МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

ЛЕКЦІЯ 3. Введення в комп'ютерний зір

Львів -- 2025

Лекція зі штучного інтелекту 2025-04

Вступ

На цьому занятті ми розглянемо комп'ютерний зір — галузь штучного інтелекту, яка дозволяє комп'ютерам "бачити" та інтерпретувати візуальну інформацію з навколишнього світу. Ми ознайомимося з основними концепціями, алгоритмами та методами обробки зображень, а також з практичними застосуваннями технологій комп'ютерного зору.

Теми, що розглядаються

1. Поняття комп'ютерного бачення та машинного навчання
2. Технологічний стек OpenCV
3. Задачі комп'ютерного зору
4. Представлення зображень та кольорів
5. Базові операції з зображеннями
6. Частотна обробка зображень
7. Фільтрація зображень
8. Виявлення країв та контурів
9. Порогова обробка (Thresholding)
10. Морфологічні перетворення
11. Зіставлення шаблонів та геометричні перетворення
12. Сегментація зображень
13. Виявлення особливих точок та дескриптори

Мова програмування Python як база до машинного навчання і комп'ютерного зору

Історія та популярність Python

Python — це високорівнева, інтерпретована мова програмування, яка набула великої популярності завдяки своїй простоті та зрозумілості. Розроблена в 1991 році Guido van Rossum, Python отримала свою назву від британського комедійного шоу "Monty Python's Flying Circus".

Чому Python?

Python став домінуючою мовою в галузі машинного навчання та комп'ютерного зору завдяки кільком ключовим перевагам:

1. **Простота та читабельність**: Синтаксис Python інтуїтивно зрозумілий і нагадує псевдокод, що дозволяє дослідникам та інженерам зосередитись на алгоритмах, а не на особливостях мови.

2. **Багата екосистема бібліотек**:

- **NumPy**: ефективні операції з багатовимірними масивами
- **Pandas**: маніпуляція та аналіз даних
- **Matplotlib/Seaborn**: візуалізація даних
- **SciPy**: наукові обчислення
- **Scikit-learn**: класичні алгоритми машинного навчання
- **OpenCV**: комп'ютерний зір
- **TensorFlow/PyTorch/Keras**: глибоке навчання

3. **Гнучкість**: Python підтримує різні парадигми програмування (об'єктно-орієнтоване, функціональне, процедурне), що дозволяє вибирати найбільш підходящий стиль для конкретної задачі.

4. **Інтерпретована природа**: Швидке прототипування та інтерактивна розробка без необхідності компіляції.

5. **Спільнота та підтримка**: Величезна спільнота розробників, дослідників та ентузіастів, які створюють документацію, навчальні матеріали та відкритий код.

Python у комп'ютерному зорі

У галузі комп'ютерного зору Python став стандартом де-факто завдяки:

- **Інтеграції з OpenCV**: Потужний Python-інтерфейс до бібліотеки OpenCV
- **Підтримці обробки зображень**: Бібліотеки як Pillow (PIL), scikit-image
- **Ефективній роботі з даними**: NumPy для швидких операцій з масивами зображень
- **Інтеграції з глибоким навчанням**: Легке використання CNN та інших архітектур для задач комп'ютерного зору

Як працює Python

Python — це інтерпретована мова, що означає:

1. **Виконання коду**: Інтерпретатор Python читає та виконує код рядок за рядком
2. **Динамічна типізація**: Типи змінних визначаються під час виконання
3. **Автоматичне керування пам'яттю**: Збирач сміття автоматично звільняє невикористовувану пам'ять
4. **GIL (Global Interpreter Lock)**: Обмеження, яке дозволяє виконувати лише один потік Python одночасно

Для обчислювально інтенсивних задач машинного навчання та комп'ютерного зору, Python часто використовується як "клей" для високопродуктивних бібліотек, написаних на C/C++.

