МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Дипломна магістерська робота

на тему

Алгоритмічне та програмне забезпечення для виявлення, розпізнавання та відстеження рухомих об'єктів

Виконав: студент групи МгЗІТ-21

спеціальності 122 Комп'ютерні науки

Дмитро ПРУДНІК

Керівник к.т.н., *доц. Вікторія РЕЗАНОВА*

Рецензент *проф. Сергій КРАСНИТСЬКИЙ*

Київ 2022

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КНТ

_____проф. Володимир ЩЕРБАНЬ

"_____" _____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Прудніку Дмитру Олександровичу

1. Тема роботи *Алгоритмічне та програмне забезпечення для розпізнавання, виявлення та відстеження рухомих об'єктів,* науковий керівник роботи *Резанова В. Г., к.т.н., доцент,*
затверджені наказом вищого навчального закладу від "28" вересня 2022 року № 180-уч

2. Строк подання студентом роботи 07.11.2022 р.

3. Вихідні дані до роботи Розробка кафедри комп'ютерних наук. *Алгоритмічне та програмне забеспечення для розпізнавання, виявлення та відстеження рухомих об'єктів*

4. Зміст дипломної роботи: РОЗДІЛ 1 математичне забезпечення; РОЗДІЛ 2 алгоритмічне забезпечення; РОЗДІЛ 3 програмне забезпечення.

5. Перелік графічного матеріалу: презентація на __ слайдах.

6. Консультанти розділів дипломної магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ1	*Вікторія РЕЗАНОВА. доц. каф КНТ*		
Розділ 2	*Вікторія РЕЗАНОВА доц. каф КНТ*		
Розділ 3	*Вікторія РЕЗАНОВА. доц. каф КНТ*		

7. Дата видачі завдання 09.2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної магістерської роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	09.10.2021	
2	Розділ 1 Математичне забезпечення	12.10.2021	
3	Розділ 2 алгоритмічне забезпечення	20.10.2021	
4	Розділ 3 програмне забезпечення	25.10.2021	
5	Висновки	25.10.2021	
6	Оформлення дипломної магістерської роботи (чистовий варіант)	30.10.2021	
7	Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту)	07.11.2021	
8	Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту)	07.12.2021	
9	Подання дипломної магістерської роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану (за 10 днів до захисту)	10.12.2021	
10	Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту)	10.12.2021	

Студент _____ Дмитро ПРУДНІК

Науковий керівник роботи _____ Вікторія РЕЗАНОВА

Директор НМЦУПФ _____ Олена ГРИГОРЕВСЬКА

АНОТАЦІЯ

Пруднік Дмитро. Тема Алгоритмічне та програмне забезпечення для розпізнавання, виявлення та відстеження рухомих об'єктів.

Дипломна магістерська робота за спеціальністю 122 - «Комп’ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2022 рік.

Мета роботи – Алгоритмічне та програмне забезпечення для розпізнавання, виявлення та відстеження рухомих об'єктів. Для реалізації мети поставлені наступні задачі:

- Математичне забезпечення.
- Алгоритмічне забезпечення.
- Програмне забезпечення.

Загальний обсяг роботи: 88 сторінок, 50 рисунки, 1 посилань, 3 додатка.

Ключові слова: Python,OPENCV, TKinter

Технології: Python, OPENCV, TKinter

ANNOTATION

Prudnik Dmytro. Algorithmic and software for recognition, detection and tracking of moving objects.

Diploma master's degree work on speciality 122 «Computer sciences». - The Kiev national university of technologies and design, Kiev, 2022 year. Purpose - Algorithmic and software for recognition, detection and tracking of moving objects. To achieve this goal the following tasks are set:

Review of object recognition methods, computer vision and machine learning.

- Mathematical security.
- Algorithmic security.
- Software security.

Total volume of work: 88 pages, 50 figures, 1 links.

Keywords: Python, OPENCV, Tkinter

Technologies: Python, OPENCV, Tkinter

ЗМІСТ

ВСТУП

Сьогодні одними з найважливіших областей досліджень та розробок сучасної прикладної математики та кібернетики є – комп’ютерний зір, розпізнавання образів, машинне навчання, а також прогнозування та аналіз даних. Прискорення темпів розвитку технологій інформаційного суспільства, розвиток концепцій “розумний дім” та “розумне місто”, розвиток інтернету та системи штучного інтелекту визначають особливе місце для цих областей в сучасному світі. У багатьох прикладних задач програмування використовуються методи збору даних, кластеризації та класифікації, статистичного висновку. У повсякденному житті активно впроваджуються технології розпізнавання образів. Раніше завдання розпізнавання, що вважалися складними, сьогодні вирішуються за допомогою звичайних мобільних пристроїв та смартфонів.

Поширені методи та підходи до вирішення завдань детектування, розпізнавання та класифікації є:

- Порівняння зі зразком – класифікація за найближчим середнім, на відстані до найближчого сусіда. Також у групу порівняння із зразком можна віднести структурні методи розпізнавання.
- Зіставлення з шаблоном – метод розпізнавання, у якому використовуються невелике зображення або шаблон для пошуку збігаються областей у збільшеному зображенні.
- Нейронні мережі – клас методів глибокого машинного навчання, який використовується для автоматичного вивчення властивостей об’єкта та його подальшої ідентифікації. Відмінною особливістю цих методів є інших є здатність вчитися.

Мета цього проекту – Створити програмне забезпечення для розпізнавання, виявлення та відстеження рухомих об'єктів на графічних зображеннях (фото-, відеодані).

Для досягнення поставленої мети в роботі були поставлені та вирішенні такі завдання:

1. Розглянути існуючі алгоритми для детектування та класифікації об’єктів.
2. Вибрати мови та середовища програмування для реалізації обраних алгоритмів.
3. Програмно реалізувати сервіс для розпізнавання об’єктів.
4. Навчити класифікатор з використанням розроблених моделей та провести його тестування.

Розділ 1. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

1.1 Введення в комп'ютерний зір

Комп'ютерний зір (computervision, машинний зір, технічний зір) – це технологія, яка в залежності від поставленого завдання, може знаходити, відстежувати, класифікувати та ідентифікувати об'єкти, витягуючи та аналізуючи отриману інформацію з зображень або відео. Цей напрямок виник у рамках штучного інтелекту. Основним завданням є розпізнавання образів, тому для повної і правильної інтерпретації того, що зображено, потрібно мати необхідну інформацію як масив пікселів, витягнутих із зображення. Зображення складається з набору пікселів. Для комп'ютера немає тоншої деталізації окрім пікселя.

Під комп'ютерним зором також розуміється автоматичне вилучення інформації з зображень. У ролі інформації може виступати 3D-моделі, положення камери, виявлення та розпізнавання об'єктів, групування зображень та пошук зображень за змістом. Застосування комп'ютерного зору набуває досить великого значимість у різних сферах нашого життя. За допомогою цієї технології реалізовано розумну систему відеоспостереження, яка обробляє вхідні відео та, відповідно до навченого алгоритму, приймає необхідні рішення.

Для вирішення завдань детектування та класифікації застосовуються різні підходи: статистичні, спеціально розроблені теорії ключових точок, застосування алгоритмами класифікації зображень за змісту, і навіть затребуваний підхід – машинне навчання.

Найбільш поширені завдання комп'ютерного зору:

- Розпізнавання - одне з базових і першорядних завдань у обробці зображень, комп'ютерному та машинному зору. Воно допомагає класифікувати та ідентифікувати об'єкти, що характеризуються певним набором властивостей та ознак.

- Відновлення зображень – це видалення шуму з використанням різних методів, наприклад, розмиття за допомогою фільтрів на основі машинного навчання (шум датчика, розмитість рухомого об'єкта і т. д.).
- Аналіз руху - завдання використовує комп'ютерний зір для оцінки швидкості руху об'єктів у відео. Також застосовується для оцінки рухів, у яких послідовність відеоданих обробляється для знаходження швидкості кожної точки зображення чи 3D сцени.
- Відновлення чи реконструкція сцени – допомагає відтворити тривимірну модель зображення або сцени, що вводиться за допомогою зображень чи відео. Найчастіше моделлю служить набір точок. тривимірного простору.
- Обробка та аналіз зображення - завдання зосереджена на перетворення одного 2D-зображення в інше. Реалізується за допомогою піксельних операцій, таких як підвищення контрастності або поворот зображення.
- Високорівнева обробка – невеликий набір даних. Вона використовує різні методи для вилучення інформації з сигналів в цілому, наприклад, набір точок або ділянка зображення, в якому імовірно знаходиться певний об'єкт, що цікавить частина даних.

1.2 Принцип роботи комп'ютерного зору

Комп'ютери інтерпретують зображення як послідовністьпікселів, кожен з яких має власний набір значень кольору. Пікселі є необробленими будівельними блоками. зображення. Кожне зображення складається з набору пікселів. піксель вважається «кольором» або «яскравістю» світла, що з'являється на зображенні. Якщо ми розглядаємо зображення як сітку, то кожен квадрат містить один піксель.

Рис. 1.2.1 Приклад зображення

Зображення на рисунку 1 має роздільну здатність 1000 × 750 пікселів, де 1000 – це ширина, а 750 – висота. Ми можемо уявити зображення у вигляді матриці. У цьому випадку наша матриця має 1000 стовпців (ширина) та 750 рядків (висота) та містить 1000 × 750 = 750 000 пікселів, які представлені двома способами: а) відтінки сірого (один канал); б) колір.

У зображеннях із градаціями сірого кольору кожен піксель є скалярним значенням від 0 до 255, де нуль відповідає "чорному" кольору, а 255 - "білому". Значення між 0 та 255 мають різні відтінки сірого, де значення ближче до 0 темніше, а значення ближче 13 до 255 світліше. Градієнтне зображення у градаціях сірого (рисунок 2) демонструє темніші пікселі з лівого боку, а світліші правою. З малюнка 2 можна зрозуміти те, як значення в градаціях сірого перетворюються на двовимірний масив цілих чисел:

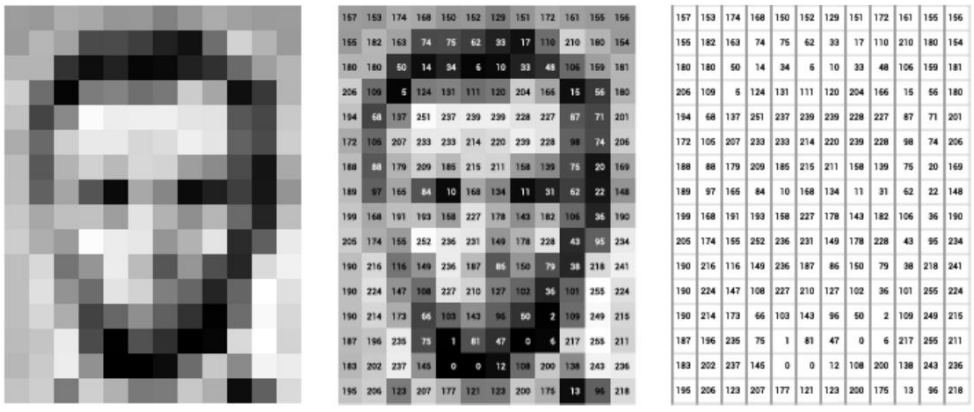


Рис. 1.2.2 Зображення та їх матричні представлення

Ряди чисел праворуч (крайній правий малюнок) – комп'ютерна вистава введеного зображення. У прикладі зображення має 12 стовпців та 16 рядків, що означає 192 вхідних значень цього зображення.

Ще необхідно описати як комп'ютер перетворює для себе кольорові зображенні у вигляді матриць. Пікселі у кольоровому просторі RGB більше не є скалярними значеннями, як було у зображеннях у градаціях сірого, на одному каналі. Натомість пікселі представлені списком з трьох значень: одне значення для компонента червоного (Red), друге для зеленого (Green) та третє для синього (Blue). Щоб визначити колір у колірної моделі RGB, все, що нам потрібно зробити, це підрахувати кількість червоного, зеленого та синього кольору, що містяться в одному пікселі. Кожен канал Red, Green та Blue може мати певні значення в діапазоні [0, 255], всього 256 відтінків, де 0 означає відсутність уявлення, а 255 - це повне уявлення. Враховуючи що значення пікселя має бути лише в діапазоні [0, 255], ми зазвичай 14 використовуємо 8 бітові цілі числа без знака для представлення яскравості.

Розглянемо приклад створення різних кольорів зображення на малюнку 1.2.3.

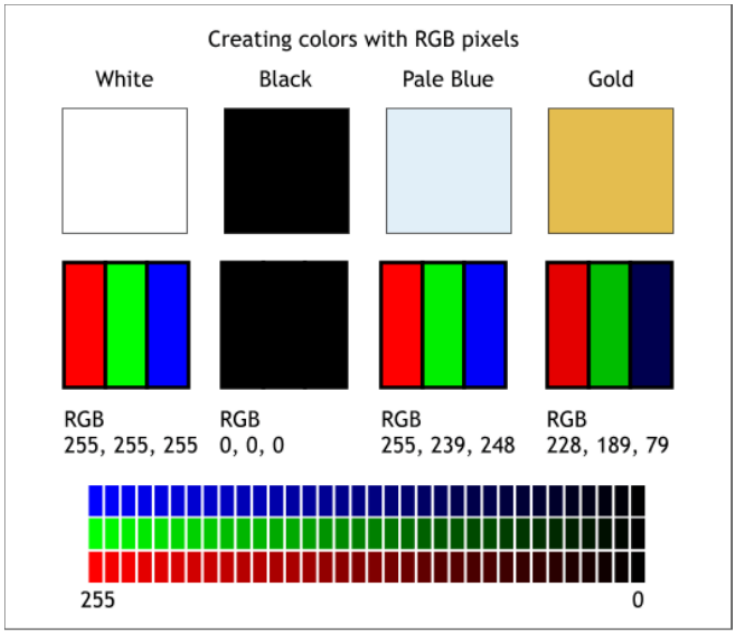


Рис. 1.2.3 Створення кольорів у RGB пікселі

З цього малюнка 1.2.3 можна зрозуміти, що кожен піксель складається з трьохрізних кольорів, при зміні їх відтінків можна отримати абсолютно різні кольори.

Для вирішення завдань детектування, розпізнавання та класифікації об'єктів на графічному зображенні необхідно попередньо обробляти зображення (наприклад, виконати операцію середнєвіднімання чи масштабування). Оскільки типи даних, які використовуються бібліотеками (наприклад, OpenCV), що завантажують зображення з диска, то їх необхідно перетворювати, перш ніж безпосередньо застосовувати алгоритминавчання до зображень. Враховуючи наші три значення Red, Green та Blue, ми можемо поєднати їх у кортеж RGB (червоний, зелений, синій). Цей кортеж представляє цей колір у колірному просторі RGB.

Ми можемо перетворити зображення RGB як три незалежних матриць, шириною W і висотою H, по одній для кожного з компонентів RGB, як показано на рисунку 1.2.3. Ми можемо об'єднати ці три матриці для отримання багатовимірного масиву з формою W × H × D, де D – 15 глибина чи кількість каналів. Для колірного простору RGBглибина D = 3.

Рис 1.2.4 Вихідне зображення та його RGB канали

1.3 Огляд методів комп'ютерного зору для розпізнавання задач об'єктів.

Для комп'ютерів завдання інтерпретація вмісту зображення менш тривіальна, ніж просто завдання відображення зображення. Все, що бачить наш комп'ютер - це велика матриця чисел. Щоб зрозуміти зміст зображення, ми повинні застосувати класифікацію зображень, що є завданням використання алгоритмів комп'ютерного зору та машинного навчання. У цій роботі були застосовані різні методи, крім глибокого навчання, що покращують комп'ютерний зір. Тим не менш, вони добре працюють для більш простих завдань, але оскільки дані стають величезними, а завдання стає складною, вони не замінюють згорткові нейронні мережі. Використання алгоритмів комп'ютерного зору та машинного навчання. У цій роботі були застосовані різні методи, крім глибокого навчання, що покращують комп'ютерний зір. Тим не менш, вони добре працюють для більш простих завдань, але оскільки дані стають величезними, а завдання стає складною, вони не замінюють згорткові нейронні мережі.

1.3 Перетворення рівня яскравості

Вважаючи за допомогою NumPy зображення в масив, ми можемо застосувати щодо нього різні математичні операції. Простий приклад перетворення рівня яскравості напівтонового зображення. Візьмемо довільну функцію f, що відображає інтервал 0...255 (або, якщо завгодно, 0…1) у собі, т. е. область значень збігається з областю визначення.

Іншим прикладом перетворення яскравості є вирівнюваннягістограми. Ця операція змінює гістограму яскравості, так щоб результуюча гістограма містила всі можливі значення яскравості та при цьому приблизно в однаковій кількості. Вона часто застосовується для нормування яскравості перед подальшою обробкою, а також для підвищення контрастності. В даному випадку для перетворення використовується функція розподілу (cumulative distribution function, CDF) значень пікселів у зображенні.

1.4 Детектор кутів Харріса

Алгоритм виявлення кутів Харріса (детектором кутів Харріса-Стівенса) один із найпростіших детекторів кутів об'єктів. Ідея полягає в тому, щоб знайти особливі точки, в околиці яких є межі у кількох напрямах, і є кутові точки.

Визначимо позитивно-напіввизначену симетричну матрицю.

$$\mathrm{M}_I = \mathrm{M}_I(x),$$

де x – точка всередині зображення:

$$\mathrm{M}_I = \nabla I \nabla I^T = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}.$$

Тут ∇I – вектор градієнта зображення, що містить похідні I x та I y (Визначення похідних були дані вище). За побудовою, M I має ранг 1, а її власні значення дорівнюють

$$\lambda_1 = |\nabla \mathrm{I}|^2, \lambda_2 = 0.$$

Таким чином, у нас є по одній матриці для кожної точки зображення.

Залежно від значень ∇I в області є три випадки:

- Якщо λ 1 і λ 2 – великі позитивні числа, то точці x є кут.
- Якщо λ 1 велике, а λ 2 ≈ 0, то існує межа і при усередненні M1 по області власні значення змінюються не сильно.
- Якщо λ 1 ≈ λ 2 ≈ 0, то в точці хнемає жодних особливостей.

Розглянемо спосіб з прикладу зображення головного корпусу ТПУ:

а) б)

Рис 1.4.1 – а) вихідне зображення; б) зображення з виявленимикутами за алгоритмом Харріса

Фільтрування контурів

Контури дуже корисні, коли ми хочемо перейти від роботи з зображенням для роботи з об'єктами на цьому зображенні. Коли об'єкт досить складний, але добре виділяється, то найчастіше єдиним методом роботи з ним є виділення його контурів.

