

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра системного аналізу та обчислювальної математики

(повне найменування кафедри)

Пояснювальна записка

до дипломної роботи

бакалавра

(ступінь вищої освіти)

на тему Побудова цифрової моделі рослини

(назва теми)

Виконав: студент 4 курсу, групи 810сп

Спеціальності 124 Системний аналіз

(код і найменування спеціальності)

Освітня програма (спеціалізація)

«Інтелектуальні технології та прийняття рішень
в складних системах»

Віталій ПЕРЕТЯТИЙ

(ПРИЗВИЩЕ та ініціали)

Керівник Еліна ТЕРЕЩЕНКО

(ПРИЗВИЩЕ та ініціали)

Рецензент Наталя МИРОНОВА

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут, факультет Факультет комп'ютерних наук і технологій
Кафедра Кафедра системного аналізу та обчислювальної математики
Ступінь вищої освіти бакалавр
Спеціальність 124 Системний аналіз
(код і найменування)
Освітня програма (спеціалізація) «Інтелектуальні технології та прийняття рішень в складних системах»
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
Еліна ТЕРЕЩЕНКО
«09» червня 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТА

Перетятому Віталію Олександровичу

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема роботи Побудова цифрової моделі рослини

керівник роботи Терещенко Еліна Валентинівна

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «18» квітня 2023 року № 121

2. Строк подання студентом роботи 05.06.2023

3. Вихідні дані до роботи масив фотоматеріалів, наданий лабораторією генетичних досліджень Інституту олійних культур НААН України

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Задачами дослідження є аналіз існуючих методів та засобів 3D моделювання; розробка етапів 3D моделювання рослини; створення бази для навчання; створення моделі для розпізнавання рослини; створення мобільного застосунку для 3D моделювання; створення остаточної 3D модель рослини.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація на 15 слайдів

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Прийняв виконане завдання
Розділ 1, Розділ 2	Терещенко Е.В., в.о. завідувача кафедри	01 березня 2023	05 червня 2023
Нормоконтроль	Ширококоряд Д.В., ст. викладач	07 червня 2023	

7. Дата видачі завдання «01» березня 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Розробка етапів 3D моделювання рослини.	2 тиждень	Розділ 1
4	Створення бази для навчання.	2 тиждень	Розділ 2
5	Створення моделі для розпізнавання рослини.	3 тиждень	Розділ 2
6	Розробка мобільного застосунку для 3D моделювання.	4 тиждень	Розділ 2
7	Створення застосунку для побудови 3D моделі.	5 тиждень	Розділ 2
8	Оформлення пояснювальної записки та відповідної документації.	6-7 тиждень	Додатки
9	Нормоконтроль та рецензування.	8 тиждень	
10	Захист дипломної роботи.	9 тиждень	

Студент

(підпис)

Перетятый В.О.

(прізвище та ініціали)

Керівник роботи

(підпис)

Терещенко Е.В.

(прізвище та ініціали)

РЕФЕРАТ

Дипломна робота: 62 с., 23 ілюстрації, 1 додаток, 14 джерел.

Об'єкт дослідження – процес 3D моделювання рослин.

Предмет дослідження – практична реалізація системи 3D моделювання зі застосуванням мобільного застосунку.

Мета роботи – створення 3D моделі рослин.

Методи дослідження – методи Deep Learning, методи створення Android додатків.

Актуальність проблеми – застосування цифрових моделей рослин для розвитку агротехнологій, наукових дослідженнях біології та сільського господарства, при аналізі розвитку та функціонування рослин.

Задачі дослідження: аналіз існуючих методів та засобів 3D моделювання; розробка етапів 3D моделювання рослини; створення бази для навчання; створення моделі для розпізнавання рослини; створення мобільного застосунку для 3D моделювання; створення остаточної 3D моделі рослини.

Результати роботи та їх новизна – згенеровано 3D модель рослини, розроблено послідовність етапів 3D моделювання рослин, створено мобільний застосунок для отримання 3D моделі різних об'єктів, включаючи рослини.

Основні висновки – результати роботи є практично корисними, розроблена програма дозволяє створювати 3D моделі довільних об'єктів, включаючи рослини.

Мобільний застосунок може бути використано в сільському господарстві, наукових дослідженнях, ландшафтному дизайні.

Робота виконана в рамках договору про співробітництво між Інститутом олійних культур НААН України та Національного університету «Запорізька політехніка».

ЦИФРОВА МОДЕЛЬ, МОБІЛЬНИЙ ЗАСТОСУНОК, 3D, PYTHON, KIVY, OPENCV, ASPOSE, СОНЯШНИК.

ЗМІСТ

Завдання	2
Реферат	4
Вступ.....	6
1 Теоретична частина.....	8
1.1 Проблема дослідження рослин	8
1.2 Етапи створення цифрової моделі	9
1.3 Розгляд технічних засобів.....	11
1.3.1 Python – мультиплатформена мова програмування	11
1.3.2 Використання Kivu для розробки мобільного додатку	12
1.3.3 Ubuntu, як окрема операційна система	14
1.3.4 Безкоштовне хмарне середовище для виконання коду Python	15
1.3.5 Buidozer, як засіб для створення мобільних застосунків	17
1.3.6 Використання OpenCV для аналізу фотозображень.....	18
1.3.7 Roboflow, створення моделей машинного навчання	20
1.3.8 Aspose – бібліотека для роботи з різними типами файлів	23
1.4 Методи штучного інтелекту при аналізі зображень	27
1.5 Актуальність проблеми побудови цифрової моделі рослини	29
1.6 Існуючі застосунки для розробки 3D моделей.....	30
2 Практична частина	34
2.1 Постановка завдання.....	34
2.2 Розробка додатку з камерою на телефон	34
2.3 Розробка додатка для виокремлення об’єктів з фото	40
2.4 Створення програми для створення 3D моделі.....	44
2.5 Підсумки зробленої роботи	49
Висновок	50
Перелік посилань.....	52
Додаток А. Програмні коди	54

ВСТУП

Рослини відіграють важливу роль у природному середовищі і мають значний вплив на екосистеми, сільське господарство, фармацевтику та інші галузі. Розуміння структури та функціонування рослин є ключовим аспектом для подальшого розвитку і вдосконалення цих галузей. Однак, вивчення рослин із використанням традиційних методів може бути часо- та працезатратним завданням.

Розуміння механізму запилення і зростання рослин грає величезне економічне значення. Якщо вченим вдасться відстежувати ці процеси, вони зможуть більш точно прогнозувати врожайності промислових культур. Крім того, багато трав і злаків досі недостатньо вивчені – дослідникам складно розібрати їх на складові частини і вивчити кожную окремо. Один з методів розв'язання цієї проблеми є створення цифрових моделей рослин та їхнього розвитку.

У зв'язку з цим, розвиток цифрових моделей рослин став актуальним напрямком досліджень. Цифрові моделі рослин надають можливість відтворити віртуальне представлення рослини, включаючи її геометричну структуру, фізіологічні процеси та інші властивості. Це дозволяє дослідникам глибше досліджувати фізичні та функціональні аспекти рослин, виконувати експерименти та передбачати їхню реакцію на різні умови [1].

Цифрові моделі рослин засновані на використанні комп'ютерного зору, обробки зображень, алгоритмів машинного навчання та інших технологій. Вони дозволяють аналізувати та візуалізувати структуру рослин, їхні фізіологічні процеси, взаємодію з довкіллям та інші характеристики.

Для побудови цифрової моделі рослини існують такі підходи.

Фотограмметрія: полягає в застосуванні фотографій рослини з різних кутів для створення тривимірної моделі. З використанням спеціального програмного забезпечення, такого як Agisoft Metashape або RealityCapture, фотографії обробляються для визначення геометричної структури рослини [2].

Лазерне сканування: використання лазерного сканування рослини і отримання точних геометричних даних. Зазвичай використовується 3D сканер, який сканує поверхню рослини і створює сукупність точок. Потім ці дані можуть бути оброблені для побудови тривимірної моделі. Програмне забезпечення, таке як CloudCompare або Autodesk ReCap, може використовуватись для обробки даних з лазерного сканування [3].

Синтез моделей: використовує математичні алгоритми та правила для синтезу цифрової моделі рослини. Він може базуватись на біологічних принципах та знаннях про рослинну структуру. Наприклад, модель L-systems застосовує контекстно-вільні граматики для генерації віток та гілок рослини [4].

Найліпше використовувати для поставлених завдань комбінований метод. Спочатку побудувати 3D модель в якості шаблону, а потім беручи дані за допомогою фотометрії змінювати готову модель. Цей підхід значно зменшить навантаження на алгоритм та потребу в великій кількості даних.

Метою даної дипломної роботи є створення 3D моделі рослини. Для досягнення цієї мети заплановано вирішення таких задач: аналіз існуючих методів та засобів 3D моделювання; розробка етапів 3D моделювання рослини; створення бази для навчання; створення моделі для розпізнавання рослини; створення мобільного застосунку для 3D моделювання; створення остаточної 3D моделі рослини.

Заплановано дослідження та розробка цифрових моделей рослин з використанням сучасних методів комп'ютерного зору та машинного навчання. Робота передбачає розробку етапів 3D моделювання для створення цифрових моделей рослин, а також їхню перевірку та оцінку за допомогою реальних даних.

Ця робота має важливе значення для подальшого розвитку наукових досліджень у галузі цифрових моделей рослин та їхнього застосування в різних сферах діяльності. Вона відкриває нові перспективи для дослідників, агрономів, біологів та інших фахівців, які працюють з рослинами, і може сприяти зростанню ефективності та сталості різних процесів, пов'язаних з рослинним світом.

