найпростіший фільтр розмиття — це усереднювальний фільтр, який базується на концепції згортки з ядром, елементи якого мають однакові ваги. Для кожного пікселя зображення *I(x,y)* нове значення визначається як середнє арифметичне інтенсивностей усіх пікселів у його локальному вікні *k*×*k*:

$$I'(x,y) = \frac{1}{k^2} \sum_{i=-r}^{r} \sum_{j=-r}^{r} I(x+i, y+j),$$

де r= $\frac{k-1}{2}$ — радіус фільтра.

Таке згладжування еквівалентне згортці з ядром:

$$H = \frac{1}{k^2} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}.$$

Цей метод ефективно зменшує випадковий шум, але має істотний недолік — він «розмазує» краї об’єктів, оскільки всі пікселі у вікні мають однакову вагу. Результатом є втрата чіткості та дрібних деталей. Усереднювальний фільтр є базовим орієнтиром для порівняння ефективності інших фільтрів.

На відміну від простого усереднення, гаусовий фільтр враховує просторову відстань між центральним пікселем і сусідніми. Він використовує ядро, побудоване за законом нормального **(**гаусового**)** розподілу, де більші ваги надаються пікселям, ближчим до центру, а віддалені мають менший вплив. Математична модель гаусового ядра в двовимірному випадку має вигляд:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

де σ — стандартне відхилення, яке визначає ступінь розмиття.

Вихідне зображення *I′(x,y)* отримується шляхом згортки:

$$I'(x,y) = \sum_{i=-r}^{r} \sum_{j=-r}^{r} I(x+i, y+j) \cdot G(i,j).$$

Гаусовий фільтр має низку важливих властивостей. Він лінійний і ізотропний, тобто згладжує зображення однаково в усіх напрямках. При цьому він ефективно пригнічує високочастотний шум, зберігаючи плавні переходи яскравості. На відміну від звичайного усереднення, цей метод менш схильний до розмиття країв, оскільки пікселі далі від центру мають менший вплив. У частотній області гаусовий фільтр еквівалентний низькочастотному фільтру, який пропускає повільні зміни інтенсивності й блокує різкі.

Параметр σ контролює ступінь згладження: мале σ зберігає деталі, велике — викликає сильне розмиття. Практично ядро обмежується до розміру 6σ+1, оскільки за межами цього діапазону вплив ваг стає незначним.

На відміну від попередніх двох, медіанний фільтр є нелінійним. Замість обчислення середнього значення інтенсивностей сусідніх пікселів, він вибирає медіану — тобто середнє значення у впорядкованій множині яскравостей у вікні. Для пікселя I(x,y) результат обчислюється як:

$$I'(x,y) = \text{median}\{I(x+i, y+j) \mid -r \le i,j \le r\}.$$

Цей метод ефективний для усунення імпульсного шуму (“сіль і перець”), при якому поодинокі пікселі мають або мінімальні, або максимальні значення (0 або 255). Оскільки медіана не залежить від поодиноких викидів, вона не спотворює сусідні пікселі. Медіанний фільтр добре зберігає краї об’єктів, що є його основною перевагою порівняно з лінійними методами. Проте при надмірному розмірі вікна може втрачатися дрібна текстура.

Медіанний фільтр не має аналітичного виразу згортки, оскільки він нелінійний і не може бути описаний простою системою ваг. Його можна розглядати як локальну статистичну операцію над множиною пікселів.

Білатеральний фільтр є більш складним методом, який поєднує принципи просторового згладжування (як у гаусовому фільтрі) з урахуванням подібності яскравостей пікселів**.** Він дозволяє згладжувати однорідні ділянки, водночас зберігаючи чіткі краї об’єктів. Його математична модель виглядає так:

$$I'(x,y) = \frac{1}{W_p} \sum_{i=-r}^{r} \sum_{j=-r}^{r} I(x+i, y+j) \cdot f_s(i,j) \cdot f_r(I(x+i, y+j) - I(x,y)),$$

$f_s(i,j) = e^{-\frac{i^2+j^2}{2\sigma_s^2}}$ — просторова (spatial) вага, що враховує відстань між пікселями,