Оператор Кенні є популярним алгоритмом виявлення кордонів та найчастіше використовується для виділення контуру об'єктів.

Алгоритм Кенні для виявлення меж об'єктів складається з п'яти етапів:

1. Пригнічення шуму;
2. Розрахунок градієнта;
3. Пригнічення країв зображення;
4. Подвійний поріг;
5. Відстеження країв по гістерезі.

Ще одна важлива річ, яку варто згадати, що алгоритм заснований на зображеннях у градаціях сірого. Отже, попередньою умовою є перетворення зображення у відтінки сірого кольору перед виконанням вищезгаданих кроків.

Коротко дамо визначення етапів алгоритму Кенні.

Придушення шуму – є один із способів позбутися шумів на зображенні. Для того щоб згладити шум застосовують розмиття по Гаусса.

Для цього застосовується метод згортки зображень з гаусовим ядром (наприклад, 3x3, 5x5, 7x7 тощо пікселів). Розмір ядра залежить від очікуваного ефекту розмиття. Здебільшого найменше ядро – це менш помітна пляма. При обробці зображення, ядро (згортка матриці) є невеликою матрицею. Воно використовується для розмиття, підвищення різкості, тиснення, виявлення країв та багато іншого.

Розрахунок градієнта - визначає інтенсивність і напрямок краю шляхом обчислення градієнта зображення з використанням операторів виявлення краю. *Градієнт* – це векторна величина, що показує напрямок якнайшвидшого зростання двовимірна функція яскравості зображення.

Краї відповідають зміні інтенсивності пікселів. Щоб виявити її, найпростіше застосувати фільтри, які виділяють це зміна інтенсивності в обох напрямках: горизонтальному (x) та вертикальному (y).

Придушення країв зображення – в ідеалі кінцеве зображення має мати тонкі краї. Таким чином, ми повинні виконати не-максимальне придушення, щоб виявити краї. Для цього можна застосувати наступний алгоритм. Обійти через усі крапки на матриці інтенсивності градієнта і знайти пікселі з максимальним значенням у напрямках ребер рис. 1.4.2

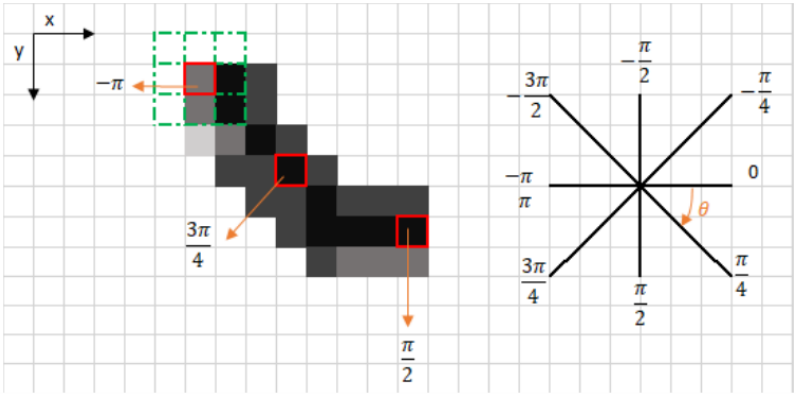


Рис. 1.4.2 а) - частина краю об'єкта у вигляді пікселів

Червоний прямокутник у верхньому лівому кутку (рисунок 1.4.1 а), представляє піксель інтенсивності оброблюваної градієнтної матриці інтенсивність. Відповідний напрямок країв позначено помаранчевим стрілкою з кутом -π радіан.

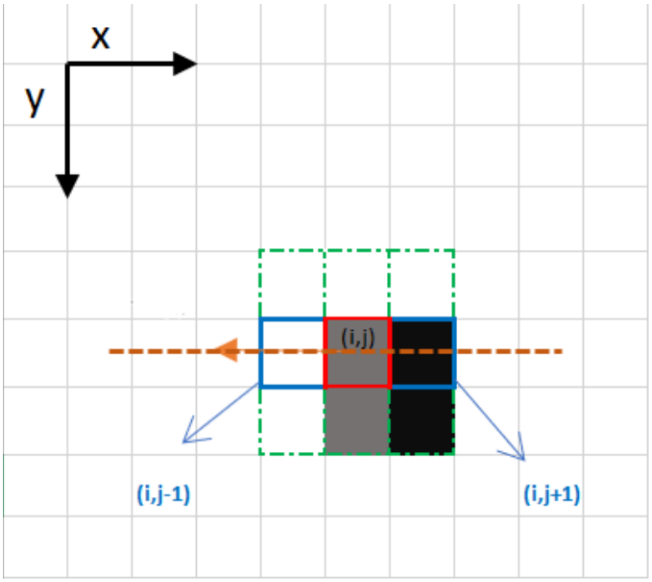


Рис 1.4.3 зображення з неповним придушенням країв

Напрямок краю – помаранчева пунктирна лінія (горизонтальна лінія зліва направо, рисунок 6). Мета алгоритму – перевірити, чи є пікселі в тому самому напрямку більш менш інтенсивними, ніж оброблювані.

У наведеному вище прикладі піксель (i, j)обробляється, і пікселі в тому ж напрямку виділяються синім кольором (i, j-1) і (i, j + 1). Якщо один з цих двох пікселів є більш інтенсивним, ніж оброблюваний, то зберігається лише інтенсивніший. Піксель (i, j-1) здається більшим інтенсивний, тому що він білий (значення 255). Отже, значення інтенсивності поточного пікселя (i, j) встановлюється 0. Якщо в напрямі краю відсутні пікселі, що мають більш інтенсивні значення, то значення поточного пікселя зберігається.

Подвійний поріг – мета застосування цієї операції спрямована на виявлення трьох видів пікселів та за допомогою них виявляти контури:

- Сильні пікселі – пікселі з високою інтенсивністю (яскравістю).
- Слабкі пікселі – це пікселі, які мають достатнього значення інтенсивності. Ці пікселі не можна вважати сильними, але їх значення інтенсивності не є маленькими, щоб їх вважати, як пікселі, що не мають відношення до країв.
- Інші пікселі вважаються такими, що не мають відношення до країв.

Відстеження країв – результати порогового значення гістерезис, складається з перетворення слабких пікселів у сильні, якщо хоча б один із пікселів навколо краю, що обробляється, є сильним.

Подивимося отримані нами результати після застосування алгоритму виявлення країв різних об'єктів (малюнки 1.4.3-1.4.4).

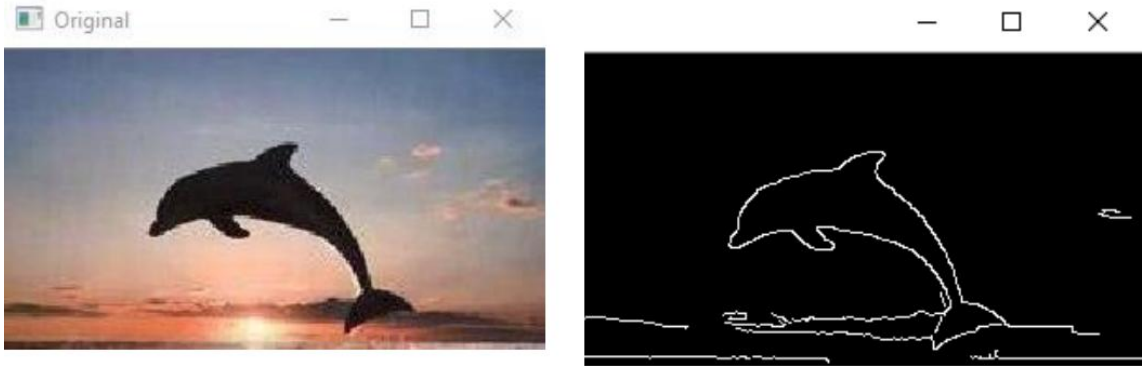


Рис 1.4.4 – а) вихідне кольорове зображення, б) результат виявлення країв об'єкта

В результаті видно, що цей метод може знайти всі краї об’єкта (Дельфін), видаляючи при цьому багато країв обєктів на задньому фоні (хмари). Далі розглянемо складніший об'єкт і застосуємо той самий алгоритм (Рисунок 1.4.5).

Рис 1.4.5 – а) вихідне кольорове зображення, б) результат виявлення країв об’єкта

З прикладів видно, що алгоритм виявлення країв Кенні дає задовільний результат, як простих, так складніших об'єктів.

Штучні нейронні мережі

Штучні нейронні мережі (нейромережі чи просто мережі) – це клас моделей машинного навчання, в основі яких лежать центральний нервової системи ссавців.

Нейронна мережа складається з кількох взаємопов'язаних різних шарів, таких як вхідний шар, щонайменше один прихований шар і вихідний шар (рис 1.4.6). Їх найкраще використовувати при виявленні об'єктів для розпізнавання образів, країв (вертикальні/горизонтальні), форми, кольори та текстури. Приховані шари є шарами згортки. У даному типі нейронної мережі, згорткові шари діє як фільтр, який спочатку отримує вхідні дані, перетворює їх, використовуючи певний алгоритм або функцію, та відправляє його на наступний шар. Основні параметри нейронів є вхідний (синій колір) та вихідний шар (зелений колір).

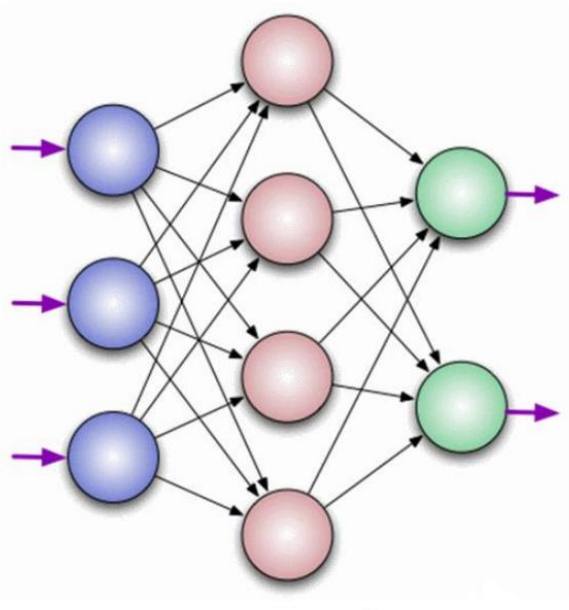


Рис 1.4.6 Модель нейронної мережі

З великою кількістю згорткових шарів, щоразу, коли новий вхід відправляється на наступний згортковий шар, він змінюється по-різному. Наприклад, у згортковому шарі фільтр може ідентифікувати форму / колір у певній області, останній згортковий шар може класифікувати об'єкт.

У загальному випадку згорткова нейронна мережа складається з великої кількості шарів. На останніх етапах зазвичай використовується один або кілька повнозв'язкових шарів.

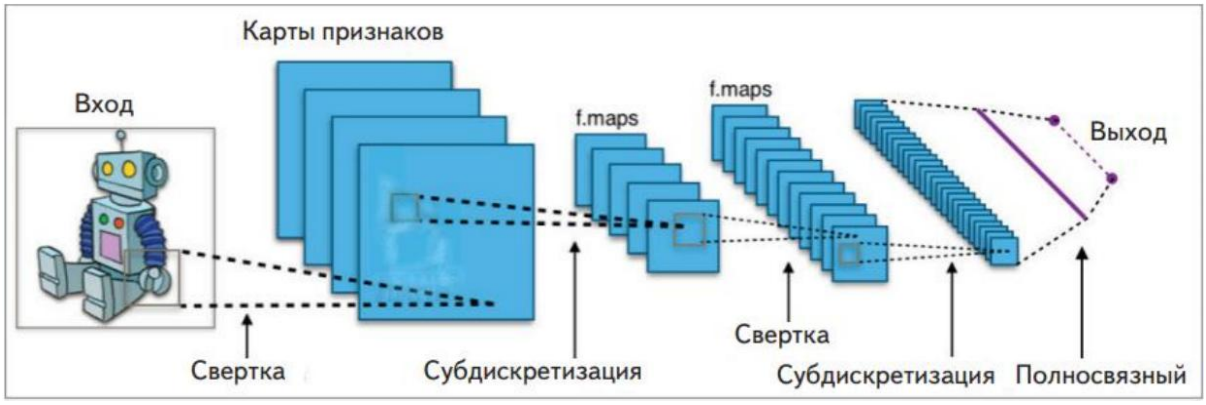


Рис 1.4.7 Топологія згорткової нейронної мережі

Згорткові нейронні мережі забеспечують часткову стійкість до змін масштабу, зсувів, поворотів, зміни ракурсу та інших спотворень. Згорткові нейронні мережі поєднуються три архітектуарні ідеї, повороту зсуву та просторових спотворень.

- Локальні рецепторні поля (запеспечують локальну двовимірну зв’язність нейронів);
- Загальні синоптичні коефіцієнти (забеспечують детектування деяких рис у будь-якому місці зображення та зменшують загальну кількість вагових коефіцієнтів);
- Ієрахічна організація із просторовими підвиборками;

Розпізнавання об'єктів за допомогою нейронної мережі

Для того, щоб навчити нейрнну мережу виявляти об’єкти на будь-якому зображенні, з приблизно однаковою формою та кольорами, слід застосувати різні фільтри (рисунку 1.7.1).

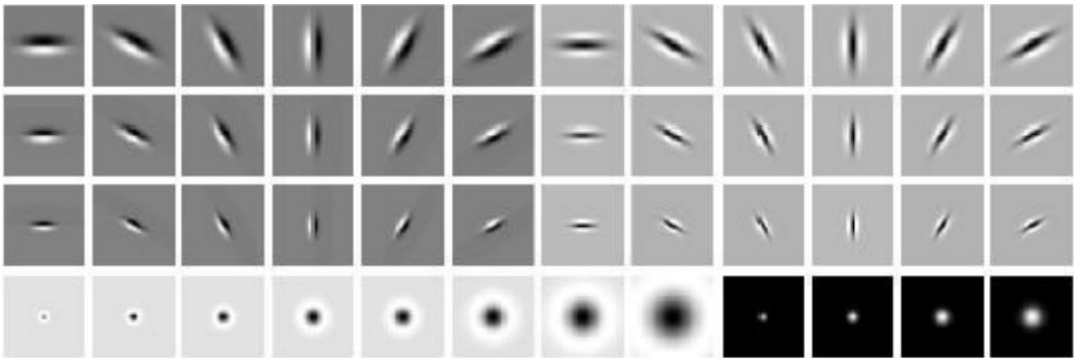


Рис 1.4.8 Фільтри для різноманітних фрагментів зображення

У цьому розділі детально досліджено види сайтів та їхні головні компонети. Також описано концептуальні основи побудови веб-сайту за допомогою фреймворку.

Для того, щоб мережам не доводилося окремо розпізнавати об'єкти в різних частинах зображення, ми розділяємо ваги, що відповідають за розпізнавання, між різними фрагментами вихідного зображення. Розглянемо зображення, у якому необхідно непросто виділити об'єкт, а встановити кількісну точність рішення чотирьох фрагментів (рисунок 1.7.2).

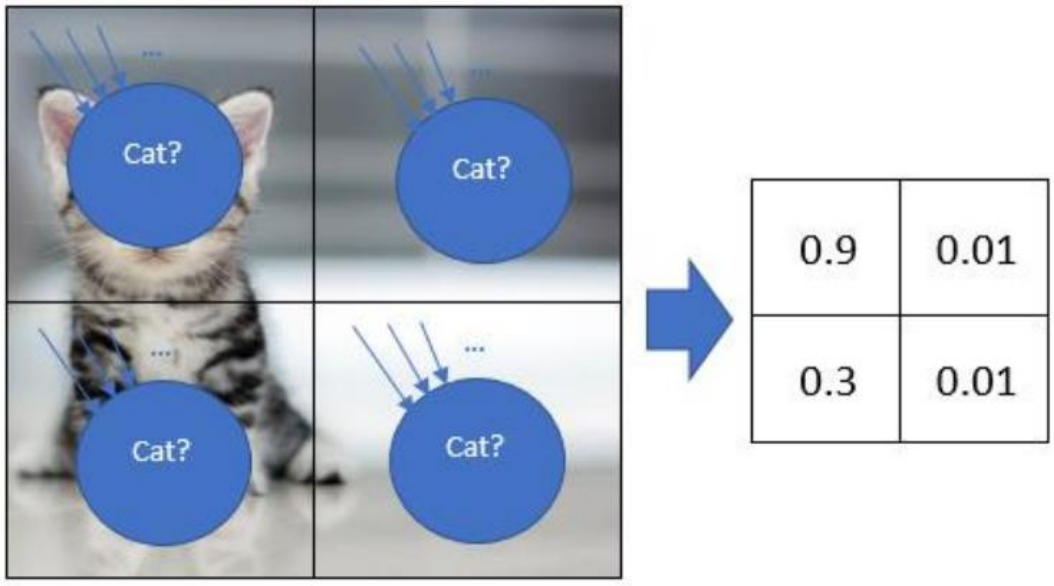


Рис 1.4.9 Розпізнавання об'єкта та характеристики точності розв'язання

У таблиці вказані кількісні характеристики рішення, що показують ступінь точності розпізнавання об'єкта у цьому фрагменті зображенні. Знаходимо максимальне значення.

max{0.9, 0.3, 0.01, 0.01}= 0.9,

яке і буде необхідною характеристикою (рисунок 1.7.3).

Отримання результату для виявленого об'єкту

Основна ідея алгоритму полягає в наступному:

- Виконуємо поділ ваг (weight sharing) для створення “фільтруючого вікна”, що пробігає по зображенню.
- Застосований до зображення фільтр допомагає виділити фрагменти, важливі для розпізнавання.
- У той час як у традиційному машинному зорі фільтри конструювали вручну нейромережі дозволяють нам сконструювати оптимальні фільтри за домопогою навчання.
- Фільтрування зображення можна поєднати з обчислсеннями нейронної мережі

Згортковий шар нейронної мережі

Згортковий шар є набір карт (карти ознак, features maps). Кожна карта має ядро, що сканує (скануюче ядро - являє собою фільтр, який ковзає по всьому зображенню і знаходить задані ознаки в будь-якому його місці). Кількість карток визначається вимогами до завдання, якщо взяти велику кількість карток, то підвищиться якість розпізнавання, але збільшиться обчислювальна складність.