Jupyter Notebook

Jupyter Notebook — це веб-додаток з відкритим кодом, який дозволяє створювати та ділитися документами, що містять:

- **Живий код**: Виконання коду Python безпосередньо в браузері

- **Візуалізації**: Вбудовані графіки та діаграми
- **Текст з розміткою**: Документація з використанням Markdown
- **Математичні формули**: Підтримка LaTeX

Переваги Jupyter Notebook для машинного навчання та комп'ютерного зору:

1. **Інтерактивна розробка**: Миттєвий зворотний зв'язок при експериментах з алгоритмами
2. **Покрокова візуалізація**: Можливість бачити проміжні результати обробки зображень
3. **Документування процесу**: Поєднання коду, пояснень та результатів в одному документі
4. **Спільна робота**: Легкий обмін експериментами з колегами
5. **Відтворюваність**: Документування всього процесу від даних до результатів

Google Colab

Google Colaboratory (Colab) — це безкоштовний хмарний сервіс на основі Jupyter Notebook, який надає:

- **Безкоштовний доступ до GPU/TPU**: Прискорення обчислень для глибокого навчання
- **Попередньо встановлені бібліотеки**: Більшість популярних бібліотек ML/CV вже доступні
- **Інтеграція з Google Drive**: Легкий доступ до даних
- **Спільна робота в реальному часі**: Як у Google Docs

Віртуальні середовища

Для ефективної розробки проектів з комп'ютерного зору рекомендується використовувати віртуальні середовища:

- **venv/virtualenv**: Стандартні інструменти Python для ізоляції залежностей
- **conda**: Потужний менеджер пакетів та середовищ, особливо корисний для наукових обчислень
- **poetry**: Сучасний інструмент для управління залежностями та пакетування

Віртуальні середовища дозволяють уникнути конфліктів між різними версіями бібліотек та забезпечують відтворюваність експериментів.

Початок роботи з Python для комп'ютерного зору

Типовий процес налаштування середовища:

1. **Встановлення Python**: Завантаження з python.org або через Anaconda
2. **Створення віртуального середовища**: `python -m venv cv_env` або `conda create -n cv_env`
3. **Активація середовища**: `source cv_env/bin/activate` (Linux/Mac) або `cv_env\Scripts\activate` (Windows)
4. **Встановлення бібліотек**: `pip install numpy opencv-python matplotlib jupyter`
5. **Запуск Jupyter**: `jupyter notebook` або використання IDE з підтримкою Jupyter (VS Code, PyCharm)

Цей підхід забезпечує ізольоване, відтворюване середовище для розробки та експериментів з комп'ютерним зором.

Приклади коду для комп'ютерного зору з Python та OpenCV

Нижче наведено базові приклади використання OpenCV для типових задач комп'ютерного зору.

Завантаження та відображення зображення

```
import cv2
image = cv2.imread('image.jpg')
cv2.imshow('Image', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Базові операції з зображеннями

```
import cv2
image = cv2.imread('image.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Зміна розміру зображення
resized = cv2.resize(image, (800, 600))

# Зміна яскравості та контрасту
enhanced = cv2.convertScaleAbs(image, alpha=1.2, beta=0)

# Збереження зображення
cv2.imwrite('processed_image.jpg', enhanced)
```

Частотна обробка зображень

```
import cv2
import numpy as np

# Зчитування зображення
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Додавання шуму
noise = np.random.normal(0, 20, image.shape)
noisy_image = image + noise

# Фільтрація шуму
denoised = cv2.GaussianBlur(noisy_image, (5, 5), 0)


### Комп'ютерний зір (Computer Vision)

Комп'ютерний зір — це галузь штучного інтелекту, яка займається розробкою методів та алгоритмів