$f_r(\Delta I) = e^{-\frac{(\Delta I)^2}{2\sigma_r^2}}$ — радіометрична (range) вага, що залежить від різниці яскравостей,
а W_p— нормалізувальний коефіцієнт:

$$W_p = \sum_{i=-r}^{r} \sum_{j=-r}^{r} f_s(i,j) \cdot f_r(I(x+i, y+j) - I(x,y)).$$

Таким чином, білатеральний фільтр зменшує вплив пікселів, які далеко від поточного або мають різку різницю в інтенсивності. Завдяки цьому краї об’єктів зберігаються, оскільки різниця інтенсивностей між фоном і краєм обмежує вагу пікселів по обидва боки межі. Цей фільтр є нелінійним і потребує більших обчислювальних витрат, ніж гаусовий, але він забезпечує одне з найкращих співвідношень між зниженням шуму та збереженням структури. Параметри σ_s (просторова дисперсія) і σ_r (радіометрична дисперсія) визначають ступінь впливу просторової близькості та подібності інтенсивностей відповідно.

У цифровій обробці зображень виявлення контурів є одним із базових завдань, яке лежить в основі більш складних процесів — сегментації, розпізнавання об’єктів, вимірювання форми, побудови моделей та аналізу сцени. Основна ідея полягає у визначенні меж між областями, які відрізняються за яскравістю або кольором.

Оператор Собеля — це простий, але ефективний метод виявлення градієнтів зображення. Він використовується для визначення напрямку та величини змін яскравості, тобто для пошуку ділянок, де інтенсивність змінюється найшвидше — це і є межі об’єктів. Sobel належить до класу градієнтних операторів, які використовують похідні першого порядку.

У дискретному вигляді похідну за напрямками X та Y можна оцінити за допомогою згортки з фільтрами:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

де G_x виявляє вертикальні краї (зміни по горизонталі), а G_y — горизонтальні (зміни по вертикалі). Для кожного пікселя зображення $I(x,y)$ обчислюється:

$$I_x = I * G_x, \qquad I_y = I * G_y,$$

Далі обчислюється **модуль градієнта**:

$$M(x,y) = \sqrt{I_x^2 + I_y^2},$$

та кут напряму:

$$\theta(x,y) = \arctan\left(\frac{I_y}{I_x}\right).$$

Яскраві ділянки на зображенні M(x,y) відповідають границям між об’єктами. Оператор Собеля добре підходить для швидкого аналізу контрастних меж, однак чутливий до шуму. Тому перед його застосуванням зображення часто згладжують гаусовим фільтром.

Виявлення країв за методом Canny – це метод вилучення корисної структурної інформації з різних об'єктів зору та значного зменшення обсягу даних, що потребують обробки. Він широко застосовується в різних системах комп'ютерного зору . Canny виявив, що вимоги до застосування виявлення країв у різних системах зору є відносно схожими. Таким чином, рішення для виявлення країв, яке відповідає цим вимогам, може бути реалізоване в широкому діапазоні ситуацій. Загальні критерії для виявлення країв включають:

1. Виявлення краю з низьким рівнем помилок, що означає, що виявлення повинно точно вловлювати якомога більше країв, показаних на зображенні
2. Крайова точка, виявлена оператором, повинна точно локалізуватися в центрі краю.
3. Даний край на зображенні слід позначати лише один раз, і, по можливості, шум зображення не повинен створювати хибних країв.

Щоб задовольнити ці вимоги, Канні використав варіаційне числення – метод, який знаходить функцію , що оптимізує заданий функціонал . Оптимальна функція в детекторі Канні описується сумою чотирьох експоненціальних членів, але її можна апроксимувати першою похідною гауссової функції .

Серед розроблених на сьогодні методів виявлення країв, алгоритм Канні є одним із найсуворіших методів, який забезпечує хороше та надійне виявлення. Завдяки своїй

оптимальності для задоволення трьох критеріїв виявлення країв та простоті процесу його реалізації, він став одним із найпопулярніших алгоритмів для виявлення країв.