1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Проблема дослідження рослин

Вивчення рослин за допомогою традиційних методів може зазнавати деяких обмежень і проблем. Наприклад, традиційні методи вивчення рослин, такі як огляди на місці, вимірювання фізичних параметрів, збір зразків та лабораторний аналіз, можуть бути дуже часо- та працезатратними завданнями. Вони вимагають багато людських ресурсів та велику кількість робочих годин.

Інші методи, наприклад, збір зразків для аналізу, можуть бути деструктивними, що означає, що рослина знищується або пошкоджується в процесі дослідження. Це може бути проблемою, особливо якщо рослина є рідкісною або захищеною.

Також традиційні методи можуть забезпечувати обмежену просторову та часову інформацію. Наприклад, огляди на місці можуть обмежуватися видимим діапазоном та доступністю до конкретних областей рослини. Це може призводити до обмеженого розуміння розподілу та взаємодії різних структур рослини [5].

Деякі аспекти, такі як фізіологічні процеси або вплив зовнішніх факторів на рослину, можуть бути складними для вимірювання за допомогою традиційних методів. Вони можуть бути залежними від часу та потребувати постійного спостереження протягом тривалого періоду часу.

Традиційні методи вивчення рослин можуть бути обмежені у масштабованості. Вони можуть бути важкими для застосування на великій території або для великої кількості рослин.

У зв'язку з цим, використання сучасних технологій, таких як цифрове моделювання рослин, може допомогти подолати ці проблеми, забезпечуючи швидке та точне збирання даних, більш повну просторову та часову інформацію та забезпечуючи більшу масштабованість досліджень [5].

1.2 Етапи створення цифрової моделі

Розробка етапів створення цифрової моделі є необхідною з кількох причин, по-перше, потрібно організувати робочий процес і визначити послідовність дій, яка потрібна для створення цифрової моделі. По-друге, кожен етап розробки моделі може бути перевірений і оцінений з точки зору якості. По-третє, можна упорядкувати інформацію та забезпечити її правильне поєднання, що сприяє отриманню повної та достовірної моделі. Останнє, можна відстежувати прогрес у виконанні завдань. Ось чому розробка етапів створення цифрової моделі сприяє систематичному та організованому підходу до процесу моделювання, що дозволяє досягти кращих результатів і забезпечити високу якість моделі [3].

Перший етап полягає у зборі даних про об'єкт, для якого буде створюватися модель, тобто для рослини. Це може включати фотографування, чи зйомка за допомогою дронів для отримання додаткової інформації. Всі отримані дані будуть зберігатися в окремій базі даних.

На другому етапі буде знайдено середні показники всіх отриманих об'єктів для створення першої шаблонної 3D моделі та створення її 3D візуалізації для подальшої обробки і використання.

Третій етап включає створення програми, яка буде відшукувати потрібні об'єкти на фото та одразу визначати їх розміри. Дані нових об'єктів буде збережено в базу даних.

Четвертий етап. Буде модифіковано шаблону 3D модель в залежності від нових даних. Нова модель буде збережена, як 3D об'єкт в окремій папці проекту.

На п'ятому етапі будуть порівнюватись нові моделі з шаблоном та їх однакові частини будуть помічатись синім кольором, щоб виокремити дані, які було згенеровано, а які отримано з зображення. Оскільки згенеровані дані не будуть точними, а лише наближеними відносно середніх показників.

Останній етап полягає у створенні мобільного додатку для полегшення збору даних для першого етапу.

В результаті опрацювання всіх розроблених етапів створення цифрової моделі (рис. 1.1) буде отримано файл 3D об'єкту.

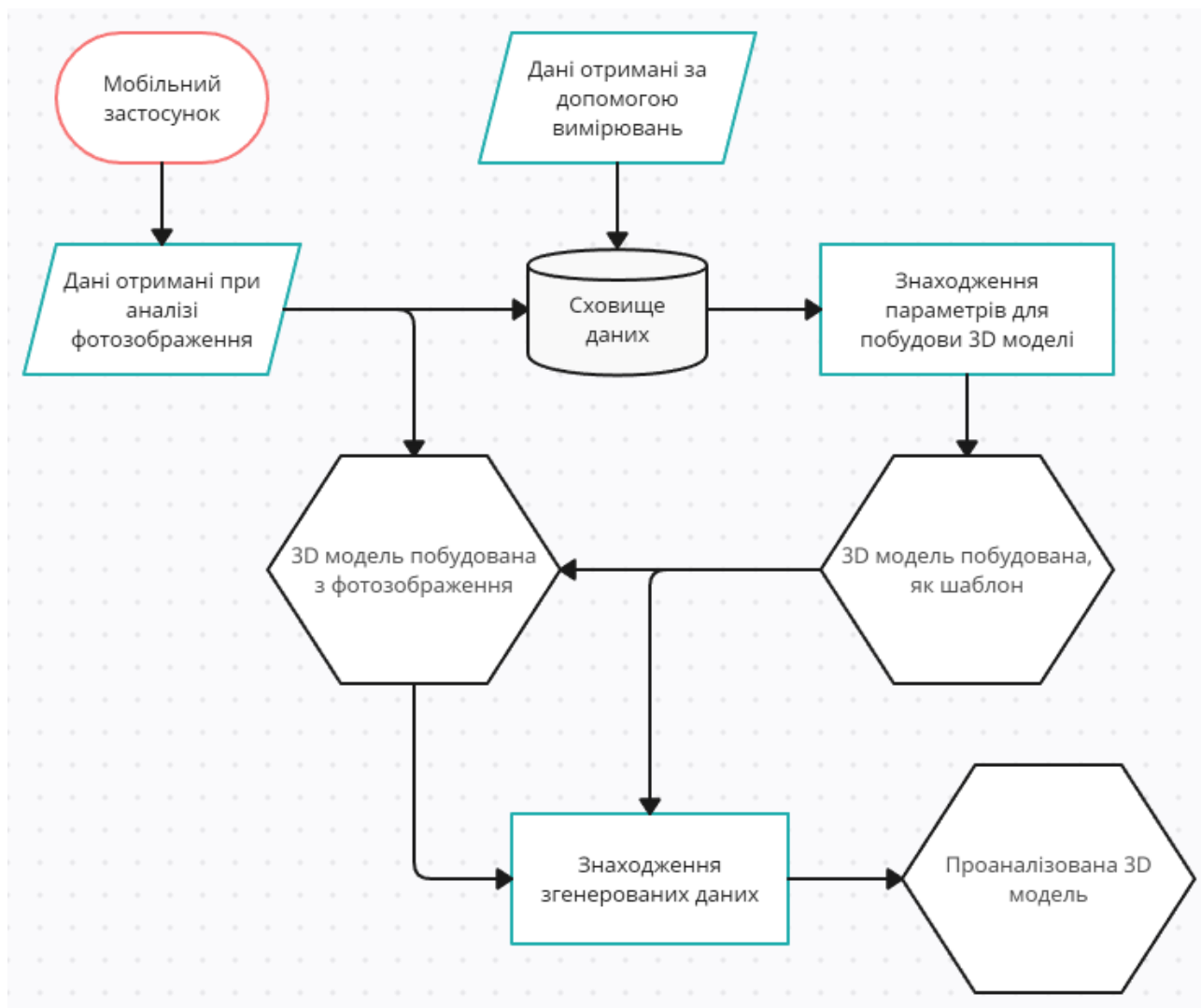


Рисунок 1.1 – Функціональна схема створення цифрової моделі

1.3 Розгляд технічних засобів

1.3.1 Python – мультиплатформена мова програмування

Python — високорівнева мова програмування, яку називають другою за популярністю в світі. Її використовують для розробки веб застосунків, програмного забезпечення, машинного навчання. Python застосовують для вирішення робочих завдань у компаніях Google, Instagram, Facebook, IBM, NASA, Dropbox, Netflix та інших. Розробники цінують цю мову програмування за простоту у вивченні, ефективність та мультиплатформність [6].

Python має легкий для вивчення синтаксис, що робить його ідеальним вибором для початківців у програмуванні. Велика кількість документації та активна спільнота розробників також полегшують процес навчання. Широкий спектр стандартних бібліотек дозволяє виконувати багато завдань безпосередньо з коробки. Крім того, існує велика кількість сторонніх бібліотек, які розширюють можливості мови для різних областей, включаючи науку, веб-розробку, обробку даних та інші.

Python є крос-платформенною мовою, що означає, що програми, написані на Python, можуть працювати на різних операційних системах, таких як Windows, macOS і Linux, безпосередньо з коробки. Це робить його універсальним і зручним для розробки на різних платформах.

Також Python може бути легко розширений за допомогою мови C/C++, що дозволяє використовувати швидкі інструменти та бібліотеки, які вимагають високої продуктивності.

Python має широкий спектр застосувань і використовується у багатьох галузях технологій та сферах діяльності. Це одна з найпопулярніших мов програмування для веб-розробки. Він використовується для створення веб-сайтів, веб-додатків, API і фреймворків, таких як Django і Flask. Python дозволяє швидко розробляти веб-додатки завдяки своїй простоті і великій кількості готових бібліотек.

Python є популярним вибором для наукових дослідників і аналітиків даних. Бібліотеки, такі як NumPy, Pandas і Matplotlib, надають потужні інструменти для роботи з числовими даними, обробки даних, візуалізації і наукових обчислень. Крім того, платформи, такі як Jupyter Notebook, сприяють інтерактивному дослідженню і розробці.