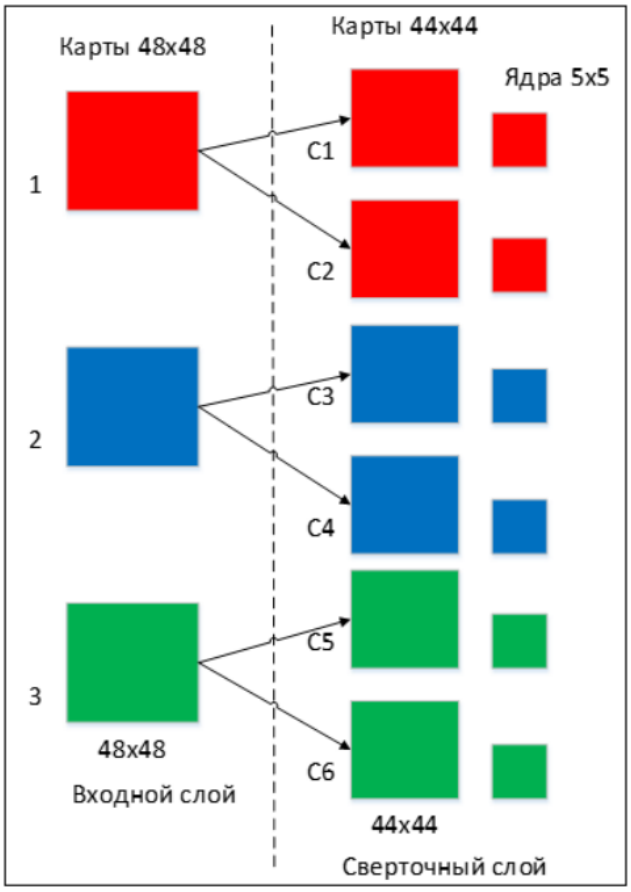


Рис 1.4.10 Організація зв'язків між картами згорткового та вхідного (попереднього) шару

Розмір у всіх карт згорткового шару однаковий і обчислюється за формулою:

$$(w, h) = (mW - kW + 1, mH - kH + 1)$$

де (x, h) – розмір обгорткової карти, що обчислюється, nW– ширина попередньої карти, nH - висота попередньої карти, lW - ширина ядра, lH - висота ядра.

Ядро ковзає по попередній карті і здійснює операцію згортка, яка часто використовується для обробки зображень:

$$(f * g)[m, n] = \sum f[m - k, n - l]\ k,l \cdot g[k, l]$$

де g - вихідна матриця зображення, g - ядро згортки. Відбувається такі: вікном розміру ядра g проходимо із заданим кроком (зазвичай 1) все

зображення f, на кожному кроці поелементно множимо вміст вікна на ядро g, результат підсумовується та записується в матрицю результату, як у зображені 15. Ковзає по попередній карті і здійснює операцію згортка, яка часто використовується для обробки зображень:

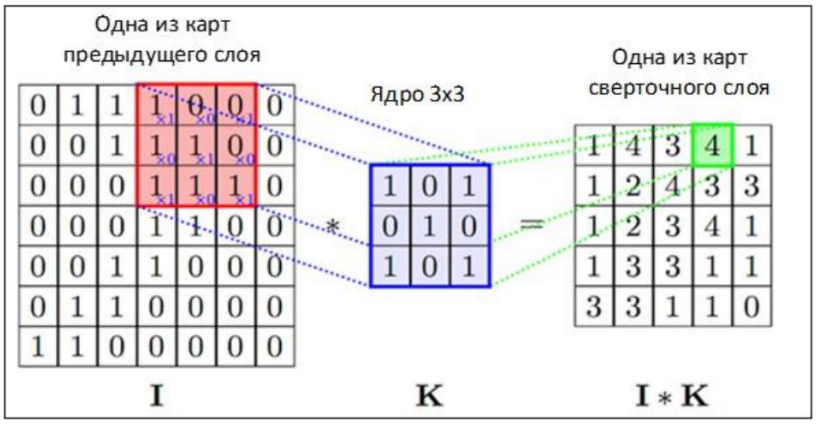


Рис 1.4.11 Операція згортки та отримання значень карти ознак

Підвибірковий шар

Підвибірковий шар також, як і згортковий, має карти, але їх кількість збігається з попереднім (згортковим) шаром. Ціль використання даного шару – зменшити розмірності карт попереднього шару. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне і воно ущільнюється до менш докладного. До того ж, фільтрація вже непотрібних деталей допомагає не перевчитися.

У процесі сканування ядром підвибіркового шару (фільтром) карти попереднього шару, скануюче ядро не перетинається на відміну від згорткового шару. Зазвичай кожна карта має ядро розміром 2x2, що дозволяє зменшити попередні карти згорткового шару в 2 рази. Вся карта ознак поділяється на комірки 2x2 елементи, у тому числі вибираються максимальні за значенням. Зазвичай, у підвиборному шарі застосовується функція активації (ReLU, Rectifiedlinearunit). Операція виборки виконується (Max-Pool – вибір максимального).

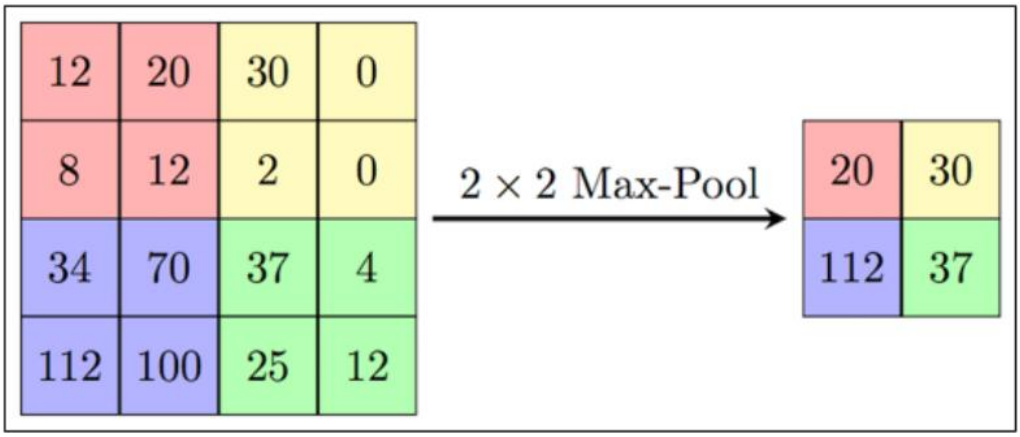


Рис 1.4.12 Формування нової карти підвиборчого шару на основі попередньої картки згорткового шару. Операція підвиборки (MaxPooling)

Шар може бути описаний формулою:

$$x^l = f(a^l \cdot subsample(x^{l-1}) + b^l),$$

Де x^l – вихід шара l, f – функція активації, a^l, b^l – коефіцієнт сдвигу шару l, subsample – операція виборки локальних максимальних значень.

Повнозв'язковий шар

Останній з типів шарів – це шар звичайного багатошарового персептрон. Мета застосування даного шару - це звернення до виходу попереднього шару та визначення властивостей, які пов'язані з певним класом.

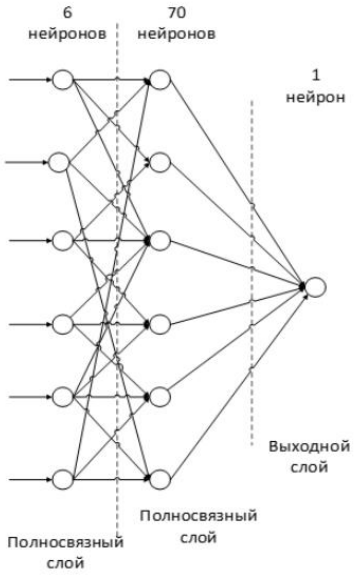


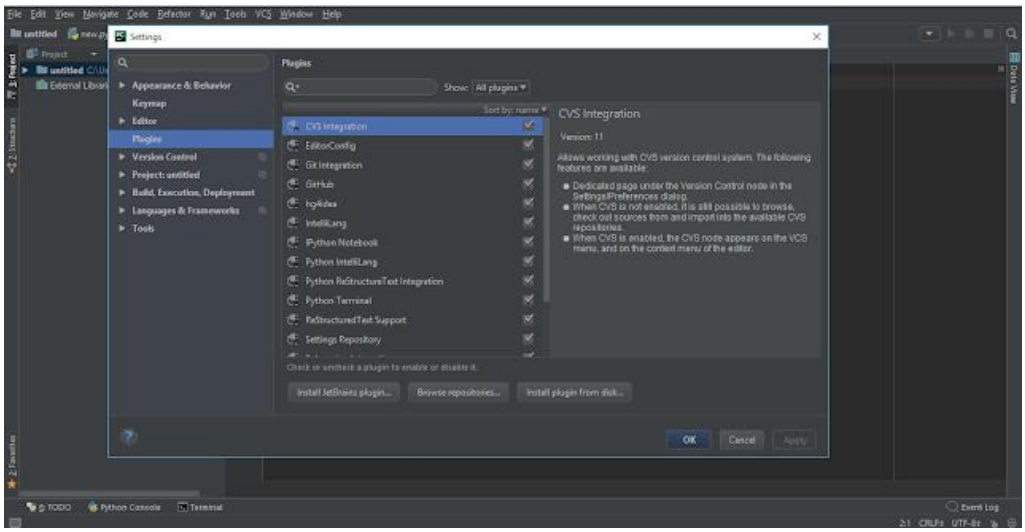
Рис 1.4.13 Приклад повнозв'язкових шарів

Нейрони кожної карти попереднього підвиборчого шару пов'язані з одним прихованим нейроном шару. Таким чином, кількість нейронів прихованого шару дорівнює числу карт підвибіркового шару, але зв'язки можуть бути не обов'язково такими, наприклад, тільки частина нейронів якоїсь із карт підвибіркового шару пов'язана з першим нейроном прихованого шару, ачастина, що залишилася з другим, або всі нейрони першої карти пов'язані знейронами 1 і 2 прихованого шару.

1.5 Опис середовища розробки ПЗ (PYCHARM)

Якими мовами підтримується PyCharm

З PyCharm можна розробляти програми на Python. Крім того, в Professional Edition можна розробляти програми Django, Flask та Pyramid. Крім того, він повністю підтримує HTML (включно з HTML5), CSS, JavaScript і XML: ці мови включені в IDE через плагіни і включені для вас за замовчуванням. Підтримка інших мов і фреймворків також може бути додана через плагіни (перейдіть до **Settings** | **Plugins** або **PyCharm** | **Preferences** | **Plugins** для користувачів MacOS, щоб дізнатися більше або встановити їх під час першого запуску IDE).



File / Settings / Plugins

На яких платформах можна запустити PyCharm

PyCharm - це крос-платформне середовище розробки, що працює у Windows, MacOS та Linux. Якщо вам потрібна допомога в установці PyCharm, див. Інструкції з інсталяції для Linux, macOS та Windows.

Навіщо мені потрібний проект?

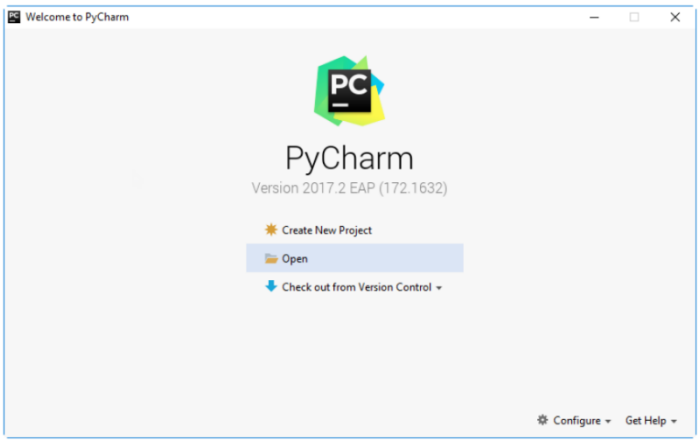
Все, що ви робите в PyCharm, виконується у контексті проекту. Він є основою підтримки кодування, рефакторингу, узгодженості стилю кодування тощо.

У вас є три варіанти почати роботу над проектом усередині середовища IDE:

1. Відкрити існуючий проект

Почніть з відкрити один з ваших існуючих проектів, що зберігаються на вашому комп'ютері. Ви можете зробити, натиснувши Відкрити проект (**Open**) на екрані привітання (або **File | Open**):

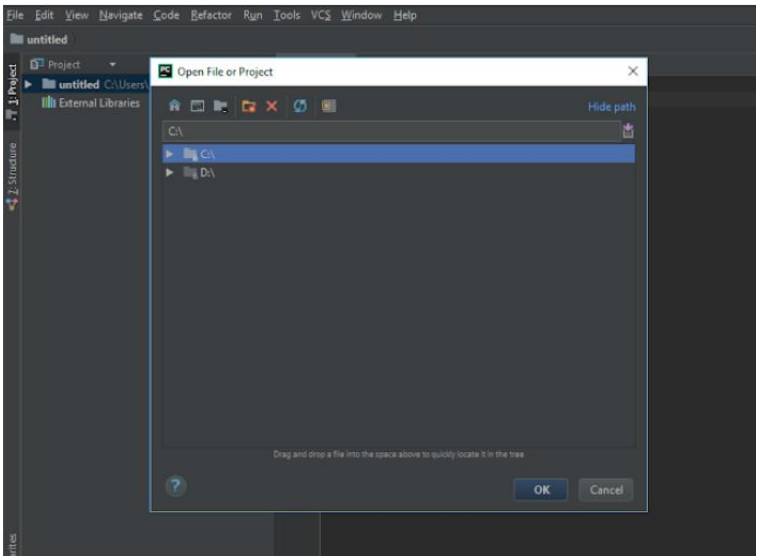
В PyCharm існують два види інтерфейса:



Світлий інтерфейс

Темний інтерфейс

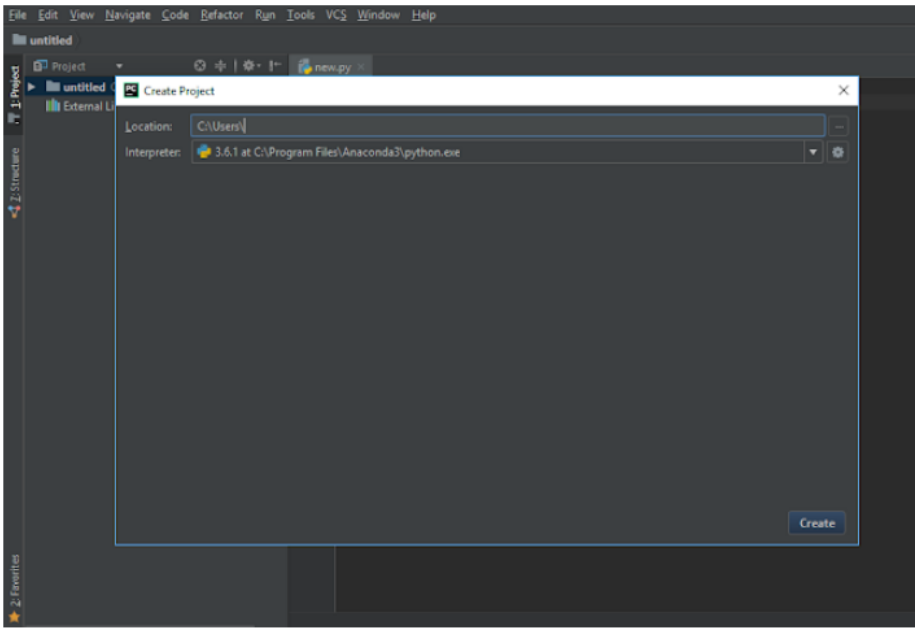
Або вибрати Open у меню File і вкажіть каталог, у якоому знаходяться ваші джерела:



Потім PyCharm створить для вас проект із джерел.

2. Створити проект із нуля

Або вибрати Open у меню File і вкажіть каталог, у якому знаходяться джерела. Якщо починати з нуля, треба натиснути New Project і на екрані ввести ім'я проекту в діалоговому вікні і буде створено проект Python.



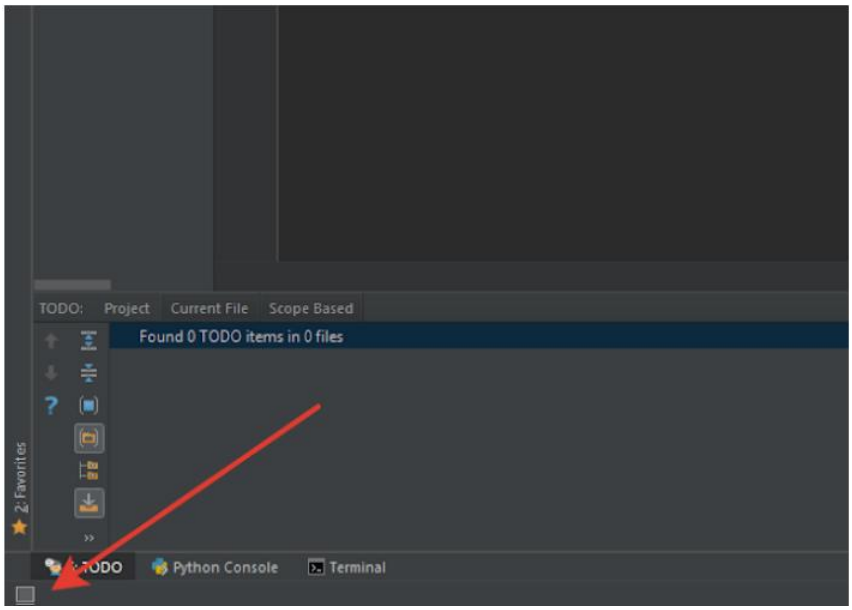
File / New Project

Коли запускається PyCharm вперше або коли немає відкритих проектів, ви бачите екран привітання . Він пропонує вам основні точки входу до середовища IDE: створення або відкриття проекту, перевірка проекту за допомогою контролю версій, перегляд документації та налаштування середовища IDE. Коли проект відкривається, ви бачите головне вікно розділене на кілька логічних областей. Розглянемо ключові елементи інтерфейсу користувача тут:

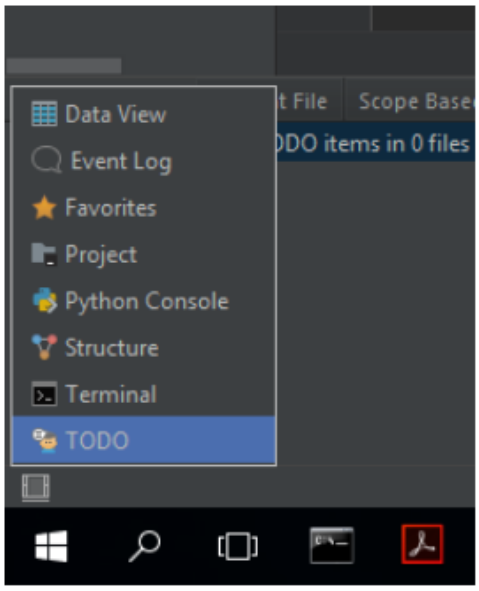
- **Project Tool Window**. *Панель інструментів проекту*. У цьому вікні відображаються файли вашого проекту.
- **PyCharm Editor**. *Редактор PyCharm* Знаходиться праворуч, де ви пишете свій код. У ньому є вкладки для зручної навігації між відкритими файлами.
- **Navigation Bar**. *Панель навігації*. Знаходиться над редактором, дозволяє швидко запускати та налагоджувати вашу програму, а також виконувати процедури контролю версій VCS.
- **Left gutter**. *Лівий стовпець*, вертикальна смуга поруч із редактором, показує брекпоінти та забезпечує зручний спосіб переходу по ієрархії коду. Він також відображає номери рядків та історію VCS.