Згладжування.
Щоб уникнути помилкових контурів через шум, зображення попередньо фільтрують Гаусом:

$$I_s = I * G_\sigma,$$

де Gσ — гаусівське ядро з дисперсією σ. Це дозволяє приглушити випадкові коливання яскравості, не знищуючи загальної структури.

Обчислення градієнтів.
Використовуючи оператор Sobel або схожі маски, обчислюють I_x,I_y, модуль градієнта M(x,y) і напрям θ(x,y).

Немаксимальне придушення (Non-Maximum Suppression).

На цьому етапі Canny залишає лише ті пікселі, які є локальними максимумами градієнта в напрямку θ. Це дозволяє отримати тонкі однопіксельні краї. Якщо значення пікселя не є максимумом уздовж напряму градієнта, воно замінюється нулем.

Подвійне порогування.
Для розрізнення істинних та хибних контурів використовуються два пороги — нижній T_L і верхній T_H :

- якщо $M(x,y) > T_H$, піксель вважається сильним краєм;
- якщо $T_L \le M(x,y) \le T_H$, він є слабким краєм;
- якщо $M(x,y) < T_L$, то цей піксель відкидається.

Відстеження контурів за гістерезисом.
Слабкі пікселі зберігаються лише якщо вони з'єднані зі сильними сусідами, що дозволяє прибрати шумові точки, зберігши зв'язні контури.

У результаті Canny формує тонкі, безперервні лінії, що точно окреслюють межі об'єктів. Цей метод вважається одним із найнадійніших для виділення контурів у комп'ютерному зорі.

Після отримання бінарного зображення з контурами (наприклад, після Canny) часто виникає потреба визначити геометричну структуру цих контурів. Для цього в OpenCV використовується функція cv.findContours(), яка аналізує бінарну маску й повертає набір контурів, кожен з яких є списком координат точок, що утворюють замкнену криву. Основна ідея полягає у пошуку меж між областями чорного (0) і білого (255) на зображенні. Алгоритм обходить усі пікселі маски та формує послідовність точок, які обмежують об'єкти. Результатом є набір контурів і, опціонально, їхня ієрархія — відношення вкладеності (наприклад, контур «дірки» всередині іншого об'єкта).

У функції можна вказати:

- **режим пошуку** (`cv.RETR_EXTERNAL`, `cv.RETR_LIST`, `cv.RETR_TREE`), який визначає, чи потрібно зберігати лише зовнішні контури, усі без ієрархії, чи повну структуру;
- **метод апроксимації** (`cv.CHAIN_APPROX_NONE`, `cv.CHAIN_APPROX_SIMPLE`), який задає рівень спрощення — чи зберігати всі пікселі контуру, чи лише ключові точки.

Після знаходження контурів OpenCV дозволяє виконувати геометричний аналіз: обчислювати площу, периметр, обмежувальні прямокутники чи кола, визначати форму

об’єкта, центроїд та моменти інерції. Наприклад, площа контуру C з вершинами (xk,yk може бути знайдена за формулою «шнурівки» (Shoelace):

$$A = \frac{1}{2}\left|\sum_{k=1}^{n}(x_k y_{k+1} - x_{k+1} y_k)\right|.$$

Розпізнавання обличь за допомогою haarcascade_frontalface_default.xml це є навченою модель для детекції облич на зображеннях, побудована на основі каскадних класифікаторів Гаара (Haar Cascade Classifier). Цей метод є класичним підходом до комп’ютерного зору, який до появи глибоких нейромереж широко використовувався в системах розпізнавання облич, жестів і об’єктів у реальному часі. Його принципи були запропоновані Полом Віолою (Paul Viola) та Майклом Джонсом (Michael Jones) у 2001 році в роботі “Rapid Object Detection using a Boosted Cascade of Simple Features”. Основна ідея методу полягає у **розпізнаванні об’єктів (зокрема, облич)** за допомогою набору простих **Haar-подібних ознак**, які обчислюються над різними ділянками зображення. Замість того, щоб безпосередньо аналізувати всі пікселі, алгоритм розглядає **відносні зміни яскравості** між сусідніми областями — наприклад, між очима (темніші зони) і щоками (світліші). Ці контрасти й утворюють характерні шаблони для певних об’єктів, у нашому випадку — для людського обличчя. **Haar-ознака** — це елементарний шаблон, який вимірює різницю середніх інтенсивностей двох або більше сусідніх прямокутних областей. Найпростіша ознака складається з двох прямокутників: один світлий, інший темний. Вона може описувати, наприклад, контраст між областю очей і верхньою частиною щік. Щоб швидко обчислювати суму пікселів у будь-якому прямокутнику, вводиться поняття інтегрального зображення (integral image).
Це проміжне представлення, у якому кожен піксель зберігає суму всіх значень пікселів, розташованих вище та лівіше від нього.