Основні відмінності від обробки зображень:
- **Обробка зображень** зосереджується на перетворенні зображень (фільтрація, покращення якості
- **Комп'ютерний зір** прагне до розуміння вмісту зображень (розпізнавання об'єктів, сцен)
```

Зв'язок з машинним навчанням

Комп'ютерний зір тісно пов'язаний з машинним навчанням, особливо з глибоким навчанням:

- **Традиційний підхід**: використання ручно розроблених алгоритмів та ознак
- **Підхід на основі машинного навчання**: автоматичне вивчення ознак та моделей з даних
- **Глибоке навчання**: використання глибоких нейронних мереж для автоматичного вилучення ієрар

Сучасні системи комп'ютерного зору часто поєднують класичні методи обробки зображень з потужним

Технологічний стек OpenCV

OpenCV (Open Source Computer Vision Library) — це відкрита бібліотека комп'ютерного зору та маш

Основні характеристики OpenCV:

- **Кросплатформеність**: підтримка Windows, Linux, macOS, Android, iOS
- **Багатомовність**: інтерфейси для C++, Python, Java та інших мов
- **Оптимізована продуктивність**: використання багатоядерної обробки та апаратного прискорення
- **Широкий функціонал**: від базової обробки зображень до складних алгоритмів комп'ютерного зо

Структура OpenCV:

- **core**: базові структури даних та функції
- **imgproc**: обробка зображень (фільтрація, перетворення, морфологія)
- **video**: аналіз відео та відстеження об'єктів
- **calib3d**: калібрування камери та 3D-реконструкція
- **features2d**: виявлення та опис особливих точок
- **objdetect**: виявлення об'єктів (обличчя, люди, автомобілі)
- **highgui**: інтерфейс користувача та введення/виведення
- **ml**: алгоритми машинного навчання

Інтеграція з іншими бібліотеками:

- **NumPy**: ефективна робота з масивами даних
- **TensorFlow/PyTorch**: інтеграція з фреймворками глибокого навчання
- **CUDA**: прискорення на GPU

Задачі комп'ютерного зору

Комп'ютерний зір вирішує широкий спектр задач, від базових до складних:

Базові задачі:
- **Обробка та покращення зображень**: фільтрація шуму, корекція кольору, підвищення контрасту
- **Виявлення країв та контурів**: знаходження меж об'єктів
- **Сегментація зображень**: розділення зображення на смислові області

Середній рівень складності:
- **Виявлення об'єктів**: знаходження конкретних об'єктів на зображенні
- **Відстеження об'єктів**: слідкування за об'єктами у відеопотоці
- **Розпізнавання облич**: ідентифікація та верифікація осіб

Складні задачі:
- **Розуміння сцени**: інтерпретація вмісту зображення в цілому
- **3D-реконструкція**: відновлення тривимірної структури з 2D-зображень
- **Семантична сегментація**: класифікація кожного пікселя зображення
- **Генерація зображень**: створення нових зображень за заданими параметрами

Представлення зображень та кольорів

Представлення пікселя

Піксель (від англ. "picture element") — це найменший елемент цифрового зображення. Зображення г

Типи зображень за глибиною кольору:
- **Бінарні зображення**: 1 біт на піксель (чорний або білий)
- **Напівтонові зображення**: 8 біт на піксель (256 рівнів сірого)
- **Кольорові зображення**: зазвичай 24 біти на піксель (8 біт на кожен канал RGB)

Простір кольорів

Простір кольорів — це спосіб організації та представлення кольорів. Різні простори кольорів вик

RGB (Red, Green, Blue)
- Адитивна модель, де кольори утворюються додаванням червоного, зеленого та синього
- Використовується в дисплеях та цифрових камерах
- Кожен піксель представлений трьома значеннями (R, G, B)

HSV (Hue, Saturation, Value)
- Відтінок (Hue): тип кольору (0-360°)
- Насиченість (Saturation): інтенсивність кольору (0-100%)
- Значення (Value): яскравість кольору (0-100%)
- Більш інтуїтивна для людського сприйняття
- Корисна для сегментації за кольором

Grayscale (Відтінки сірого)
- Одноканальне зображення, де кожен піксель має значення від 0 (чорний) до 255 (білий)
- Часто використовується як проміжний крок у багатьох алгоритмах

Інші простори кольорів
- **LAB**: розділяє яскравість (L) та кольорові компоненти (a, b)
- **YCrCb**: використовується у відеокодеках
- **CMYK**: субтрактивна модель для друку

Перетворення між просторами кольорів є важливою операцією в комп'ютерному зорі, оскільки різні

Базові операції з зображеннями

Точкові операції
- **Зміна яскравості**: додавання константи до всіх пікселів
- **Зміна контрасту**: множення пікселів на коефіцієнт
- **Гамма-корекція**: нелінійне перетворення яскравості
- **Порогова обробка**: перетворення в бінарне зображення

Геометричні перетворення
- **Масштабування**: зміна розміру зображення
- **Обертання**: поворот зображення на заданий кут
- **Зсув**: переміщення зображення
- **Відображення**: дзеркальне відображення

Арифметичні операції
- **Додавання зображень**: поєднання двох зображень
- **Віднімання зображень**: виявлення різниці між зображеннями
- **Множення зображень**: маскування

- **Логічні операції**: AND, OR, XOR для бінарних зображень

Гістограмні операції
- **Обчислення гістограми**: розподіл яскравості пікселів
- **Вирівнювання гістограми**: покращення контрасту
- **Розтягнення гістограми**: нормалізація діапазону яскравості

Частотна обробка зображень

Ряд Фур'є та перетворення Фур'є

Перетворення Фур'є дозволяє представити зображення як суму синусоїдальних компонент різних част

Дискретне перетворення Фур'є (DFT)

Для двовимірного зображення розміром M×N, DFT визначається як:

$$
F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \cdot e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}
$$

де:
- $f(x,y)$ — значення пікселя в просторовій області
- $F(u,v)$ — значення в частотній області
- u та v — частотні координати

Швидке перетворення Фур'є (FFT)

FFT — це ефективний алгоритм обчислення DFT, який значно зменшує обчислювальну складність:
- Складність DFT: O(N^2)
- Складність FFT: O(N log N)

Застосування частотної обробки

- **Аналіз частотних компонент**: розуміння структури зображення
- **Фільтрація в частотній області**: видалення шуму, виділення країв
- **Стиснення зображень**: видалення високочастотних компонент
- **Відновлення зображень**: корекція розмиття та інших спотворень

Фільтрація зображень

Фільтрація — це процес модифікації або покращення зображення шляхом застосування різних операто

Просторова фільтрація

Низькочастотна фільтрація (Low-pass)
Згладжує зображення, видаляючи високочастотні компоненти (шум, деталі):

Гаусівський фільтр (Gaussian filter)
- Використовує ядро, засноване на функції Гауса
- Надає більшу вагу центральному пікселю та меншу — віддаленим
- Ефективно видаляє гаусівський шум
- Ядро Гаусівського фільтра 3×3:

$$
\frac{1}{16} \begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
```
```