- **Right gutter**. Правий стовпець, праворуч від редактора. PyCharm постійно контролює якість вашого коду та постійно показує результати перевірки у правому стовпці: помилки, попередження тощо. Індикатор у верхньому правому куті показує загальний статус перевірки коду для всього файлу.
- **PyCharm Tool Windows**. Панель інструментів PyCharm Це спеціальні вікна, прикріплені до низу та сторін робочої області, які забезпечують доступ до типових завдань, таких як управління проектами, пошук та навігація за вихідним кодом, інтеграція із системами контролю версій тощо.
- **Status Bar**. Рядок стану. Вказує стан вашого проекту та показує різні попередження та інформаційні повідомлення.

Крім того, у нижньому лівому куті вікна PyCharm у рядку стану ви побачите кнопку. Ця кнопка перемикає показ панелей інструментів. Якщо ви наведете вказівник миші на цю кнопку, з'явиться список доступних на даний момент панелей:

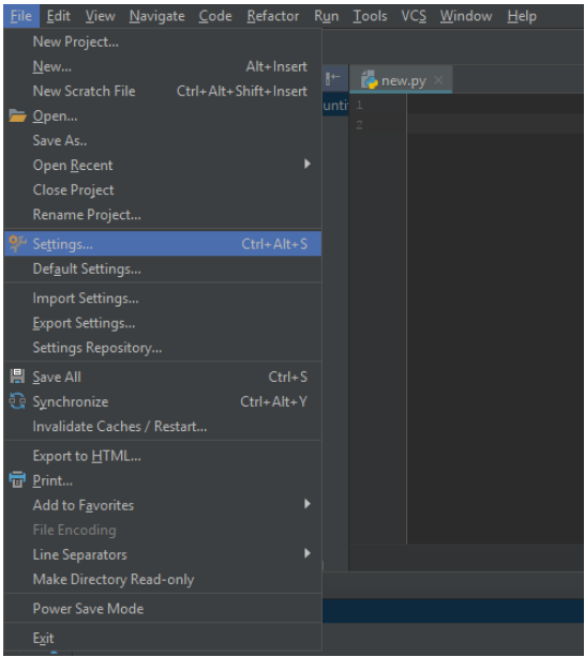


Ця кнопка перемикає показ панелей інструментів

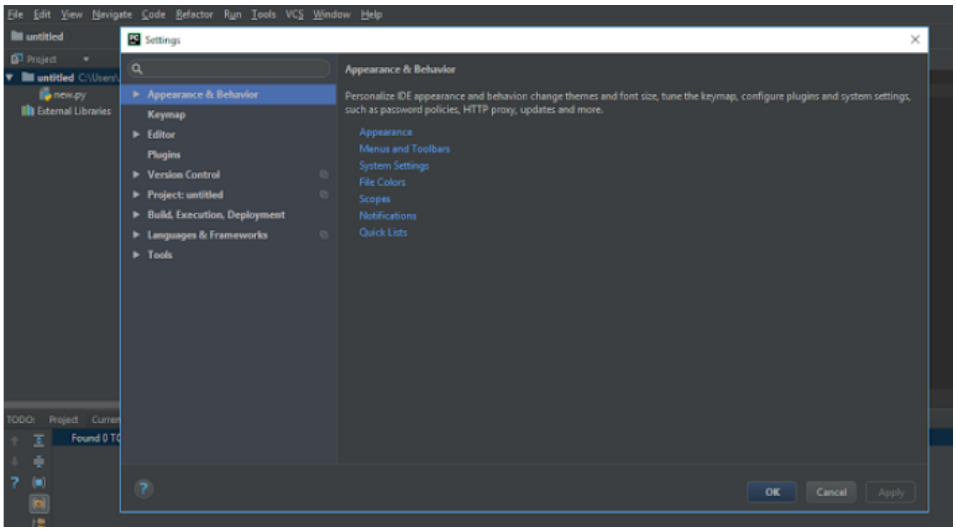


Перемикання між панелями

Можна налаштувати середовище IDE, щоб воно ідеально відповідало вашим потребам і було зручним для Вас. Зайдіть в меню **File/Settings**, щоб переглянути список доступних параметрів налаштування.



File / Settings PyCharm



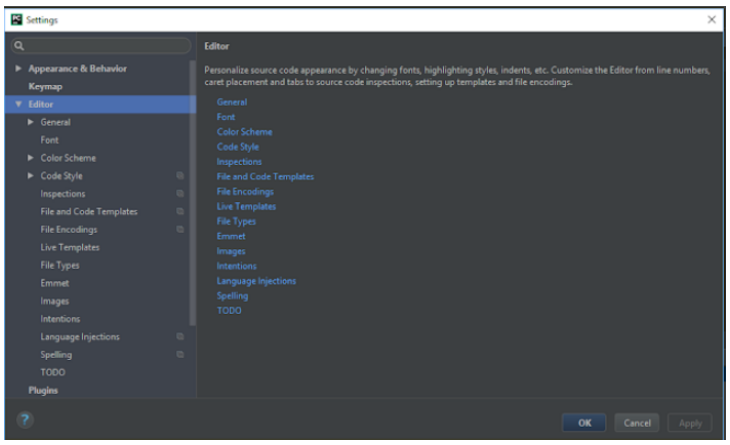
Settings PyCharm

Зовнішній вигляд IDE

Перше, що потрібно підлаштувати, - це загальний “зовнішній вигляд”. Зайдіть в меню **File / Settings / Appearance and Behavior /Appearance**, щоб вибрати Тему IDE : тема за замовчуванням, або Dracula, якщо ви віддаєте перевагу більш темному налаштуванню.

Редактор

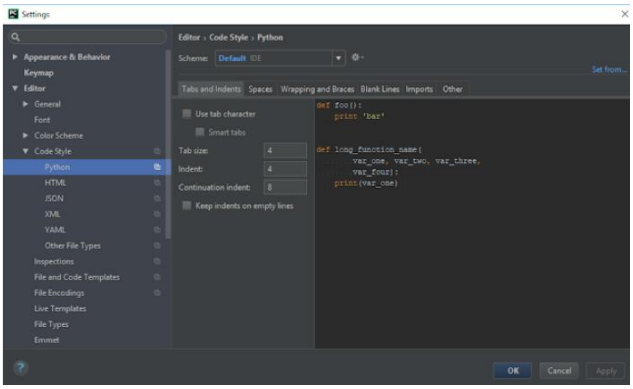
Безліч вкладок, доступних у меню File/Settings/Editor допоможуть вам налаштувати кожен аспект поведінки редактора. Тут є безліч опцій, починаючи із загальних налаштувань (наприклад, за допомогою функції Drag'n'Drop, конфігурації прокручування і т.д.). Для налаштування кольору для кожної доступної мови та варіанта використання, для вкладок та налаштувань згортки коду, для поведінки завершення коду тощо.



File / Settings / Editor

Стиль коду

Стиль коду може бути визначений для кожної мови File/Settings/Editor/Code Style. Ви також можете створити та зберегти свій власний стиль коду.



File / Settings / Editor / Code Style

Розкладка

PyCharm використовує підхід, орієнтований на клавіатуру, що означає, що майже всі дії, доступні серед IDE, зіставляються з комбінаціями клавіш.

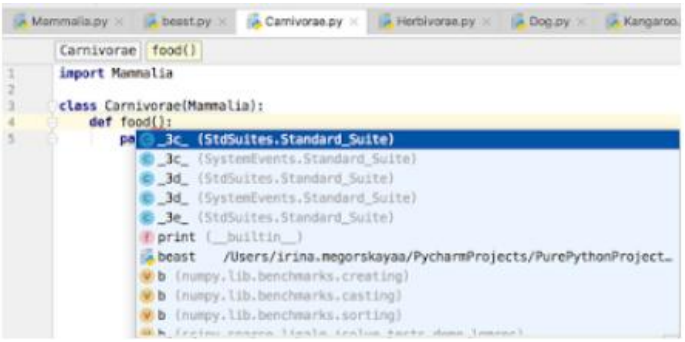
Гарячі клавіші, з якими ви працюєте, – одна з ваших особистих звичок – ваші пальці «пам'ятають» певні комбінації клавіш, і змінити ці звички досить складно.

PyCharm надає вам стандартну розкладку (**Help/Keymap Reference** у головному меню), роблячи ваше кодування дійсно продуктивним та зручним. Однак ви завжди можете її змінити **File/Settings/Keymap**.

Також є деякі попередньо визначені розкладки клавіатури (такі як Emacs, Visual Studio, Eclipse, NetBeans і т.д.), і ви можете створити свою власну розкладку на основі існуючої.

Завершення коду

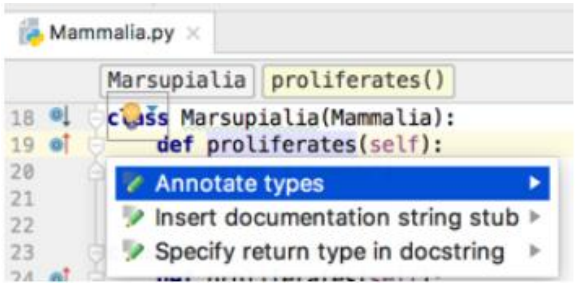
Автозаповнення коду (Auto-Completing Code) – відмінна економія часу, незалежно від типу файлу, з яким ви працюєте. Завершення працює в міру введення та завершення будь-якого імені миттєво. Інтелектуальне введення аналізує контекст, у якому ви зараз працюєте, і пропонує більш точні пропозиції, що базуються на цьому аналізі.



Auto-Completing Code PyCharm

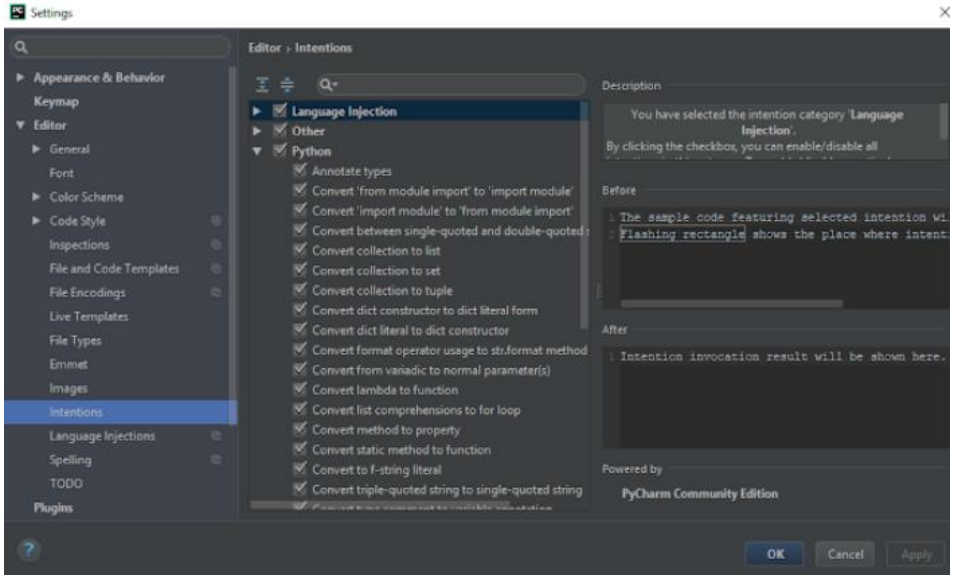
Intention Actions

PyCharm стежить за тим, що ви зараз робите, і робить розумні пропозиції, які називають Intention Actions. При вказівці з лампочкою Intention Actions дозволяють застосувати автоматичні зміни коду.



Intention Actions

Повний список доступних Intention Actions в File/Settings/Editor/Intentions.

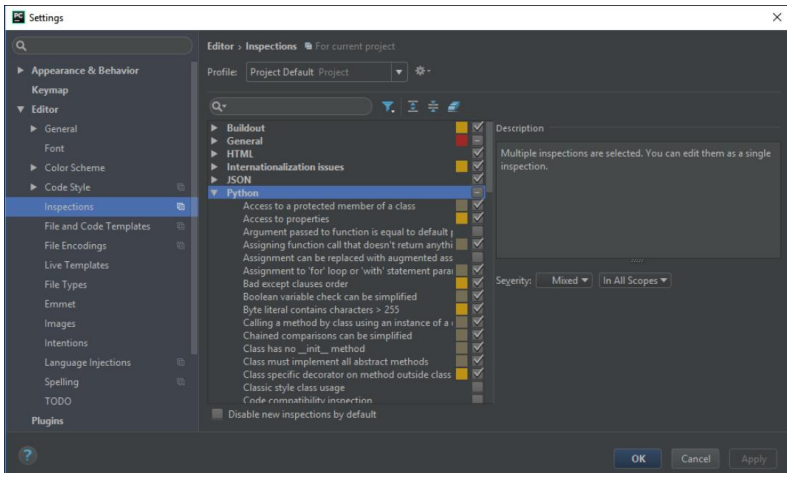


File / Settings / Editor / Intentions

PyCharm контролює ваш код і намагається зберегти його точним та чистим. Він виявляє потенційні помилки та проблеми та пропонує швидкі виправлення (quick-fixes) для них.

Щоразу, коли середовище IDE знаходить невикористаний код, нескінченний цикл та багато іншого, що , ймовірно, вимагатиме вашої уваги, ви побачите лампочку. Натисніть або натисніть Alt+Enter, щоб застосувати виправлення.

Повний список доступних перевірок можна знайти в розділі меню **File/Settings/Editor/Inspections**. Вимкніть деякі з них або увімкніть інші, а також налаштуйте рівень серйозності кожної перевірки. Ви вирішуєте, чи слід вважати це помилкою чи просто попередженням.



File / Settings / Editor / Inspections

Написання коду може бути набагато простіше і швидше, якщо ви використовуєте варіанти створення коду, доступні в PyCharm. The Code / Generate (Alt+Insert) допоможе вам створити символи, а також запропонує перевизначити/реалізувати деякі функції:

Висновки до розділу 1

У цьому розділі ми детально розібрали що таке компютерний зір, що це за технологія, яка залежить від поставленної задачі, що вона вміє робити, які має основні функції. Було розглянуто математичні моделі для опису різних алгоритмів. Також ми детально розглянули програму для редагування коду PyCharm.

РОЗДІЛ 2. АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Алгоритмічне забеспечення

На даний момент розроблено велику кількість алгоритмів пошуку об’єкта інтересу на зображенні. Як правило, неможливо побудувати алгоритм, здатний покрити всі можливі класи об'єктів. У зв'язку з цим огляд існуючих методів є одним з перших етапів розв'язання задачі обробки зображень.

Розглянуто найбільш популярні групи алгоритмів виділення інформації зображенні, описані їх переваги, недоліки та сфера застосування. Виділено 3 групи алгоритмів:

- Color-based (виділення області інтересу на основі кольору);
- Shape-based (основним критерієм є аналіз форми та контуру об'єкта);
- Learn-based (машинне навчання – основний інструмент аналізу).

У питаннях детекції та аналізу розташування дорожніх знаків досить часто знаходить своє застосування підхід, заснований на пошуку кольору інтересу. Як правило, інформація про знак закріплена в різних правових актах, у зв'язку з цим можна бути впевненим, що знак завжди зберігає свої кольори та форму. Однак, алгоритми цієї групи схильні високого впливу зміни освітленості об'єкта, через що потрібно застосування динамічно визначуваних параметрів. Дані алгоритми стають суттєво стійкішими при переході від RGB колірної схеми (Red, Green and Blue) у простір HSV (Hue, Saturation and Value). У цій колірної схеми можна встановити жорсткі обмеження на тон інтересу (hue), ірізні пороги насиченості (Saturation) та яскравості (Value) кольору. Детальний опис колірних просторів та методів переходу між ними.

Рис 2.1 Результат виділення синьої компоненти на зображенні

Алгоритми цієї групи застосовні завдання попереднього виділення області з підвищеною ймовірністю знаходження об’єкта інтересу. Результати можна передавати на наступний етап як гіпотезу про його місцезнаходження. Аналіз форми об'єкта ґрунтується на обчисленні градієнта зміни кольори для кожного пікселя. Далі обчислені градієнти аналізуються з припущенням, що об'єкт інтересу може бути описаний деяким регулярним правилом. Наприклад, для пошуку трикутника можна стверджувати, що градієнт буде спрямований за нормаллю до цього багатокутника. При цьому нормалі до сторін трикутника підпорядковуються співвідношенню, наведеному малюнку 2.1.2:

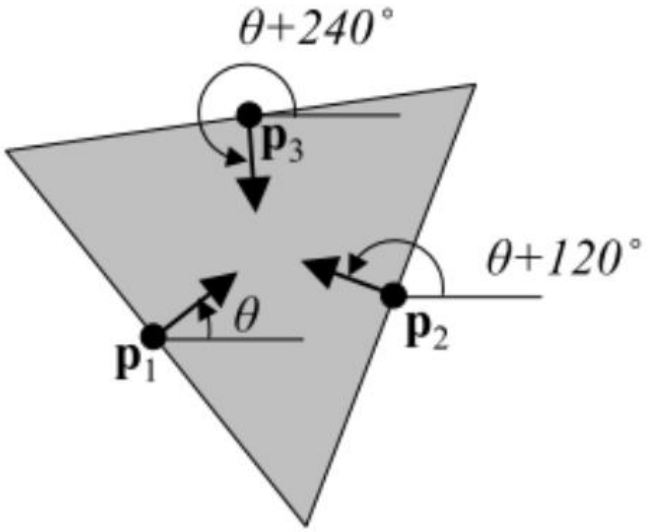


Рис 2.1.2 Співвідношення напрямок нормалей у рівносторонньому трикутнику

Це означає, що чим більше груп із трьох таких нормалей вдалося виділити в обмеженій області, тим вища ймовірність перебування там аналізованого об'єкта.

Ймовірно, найбільш популярним алгоритмом цієї групи є перетворення Хафа та її узагальнення, запропоноване 1981 року D.H. Ballard. Класичний алгоритм перетворення Хафа пов'язаний з ідентифікацією прямих у зображенні, але пізніше алгоритм було розширено можливістю ідентифікації позиції довільної фігури, найчастіше еліпсів та кіл.