Також Python має велику кількість бібліотек і фреймворків, які підтримують розробку моделей штучного інтелекту і машинного навчання. TensorFlow, Keras, PyTorch і Scikit-learn - це лише декілька прикладів популярних інструментів, які допомагають створювати, тренувати і використовувати моделі для класифікації, прогнозування, обробки природної мови та багатьох інших завдань.

Python є потужним і гнучким інструментом для розробки різноманітних додатків, від простих скриптів до великих проектів. Він надає розробникам зручність і продуктивність, що робить його одним з найпопулярніших виборів для програмістів у всьому світі.

1.3.2 Використання Kivu для розробки мобільного додатку

Kivu (рис. 1.2) – це відкрита платформа для розробки графічних користувацьких інтерфейсів (GUI) на мові програмування Python. Вона була спеціально розроблена для створення кросплатформених додатків, що працюють на різних операційних системах, таких як Windows, macOS, Linux, Android та iOS.

Однією з основних переваг Kivu є те, що вона забезпечує розробникам можливість створювати багатоплатформенні додатки з єдиним кодом. Це означає, що ви можете написати програму один раз і запускати її на різних платформах без необхідності внесення значних змін в код [7].



Рисунок 1.2 – Логотип Kivy

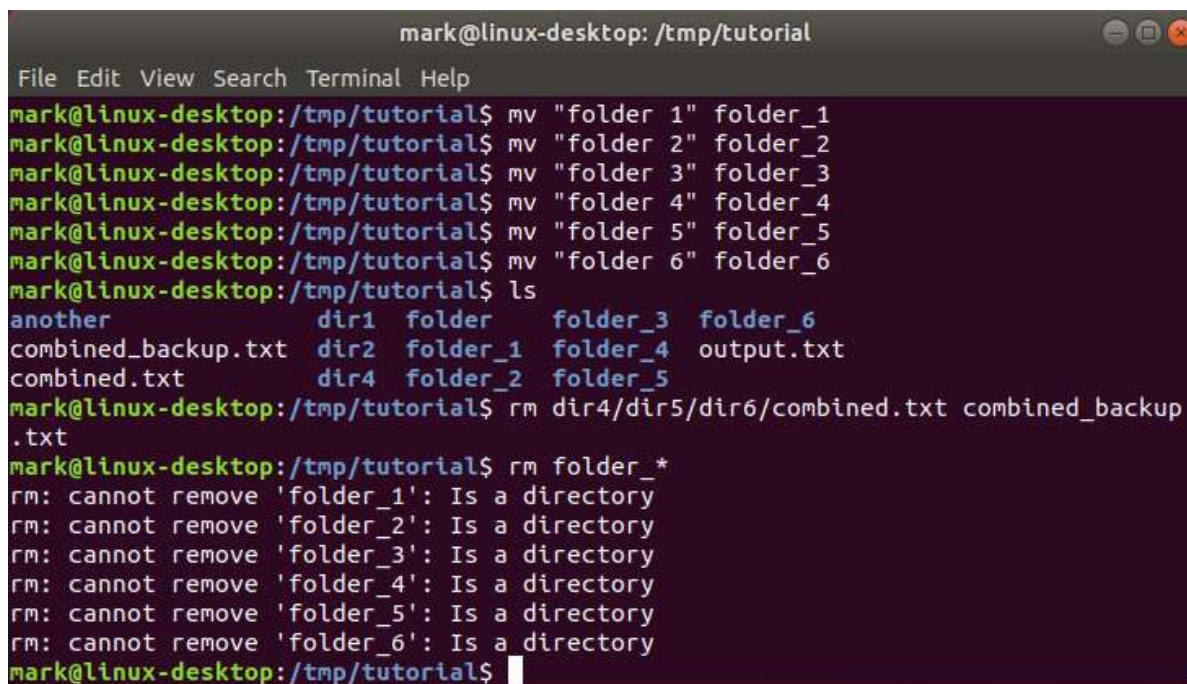
Основні риси та можливості Kivy включають:

- Кросплатформенність: Kivy дозволяє створювати додатки, що працюють на різних платформах, включаючи настільні комп'ютери, мобільні пристрої та навіть Raspberry Pi.
- Мультитач-інтерфейс: Kivy підтримує роботу з сенсорними пристроями та мультитач-жестами, що дозволяє створювати інтуїтивні та зручні інтерфейси для дотикового вводу.
- Графічні можливості: Kivy надає розширені графічні можливості, такі як анімація, відтворення відео, відображення зображень та 3D-графіка.
- Можливості взаємодії: Kivy підтримує різні способи взаємодії з користувачем, включаючи кнопки, текстові поля, списки, повзунки та багато іншого.
- Язикові ресурси: Kivy підтримує багатомовність і дозволяє використовувати різні мови програмування для створення інтерфейсів.
- Розширюваність: Kivy має модульну архітектуру, яка дозволяє розширювати функціональність за допомогою сторонніх пакетів та бібліотек Python.

Kivy також надає можливість використовувати мови розмітки, такі як Kivy Language (KV), яка дозволяє окремо визначати структуру та вигляд інтерфейсу без прив'язки до Python-коду [7].

1.3.3 Ubuntu, як окрема операційна система

Ubuntu (рис 1.3) – це безкоштовна та відкрита операційна система, яка базується на ядрі Linux. Вона розробляється та підтримується спільнотою розробників, яка працює над покращенням функціональності, безпеки та зручності користування [8].



```

mark@linux-desktop: /tmp/tutorial
File Edit View Search Terminal Help
mark@linux-desktop:/tmp/tutorial$ mv "folder 1" folder_1
mark@linux-desktop:/tmp/tutorial$ mv "folder 2" folder_2
mark@linux-desktop:/tmp/tutorial$ mv "folder 3" folder_3
mark@linux-desktop:/tmp/tutorial$ mv "folder 4" folder_4
mark@linux-desktop:/tmp/tutorial$ mv "folder 5" folder_5
mark@linux-desktop:/tmp/tutorial$ mv "folder 6" folder_6
mark@linux-desktop:/tmp/tutorial$ ls
another          dir1  folder  folder_3  folder_6
combined_backup.txt  dir2  folder_1  folder_4  output.txt
combined.txt       dir4  folder_2  folder_5
mark@linux-desktop:/tmp/tutorial$ rm dir4/dir5/dir6/combined.txt combined_backup
.txt
mark@linux-desktop:/tmp/tutorial$ rm folder_*
rm: cannot remove 'folder_1': Is a directory
rm: cannot remove 'folder_2': Is a directory
rm: cannot remove 'folder_3': Is a directory
rm: cannot remove 'folder_4': Is a directory
rm: cannot remove 'folder_5': Is a directory
rm: cannot remove 'folder_6': Is a directory
mark@linux-desktop:/tmp/tutorial$

```

Рисунок 1.3 – Термінал Ubuntu

Перша версія Ubuntu з'явилася у жовтні 2004 року і мала назву "Ubuntu 4.10 Warty Warthog". Це була перша з серії регулярних випусків, які відбуваються кожні шість місяців. Назва "Ubuntu" походить з африканського слова, що означає "людство до людей", і відображає філософію проекту співпраці та спільноти.

Ubuntu швидко набула популярності серед користувачів Linux завдяки своїй простоті використання, зручному інтерфейсу та активній спільноті розробників. Операційна система була орієнтована на настільні комп'ютери та ноутбуки, забезпечуючи надійну та функціональну альтернативу іншим комерційним операційним системам.

Сьогодні Ubuntu є однією з найпопулярніших дистрибутивів Linux, використовується мільйонами користувачів та має велику спільноту розробників, що надають підтримку та розвивають екосистему. Ubuntu продовжує розширювати свої можливості та впроваджувати нові технології, підтримуючи принципи відкритості та доступності [8].

1.3.4 Безкоштовне хмарне середовище для виконання коду Python

Google Colab (або Colaboratory) (рис 1.4) – це безкоштовне хмарне середовище для виконання коду Python, яке надається Google. Воно базується на сервісі хмарних обчислень Google Cloud та має інтегровану підтримку для Jupyter Notebook [9].