Оскільки не всі ознаки інформативні, алгоритм **AdaBoost (Adaptive Boosting)** використовується для вибору невеликої кількості найефективніших ознак і побудови з них **сильного класифікатора**.

- Кожна Haar-ознака розглядається як **слабкий класифікатор**, який лише частково відрізняє обличчя від не-облич.
- Алгоритм поступово комбінує їх, надаючи більшу вагу тим, що найкраще класифікують приклади, і зменшуючи вагу менш інформативних.
- У результаті формується ланцюг послідовних класифікаторів, які разом дають високу точність розпізнавання.

Процес навчання вимагає великої кількості позитивних прикладів (зображень облич) і негативних прикладів (фонових зображень без облич), з яких формується навчальний набір. Сам файл haarcascade_frontalface_default.xml містить уже навчені ваги та параметри цих класифікаторів, тому його можна одразу використовувати без повторного навчання.

Щоб знайти обличчя, алгоритм послідовно сканує зображення ковзним вікном (sliding window) різного розміру та масштабу. Для кожного положення обчислюються ознаки Гаара, і вікно проходить крізь каскад класифікаторів.

Якщо ділянка проходить усі рівні, вона вважається обличчям. Результати часто перекриваються, тому після виявлення застосовується NMS (Non-Maximum Suppression), щоб об’єднати близькі прямокутники в один.

Таким чином, результатом роботи алгоритму є координати прямокутників (x,y,w,h), які позначають положення облич на зображенні.

```
face_xml = ensure_cascade("haarcascade_frontalface_default.xml")
```

```
eye_xml  = ensure_cascade("haarcascade_eye.xml")

face_cascade = cv.CascadeClassifier(face_xml)

eye_cascade  = cv.CascadeClassifier(eye_xml)
```

Визначення ключових точок виявлення на зображенні ключових ознак (features), які є унікальними і добре розпізнаваними. Це можуть бути кути, точки з високим контрастом або локальні фрагменти текстури. Такі точки мають бути стійкими до змін масштабу, кута огляду, освітлення чи шуму. Алгоритми на зразок SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features) або ORB (Oriented FAST and Rotated BRIEF) аналізують градієнти яскравості в околі кожного пікселя, визначаючи ті, де зміни інтенсивності найвиразніші. Сенс цього етапу полягає у тому, щоб знайти невелику, але інформативну підмножину пікселів, які репрезентують структуру сцени. Кожна така точка описується дескриптором — вектором числових характеристик, що описує локальне оточення точки. Наприклад, у SIFT дескриптор описує розподіл напрямків градієнта навколо точки, що робить його стійким до поворотів і масштабування.

```
sift = cv2.SIFT_create()

kp1, des1 = sift.detectAndCompute(img1, None)

kp2, des2 = sift.detectAndCompute(img2, None)
```