У підходах, описаних вище, основні знання про об'єкт інтересу заздалегідь закладаються у алгоритм виявлення. Однак, дана інформація може бути одержана за допомогою машинного навчання.

Дослідження Віоли та Джонса продемонстрували значну віху в комп'ютерному баченні. Віола та Джонс розробили алгоритм, що дозволяє виявляти об'єкти дуже надійно та в режимі реального часу. Даний алгоритм використовує примітиви Хаара для визначення ймовірності знаходження об'єкта у цій галузі. Використання інтегрального подання зображення дозволяє сильно прискорити процес обчислення примітивів Хаара, прості ознаки посилюються при за допомогою технології AdaBoost (Adaptive Boosting), а каскадна архітектура класифікатор дозволяє швидко відкидати вікна, в яких немає об'єкт інтересу.

Також не можна не відзначити активне застосування нейронних мереж у галузі комп'ютерного зору. Наприклад, у задачі детекції та розпізнавання дорожніх знаків часто застосовується згорткова архітектура нейронних мереж, запропонована Яном Лекуном у 1988 році. Ідея даних багатошарових мереж заснована на виконанні багаторазової операції згортки між зображенням та ядром згортки. Вони призначені для виділення абстрактних понять зображення, причому самі ці поняття вибудовуються у процесі навчання мережі.

Нейронні мережі є потужним інструментом у сфері комп'ютерного зору. Як об'єкт інтересу можуть виступати різні класи предметів. Однак

нейронні мережі, як правило, вимагають Великі обчислювальні витрати, особливо на етапі навчання. У зв'язку з цим, у цій роботі було вирішено використати метод Віоли-Джонса.

2.2 Метод Віоли-Джонса

Основні принципи, на яких заснований метод:

- Використовуються ознаки Хаара, за допомогою яких відбувається пошук необхідного об’єкта;
- Використовуються інтегральне уявлення зображення, що дозволяє швидко необхідні ознаки;
- Окремі ознаки Хаара поєднуються в єдиний алгоритм при допомоги технології бустингу (від. англ boost – покращення, посилення) для вибору, що найбільше підходить на даній ділянці зображення ознаки;
- Використовуються каскади ознак для швидкого відкидання вікон, де об’єкт не знайдено.

Класифікатор Віоли-Джонса призначений для роботи з зображеннями у відтінках сірого. Це обмеження може призводити до хибним спрацьованням у разі повторюваних форм об’єктів.

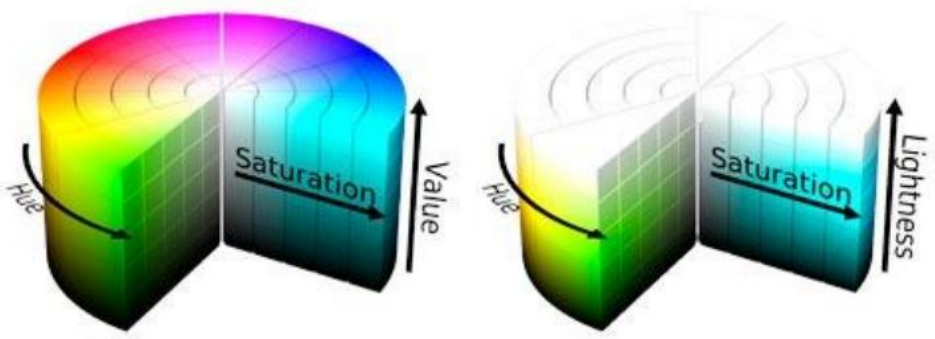


Рис 2.2 Подання колірних просторів HSV та HSL у формі циліндра

2.3 Основні алгоритми бібліотеки?

Встановлення:

```
pip install opencv-python
```

Бібліотека комп'ютерного зору та машинного навчання з відкритим вихідним кодом. До неї входять аж 2500 алгоритмів, у яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання. Ця бібліотека має інтерфейси на різних мовах програмування серед яких є Python (у цій статті використовуємо його), Java, C++ та Matlab.

2.3.1 Імпорт та перегляд зображень

```
import cv2
image = cv2.imread("./путь/к/изображению.расширение")
cv2.imshow("Image", image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

При читанні способом вище зображення знаходиться у просторі не RGB (як усі звикли), а BGR. Можливо, на початку це не так важливо, але як тільки ви почнете працювати з кольором, варто знати про цю особливість. Є 2 шляхи вирішення:

1. Поміняти місцями 1-й канал (R – червоний) з 3-м каналом (B – синій), і тоді червоний колір буде (0,0,255), а не (255,0,0).
2. Змінити колір простір на RGB:

```
rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

І тоді в коді рабить вже не с image, а с rgb_image.

Щоб закрити вікно, в якому відображається зображення, натисніть будь-яку клавішу. Якщо використовувати кнопку закриття вікна, можна натрапити на підвисання.

Протягом статті для відображення зображень буде використовуватися наступний код:

```
1 import cv2
2 def viewImage(image, name_of_window):
3   cv2.namedWindow(name_of_window, cv2.WINDOW_NORMAL)
4   cv2.imshow(name_of_window, image)
5   cv2.waitKey(0)
6   cv2.destroyAllWindows()
```

Кадрування

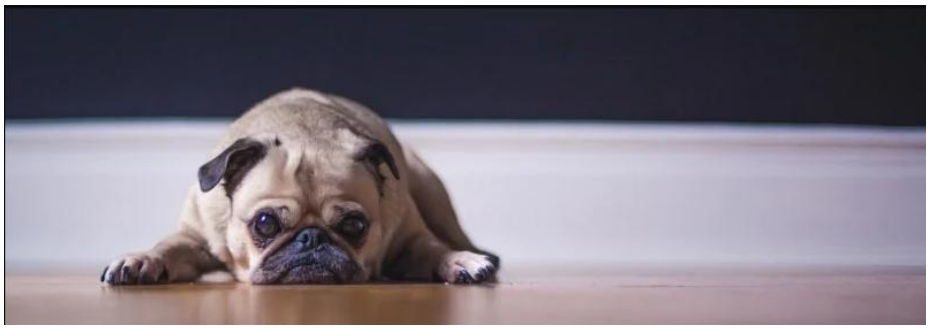


Рис 2.3.1 Зображення після кадрування

```
1 import cv2
2 cropped = image[10:500, 500:2000]
3 viewImage(cropped, "Пес после кадрування")
```

Де image[10:500, 500:2000] – це image[y:y + висота, x:x + ширина]

2.3.2 Зміна розміру

Рис 2.3.2 Після зміни розміру на 20%

```
1 import cv2
2 scale_percent = 20 # Процент от изначального размера
3 width = int(img.shape[1] * scale_percent / 100)
4 height = int(img.shape[0] * scale_percent / 100)
5 dim = (width, height)
6 resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
7 viewImage(resized, "Після зміни розміру на 20 %")
```

Ця функція враховує співвідношення сторін оригінального зображення.

2.3.3 Поворот

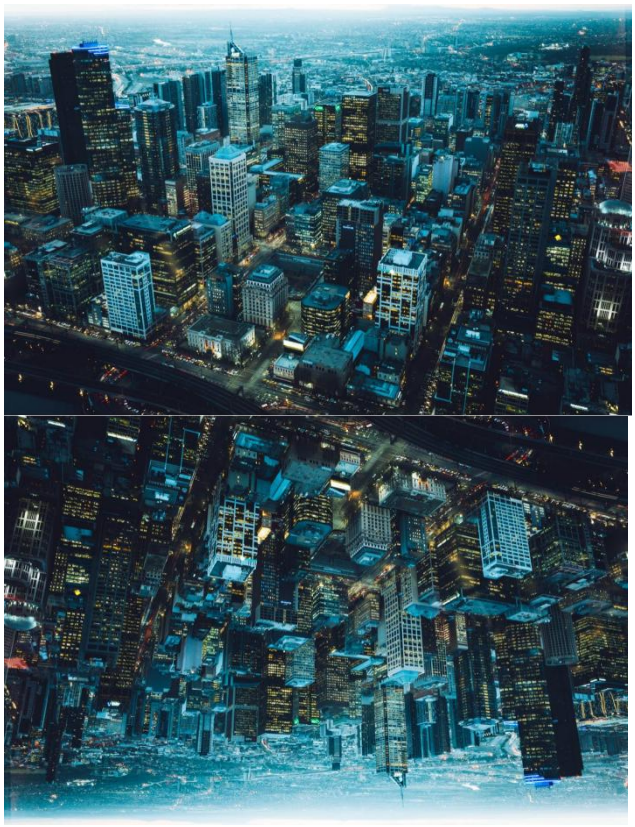


Рис 2.3.3 Зображення після повороту на 180 градусів

```
1 import cv2
2 (h, w, d) = image.shape
3 center = (w // 2, h // 2)
4 M = cv2.getRotationMatrix2D(center, 180, 1.0)
5 rotated = cv2.warpAffine(image, M, (w, h))
6 viewImage(rotated, "Пес після повороту на 180 градусів")
```

Image.shape повертає висоту, ширину та канали. M – матриця повороту – повертає зображення на 180 градусів навколо центру. -ve — це кут повороту зображення за годинниковою стрілкою, а +ve відповідно проти годинникової.

2.3.4 Переклад у градації сірого та у чорно-біле зображення на порозі

Рис 2.3.4 Переклад у градації сірого та у чорно-біле зображення на порозі

```
1 import cv2
2 gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
3 ret, threshold_image = cv2.threshold(im, 127, 255, 0)
4 viewImage(gray_image, "Пес в градаціях сірого")
5 viewImage(threshold_image, "Чорно-білий пес")
```

gray_image – це одноканальна версія зображення.

Функція threshold повертає зображення, в якому всі пікселі, які темніші (менше) 127 заменені на 0, а всі, які яскравіші (більше) 127, - на 255.

Для ясності інший приклад:

```
ret, threshold = cv2.threshold(im, 150, 200, 10)
```

Тут все, що темніше за 150, замінюється на 10, а все, що яскравіше, — на 200.

2.3.5 Розмиття/згладжування

Рис 2.3.5 Розмите зображення

```
1 import cv2
2 blurred = cv2.GaussianBlur(image, (51, 51), 0)
3 viewImage(blurred, "Розмите зображення")
```

Функція GaussianBlur (розмиття по Гаусс) приймає 3 параметри:

1. Початкове зображення.
2. Кортеж із двох позитивних непарних чисел. Чим більше числа, тим більша сила згладжування.
3. sigmaX та sigmaY. Якщо ці параметри залишити рівними 0, їх значення буде розраховано автоматично.

2.3.6 Малювання прямокутників

Рис 2.3.6 Обводимо прямокутником

2.3.7 Малювання ліній

Рис 2.3.7 Розділені лінією

```
1 import cv2
2 output = image.copy()
3 cv2.rectangle(output, (2600, 800), (4100, 2400), (0, 255, 255), 10)
4 viewImage(output, "Обводим прямоугольником лицо пёсика")
```

Ця функція приймає 5 параметрів:

1. Саме зображення;
2. Координата верхнього лівого кута (x1, y1);
3. Координата нижнього правого кута (x2, y2);
4. Колір прямокутника (GBR/RGB в залежності від обраного колірної моделі);
5. Товщина прямокутника лінії

2.3.7 Текст на зображенні

Рис 2.3.8 Зображення с текстом

```
1 import cv2
2 output = image.copy()
3 cv2.putText(output, "We <3 Subject", (1500, 3600),cv2.FONT_HERSHEY_SIMPLEX, 15, (30, 105, 210), 40)
4 viewImage(output, "Изображение с текстом")
```

Функція putText приймає 7 параметрів:

1. Беспосередньо зображення;
2. Текст для зображення;
3. Координата нижнього лівого кута початку тексту (x, y);
4. Шрифт, що використовується;
5. Розмір шрифту;
6. Колір тексту (GBR/RGB залежно від вибраної моделі кольорів);
7. Товщина ліній букв.

2.3.9 Розпізнавання осіб

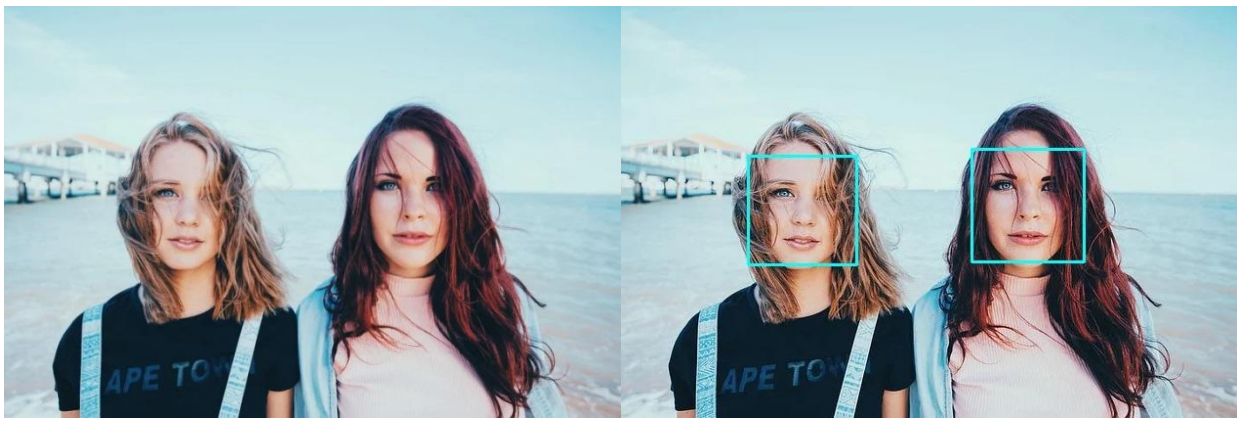


Рис 2.3.9 Облич виявлено: 2

```
1 import cv2
2 image_path = "./шлях/до/фото.розширення"
3 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
3 image = cv2.imread(image_path)
4 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
5 faces = face_cascade.detectMultiScale(
6 gray,
7 scaleFactor= 1.1,
8 minNeighbors= 5,
9 minSize=(10, 10)
10
11 faces_detected = "облич виявлено: " + format(len(faces))
12 print(faces_detected)
13 # Малюєм квадрати навкруги лиць
14 for (x, y, w, h) in faces:
15   cv2.rectangle(image, (x, y), (x+w, y+h), (255, 255, 0), 2)
16  viewImage(image,faces_detected)
```

detectMultiScale — загальна функція для розпізнавання як облич, так і об'єктів. Щоб функція шукала саме обличчя, ми передаємо відповідний каскад.

Функція detectMultiScale приймає 4 параметри:

1. Оброблюване зображення в сірому градації;

2. Параметр scaleFactor. Деякі особи можуть бути більшими за інших, оскільки знаходяться ближче, ніж інші. Цей параметр компенсує перспективу;
3. Алгоритм розпізнавання використовує ковзне вікно під час розпізнавання об'єктів. Параметр minNeighbors визначає кількість об'єктів навколо обличчя. Тобто чим більше значення цього параметра, тим більше аналогічних об'єктів необхідно алгоритму, щоб визначив поточний об'єкт, як особа.Занадто маленьке значення збільшить кількість хибних спрацьовувань, а надто велике зробить алгоритм більш вимогливим;
4. minSize – безпосередньо розмір цих областей.

Розпізнавання об'єктів здійснюється за допомогою сегментації кольору зображення. Для цього є дві функції: cv2.findContours та cv2.drawContours.

Ця бібліотека є дуже важливою для тих, хто розробляє проекти, пов'язані з машинним навчанням в області зображень.

2.3.10 Збереження зображення

```
1 import cv2
2 image = cv2.imread("./імпорт/шлях.розширення")
3 cv2.imwrite("./ескорт/шлях.розширення", image)
```

2.3.11 Заключення

OPENCV – відмінна бібліотека з легкими алгоритмами, які можуть використовуватися в 3D – рендері, просунутоум редагуванні зображень і відео, відстеження та ідентифікації об’єктів і людей на відео, пошуку ідентичних зображень з набору і для багато чого ще.

Як основний програмний інструмент, було обрано мову програмування Python. Ця мова програмування є кросплатформною і дозволяє працювати швидше та ефективніше інтегрувати в різних операційних системах. Більш того, Python широко застосовується у сфері машинного навчання, аналіз великих даних, комп'ютерного зору тощо.

В даний час існує кілька сучасних фреймворків для роботи із зображеннями, що підтримують глибоке навчання:

TensorFlow – відкрита програмна бібліотека для машинного навчання, розроблена Google для вирішення завдань побудови та тренування нейронної мережі з метою автоматичного знаходження та класифікації образів, досягаючи якості людського сприйняття.

OpenCV – бібліотека з відкритим вихідним кодом використовується для всіх видів обробки та аналізу зображень і відео, яка включає кілька сотень алгоритмів комп'ютерного зору: інтерпретації зображень, калібрування камери по еталону, усунення оптичних спотворень, визначення подібності, аналіз переміщення об'єкта, визначення форми об'єкта і стеження за об'єктом, 3D- реконструкція, сегментація об'єкта, розпізнавання жестів тощо.

Keras – це мінімалістична відкрита нейромережева бібліотека, написана мовою Python для глибокого навчання, яка може працювати поверх TensorFlow. Дана бібліотека націлена для оперативної роботи з мережами глибокого навчання, при цьому спроектована так, щоб бути компактною, модульною та розширюваною.

NumPy – це бібліотека, з відкритим кодом для мови програмування Python, з підтримкою багатовимірних масивів (включаючи матриць) і підтримкою високорівневих математичних функцій, призначених для роботи з багатовимірними масивами.

2.4 Tkinter

Tkinter (від англ. Tk interface) – це графічна бібліотека, що дає змогу створювати програми з віконним інтерфейсом. Ця бібліотека є інтерфейсом до популярної мови програмування та інструменту створення графічних додатків tcl/tk. Tkinter, як і tcl/tk, є кросплатформеною бібліотекою і може бути використана в більшості поширених операційних систем (Windows, Linux, Mac OS X та ін.).

Починаючи з версії python-3.0 бібліотека перейменована відповідно до PEP 8 в tkinter (з маленької літери).

Імпортується вона як і будь-яка інша бібліотека:

```
from tkinter import *
```

У Tkinter візуальні контроли називаються віджетами (widget, від англ. window gadget) – стандартизований компонент графічного інтерфейсу, з яким взаємодіє користувач.

Tk є базовим класом будь-якої програми Tkinter. Під час створення об'єкта цього класу запускається інтерпретатор tcl/tk і створюється базове вікно програми.

Tkinter є подієво-орієнтованою бібліотекою. У програмах такого типу є головний цикл обробки подій. У Tkinter такий цикл запускається методом mainloop. Для явного виходу з інтерпретатора та завершення циклу обробки подій використовується метод quit.