```
1 & 2 & 1
\end{bmatrix}
\]

##### Фільтр середнього (Box filter)
- Замінює кожен піксель середнім значенням його околиці
- Простий в реалізації, але може розмивати краї
- Ядро фільтра середнього 3×3:
\[
\frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]

#### Високочастотна фільтрація (High-pass)
Підкреслює різкі зміни в зображенні (краї, деталі):

##### Фільтр Собеля (Sobel filter/operator)
- Обчислює градієнт яскравості зображення
- Підкреслює краї в горизонтальному та вертикальному напрямках
- Ядра Собеля:
\[
G_x = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}, \quad
G_y = \begin{bmatrix}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
\]

##### STD фільтр (Standard Deviation filter)
- Обчислює стандартне відхилення в околиці кожного пікселя
- Виділяє області з високою варіацією (текстури, краї)
- Корисний для аналізу текстур

### Детектор країв Кенні (Canny edge detector)

Алгоритм Кенні — це багатоетапний алгоритм виявлення країв, який забезпечує хороші результати:

1. **Згладжування**: застосування Гаусівського фільтра для зменшення шуму
2. **Обчислення градієнтів**: використання фільтрів Собеля для знаходження величини та напрямку
3. **Придушення немаксимумів**: залишаються тільки локальні максимуми градієнта
4. **Подвійна порогова обробка**: класифікація пікселів як "сильні" та "слабкі" краї
5. **Трасування країв**: включення слабких країв, пов'язаних з сильними

Переваги детектора Кенні:
- Хороше виявлення: низька ймовірність пропуску реальних країв
- Хороша локалізація: виявлені краї близькі до реальних
- Мінімальна відповідь: кожен край виявляється тільки один раз