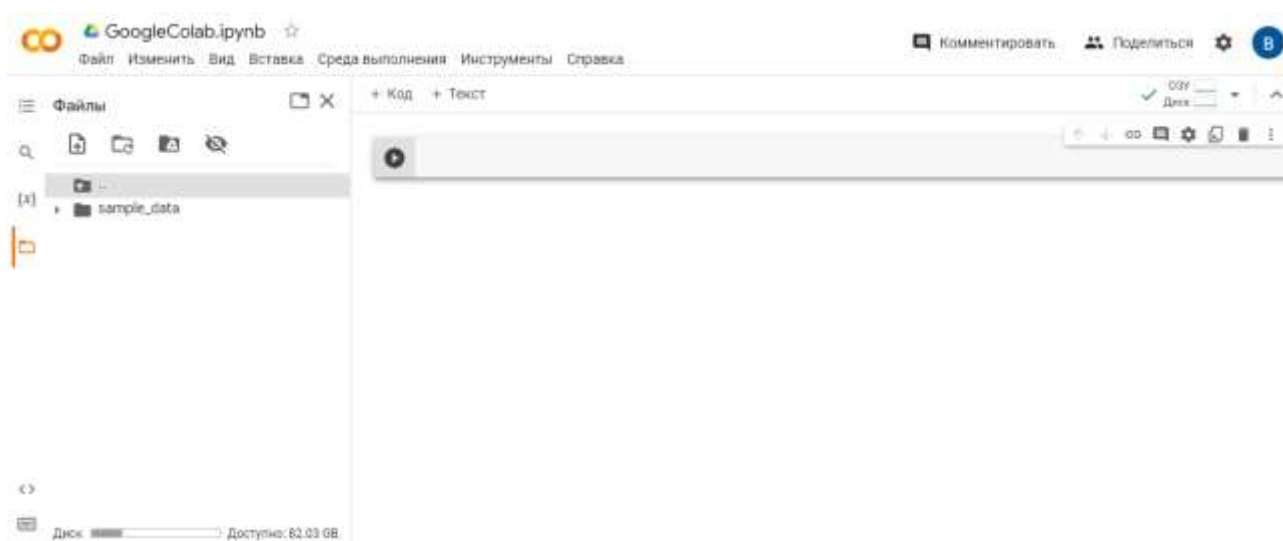


Рисунок 1.4 – Інтерфейс Google Colaboratory

Основні особливості Google Colab:

- Хмарне середовище: Google Colab дозволяє виконувати код Python безпосередньо у хмарі, що означає, що вам не потрібно встановлювати або налаштовувати локальне середовище. Ви можете працювати з ним з будь-якого пристрою, що має доступ до Інтернету.
- Jupyter Notebook: Google Colab підтримує Jupyter Notebook, що дозволяє вам створювати документи, які поєднують виконуваний код Python, текстові описи та візуалізації. Це робить його ідеальним інструментом для наукових досліджень, розробки алгоритмів та вивчення мови програмування Python.
- Бібліотеки та середовища: Google Colab має попередньо встановлені багато популярних бібліотек та середовищ, таких як NumPy, Pandas, Matplotlib, TensorFlow та багато інших. Вам не потрібно встановлювати їх самостійно – вони готові до використання.
- Ресурси хмари: Google Colab надає можливість використовувати ресурси хмари, такі як графічні процесори (GPU) та тензорні процесори (TPU), для прискорення обчислень. Це особливо корисно для завдань, пов'язаних з машинним навчанням та глибинним навчанням.
- Спільне використання та співпраця: Google Colab дозволяє легко спільно використовувати свої ноутбуки з іншими користувачами. Ви можете надати доступ до ноутбука для перегляду або редагування, спільно працювати над проектами та обмінюватися ідеями.

Google Colaboratory – це інтуїтивно зрозуміле та просте у використанні середовище. Також у ньому є покажчик з документацією та довідкою, та кілька прикладів, щоб написати нові, або змінити вже створені коди та перейти до тестування [9].

Найвідомішими функціями Google Colaboratory є:

- редагування та запуск коду Python;
- збереження проектів на Google Drive (GDrive);

- завантаження кодів з GitHub;
- імпорт файлів Jupyter або IPython.

1.3.5 Buidozер, як засіб для створення мобільних застосунків

Buidozер – це інструмент командного рядка, розроблений для спрощення процесу збирання і збірки Android-програм, написаних на Python. Він дає змогу перетворити ваш Python-проект в APK-файл, який може бути встановлений на Android-пристрої.

Історія Buidozер починається у 2013 році, коли Mathieu Virbel, розробник фреймворку Kivu, створив цей інструмент. Buidozер був спеціально розроблений для полегшення процесу збірки та розгортання Android-програм, написаних на Python.

Buidozер був створений з урахуванням потреб розробників, які використовують фреймворк Kivu для створення крос-платформових додатків на основі Python. Однак, з часом Buidozер також став популярним серед розробників, що використовують інші фреймворки та бібліотеки Python для створення Android-додатків.

Протягом років розвитку Buidozер отримав підтримку від активної спільноти розробників. Вони додавали нові функції, виправляли помилки та покращували його продуктивність і надійність. З часом Buidozер став важливим інструментом для розробників Python, що прагнуть швидко та зручно розгорнути свої програми на платформі Android.

Buidozер надавав можливість автоматизувати процеси збірки, пакування та налаштування APK-файлів, що значно спрощувало розгортання програм на Android-пристроях. Інструмент надавав можливість керувати залежностями, встановлювати потрібні бібліотеки та модулі Python, а також налаштовувати параметри APK-файлів.

Buildozer продовжує активно розвиватися, вдосконалюючись і підлаштовуючись під нові вимоги розробників. Він залишається популярним в індустрії розробки мобільних додатків на базі Python і продовжує сприяти швидкому та зручному розгортанню програм на платформі Android.

1.3.6 Використання OpenCV для аналізу фотозображень

OpenCV (Open Source Computer Vision Library) – це відкрите програмне забезпечення, яке надає набір функцій і алгоритмів комп'ютерного зору та обробки зображень. Воно розроблене для реалізації різноманітних завдань у сферах комп'ютерного зору, розпізнавання об'єктів (рис 1.5), аналізу зображень та роботи з відео [10].

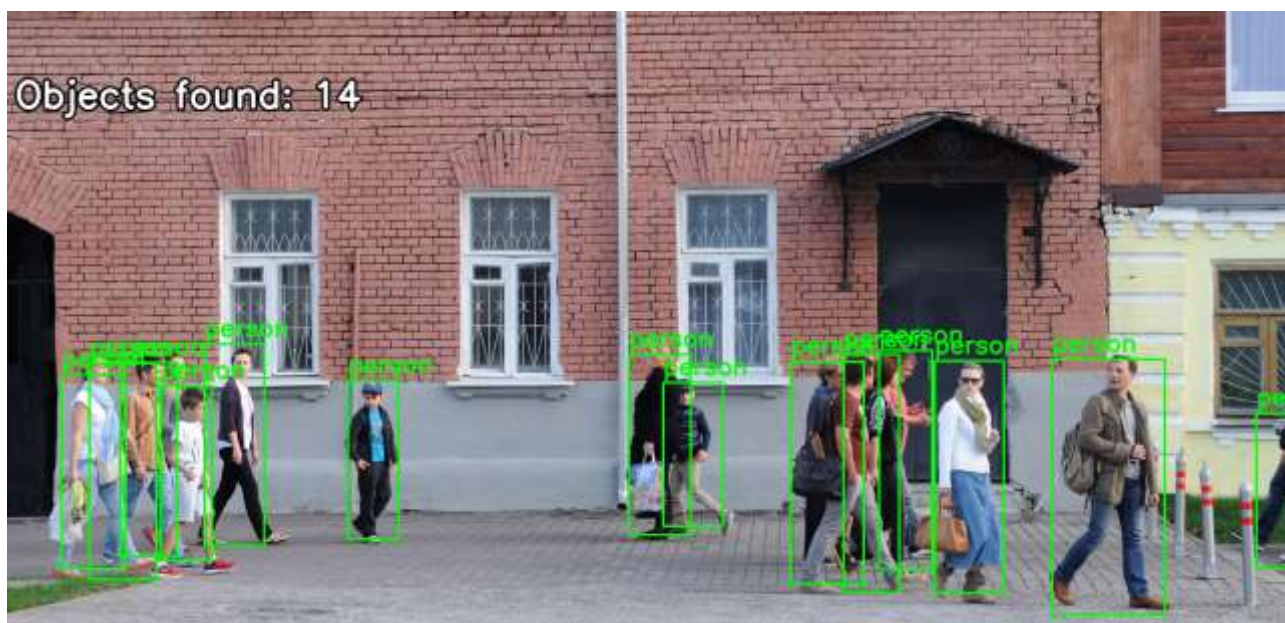


Рисунок 1.5 – Приклад роботи OpenCV по розпізнаванню людей

OpenCV написано на мові програмування C++, але має інтерфейси для багатьох мов, включаючи Python. Бібліотека пропонує велику кількість функцій

для обробки зображень і відео, таких як фільтрація, сегментація, детекція об'єктів, відстеження руху, розпізнавання обличчя та багато іншого.

Деякі основні можливості OpenCV включають:

- Завантаження та збереження зображень і відео: OpenCV надає функції для завантаження та збереження зображень та відео у різних форматах, включаючи JPEG, PNG, TIFF, AVI та інші.
- Обробка зображень: OpenCV має набір функцій для обробки зображень, таких як зміна розміру, обрізка, поворот, зміна контрастності та яскравості, фільтрація (рис 1.6), видалення шуму, виявлення країв та інші операції [10].