Таким чином мінімальний додаток на Tkinter буде таким:

```
from tkinter import *
root = Tk()
root.mainloop()
```

У програмі можна використовувати кілька інтерпретаторів tcl/tk. Так як після виклику методу mainloop подальші команди python виконуватися не будуть до виходу з циклу обробки подій, необхідно метод mainloop всіх інтерпретаторів, крім останнього, здійснювати у фоновому режимі. Приклад запуску двох інтерпретаторів:

```
from tkinter import *
root1 = Tk()
root2 = Tk()
root1.after(500, root1.mainloop) # первый цикл запускаем в фоне
root2.mainloop()
```

При використанні двох і більше інтерпретаторів слід стежити щоб об'єкти, створені одному інтерпретаторі, використовувалися лише у ньому. Наприклад, зображення, створене в першому інтерпретаторі, може бути використане багато разів у цьому інтерпретаторі, але не може бути використане в інших інтерпретаторах. Необхідність запуску декількох інтерпретаторів в одному додатку виникає вкрай рідко. Для створення додаткового вікна програми в більшості випадків достатньо віджету Toplevel.

2.4.1 Основні віджети

Toplevel

Toplevel – вікно верхнього рівня. Зазвичай використовується створення багатовіконних програм, і навіть для діалогових вікон.

Методи віджету

- **title** – заголовок вікна;
- **overrideredirect –** вказівка віконному менеджеру ігнорувати це вікно. Аргументом є True чи False. У разі, якщо аргумент не вказано – отримуємо поточне значення. Якщо аргумент дорівнює True, таке вікно буде показано віконним менеджером без обрамлення (без заголовку і бордюру). Може бути використано, наприклад, для створення splashscreen при старті програми;
- **iconify/deiconity –** згорнути/розгорнути вікно;
- **withdraw –** "заховати" (зробити невидимим) вікно. Для того, щоб знову показати його, треба використовувати метод deiconify;
- **minsize та maxsize –** мінімальний/максимальний розмір вікна. Методи приймають два аргументи – ширина та висота вікна. Якщо аргументи не вказані – повертають поточне значення;
- **state –** отримати поточне значення стану вікна. Може повертати такі значення: normal (нормальний стан), icon (показано у вигляді іконки), iconic (згорнуто), withdrawn (не показано), zoomed (розгорнуто на повний екран, тільки для Windows та Mac OS x) ;

- **resizable** – чи може користувач змінювати розмір вікна. Приймає два аргументи – можливість зміни розміру по горизонталі та по вертикалі. Без аргументів повертає значення;
- **geometry** – встановлює геометрію вікна у форматі ширинахвисота+x+y (приклад: geometry("600x400+40+80") – помістити вікто в точку з координатами 40,80 і встановити розмір 600x400). Розмір або координати можуть бути опущені (geometry("600x400") – тільки змінити розмір, geometry("+40+80") - лише перемістити вікно;
- **transient –** зробити вікно залежним від іншого вікна, вказаного в аргументі. Згортатиметься разом із зазначеним вікном. Без аргументів повертає значення;
- **protocol –** отримує два аргументи: назва події та функцію, яка буде викликатись при настанні зазначеної події. Події можуть називатися WM_TAKE_FOCUS (отримання фокусу), WM_SAVE_YOURSELF (необхідно зберегтися, зараз є застарілим), WM_DELETE_WINDOW (видалення вікна) ;
- **tkraise** (синонім **lift**) і **lower –** піднімає (розміщує поверх всіх інших вікон) чи опускає вікно. Методи можуть приймати один необов'язковий аргумент: над яких вікном розмістити поточне;
- **grab_set** – встановлює фокус на вікно, навіть за наявності відкритих інших вікон;
- **grab_release** – знімає монопольне володіння фокусом уведення з вікна.

Ці методи можуть бути використані для кореневого (root) вікна.

Приклад:

```
from Tkinter import *
def window_deleted():
   print u'Вікно закрито'
   root.quit() # явне указание на вихід із програми
root=Tk()
```

```
root.title(u'Приклад приложения')
root.geometry('500x400+300+200') # ширина=500, высота=400, x=300, y=200
root.protocol('WM_DELETE_WINDOW', window_deleted) # обработчик закрытия окна
root.resizable(True, False) # размер окна может быть изменён только по горизонтали
root.mainloop()
```

У такий спосіб можна запобігти закриттю вікна (наприклад, якщо закриття вікна призведе до втрати введених користувачем даних).

Button

Віджет **Button** – звичайна кнопка, яка використовується в тисячах програм. Приклад коду:

```
from Tkinter import *
root=Tk()
button1=Button(root,text='ok',width=25,height=5,bg='black',fg='red',font='arial 14')
button1.pack()
root.mainloop()
```

Розберемо цей маленький код. За створення, власне, вікна відповідає клас Tk(), і насамперед потрібно створити екземпляр цього класу. Цей екземпляр прийнято називати root, хоча ви можете назвати його як завгодно. Далі створюється кнопка, причому ми вказуємо її характеристики (починати необхідно з вказівки вікна, у прикладі - root). Тут перераховані деякі з них:

- **text –** який буде відображено на кнопці (у прикладі - ок);
- **width,height –** відповідно, ширина та довжина кнопки;
- **bg –** колір кнопки (скорочено від background, у прикладі колір - чорний);
- **fg –** колір тексту на кнопці (скорочено від foreground, у прикладі колір - червоний)
- **font** - шрифт та його розмір (у прикладі - arial, розмір - 14)

Далі нашу кнопку необхідно розмістити на вікні. Для цього, в Tkinter використовуються спеціальні пакувальники (pack(), place(), grid()). Детальніше про пакувальників дізнаємось пізніше. Поки, щоб розмістити кілька віджетів на вікні, будемо застосовувати найпростіший пакувальник pack().Наприкінці програми потрібно використовувати функцію mainloop (див. приклад), інакше вікно не буде створено.

Label

Label - це віджет, призначений для відображення будь-якого напису без можливості редагування користувачем. Має самі властивості, як і перелічені характеристики кнопки.

Entry

Entry - це віджет, який дозволяє користувачеві ввести один рядок тексту. Має додаткову властивість bd (скорочено від borderwidth), що дозволяє регулювати ширину кордону.

- **borderwidth –** ширина бордюру елемента
- **bd** - скорочення від borderwidth
- **width** - задає довжину елемента у знайомостах.
- **show** - задає символ, що відображається.

Text

Text - це віджет, який дозволяє користувачеві ввести будь-яку кількість текстів. Має додаткову властивість wrap, що відповідає за перенесення (щоб, наприклад, переносити за словами, потрібно використовувати значення WORD. Наприклад:

```
text1=Text(root,height=7,width=7,font='Arial 14',wrap=WORD)
text1.pack()
```

Методи insert, delete і get додають, видаляють або витягують текст. Перший аргумент - місце вставки як 'x.y', де x – це рядок, а y – стовпець. Наприклад:

```
text1.insert(1.0,'Добавити Текст\n\ в початок першої стрічки')
text1.delete('1.0', END)   # Видалити все
text1.get('1.0', END)      # Вилучити все
```

Listbox

Listbox - це віджет, який є списком, з елементів якого користувач може вибирати один або кілька пунктів. Має додаткову властивість selectmode, яка, при значенні SINGLE, дозволяє користувачеві вибрати тільки один елемент списку, а при значенні EXTENDED - будь-яку кількість. Приклад:

```
from Tkinter import *
root=Tk()
listbox1=Listbox(root,height=3,width=15,selectmode=EXTENDED)
list1=[u"Київ",u"Харків",u"Одесса"]
for i in list1:
   listbox1.insert(END,i)
listbox1.pack()
root.mainloop()
```

Frame

Віджет Frame (рамка) призначений для організації віджетів у вікні. Розглянемо приклад:

```
from tkinter import *
root=Tk()
frame1=Frame(root,bg='green',bd=5)
frame2=Frame(root,bg='red',bd=5)
button1=Button(frame1,text=u'Первая кнопка')
button2=Button(frame2,text=u'Вторая кнопка')
frame1.pack()
frame2.pack()
button1.pack()
```

```
button2.pack()
root.mainloop()
```

Властивість bd відповідає за товщину краю рамки.

Checkbutton

Checkbutton - це віджет, який дозволяє відзначити галочкою певний пункт у вікні. При використанні кількох пунктів потрібно кожному присвоїти свою змінну. Розберемо приклад:

```
from tkinter import *
root=Tk()
var1=IntVar()
var2=IntVar()
check1=Checkbutton(root,text=u'1 пункт',variable=var1,onvalue=1,offvalue=0)
check2=Checkbutton(root,text=u'2 пункт',variable=var2,onvalue=1,offvalue=0)
check1.pack()
check2.pack()
root.mainloop()
```

IntVar() - спеціальний клас бібліотеки до роботи з цілими числами. variable - властивість, яка відповідає за прикріплення до віджету змінної. onvalue, offvalue - властивості, які надають прикріпленої до віджету змінної значення, що залежить від стану (onvalue - при вибраному пункті, offvalue - при невибраному пункті).

Radiobutton

Віджет Radiobutton виконує функцію, схожу на функцію віджета Checkbutton. Різниця в тому, що у віджеті Radiobutton користувач може вибрати лише один із пунктів. Реалізація цього віджету дещо інша, ніж віджета Checkbutton:

```
from tkinter import *
root=Tk()
var=IntVar()
rbutton1=Radiobutton(root,text='1',variable=var,value=1)
```

```
rbutton2=Radiobutton(root,text='2',variable=var,value=2)
rbutton3=Radiobutton(root,text='3',variable=var,value=3)
rbutton1.pack()
rbutton2.pack()
rbutton3.pack()
root.mainloop()
```

У цьому віджеті використовується вже одна змінна. Залежно від того, який пункт вибрано, вона змінює своє значення. Найцікавіше, що якщо присвоїти цій змінній будь-яке значення, зміниться і вибраний віджет. На цьому ми перервемо вивчення типів віджетів (потім до них обов'язково повернемося).

Scale

Scale (шкала) - це віджет, що дозволяє вибрати будь-яке значення із заданого діапазону. Властивості:

- **orient –** як розташована шкала на вікні. Можливі значення: HORIZONTAL, VERTICAL (горизонтально, вертикально);
- **length** – довжина шкали;
- **from_ -** з якого значення починається шкала;
- **to –** яким значенням закінчується шкала;
- **tickinterval -** інтервал, через який відображаються позначки шкали;
- **resolution –** крок пересування (мінімальна довжина, яку можна пересунути двигун);

```
from tkinter import *
root = Tk()
def getV(root):
   a = scale1.get()
   print "Значення", a
scale1 = Scale(root,orient=HORIZONTAL,length=300,from_=50,to=80,tickinterval=5,
          resolution=5)
button1 = Button(root,text=u"Отримати значення")
```

```
scale1.pack()
button1.pack()
button1.bind("<Button-1>",getV)
root.mainloop()
```

Тут використовується спеціальний метод get(), який дозволяє зняти з віджету певне значення і використовується не тільки в Scale.

ScrollBar

Цей віджет дає можливість "прокрутити" інший віджет (наприклад текстове поле) і часто буває корисним. Використання цього віджету досить нетривіальне. Необхідно зробити дві прив'язки: command смуги прокручування прив'язуємо до методу xview/yview віджету, а xscrollcommand/yscrollcommand віджету прив'язуємо до методу set смуги прокручування.

```
from tkinter import *
root = Tk()
text = Text(root, height=3, width=60)
text.pack(side='left')
scrollbar = Scrollbar(root)
scrollbar.pack(side='left')
# первая привязка
scrollbar['command'] = text.yview
# вторая привязка
text['yscrollcommand'] = scrollbar.set
root.mainloop()
```

2.4.2 Пакувальник

Пакувальник (менеджер геометрії, менеджер розташування) це спеціальний механізм, який розміщує (пакує) віджети на вікні. У Tkinter є три пакувальники: pack, place, grid. Зверніть увагу, що в одному віджеті можна використовувати тільки один тип упаковки, при змішуванні різних типів упаковки програма, швидше за все, не працюватиме.

Розберемо кожен із них по порядку:

pack()

Результат роботи можна побачити на скріншоті праворуч.

Для створення складної структури з використанням цього пакувальника зазвичай використовують Frame, вкладені один в одного.

Аргументи

При застосуванні цього пакувальника можна вказати такі аргументи:

- **side** ("left"/"right"/"top"/"bottom") – до якої сторони повинен примикати віджет, що розміщується;
- **fill** (None/"x"/"y"/"both") – чи необхідно розширювати простір, що надається віджету;
- **expand** (True/False) – чи необхідно розширювати сам віджет, щоб він зайняв простір, що йому надається.
- **in_** - явна вказівка у який батьківський віджет має бути поміщений.

Додаткові функції

Крім основної функції віджети мають додаткові методи для роботи з пакувальниками.

- **pack_configure –** синонім для pack;
- **pack_slaves** (синонім **slaves**) – повертає список усіх дочірніх упакованих віджетів;

- **pack_info** – повертає інформацію про конфігурацію упаковки.
- **pack_propagate** (синонім **propagate**) (True/False) – включає відключає поширення інформації про геометрію дочірніх віджетів. За замовчуванням віджет змінює розмір відповідно до розміру своїх нащадків. Цей метод може вимкнути таку поведінку (pack_propagate(False)). Це може бути корисно, якщо необхідно, щоб віджет мав фіксований розмір і не змінював його за забаганки нащадків.
- **pack_forget** (синонім **forget**) – видаляє віджет та всю інформацію про його розташування з пакувальника. Пізніше цей віджет може бути знову розміщенний.

grid()

Цей пакувальник є таблицею з осередками, в які поміщаються віджети. Крім основної функції віджети мають додаткові методи для роботи з пакувальниками.

Аргументи

- **row –** номер рядка, в який розміщуємо віджет;
- **rowspan** – скільки рядків займає віджет;
- **column** – номер стовпця, в який розміщуємо віджет;
- **columnspan** – скільки стовпці займає віджет;
- **padx/pady** – розмір зовнішнього кордону (бордюру) по горизонталі та вертикалі;
- **ipadx/ipady** – розмір внутрішнього кордону (бордюру) по горизонталі та вертикалі. Різниця між pad і ipad в тому, що при вказівці pad розширюється вільний простір, а при ipad розширюється віджет, що розміщується;
- **sticky** ("n", "s", "e", "w" або їх комбінація) – вказує до якого кордону "приклеювати" віджет. Дозволяє розширювати віджет у вказному напрямку. Межі названі відповідно до сторін світла. "n" (північ) - верхня межа, "s" (південь) - нижня, "w" (захід) - ліва, "e" (схід) - права.

- **in_** - явна вказівка у який батьківський віджет має бути поміщений.

Для кожного віджету вказуємо, в якому він знаходиться рядку, і в якому стовпі. Якщо потрібно, вказуємо, скільки осередків він займає (якщо, наприклад, нам потрібно розмістити три віджети під одним, необхідно “розтягнути” верхній на три осередки). Приклад:

```
entry1.grid(row=0,column=0,columnspan=3)
button1.grid(row=1,column=0)
button2.grid(row=1,column=1)
button3.grid(row=1,column=2)
```

place()

place є простим пакувальником, що дозволяє розміщувати віджет у фіксованому місці з фіксованим розміром. Також він дозволяє вказувати координати розміщення у відносних одиницях для реалізації гумового розміщення. При використанні цього пакувальника нам необхідно вказувати координати кожного віджету. Наприклад:

```
button1.place(x=0,y=0)
```

Цей пакувальник, хоч і здається незручним, надає повну свободу розміщення віджетів на вікні.

Аргументи

- **anchor** ("n", "s", "e", "w", "ne", "nw", "se", "sw" або "center") – який кут або сторона віджету, що розміщується, буде вказана в аргументах x /y/relx/rely. За промовчанням "nw" - лівий верхній;
- **bordermode** ("inside", "outside", "ignore") – визначає якою мірою враховуватимуться межі при розміщенні віджету;
- **in_** - явна вказівка у який батьківський віджет має бути поміщений;
- **x** та **y** – абсолютні координати (у пікселях) розміщення віджету;

- **width** і **height** - абсолютна ширина і висота віджету;
- **relx** та **rely** – відносні координати (від 0.0 до 1.0) розміщення віджету;
- **relwidth** і **relheight** - відносні ширина і висота віджету.

Відносні та абсолютні координати (а також ширину та висоту) можна комбінувати. Так наприклад, relx=0.5, x=-2 означає розміщення віджету в двох пікселях зліва від центру батьківського віджету, relheight=1.0, height=-2 - висота віджета на два пікселі менше висоти батьківського віджету.

2.4.3 Прив'язка подій

Command

Для більшості віджетів, що реагують на дію користувача, активацію віджету (наприклад, натискання кнопки) можна прив'язати за допомогою опції command. До таких віджетів відносяться: Button, Checkbutton, Radiobutton, Spinbox, Scrollbar, Scale. Вище ми вже неодноразово користувалися цим способом:

```
button = Button(command=callback)
```

Такий спосіб є кращим та найбільш зручним способом прив'язки.

bind()

Метод bind прив'язує подію до дії (натискання кнопки миші, натискання клавіші на клавіатурі і т.д.). bind приймає три аргументи:

- назва події;
- функцію, яка буде викликана при настанні події;
- Третій аргумент (необов'язковий) – рядок "+" – означає, що ця прив’язка додається до існуючих. Якщо третій аргумент опущений або дорівнює порожньому рядку – привязка замінює всі інші прив’язки цієї події до віджету.

Метод bind повертає ідентифікатор прив'язки, який може бути використаний у функції unbind.

Зверніть увагу, що якщо bind прив'язаний до вікна верхнього рівня, то Tkinter оброблятиме події всіх віджетів цього вікна (див. також bind_all нижче).

Функція, що викликається під час настання події, має приймати один аргумент. Це об'єкт класу Event, в якому описано подію, що настала. Об'єкт класу Event має такі такі атрибути (у дужках вказані події, для яких цей атрибут встановлено):

- **serial –** серійний номер події (всі події);
- **num** – номер кнопки миші (ButtonPress, ButtonRelease);
- **focus** - чи має вікно фокус (Enter, Leave);
- **height** і **width –** ширина та висота (Configure, Expose);
- **keycode** – код натиснитої клавіші (KeyPress, KeyRelease);
- **state** – стан події (для ButtonPress, ButtonRelease, Enter,KeyPress, KeyRelease, Leave, Motion - у вигляді числа; для Visibility - у вигляді рядка);
- **time** – час настяння події (всі події);
- **x** та **y** – коодинати миші;
- **x_root** та **y_root** - координати миші на екрані (ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion);
- **char –** набраний на клавіатурі символ (KeyPress, KeyRelease);
- **send_event** – див. Документація X/Windows;
- **keysym** – набраний на клавіатурі символ (KeyPress, KeyRelease);
- **keysym_num** - набраний на клавіатурі символ у вигляді числа (KeyPress, KeyRelease);
- **type** – тип події у вигляді числа (всі події);
- **widget** – віджет, який отримав поді. (всі події);
- **delta** – зміна при обертанні колеса миші (MouseWheel);

Ця функція може повертати рядки "continue" та "break". Якщо функція повертає "continue", то Tkinter продовжить обробку інших прив'язок цієї події, якщо "break" - обробка цієї події припиняється. Якщо функція нічого не повертає (якщо повертає None), то обробка подій продовжується (тобто це еквівалентно поверненню "continue").