## Порогова обробка (Thresholding)
```

Порогова обробка — це метод сегментації зображення, який перетворює напівтонове зображення в бі

Глобальна порогова обробка

Використовує єдине порогове значення для всього зображення:

$$
g(x,y) =
\begin{cases}
1, & \text{якщо } f(x,y) > T \\
0, & \text{інакше}
\end{cases}
$$

де:
- $f(x,y)$ — вхідне зображення
- $g(x,y)$ — бінарне зображення
- T — порогове значення

Метод Оцу (Otsu method)

Автоматично визначає оптимальне порогове значення, максимізуючи міжкласову дисперсію:

1. Обчислення гістограми зображення
2. Для кожного можливого порогового значення:
   - Розділення пікселів на два класи
   - Обчислення дисперсії між класами
3. Вибір порогового значення з максимальною міжкласовою дисперсією

Метод Оцу ефективний для зображень з бімодальною гістограмою (два піки).

Адаптивна порогова обробка

Використовує різні порогові значення для різних областей зображення:

1. **Адаптивний метод середнього**: поріг визначається як середнє значення в околиці пікселя
2. **Адаптивний метод Гауса**: поріг визначається як зважене середнє (з гаусівськими вагами) в

Адаптивна порогова обробка ефективна для зображень з нерівномірним освітленням.

Морфологічні перетворення

Морфологічні операції — це набір операцій обробки зображень, заснованих на формі. Вони зазвичай

Базові морфологічні операції

Ерозія (Erosion)
- Зменшує розмір об'єктів
- Видаляє малі об'єкти
- Розділяє об'єкти, з'єднані тонкими лініями
- Математично: $A \ominus B = \{z \mid B_z \subseteq A\}$

Дилатація (Dilation)
- Збільшує розмір об'єктів
- Заповнює малі отвори
- З'єднує близькі об'єкти

- Математично: $A \oplus B = \{z | (B_z \cap A) \neq \emptyset\}$

Складні морфологічні операції

Відкриття (Opening)
- Ерозія, за якою слідує дилатація
- Видаляє малі об'єкти та шум
- Згладжує контури
- Математично: $A \circ B = (A \ominus B) \oplus B$

Закриття (Closing)
- Дилатація, за якою слідує ерозія
- Заповнює малі отвори та розриви
- Згладжує контури
- Математично: $A \bullet B = (A \oplus B) \ominus B$

Морфологічний градієнт
- Різниця між дилатацією та ерозією
- Виділяє контури об'єктів
- Математично: $G(A) = (A \oplus B) - (A \ominus B)$

Розширення для напівтонових зображень

Морфологічні операції можуть бути розширені для роботи з напівтоновими зображеннями:

- **Напівтонова ерозія**: мінімум в околиці
- **Напівтонова дилатація**: максимум в околиці
- **Напівтонове відкриття та закриття**: комбінації вищезазначених операцій

Властивості базових операторів

- **Ідемпотентність**: повторне застосування не змінює результат (для відкриття та закриття)
- **Екстенсивність**: дилатація збільшує, ерозія зменшує об'єкти
- **Антиекстенсивність**: ерозія $\subseteq$ оригінал $\subseteq$ дилатація
- **Інваріантність до зсуву**: результат не залежить від положення об'єктів

Зіставлення шаблонів та геометричні перетворення

Зіставлення шаблонів (Template matching)

Зіставлення шаблонів — це метод пошуку частин зображення, які відповідають заданому шаблону.

Алгоритм зіставлення шаблонів:
1. Ковзання шаблону по зображенню
2. Обчислення міри схожості для кожної позиції
3. Знаходження позицій з найкращою відповідністю

Методи обчислення схожості:
- **Квадрат різниці (SSD)**: $\sum_{x,y} [T(x,y) - I(x+u,y+v)]^2$
- **Нормалізована кореляція (NCC)**: $\frac{\sum_{x,y} T(x,y) \cdot I(x+u,y+v)}{\sqrt{\sum_{x

Афінні перетворення (Affine transformation)

Афінні перетворення зберігають паралельність ліній та співвідношення відстаней вздовж ліній.

Матриця афінного перетворення 2×3:
$$

```
M = \begin{bmatrix}
a_{00} & a_{01} & b_0 \\
a_{10} & a_{11} & b_1
\end{bmatrix}
\]