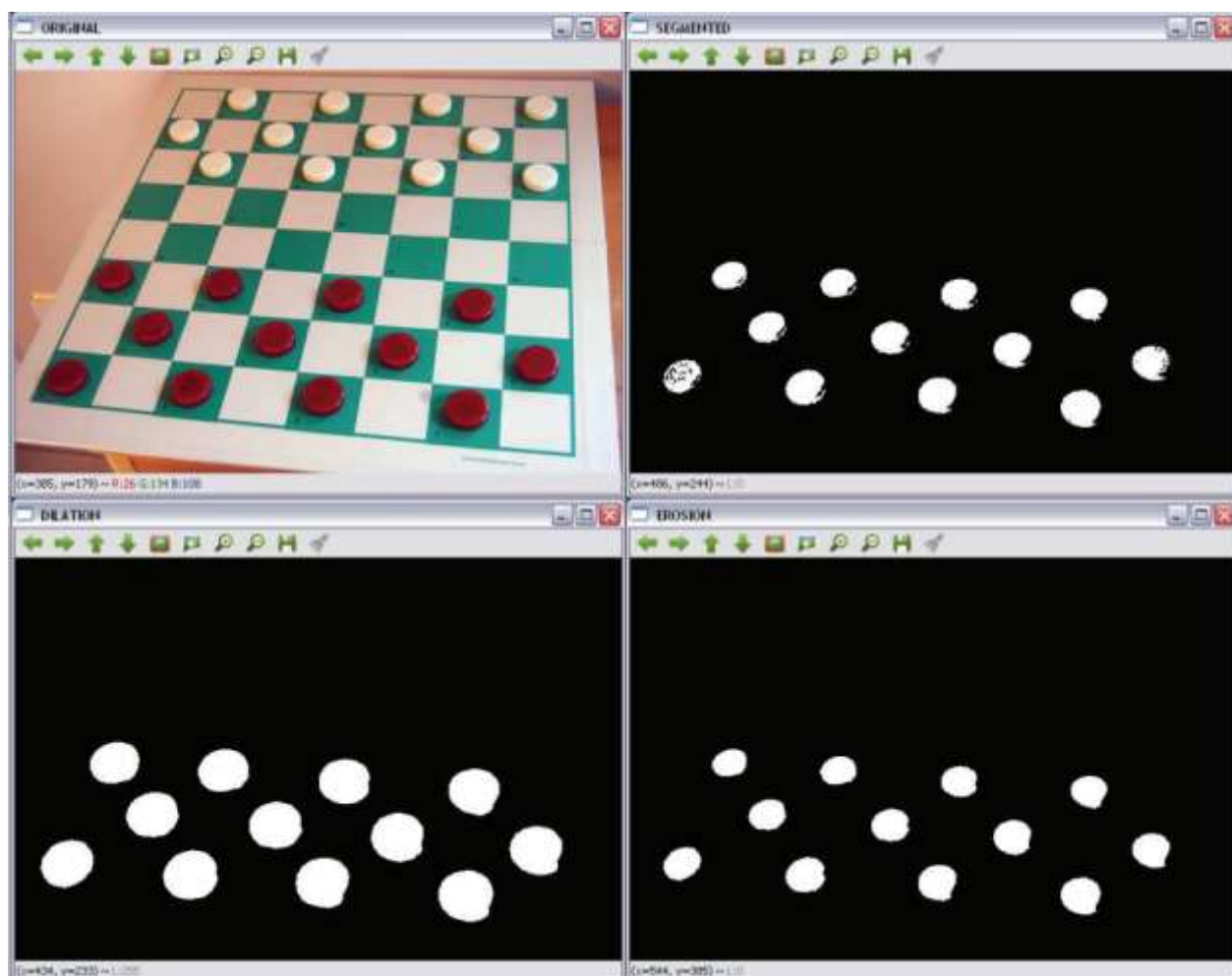


Рисунок 1.6 – Приклад фільтрації зображень за допомогою OpenCV

- Детекція об'єктів: OpenCV надає алгоритми для детекції об'єктів на зображеннях, таких як обличчя, люди, автомобілі, дорожні знаки та інші. Це відкриває можливості для розробки систем відеоспостереження, робототехніки, автоматичного розпізнавання та інших застосувань.

- Розпізнавання та відстеження руху: OpenCV містить алгоритми для виявлення та відстеження руху на відео. Це дозволяє створювати системи відстеження об'єктів, аналізу поведінки та інші рухові додатки.

- Машинне навчання: OpenCV має підтримку машинного навчання і надає інструменти для навчання класифікаторів та регресорів на основі зображень та відео.

OpenCV є однією з найпопулярніших бібліотек для комп'ютерного зору та обробки зображень. Вона знаходить широке застосування у багатьох галузях, включаючи комп'ютерну графіку, медичну діагностику, автономну навігацію, робототехніку та багато інших. Інтеграція з Python робить її потужним і зручним інструментом для розробників, що працюють зі зображеннями та відео [10].

1.3.7 Roboflow, створення моделей машинного навчання

Roboflow є онлайн-платформою для розробки та управління комп'ютерним зором і навчанням моделей машинного навчання. Вона надає інструменти для створення, анотування, побудови та оптимізації даних, а також для навчання та оцінювання моделей комп'ютерного зору (рис.1.7–1.8).



Рисунок 1.7 – Можливості Roboflow у створенні моделей

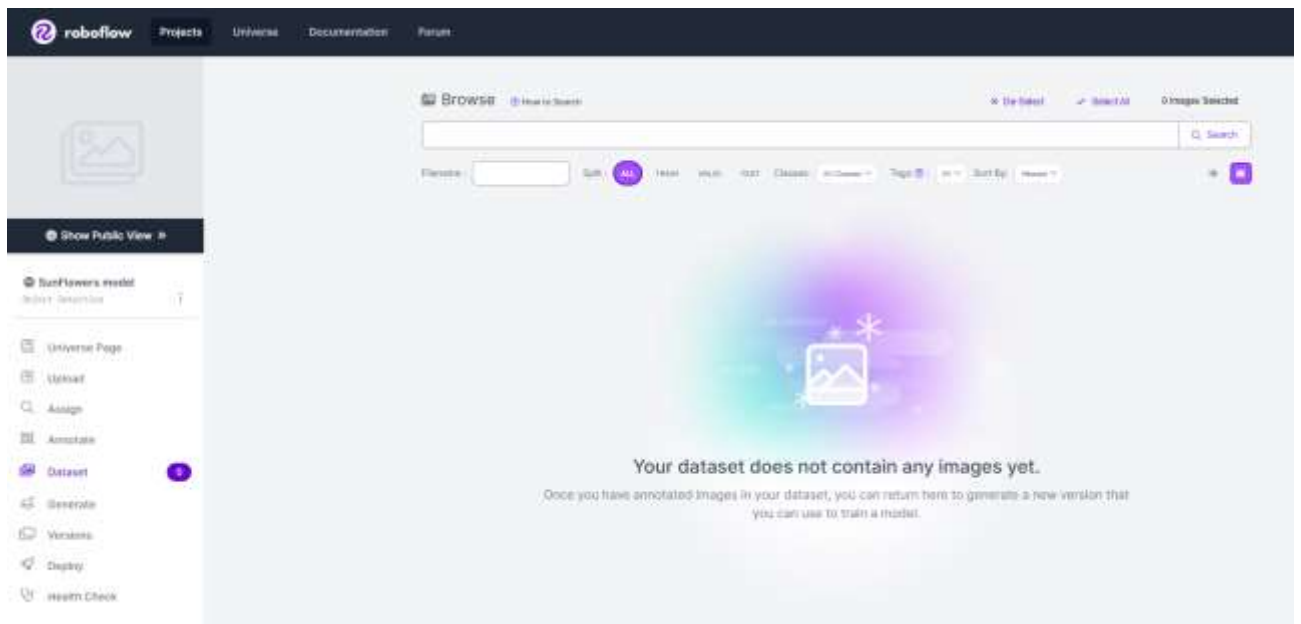


Рисунок 1.8 – Інтерфейс сайту Roboflow

- Roboflow дозволяє анотувати зображення та відео, додавати мітки, рамки, полігональні області та інші анотації для підготовки даних для тренування моделей комп'ютерного зору.

- Платформа надає інструменти для завантаження, каталогізації, організації та керування даними, зокрема зображеннями, анотаціями та метаданими.

- Roboflow дозволяє об'єднувати дані з різних джерел і форматів, створюючи комплексні датасети для тренування моделей.

- Платформа надає інструменти для тренування моделей машинного навчання з використанням навчальних даних. Вона підтримує різні архітектури моделей, такі як YOLO, SSD, RetinaNet та інші.

- Roboflow надає можливості для оцінки моделей, включаючи валідацію, вимірювання показників точності, розпізнавання об'єктів і тестування на нових даних.

- Платформа дозволяє експортувати навчені моделі в різних форматах, таких як TensorFlow, PyTorch, ONNX та інші, для подальшого використання у власних проектах або інтеграції з іншими системами.

Roboflow є потужним інструментом для розробки рішень комп'ютерного зору, спрощуючи процес підготовки даних, тренування моделей і оцінки їх ефективності. Вона знаходить застосування в різних галузях, включаючи автономні автомобілі, медицину, робототехніку, виробництво та інші [11].

Roboflow використовує нейронну мережу Convolutional Neural Networks (CNN), для завдань розпізнавання об'єктів, виявлення контурів, класифікації та сегментації зображень.

Convolutional Neural Networks (CNNs) – це тип нейронних мереж, які широко використовуються для обробки зображень і завдань комп'ютерного зору. Вони були спеціально розроблені для виявлення властивостей зображень, таких як краї, форми, текстури та об'єкти.

Головна особливість CNN полягає в застосуванні операції згортки (convolution) для аналізу локальних областей зображення. Ця операція дозволяє

виявляти різні характеристики зображень, такі як риси обличчя, краї об'єктів чи текстурні орнаменти. Крім того, CNN використовує пулінг (pooling) для зменшення розміру представлення зображення і забезпечення інваріантності до масштабу та незначних зміщень.

Структура CNN складається з вхідного шару, шарів згортки, шарів пулінгу та пов'язаних шарів, що виконують класифікацію. Після тренування на великій кількості зображень, CNN може вивчити представлення характеристик зображень та використовувати ці знання для класифікації або виявлення об'єктів на нових зображеннях.