Назви подій

Існують три форми назви подій. Найпростіший випадок це символ ASCII. Так описуються події натискання клавіш на клавіатурі:

```
widget.bind("z", callback)
```

callback викликається кожного разу, коли буде натиснуто клавішу "z".

Другий спосіб довший, але дозволяє описати більше подій. Він має наступний синтаксис:

```
<modifier-modifier-type-detail>
```

Назва події укладено у кутові дужки. Усередині є нуль або більше модифікаторів, тип події та додаткова інформація (номер натиснутої клавіші миші або символ клавіатури). Поля поділяються дефісом або пробілом. Приклад (прив'язуємо одночасне натискання Ctrl+Shift+q):

```
widget.bind("<Control-Shift-KeyPress-q>", callback)
```

2.4.4 Зображення

Конструктор класу приймає такі аргументи:

- **background** і **foreground** – кольори фону та попереднього плану для зображення. Оскільки зображення двокольорове, ці параметри визначають відпоідно чорний і білий колір;
- **file** і **maskfile** – шляхи до файлу із зображенням і масці (зображенню, що вказує які пікселі будуть прозорими);
- **data** та **maskdata** – замість шляху до файлу можна вказати вже завантажені в пам'ять дані зображення. Ця можливість зручна для вбудовування зображення у програму.

```
from Tkinter import *
data = '''#define image_width 15
#define image_height 15
static unsigned char image_bits[] = {
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x38, 0x1c, 0x30, 0x0c, 0x60, 0x06,
  0x60, 0x06, 0xc0, 0x03, 0xc0, 0x03, 0x60, 0x06, 0x60, 0x06, 0x30, 0x0c,
  0x38, 0x1c, 0x00, 0x00, 0x00, 0x00 };'''
root=Tk()
image = BitmapImage(data=data, background='red', foreground='green')
button=Button(root, image=image)
button.pack()
root.mainloop()
```

PhotoImage

PhotoImage дозволяє використовувати повнокольорове зображення. Крім того, у цього класу є кілька (досить примітивних) методів для роботи із зображеннями. PhotoImage гарантовано розуміє формат GIF.

Аргументи конструктора:

- **file**– шляхи до файлу із зображенням;
- **data** - замість шляху до файлу можна вказати вже завантажені на згадку дані зображення. Зображення у форматі GIF можуть бути закодовані за

допомогою base64. Ця можливість зручна для вбудовування зображення у програму.

- **format** – явна вказівка формату зображення;
- **width, height** – ширина та висота зображення;
- **gamma** – корекція гами;
- **palette** – зображення палітри;

Combobox

```
import tkinter as tk
import tkinter.ttk as ttk
root = tk.Tk()
frame = tk.Frame(root)
frame.grid()
combobox = ttk.Combobox(frame,values = [u"ОДИН",u"ДВА",u"ТРИ"],height=3)
#frame - задає батьківський віджет, на його території розташовуватиметься Combobox
#values - задає набір значень, які будуть у Combobox спочатку
#height - задає висоту списку, що випадає. Якщо кількість елементів списку менше 11, можна не ставити.
#Якщо не задано при кількості елементів більше 10, то праворуч з'явиться смуга прокручування.
#Якщо в нашому прикладі задати значення height менше трьох, то з правого боку з'явиться смуга прокручування,
#Але вона буде недоступна, а всі елементи будуть відображатися одночасно.
combobox.set(u"ОДИН")# за допомогою цього рядка ми встановимо Combobox у значення ОДИН спочатку
combobox.grid(column=0,row=0)# Позиціонуємо Combobox на формі
root.mainloop()
```

Progressbar

Віджет відображає рівень завантаження.

- **length –** довжина смуги.

Start

Запуск нескінченного циклу завантаження. Крок довжиною 1 виконується один раз у вказаний час (у мілісекундах).

Stop

Зупиняє цикл завантаження.

Step

Просуває завантаження на задану кількість кроків.

Висновки до розділу 2

У даному розділі ми детально розписали алгоритмічний Метод Віоли-Джонса. Присутній детальний опис бібліотеки (OPENCV) та дано алгоритми для роботи з бібліотекою. Описана бібліотека для графічного інтерфейсу (Tkinter).

Присутній детальний опис з поясненням технологічного забезпечення який буде кінцевим для створення даної платформи.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Використання розробленного сервісу

Для того щоб запустити програму потрібно ввести команду: python window.py

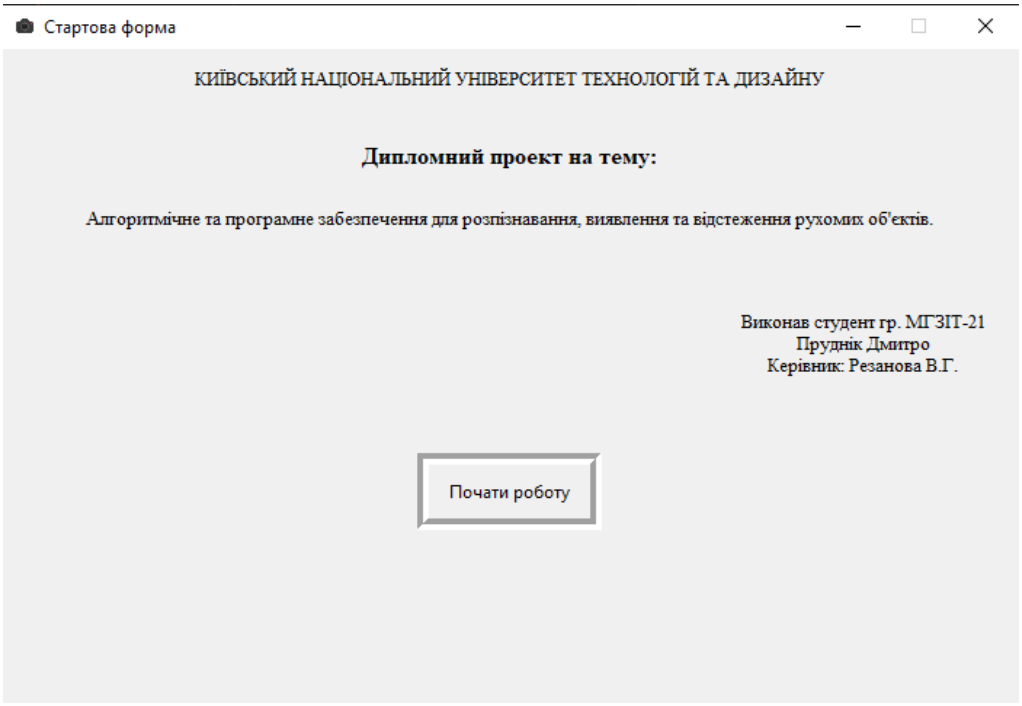


Рис 3.1 Стартова форма

Рис 3.2 Mask

Рис 3.3 Frame

Рис 3.4 Основний дисплей

Для закриття програми натисніть клавішу e (англ.) на клавіатурі

3.2 У яких сферах застосовується технологія комп'ютерного зору?

Охорона здоров'я та медицина

Технологія комп'ютерного зору використовується в медичній промисловості для виявлення онкозахворювань. А саме вона дозволяє виявляти найменші відмінності між зображеннями злоякісних і доброякісних пухлин, а також діагностувати дані сканування магнітно-резонансної томографії (МРТ) та об'єктів на зображеннях.

Аналіз руху

Деякі неврологічні та захворювання опорно-рухового апарату можуть бути виявлені за допомогою глибоких моделей навчання та комп'ютерного зору навіть без аналізу з боку лікаря. Комп'ютерний зір аналізує рухи тіла пацієнта та допомагає лікарям з більшою точністю діагностувати захворювання.

Виявлення різних хвороб

Пошкодження мозку можна побачити за допомогою результатів МРТ. Також вони часто виявляються за допомогою глибоких нейронних мереж. Програмне забезпечення виявлення пошкоджень, що використовує глибоке

навчання, має вирішальне значення для медичної індустрії, оскільки воно може виявляти пошкодження з високою точністю, що допомагає лікарям поставити діагноз. При цьому постійно розробляються нові методи підвищення точності діапазонів.

Сегментація зображень за сканованими зображеннями

Потенціал комп'ютерного зору охорони здоров'я величезний, а можливості його застосування незліченні. Медична діагностика багато в чому ґрунтується на вивченні різних зображень, сканів та фотографій. Аналіз ультразвукових зображень, МРТ та КТ – невід'ємна частина стандартного набору сучасної медицини. Очікується, що технології комп'ютерного зору не лише спростять ці процеси, а й навчаться запобігати помилковим діагнозам та знижувати витрати на лікування. Варто уточнити, що призначення компютерного зору полягає не в заміні медичних фахівців, а в полегшенні їх роботи та підтримці прийняття рішень.

2. Сільське господарство та землеробство

Виявлення пошкодженої продукції

За допомогою алгоритмів машинного навчання можна виявити пошкоджену продукцію у процесі її обробки. Алгоритми навчаються на безлічі даних і вони здатні розрізняти стиглі та зіпсовані продукти на основі їх характерних відмінностей.

Підрахунок продукції

Через систему обробки зображень можуть проходити маси зображень для швидкого і точного підрахунку кількості об’єктів. Це дозволяє фермерам мати точні дані про обсяг продукції, що вирощується, і ця інформація дозволяє їм розрахувати оптимальну вартість продукції і ефективніше планувати роботи.

Розпізнавання рослин

За допомогою додатків, що використовуються технологію компютерного зору, фермери можуть легко визначати бур’яни та інші

шкідники. Подібні програми легко ідентифікуються рослини та тварини на рівні виду по фотографії.

Моніторинг тварин

За тваринами можна спостерігати за допомогою нових методів, які були навчені визначати тип тварини та її дії. Віддалений моніторинг домашніх тварин може бути корисним у сільському господарстві для виявлення захворювань, зміни поведінки або, наприклад, пологів. Також вчені, які займаються вивченням дикої природи, можуть безпечно спостерігати диких тварин на відстані.

Автоматизація ферм

Роботи для збирання врожаю та інших сільськогосподарських робіт, автономні трактори та дрони для моніторингу умов на фермі та розповсюдження добрив можуть підтримувати і навіть збільшити кількість врожаю за нестачі робочої сили. Сільське господарство також може бути прибутковішим, якщо мінімізувати міру впливу людини на навколишнє середовище — так званий екологічний слід. Сучасні технології полегшують збирання різних значень, параметрів та статистичних даних, які можна відстежувати автоматично. Наприклад, дрони та віддалені датчики збирають великі обсяги даних, які потім обробляються. З їх допомогою фермерські господарства можуть постійно стежити за всіма умовами: станом грунту, рівнем зрошення, температурою та іншими погодними умовами, здоров'ям рослин. Далі алгоритми машинного навчання аналізують отримані дані та ферма може використовувати цю інформацію для реагування на потенційні проблемні місця на їхній ранній стадії та ефективного розподілу існуючих ресурсів.

Сучасні технології полегшують збирання різних значень, параметрів та статистичних даних, які можна відстежувати автоматично. Наприклад, дрони та віддалені датчики збирають великі обсяги даних, які потім обробляються. З їх допомогою фермерські господарства можуть постійно стежити за всіма умовами: станом грунту, рівнем зрошення, температурою та іншими погодними умовами, здоров'ям рослин. Далі алгоритми машинного навчання аналізують отримані дані та ферма може використовувати цю інформацію для реагування на потенційні проблемні місця на їхній ранній стадії та ефективного розподілу існуючих ресурсів.

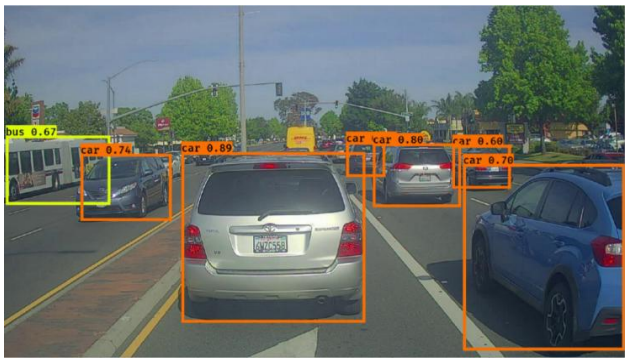
3.Транспорт

Розпізнавання номерних знаків

Автоматичне розпізнавання символів на зображеннях номерних знаків реалізує можливість отримання даних про місцезнаходження транспортного засобу. Цю технологію можна використовувати в камерах на дорогах для відстеження порушень, у системі електронного збору плати за проїзд або в інших правоохоронних цілях, наприклад, у кримінальних розслідуваннях.

Виявлення мертвого кута

Система бокового сканування здатна розпізнавати велосипедистів, транспортні засоби або людей, які знаходяться в небезпечній близькості від об'єкта, та сигналізувати про це за допомогою світлодіода або попереджати звуковим сигналом.



Цікавий кейс: розпізнавання та класифікація об'єктів у дорожньому русі. Безпілотний автомобіль, що є яскравим втіленням ідеї автономного руху, є прикладом використання штучного інтелекту. У нього упаковано кілька завдань машинного навчання, і комп'ютерний зір є важливою частиною їх вирішення.

Алгоритм, що керує безпілотним автомобілем, повинен постійно отримувати інформацію про навколишнє оточення. Алгоритму необхідно знати, як проходить дорога, де поблизу знаходяться інші об'єкти і з якою швидкістю вони рухаються, як далеко до потенційних перешкод і багато іншого, щоб постійно адаптуватися до умов, що постійно змінюються. З цією метою автономні транспортні засоби оснащені великими камерами з усіх

боків, які знімають навколо. Алгоритм обробляє безперервний потік зображень та класифікує дані.

Розпізнавання та класифікація об'єктів у дорожньому русі

Нетривіальність завдання полягає у складності та непостійності дорожнього руху, тому алгоритм необхідно навчати таким чином, щоб виключити ймовірність збоїв у складних виняткових ситуаціях. З цього випливає така складність - потреба у великих обсягах навчальних даних, отримання яких в умовах дорожнього руху пов'язане з високими витратами.

4. Роздрібна торгівля

У секторі роздрібної торгівлі комп'ютерний зір дозволяє покращити процес здійснення покупок, уникнути втрати чи крадіжки товарів та своєчасно виявляти нестачу товарів на полицях. Комп'ютерний зір також використовується для прискорення процесу оплати - на касах самообслуговування або в поєднанні з машинним навчанням для автоматичного виставлення рахунків, що дозволяє уникнути процесу оплати в магазині.

Відстеження поведінки відвідувачів

Великі глобальні інтернет магазини, такі як Amazon, вже давно можуть скористатися аналітичними можливостями своєї цифрової платформи. Поведінку клієнтів можна детально проаналізувати, а досвід користувача можна оптимізувати і поліпшити.

Алгоритми можуть оцінювати відеоматеріал, знятий за допомогою звичних камер спостереження та вивчати поведінку покупців у магазині. Для маркетингу є цікавою можливість аналізу поведінки окремих споживачів, наприклад, обраний маршрут магазином та окремим відділам. Це дозволяє поліпшити методику викладення товару, набори товарів, розташування і структуру відділів, що продаються, уникнути надмірного завантаження популярних відділів, словом, оптимізувати маркетингові складові і покращити загальний користувальницький досвід відвідувачів магазину.

5. Виробництво

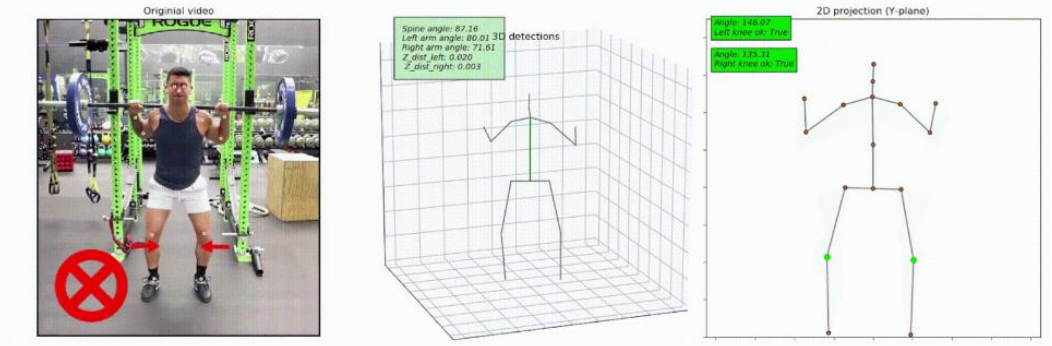
Виробничі компанії можуть використовувати комп'ютерне зір виявлення дефектів продукції режимі реального часу. Коли продукція залишає виробничу лінію, комп'ютер обробляє зображення чи відео і може виявити різні дефекти навіть у найменших компонентах.

6. Фітнес

Оцінка пози людини

Індустрія фітнесу останніми роками перебуває у активному процесі цифрової трансформації. Прогрес тренувань відстежується та оцінюється за допомогою додатків, а віртуальні та домашні тренування стали користуватися особливою популярністю під час кризи пандемії. Зокрема, під час силових тренувань особливо важливо дотримуватись певного положення тіла, щоб уникнути ризику отримання травми. Зараз це стало можливим без спостереження фітнес-тренера — комп'ютерний зір дозволяє оцінювати та оцінювати відеоматеріал точніше, ніж людське око в цій галузі, відбувається це за допомогою перевірки постави та положення тіла за допомогою відео. І тому визначається становище суглобів та його становище стосовно друг до друга. Алгоритм знає, як має виглядати ідеальне та безпечне виконання фітнес-вправи, і відхилення від нього можна автоматично розпізнавати та виділяти. Таким чином, програма своєчасно попереджає про небезпечні помилки, що робить тренування без спостереження тренера безпечнішими.