Афінне перетворення точки (x, y):
\[
\begin{bmatrix}
x' \\
y'
\end{bmatrix} =
\begin{bmatrix}
a_{00} & a_{01} \\
a_{10} & a_{11}
\end{bmatrix}
\begin{bmatrix}
x \\
y
\end{bmatrix} +
\begin{bmatrix}
b_0 \\
b_1
\end{bmatrix}
\]

#### Типи афінних перетворень:
- **Масштабування**: зміна розміру
- **Обертання**: поворот навколо точки
- **Зсув**: переміщення
- **Відображення**: дзеркальне відображення
- **Зсув (shear)**: деформація

### Перспективне перетворення (Homography)

Перспективне перетворення моделює проекцію 3D-сцени на 2D-площину. Воно може змінювати паралель

Матриця перспективного перетворення 3×3:
\[
H = \begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\]

Перспективне перетворення точки (x, y):
\[
\begin{bmatrix}
x' \\
y' \\
w'
\end{bmatrix} =
\begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
```

```
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Після перетворення координати нормалізуються:
\[
(x'', y'') = (\frac{x'}{w'}, \frac{y'}{w'})
\]

#### Застосування гомографії:
- **Вирівнювання зображень**: корекція перспективи
- **Панорамне зшивання**: об'єднання кількох зображень
- **Доповнена реальність**: накладання віртуальних об'єктів на реальну сцену

## Сегментація зображень

Сегментація — це процес розділення зображення на смислові області або об'єкти.

### K-means для сегментації зображень

K-means — це алгоритм кластеризації, який розділяє пікселі на K груп:

1. Ініціалізація K центроїдів (випадково або евристично)
2. Призначення кожного пікселя до найближчого центроїда
3. Оновлення положення центроїдів як середнього значення пікселів у кластері
4. Повторення кроків 2-3 до збіжності

#### Застосування K-means у різних просторах кольорів:

- **RGB**: сегментація за кольором, але чутлива до освітлення
- **HSV**: краща сегментація за кольором, менш чутлива до освітлення
- **LAB**: перцептивно рівномірний простір, хороший для сегментації

#### Переваги та недоліки K-means:
- **Переваги**: простота, ефективність, інтуїтивність
- **Недоліки**: необхідність заздалегідь задавати K, чутливість до ініціалізації, не враховує г

### Інші методи сегментації:
- **Watershed**: сегментація на основі водорозділу
- **GrabCut**: інтерактивна сегментація переднього плану
- **Mean Shift**: кластеризація на основі оцінки щільності
- **Superpixels**: групування пікселів у перцептивно значущі атоми

## Виявлення особливих точок та дескриптори

### Детектор кутів Харріса (Harris corner detection)

Детектор Харріса виявляє кути на зображенні, аналізуючи зміни градієнта в різних напрямках:

1. Обчислення градієнтів зображення (Ix, Iy)
2. Обчислення матриці структури M для кожного пікселя:
   \[
   M = \begin{bmatrix}
```

```
   \sum I_x^2 & \sum I_x I_y \\
   \sum I_x I_y & \sum I_y^2
   \end{bmatrix}
   \]
3. Обчислення міри відгуку кута:
   \[
   R = \det(M) - k \cdot \text{trace}(M)^2
   \]
4. Застосування порогової обробки та придушення немаксимумів

Кути мають високі значення R, оскільки градієнт змінюється в обох напрямках.

### SIFT (Scale-Invariant Feature Transform)

SIFT — це алгоритм для виявлення та опису локальних особливостей зображення, інваріантний до ма

#### Етапи SIFT:

1. **Виявлення екстремумів у масштабному просторі**:
   - Побудова піраміди зображень з різними масштабами (октави)
   - Застосування різниці гаусіанів (DoG) для виявлення потенційних особливих точок
   - Пошук локальних екстремумів у просторі та масштабі

2. **Локалізація ключових точок**:
   - Уточнення положення та масштабу за допомогою інтерполяції
   - Відкидання точок з низьким контрастом
   - Відкидання точок на краях (використовуючи матрицю Г
```