1.3.8 Aspose – бібліотека для роботи з різними типами файлів

Aspose (рис 1.9) є набором програмних бібліотек і інструментів, які надають розширені можливості для роботи з різними типами файлів у програмах на платформі .NET, Java, C++, Android та інших. Цей набір програмних засобів розроблений компанією Aspose і відомий своєю широкою підтримкою різних форматів файлів і функціональністю [12].



Рисунок 1.9 – Логотип Aspose

Aspose дозволяє розширити можливості програмного забезпечення шляхом взаємодії з файлами, такими як документи, презентації, електронні таблиці, зображення, PDF-файли, файли звуку та багато інших. Він надає широкий спектр функцій, таких як створення, зчитування, редагування, конвертація, валідація, обрізка, розпізнавання символів, генерація звітів та багато іншого.

Aspose є особливо корисним для розробників програмного забезпечення, які працюють з бізнес-документами або потребують взаємодії з різними типами файлів у своїх додатках. Наприклад, ви можете використовувати Aspose для створення звітів у форматі PDF, зчитування та редагування даних у електронних таблицях, конвертації зображень між різними форматами, витягування тексту з документів та багато іншого.

Aspose надає потужні інструменти, які спрощують роботу з файлами, забезпечуючи гнучкість, швидкість та точність обробки. Він широко

використовується в різних галузях, включаючи фінанси, логістику, медицину, право, науку, освіту та інші.

Загальна філософія Aspose полягає в наданні програмістам потужних інструментів для роботи з різними типами файлів, щоб вони могли ефективно розробляти свої додатки і розв'язувати складні завдання обробки даних.

Aspose.3D for Python – універсальне, багатофункціональне та високоефективне рішення для 3D-моделювання для розробників Python. Він пропонує повний набір функцій, які дозволяють користувачам легко створювати, читати та змінювати 3D-моделі. Деякі ключові особливості цієї 3D-бібліотеки Python включають:

- Підтримка популярних форматів 3D-файлів, таких як FBX, STL та ін.
- Високопродуктивні можливості 3D-рендерингу та візуалізації.
- Розширені інструменти управління та маніпулювання сценою.
- Велика документація та приклади коду, які допоможуть розробникам швидко розпочати роботу.

Aspose.3D є частиною сімейства продуктів Aspose і надає розширені можливості для роботи з тривимірними моделями і 3D-даними. Вона дозволяє створювати, редагувати, конвертувати та взаємодіяти з 3D-моделями у різних форматах.

За допомогою Aspose.3D (рис 1.10) на Python ви зможете виконувати різноманітні операції з 3D-моделями, такі як зчитування та запис моделей у різних форматах, зміна їх геометрії, текстур, матеріалів та інших атрибутів. Ви також зможете виконувати рендеринг моделей, створювати анімації, взаємодіяти зі сценою та виконувати інші операції, що стосуються 3D-графіки [12].

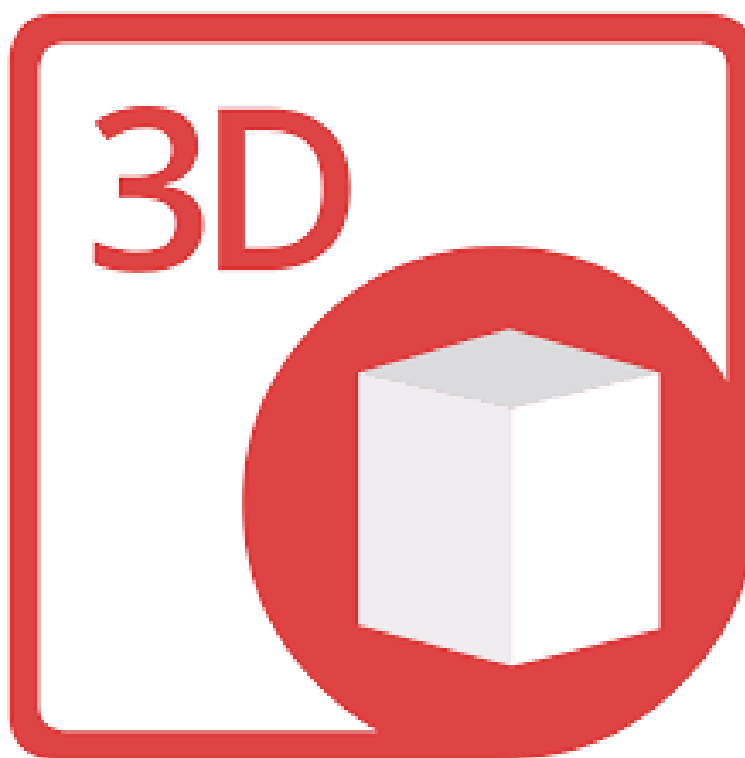


Рисунок 1.10 – Aspose.3D

Aspose.3D на Python може підтримувати різні формати 3D-моделей, такі як STL, FBX, Collada, 3DS та багато інших. Це дає вам гнучкість у роботі з різноманітними типами 3D-моделей у вашому проекті.

Generative Adversarial Networks (GANs) – це тип нейронних мереж, що використовується для генерації нових зразків, таких як зображення, звуки або текст, що мають схожі характеристики з реальними даними. GANs складаються з двох основних компонентів: генератора і дискримінатора.

Генератор має за мету створювати нові зразки, використовуючи набір випадкових змінних як вхід. Він навчається генерувати зразки, які неможливо відрізнити від реальних зразків. Дискримінатор, з іншого боку, навчається визначати, чи є вхідний зразок реальним (взятий з навчального набору даних) чи згенерованим генератором.

Генератор і дискримінатор грають у гру, де генератор намагається зманіпулювати свої вихідні дані, щоб дискримінатор помилився, а дискримінатор намагається розрізнити справжні зразки від згенерованих. Цей процес триває до досягнення рівноваги між генератором і дискримінатором, де генератор стає

досить вправним у створенні реалістичних зразків, а дискримінатор не може ефективно розрізнити їх.

GANs використовуються в багатьох задачах, таких як генерація зображень, синтез музики, створення відео та генерація тексту. Вони можуть створювати нові зразки, що мають великий потенціал у творчому процесі, варіативному дизайні, візуальних ефектах та багатьох інших областях, де потрібна генерація нових даних з визначеними характеристиками.

1.4 Методи штучного інтелекту при аналізі зображень

Штучний інтелект (ШІ) грає значну роль в аналізі зображень, де його використовують для розпізнавання, класифікації, визначення об'єктів та багатьох інших завдань, пов'язаних з обробкою та розумінням зображень.

Комп'ютерний зір – вид діяльності, в якій для отримання даних застосовують статистичні методи та використовують моделі, які побудовані за допомогою геометрії, фізики і теорії навчання, які є базою та також частиною освітньої програми. Комп'ютерний зір застосовують досить широко в управлінні мобільними роботами, військових додатках, промислових засобах спостереження, а також у сфері взаємодії людина/комп'ютер, пошуку зображення в бібліотеках, аналізі медичних зображень та реалістичному передаванні змодельованих сцен у комп'ютерній графіці.

Особливість комп'ютерного зору – це отримання описів із зображень або послідовності зображень. За допомогою невеликої кількості фотографій можна отримати хороші, прості, точні та зручні моделі. Проводити серйозні дослідження та вирішувати повсякденні завдання (наприклад, упорядкувати колекцію фотографій, створити тривимірну модель навколишнього світу, керувати та вносити зміни в колекцію відеозаписів) тепер можна за допомогою методів комп'ютерного зору. Сьогодні комп'ютерний зір перебуває на піку свого

розвитку та набуває популярності, що в свою чергу збільшує попит на фахівців саме в цій галузі [13].

ШІ може бути навчений розпізнавати та класифікувати об'єкти на зображеннях. За допомогою навчання з нагляду, моделі можуть розпізнавати та відрізняти різні класи об'єктів, такі як тварини, транспортні засоби, люди, рослини тощо.

Також ШІ може бути використаний для виявлення та локалізації об'єктів на зображеннях. Це важливо, наприклад, для систем відеоспостереження, автоматичного водіння або робототехніки. Моделі можуть виявляти обличчя, автомобілі, дорожні знаки, предмети у супермаркетах тощо.

Застосування штучного інтелекту у виявленні об'єктів на зображеннях має великий потенціал і знаходить широке застосування в різних галузях. Ось деякі з них:

- Автоматичне водіння: Штучний інтелект використовується у системах автоматичного водіння для виявлення та відстеження об'єктів на дорозі, таких як автомобілі, пішоходи, велосипедисти тощо. Це допомагає автономним автомобілям приймати рішення про безпечне рухання в середовищі з рухомими об'єктами.

- Системи безпеки: В системах відеоспостереження штучний інтелект використовується для виявлення підозрілих об'єктів або поведінки на основі зображень з камер спостереження. Це може включати виявлення вторгнень, небажаної поведінки або небезпечних ситуацій.

- Медична діагностика: Штучний інтелект використовується для автоматичного виявлення патологій на медичних зображеннях, таких як рентгенівські знімки, КТ або МРТ. Він може допомогти лікарям у виявленні ракових ознак, захворювань серця, неврологічних станів та інших медичних відхилень.

- Оптичне розпізнавання символів: Штучний інтелект використовується для оптичного розпізнавання символів (OCR) у документах або на зображеннях. Це може бути корисно для автоматизації обробки документів,

розпізнавання номерів автомобілів, сканування тексту та інших подібних завдань.

– Комерційні застосування: Штучний інтелект використовується у різних комерційних сферах для виявлення об'єктів. Наприклад, у роздрібній торгівлі він може допомагати виявляти товари на полицях, аналізувати покупців та їхню поведінку, визначати тенденції споживання та забезпечувати персоналізований сервіс.