Аналіз становища та рухів тіла в режимі реального часу за допомогою комп'ютерного зору



Висновки до розділу 3

У розділі 3 ми описали в яких сферах використовується комп’ютерний зір. У кожній із галузей докладаються великі зусилля, щоб зробити існуючі процеси ефективнішими за допомогою технологій комп’ютерного зору. OpenCV надає можливість використання серії алгоритмів для розпізнавання і відстеження переміщення особи, тіла і жестів, автоматизації відеоспостереження автоматичних систем допомоги водієві, реконструкції об’єктів та сцен, доповненої реальності, візуального огляду, роботехніки та багатьох інших застосувань. Ми написали алгоритм для розпізнавання, виявлення, та відстеження рухомих об’єктів, в нашому прикладі це автомобілі. Поданий приклад це лише програма на початку розробки, в наш час докладаються великі зусилля для покращення якості цих алгоритмів.

Висновки

За результатами виконання дипломної магістерської роботи отримано наступні результати:

Проведений огляд основних компонентів побудови сервісу для розпізнавання, виявлення та відстеження рухомих об’єктів. Для цього було зроблено аналіз існуючих систем та пошук літературних джерел, які допомагають у створенні таких систем, що дозволило визначити потрібні компоненти для працездатності системи та уникнути недоліків існуючих систем.

Згодом було визначено основне призначення сервісу разом з усіма необхідними компонентами системи. Вибрано технічне забезпечення для створення програми, яке основане на дослідженнях в першому розділі.

Поданий детальний покроковий опис розробки системи результатом якого є сама система.

Також визначено вимоги до рівня підготовки користувача, що допомагає зрозуміти потенційних користувачів програми, а також визначено вимоги до апаратного та програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абламейко С.В., Лагуновський Д.М. Обробка зображень: технологія, методи застосування. Навчальний посібник. М: Амалфея, 2000. 18-44 стор.

2. Браммер К., Зіффлінг Г. Фільтр Калмана-Бьюсі. Детерміноване спостереження та стохастична фільтрація. 1982 : Наука. Головна редакціяфізико-математичної літератури.

3. Власенко С. Системи АЛС для високошвидкісного сполучення // Автоматика зв'язок інформатики. 011. No. 3. P. 39.

4. Іванов Ю.А. Розробка локомотивної системи технічного зору. Дисертація на здобуття наукового ступеня кандидата технічних наук. М.: ФДБОУ ВПО "Київський авіаційний інститут" (MAI), 2014.

5. Рязанов С.М. Транспортна безпека об'єктів залізничної інфраструктури / / Автоматика, зв'язок, інформатика. 2011. Вип. 6. Стор. 23-25.

6. Труфанов М.І., С.В. П. Спосіб виявлення перешкод перед транспортним засобом із використанням бінокулярної системи технічного зору.

7. Усилін С.А. Алгоритмічний розвиток Віола-Джонсівських детекторів для розв'язання прикладних завдань розпізнавання зображень. М. 2017.

8. Ballard D.H. Generalizing the Hough transform to detect arbitrary shapes // Pattern Recognition 13, 2, April 1981.

9. Bileschi S.M. StreetScenes: Towards scene understanding in still images. PhD thesis. Citeseer. 2006.

10. Canny J. Computational Approach To Edge Detection, IEEE Trans // Pattern Analysis and Machine Intelligence, No. 8 (6), 1986. pp. 679-698.

11. Filipowicz A. Driving School II. Video Games for Autonomous Driving. 2016.

12. Geiger A., Lenz P., Urtasun R. Чи є ready for autonomous driving? the kitt vision benchmark suite. // Conference on Computer Vision and Pattern Recognition (CVPR).

13. Hough P. Метод і методи для recognizing complex patterns. U.S. Patent 3,069,654. December 18, 1962.

14. Kalman R.E. Нові приклади до linear filtering and prediction problems // Journal of Basic Engineering 82 (1), 1960. pp. 35-45.

15. Karakose M., Yaman O., Baygin M., Murat K., Akin E. A New Computer Vision Based Method for Rail Track Detection and Fault Diagnosis in Railways // International Journal of Mechanical Engineering and Robotics Research, Vol. 6, No. 1, January 2017. pp. 22-27.

16. Karla Brkie. An overview of traffic sign detection methods. Zagreb: Department of Electronics, Microelectronics, Computer and Intelligent Systems. Faculty of Electrical Engineering and Computing.

17. LeCun Y., Boser B., Denker J.S., Henderson D., Howard R.E., Hubbard W., Jackel L.D. Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4):541-551, Winter 1989.

18. Ліенхарт Р., Куранов А., Пісаревскій V. Empirical Analysis of Detection Cascades Boosted Classifiers для Rapid Object Detection With an Extended Set of Haar-like Features – Intel Technical Report. 2002.

19. Nadra Ben Romdhane, Hazar Mliki, Mohamed Hammami. A new approach to traffic sign recognition через primary visual characteristics // Computer and Information Science (ICIS) 2016 IEEE/ACIS 15th International Conference on 2016.pp. 1-6.

20. Puttemans S. OpenCV Tutorials. Object Detection. Cascade Classifier Training. https://docs.opencv.org/3.3.0/dc/d88/tutorial_traincascade.html.

21. Ruta A., Li Y., Liu X. Real-time traffic sign recognition from video by class-specific discriminative features // Pattern recognition, Vol. 43, No. 1, 2010. pp. 416- 430.

22. Sobel I., Feldman G. A 3x3 Isotropic Gradient Operator для Image Processing (Неопубл.), 1968.

23. Viola P., Jones M. Robust real-time object detection // International Journal of Computer Vision, 2001.

24. NVidia Driver Assistance. http://www.nvidia.com/object/advanced-driver assistance-systems.html.

25. Train Simulator http://store.steampowered.com/app/24010/Train_Simulator/

26. Vector. https://vector.com/vi_adas_en.html.

```
from tkinter import *
#from child_window import ChildWindow
import cv2
from tracker import *
import math

def tracker():
    cap = cv2.VideoCapture("bombay traffic.mp4")
    tracker = DistTracker()

    object_detector = cv2.createBackgroundSubtractorMOG2(history=100, varThreshold=40)

    while True:
        ret, main = cap.read()
        height, width, _ = main.shape

        frame = main[340: 720, 500: 800]

        mask = object_detector.apply(frame)
        _, mask = cv2.threshold(mask, 254, 255, cv2.THRESH_BINARY)
        contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        detections = []
        for contour in contours:
            area = cv2.contourArea(contour)
            if area > 100:
                x, y, w, h = cv2.boundingRect(contour)

                detections.append([x, y, w, h])

        boxes_ids = tracker.update(detections)
        for box_id in boxes_ids:
            w, y, w, h, id = box_id
            cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 3)

        cv2.imshow('mask', mask)
        cv2.imshow('main', main)
        cv2.imshow('frame', frame)

        key = cv2.waitKey(30)
        if key & 0xFF == ord('e'):
            break
```

```
    cap.release()
    cv2.destroyAllWindows()


class Window:
    def __init__(self, width, height, title="Стартова форма", resizable=(False, False), icon="camera.ico"):
        self.root = Tk()
        self.root.title(title)
        self.root.geometry(f"{width}x{height}+600+300")
        self.root.resizable(resizable[0], resizable[1])
        if icon:
            self.root.iconbitmap(icon)


    def draw_widgets(self):
        self.label1 = Label(self.root, text="КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ", font=("Times New Roman", 10, "normal")).pack(pady=10)
        self.label2 = Label(self.root, text="Дипломний проект на тему:", font=("Times New Roman", 12, "bold")).pack(pady=20)
        self.label3 = Label(self.root, text="Алгоритмічне та програмне забезпечення для розпізнавання, виявлення та відстеження рухомих об'єктів.",font=("Times New Roman", 10, "normal")).pack()

        self.label4 = Label(self.root, text="Виконав студент гр. МГЗІТ-21\nПруднік Дмитро\nКерівник: Резанова В.Г.", font=("Times New Roman", 10, "normal")).pack(anchor=E, padx=20, pady=50)

        self.button = Button(self.root, width=15, height=2, text="Почати роботу", relief=GROOVE, bd=8, command=tracker).pack()


    def run(self):
        self.draw_widgets()
        self.root.mainloop()


    #def create_child(self, width, height, title="Camera", resizable=(False, False), icon=None):
        #ChildWindow(self.root, width, height, title, resizable, icon)
if __name__ == "__main__":
    window = Window(700, 450, "Стартова форма", )
    #window.create_child(600, 300, title="Camera", icon="camera.ico")
    window.run()
```

```
class DistTracker:
    def __init__(self):
        self.center_points = {}
        self.id_count = 0

    def update(self, object_rect):
        objects_bbs_ids = []

        for rect in object_rect:
            x, y, w, h = rect
            cx = (x+x+w) // 2
            cy = (y+y+h) // 2

            same_object_detected = False
            for id, pt in self.center_points.items():
                dist = math.hypot(cx - pt[0], cy - pt[1])

                if dist < 25:
                    self.center_points[id] = (cx, cy)
                    print(self.center_points)
                    objects_bbs_ids.append([x, y, w, h, id])
                    same_object_detected = True
                    break

            if same_object_detected is False:
                self.center_points[self.id_count] = (cx, cy)
                objects_bbs_ids.append([x, y, w, h, self.id_count])
                self.id_count += 1

        new_center_points = {}

        for obj_bb_id in objects_bbs_ids:
            _, _, _, _, object_id = obj_bb_id
            center = self.center_points[object_id]
            new_center_points[object_id] = center

        self.center_points = new_center_points.copy()
        return objects_bbs_ids
```

Додаток Б

1. Стаття

2. Ивченко Г.И., Медведев Ю.И. Введение в математическую статистику. М.: Изд-во ЛКИ, 2010. —600 с.

3. Осипов Д. Л. Delphi. Программирование для Windows, OS X, iOS и Android // СПб.: БХВ-Петербург, - 2014. – 464 с.

РЕЗАНОВА В.Г., ПРУДНІК Д.О.

ДОСЛІДЖЕННЯ ТА РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕВІРКИ АДЕКВАТНОСТІ МАТЕМАТИЧНОЇ МОДЕЛІ УТВОРЕННЯ МІКРОФІБРИЛЯРНИХ СТРУКТУР

REZANOVA V.G., PRUDNIK D.O.

RESEARCH AND DEVELOPMENT OF SOFTWARE FOR VERIFICATION OF ADEQUACY OF MATHEMATICAL MODEL OF FORMATION OF MICROFIBRIBRAL STRUCTURES

Purpose and tasks. The purpose of the work is to create software for checking the adequacy of regression mathematical models for the study of three-component polymer mixtures for the implementation of the process of specific fiber formation

The task is to study the adequacy of the model by the method of checking the proper linear hypothesis. Software development in C ++ language in the Borland Builder environment.

Object and subject of research. Object of research - specific fiber formation. It is realized under appropriate conditions under the flow of molten polymer mixtures. It is based on micro-regional processes - such as the deformation of the droplets of the disperse phase component and the combining of liquid jets in the direction of flow.

Subject of research - the process of automated verification of the adequacy of the model.

The adequacy of the mathematical model of formation of microfibrillar structures is checked by checking the corresponding linear hypotheses. The model turned out to be adequate, which gives grounds for its use in further research, in particular - for predicting the behavior of the system, as well as for optimizing its parameters. In addition, the created software can be applied to a wider class of tasks.

Вступ

Світовий досвід свідчить, що раціональним рішенням проблеми створення нових матеріалів із унікальними характеристиками є змішування полімерів.

Утворення мікрофібрилярних структур реалізується у відповідних умовах при течії розплавів сумішей полімерів. Результати авторів з дослідження цього процесу відносяться до питань математичного моделювання поведінки кількісних характеристик специфічного волокноутворення, що надає можливість описувати вже існуючі закономірності, а також прогнозувати поведінку процесу на інших (ще не досліджених) сумішах полімерів. Зауважимо, що практичне використання математичної моделі можливе лише після перевірки її адекватності

Постановка завдання

Об'єктом дослідження є багатокомпонентна сумішева система, яка складається із двох полімерів (волокноутворюючий та матричний) та добавки (добавок). Сумарний вміст компонентів суміші дорівнює одиниці. Контроль якості отриманого полімерного композиту відбувається за вихідними показниками якості.

Задача математичного моделювання в даній роботі полягає у встановленні залежності між однією групою змінних (незалежних змінних, факторів) та іншою групою змінних (залежних змінних, функцій відгуку). Оцінюємо параметри обраної моделі, перевіряємо її адекватність і, у випадку позитивного вирішення останнього питання, робимо висновок про можливість застосування побудованої моделі до тих питань, для яких вона і була побудована.

Основна частина

Відповідно до методики планування експерименту із сумішами використовуємо симплексно-гратковий план. Останній забезпечує рівномірний розкид експериментальних точок на області, що являє собою симплекс відповідної розмірності (для трикомпонентної суміші це правильний трикутник на площині). Для побудови моделі обираємо неповний кубічний поліном, оскільки дані літератури свідчать, що такі функції досить якісно описують поведінку трикомпонентних сумішевих систем. Оскільки на вміст компонентів суміші умовами задачі накладаються певні обмеження, на повному симплексі програмним чином виділяємо підобласть, що відповідає цим обмеженням, а потім всередині виділеної підобласті обираємо область, «подібну» вихідному симплексу, тобто трикутник (хоча і не обов'язково правильний). Відповідно до симплексно-граткового підходу для неповної кубічної моделі, для побудованого трансформованого симплексу маємо сім точок плану експерименту. Для забезпечення можливості в подальшому використовувати методи регресійного аналізу моделі, зокрема – перевірки її адекватності, додаємо до плану ще одну точку.

Використовуємо метод перевірки адекватності лінійної моделі, що полягає в порівнянні оцінок дисперсій похибок, що одержані, з одного боку, з застосуванням даної моделі, а з іншого — незалежним шляхом. Це є еквівалентним перевірці деякої лінійної гіпотези за допомогою обчислення і аналізу відповідного F-відношення Фішера.

Нехай $x_1, x_2, \ldots, x_m$ — різні точки спостережень (вектори-рядки), причому хоч в одній з них кількість спостережень більше ніж 1. Зазначена F-статистика має вигляд $F = \frac{S_1^2}{S_2^2}$ де $S_1^2 = \frac{1}{m-p}\sum_{i=1}^{m} n_i(\hat{y}_i - \bar{y}_i)^2$,

$S_2^2 = \frac{1}{n-m}\sum_{i=1}^{m}\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_i)^2$ $y_{i1}, \ldots, y_{in_i}$, $i = 1, \ldots, m$ – значення вихідної змінної, що спостерігались в точці $x = x^i$; n_i - кількість дослідів i-й точці. Якщо $m > p$, то відношення вигляду $\frac{S_1^2}{S_2^2}$ (варіант з сукупності F- відношень) має розподіл Фішера $F(m-p, n-m)$ [4]. Згідно з загальними положеннями [4] гіпотеза про адекватність моделі $\hat{y}$ не приймається при рівні значущості α, якщо вказане відношення перевищує квантиль рівня $1 - \alpha$ вказаного розподілу. Зазначені дії реалізуються розробленим програмним забезпеченням.

Знаходимо F-відношення для моделі. Для всіх вихідних змінних моделі за допомогою створеного програмного додатку отримуємо значення, що наведено на рис. 1.

а) б) в)

Рисунок. 1- Відношення $\frac{S_1^2}{S_2^2}$, отримане в програмному додатку для y_1 (а), y_2 (б) та y_3 (в)

Далі приймаємо рішення з приводу гіпотези про адекватність нашої математичної моделі. У даному випадку маємо: для $\alpha = 0.01$ $F(m-p, n-m) = F(8-7, 24-8) = F(1, 16)=8.531$. Бачимо, що для всіх y з тестовоїмоделі розраховане відношення $\frac{S_1^2}{S_2^2}$ менше за значення $F(m-p, n-m)$. Отже, гіпотеза про адекватність моделі може бути прийнята.

Висновки

Здійснено перевірку адекватності математичної моделі утворення мікрофібрилярних структур шляхом перевірки відповідних лінійних гіпотез. Модель виявилась адекватною, що дає підстави для її використання у подальших дослідженнях, зокрема – для прогнозування поведінки системи, а також для оптимізації її параметрів. Крім того, створене програмне забезпечення може бути застосоване до більш широкого класу задач.

Ключові слова: програмне забезпечення, математична модель, адекватність.

Література

1. Резанова В.Г., Резанова Н.М. Програмне забезпечення для дослідження полімерних систем. Монографія. – К.: АртЕк, 2020. – 358 с.
2. Rezanova N.M., Rezanova V.G., Plavan V.P., Viltsaniuk O.O. The influence of nano-additives on the formation of matrix-fibrillar structure in the polymer mixture melts and on the properties of complex threads // Vlákna a textil (Bratislava, Slovak Republic) - №2, 2017. - p. 37-42
3. Резанова В.Г. Програмне забезпечення для математичного моделювання специфічного волокноутворення // Інформаційні технології в науці, виробництві та підприємництві. Збірник наукових праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектування. – К. : Освіта України, 2017
4. Сидняев Н. Теория планирования эксперимента и анализ статистических данных. – М.: Юрайт, 2012, 400 с.
5. Stroustrup B. Programming: Principles and Practice Using C++ (2nd Edition). Addison-Wesley Professional, 2014. – 1312 p.
6. Мейерс С. Эффективный и современный C++. М.: Вильямс, 2016. - 304 с.

РЕЗАНОВА В.Г.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ДОСЛІДЖЕННЯ ВПЛИВУ ТЕХНОЛОГІЧНИХ ПАРАМЕТРІВ НА ФОРМУВАННЯ МІКРОФІБРИЛЯРНИХ СТРУКТУР

REZANOVA V.G.

A SOFTWARE DEVELOPMENT TO INVESTIGATE THE INFLUENCE OF TECHNOLOGICAL PARAMETERS ON THE FORMATION OF MICROFIBRIBRAL STRUCTURES

An important task for science today is to conduct theoretical and experimental research that opens up fundamentally new ways to obtain materials with specified properties and the creation and introduction into industry of new waste-free environmentally friendly low-energy technologies.

The aim is to study the mechanisms of processes and phenomena observed in the processing of melts of polymer mixtures is important and relevant and is subject to further study. Investigation of the influence of technological parameters on the formation of microfibrillar structures.

Thus, software has been developed that optimizes the composition of the three-component polymer mixture, which makes it possible to implement formation of microfibrillar structures in the best way.

Вступ

Важливим завданням для науки на сьогодні є проведення теоретичних та експериментальних досліджень, що відкривають

2.Тези доповіді на конференції

Резанова В. Г., Опаленик В.В., Прудник Д.С. Програмне забезпечення для визначення впливу добавок на властивості мікроволокон // Мехатронні системи: інновації та інжиніринг : тези доповідей VI Міжнародної науково-практичної конференції, 24 листопада 2022 р., м. Київ : КНУТД, 2022

Ї