Після того, як ключові точки знайдені на двох зображеннях (наприклад, у лівій і правій частині стереопари), потрібно визначити, які з них відповідають одна одній — тобто знайти пари відповідних точок. Цей процес називається feature matching. Для кожної точки з першого зображення шукається точка на другому, дескриптор якої є найбільш схожим. Схожість визначається за відстанню між дескрипторами, зазвичай за евклідовою або мангеттенською метрикою. Після первинного підбору кореспонденцій застосовується відсів хибних відповідностей. Один із найвідоміших критеріїв — ratio test (тест відношення): якщо відношення відстані до найближчого сусіда до відстані до другого за близькістю перевищує певний поріг, відповідність вважається ненадійною і відкидається. Це дозволяє залишити лише ті збіги, які мають дійсну геометричну природу.

```
FLANN_INDEX_KDTREE = 1

index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=5)

search_params = dict(checks=50)

flann = cv2.FlannBasedMatcher(index_params, search_params)

matches = flann.knnMatch(des1, des2, k=2)

good = []

pts1 = []

pts2 = []

matches = sorted(matches, key=lambda x: x[0].distance)

for i, (m, n) in enumerate(matches):

    if m.distance < 0.6 * n.distance:

        good.append(m)

        pts2.append(kp2[m.trainIdx].pt)

        pts1.append(kp1[m.queryIdx].pt)
```

Після визначення пар ознак можна дослідити епіполярну геометрію — просторове співвідношення між двома камерами. Вона описується так званою фундаментальною матрицею, яка відображає зв'язок між координатами точок на першому та другому зображеннях. Цей зв'язок гарантує, що кожна точка на одному зображенні має відповідати прямій на іншому, що проходить через епіполю. Таким чином, замість пошуку у двох вимірах задача звужується до пошуку вздовж однієї лінії — епіполярної прямої. На практиці це дозволяє суттєво зменшити кількість можливих хибних відповідностей і зробити обчислення швидшими. Для обчислення фундаментальної матриці зазвичай використовується алгоритм RANSAC, який відкидає точки, що не відповідають єдиній геометричній моделі.

```
F, mask = cv2.findFundamentalMat(pts1, pts2, cv2.FM_RANSAC)
```

Коли відомо, які точки на двох зображеннях є відповідними, можна виміряти їхню диспаратність — різницю в положенні точки на лівому та правому кадрах. У стереопарі об'єкти, які ближче до камери, зміщуються більше, а ті, що далі, — менше. Саме ця різниця і є ключем до відновлення глибини. Глибина кожної точки обчислюється з використанням параметрів камери — фокусної відстані та базису (відстані між камерами). Чим менша диспаратність, тим більша відстань до об'єкта. Результатом є карта глибини (depth map) — зображення, у якому яскравість кожного пікселя відповідає його відстані від камери. На таких картах світлі зони означають близькі об'єкти, а темні — далекі.

```
stereo = cv2.StereoBM_create(numDisparities=16*6, blockSize=11)

disparity2 = stereo.compute(img2, img1)
```

3.2 Порядок виконання лабораторної роботи

1. Обрати зображення відповідно до варіанту;
2. Побудувати гістограму яскравості до зображення;
3. Застосувати фільтри blur, GaussianBlur, medianBlur, bilateralFilter та порівняти вплив на зображення;
4. До обраного зображення застосувати детектор контурів Sobel, Canny знайти контури за допомогою findContours;
5. Обрати нове зображення де присутні люди та видно їх обличчя;
6. Завантажити haarcascade_frontalface_default.xml файл та за допомогою нього виявити обличчя та очі людей які зображенні на фото. Після чого порахувати кількість виявлених обличь та нанести рамки;
7. Обрати стереопару зображень та визначити ключові точки за допомогою доступних алгоритмів та з'єднати їх;
8. На основі п.7 побудувати карту глибини використовуючи StereoBM і StereoSGBM.
9. Зробити висновки.

3.3 Зміст звіту

1. Найменування і мета роботи;
2. Код по кожному пункту порядку виконання роботи;
3. Результати роботи по кожному пункту виконання роботи;
4. Висновки.

3.4 Контрольні запитання

1. У чому суть алгоритму SIFT? Які етапи він включає?
2. Що таке дескриптор ознаки? Як він формується?
3. Як працює алгоритм Canny і чим він відрізняється від простих фільтрів?
4. Що показує гістограма інтенсивності зображення?
5. Що таке диспаратність і як вона пов'язана з відстанню до об'єкта?