Завдяки постійному розвитку технологій глибинного навчання та обробки зображень, ця область продовжує розширюватись, пропонуючи нові можливості та використання у різних сферах життя [2].

Можна проводити сегментацію зображень, тобто виділяти окремі області на зображенні, які належать до різних класів. Наприклад, в медицині це може використовуватися для виділення ракових клітин на зображеннях біоптатів.

Глибинні нейронні мережі є потужним інструментом для аналізу зображень. Вони можуть навчатися визначати складні ознаки та шаблони на зображеннях, що дозволяє їм ефективно вирішувати задачі аналізу зображень.

ШІ дає змогу автоматизувати та полегшити обробку та розуміння зображень, що має великий потенціал в різних сферах, включаючи медицину, автоматичне водіння, розпізнавання об'єктів та багато інших [4].

1.5 Актуальність проблеми побудови цифрової моделі рослини

Проблема побудови цифрової моделі рослини є актуальною в наш час, оскільки вона має великий потенціал і може знайти застосування у різних галузях. Цифрові моделі рослин використовуються у наукових дослідженнях для вивчення рослинної біології, фізіології та екології. Вони дозволяють вченим аналізувати та розуміти різні аспекти росту, розвитку та функціонування рослин.

Моделі рослин можуть бути використані для оптимізації сільського господарства. Допомагають визначати оптимальний час поливу, добрив та захисту рослин, що сприяє збільшенню урожайності та зниженню витрат.

Дозволяють візуалізувати та планувати ландшафтний дизайн, допомагають вибрати та розташувати рослини відповідно до задуманого проекту. Вони дають змогу побачити, як рослини будуть виглядати у різних часах року та допомагають створити гармонійний дизайн.

Цифрові моделі можуть бути використані в освіті для навчання студентів біології та ботаніки. Вони дозволяють студентам досліджувати та вивчати різноманітні види рослин, їх структуру та функціонування, не обмежуючись фізичними зразками. Також вони допомагають прогнозувати зміни в рослинному покриві та оцінювати його важливість для збереження природних ресурсів.

Застосування цифрових моделей рослин може принести значні переваги у вивченні та застосуванні рослинного світу, тому ця проблема продовжує бути актуальною для дослідників і фахівців у відповідних галузях.

1.6 Існуючі застосунки для розробки 3D моделей

Reality Capture — це програмне забезпечення для створення 3D моделей реальності. Він використовує фотографії, лазерне сканування та інші дані для автоматичного збирання інформації про оточення та створення 3D-моделі з високою точністю (рис 1.11). Це як варіант можна використовувати для створення копій середовища для гейміфікації [14].



Рисунок 1.11 – Створення 3d моделі за допомогою Reality Capture

Reality Capture використовує техніку фотограмметрії. Техніка фотограмметрії включає такі основні етапи:

1. Знімання фотографій: фотографування об'єкта з різних ракурсів, забезпечуючи достатнє перекриття між фотографіями. Це забезпечує наявність достатньої кількості спільних ознак між сусідніми зображеннями.

2. Вирівнювання фотографій: на цьому етапі програми фотограмметрії знаходять на фотографіях спільні точки (ознаки) та вирівнюють знімки відповідно до їх відносного положення й орієнтації.

3. Відтворення 3D-моделі: програма використовує збіги між зображеннями для відтворення тривимірної моделі об'єкта у вигляді хмари точок або полігональної сітки.

4. Текстурування: на цьому етапі програма накладає текстури з фотографій на відтворену 3D-модель, створюючи візуально реалістичний зовнішній вигляд моделі.

5. Експорт та оптимізація: залежно від потреб користувача, 3D-модель може бути експортована у різні формати, а також оптимізована для подальшого

використання в програмах візуалізації, аналізу, або створення контенту для застосунків AR/VR.

Складнощі фотограмметрії:

- Необхідність багатьох фотографій: як митець, який намагається зобразити об'єкт з різних ракурсів, вам потрібно мати багато зображень, щоб забезпечити достатній рівень деталізації.
- Залежність від якості фотографій тож зовнішні умови, такі як освітлення, погода або рух, можуть впливати на якість знімків і ускладнити процес фотограмметрії.
- Нестача ознак для об'єднання подібних фото: Це як спробувати знайти своїх друзів на фестивалі, де всі одягнені в однакові куртки — важко розібратися, хто є хто.
- Велика кількість фотографій і складність об'єктів можуть призвести до довгого часу обробки даних.

Переваги фотограмметрії:

- Низькі витрати на обладнання: Фотограмметрія може виконуватися за допомогою стандартного фотоапарата, тому витрати на обладнання менші, ніж в випадку лазерного сканування.
- Зручний багаж: Замість того щоб нести важкі сканери, все, що вам потрібно, це камера і, можливо, штатив. Таким чином, вам не доведеться ставати шерпом, щоб носити важке обладнання
- Висока роздільна здатність текстури 3D-моделей, що може бути використана для відображення дрібних деталей на фресках та картинах інтер'єрів.

Cognex VisionPro – програмне забезпечення для ПК, призначене для налаштування та розгортання прикладних завдань машинного зору – для зображень, як отриманих безпосередньо від цифрової камери, так і через вбудовану плату захоплення зображень з ПК. З VisionPro користувачі можуть реалізовувати широкий спектр функцій, від позиціонування та контролю геометричних об'єктів до ідентифікації, вимірювання та вирівнювання, а також

спеціалізованих функцій у прикладних задачах напівпровідникової та електронної промисловості.

З VisionPro користувачі отримують доступ до серйозної бібліотеки інструментів – порівняння з шаблоном, хмара точок, штангенциркуль, положення лінії, фільтри зображення, розпізнавання тексту (OCR) та відео інструменти (OCV), зчитування 1D та 2D-кодів – для вирішення широкого спектру завдань від перевірки до ідентифікації та вимірювання. VisionPro повністю інтегровано з великою бібліотекою класу .NET і елементами управління користувача (рис 1.12).

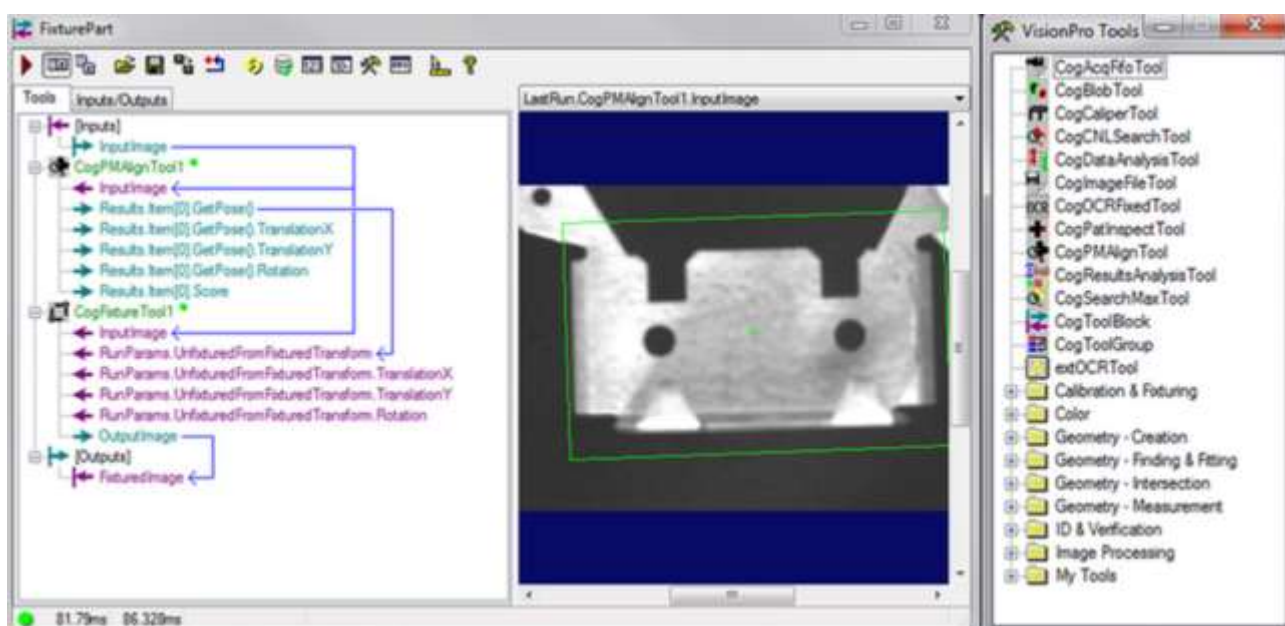


Рисунок 1.12 – Інтерфейс VisionPro

Середовище швидкого прототипування VisionPro QuickBuild поєднує потужність та гнучкість передового програмування з простотою розробки. Надає змогу завантажувати та виконувати автоматично або вручну налаштування інструментів для кожного окремого коду.

Перетягування – зв'язок між інструментами дозволяє швидко передавати значення, результати та зображення

Сценарії – розробка керованих додатків мовами C# або VB

Програмування – налаштування збору даних, вибір та оптимізація візуальних інструментів, а також ухвалення рішень про проходження тесту [14].

2 ПРАКТИЧНА ЧАСТИНА

2.1 Постановка завдання

Потрібно розробити мобільний додаток з якого можна швидко отримувати дані та відправляти їх на робочу машину.

Також потрібно створити програму, яка за отриманими даними буде здатна визначати потрібні об'єкти з фото та відео, а також обробляти та зберігати їх.

Та створити програму, яка з отриманих даних буде створювати цифрову 3D модель.

2.2 Розробка додатку з камерою на телефон

Перший етап розробки включає створення додатку на Android, який фотографуватиме та одразу зберігатиме фото в папку на комп'ютері через Wi-Fi з'єднання.

Було обрано використання бібліотеки kivy. Створимо додаток, здатний роботи зображення (Лістинг 2.1):

Лістинг 2.1:

```
class CameraClick(FloatLayout):
    def __init__(self, **kwargs):
        super(CameraClick, self).__init__(**kwargs)
        self.cam = Camera(index = 0, play=True, size_hint=(1, .8),
pos_hint={'x':0, 'y':.18})
        self.add_widget(self.cam)
```

```

self.butSave = Button(text="Зберегти", size_hint=(0.3, .15),
pos_hint={'x':.35, 'y':.01}, font_size=40)
self.butSave.bind(on_press=self.capture)
self.add_widget(self.butSave)
def capture(self, instance):
    camera = self.cam
    timestr = time.strftime("%Y%m%d_%H%M%S")
    camera.export_to_png("IMG_{}.png".format(timestr))

```

Зовні він виглядатиме так (рис 2.1) та зберігатиме зображення в форматі png, при наступній розробці додамо сканування, щоб додаток обробляв все у режимі реального часу.

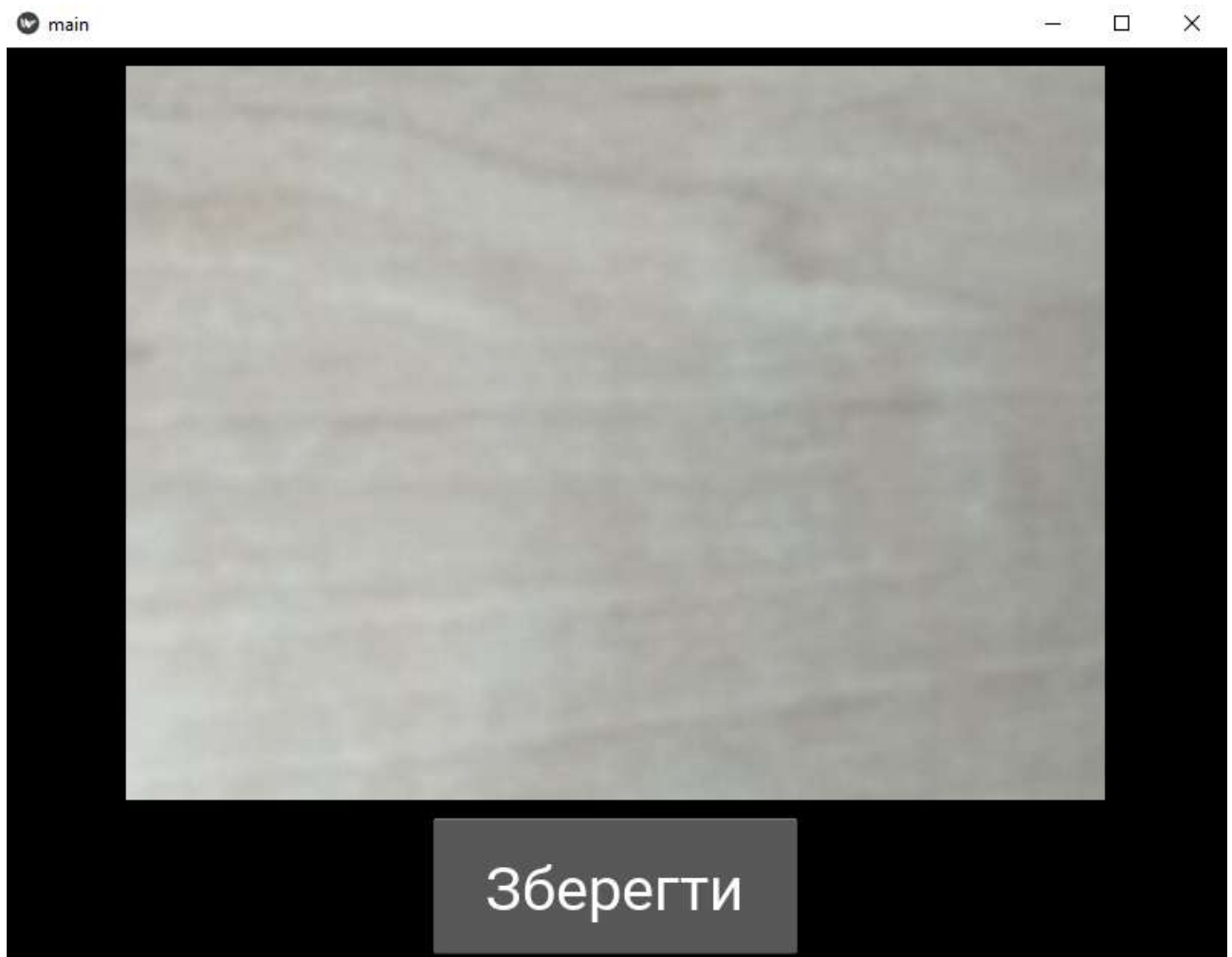


Рисунок 2.1 – Інтерфейс додатку

Далі потрібно скомпілювати його в Android-файл з розширенням `apk`. Для цього бажано скористатися операційною системою Ubuntu, оскільки саме вона рекомендується для компіляції `apk`-файлів, а також інструментом командного рядка Buildozer. Одним з варіантів встановлення Ubuntu поверх Windows є Microsoft Store (рис 2.2).

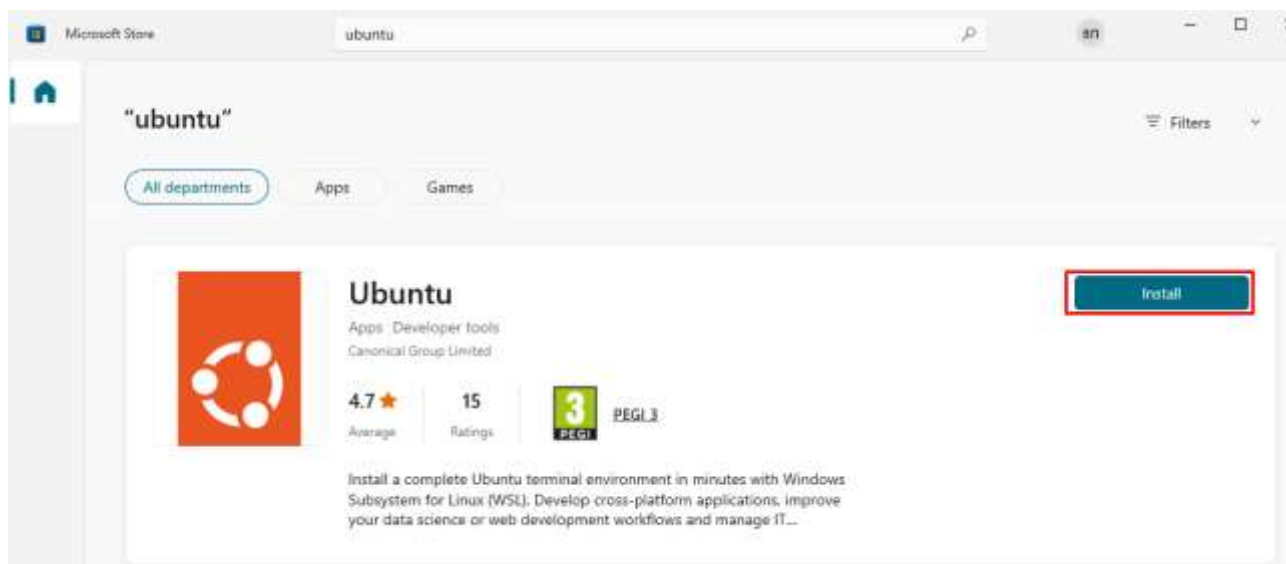


Рисунок 2.2 – Магазин Microsoft Store

Також можна скористатися безкоштовним хмарним середовищем для виконання коду Python – Google Colab. Воно може замінити Ubuntu, але має значно менший функціонал. В цій роботі буде використано його, але команди для компіляції та передналаштування схожі.

Для початку роботи потрібно встановити всі потрібні бібліотеки (Лістинг 2.2):

Лістинг 2.2:

```
!pip install buildozer
!pip install cython==0.29.19
!sudo apt-get install python3-pip
!sudo apt-get install build-essential
!sudo apt-get install git
```

```
!sudo apt-get install python3
!sudo apt-get install python3-dev
!sudo apt-get install ffmpeg
!sudo apt-get install libsdl2-dev
!sudo apt-get install libsdl2-image-dev
!sudo apt-get install libsdl2-mixer-dev
!sudo apt-get install libsdl2-ttf-dev
!sudo apt-get install libportmidi-dev
!sudo apt-get install libswscale-dev
!sudo apt-get install libavformat-dev
!sudo apt-get install libavcodec-dev
!sudo apt-get install zlib1g-dev
!sudo apt-get install libgstreamer1.0
!sudo apt-get install gstreamer1.0-plugins-base
!sudo apt-get install gstreamer1.0-plugins-good
!sudo apt-get install build-essential
!sudo apt-get install libsqlite3-dev
!sudo apt-get install sqlite3
!sudo apt-get install bzip2
!sudo apt-get install libbz2-dev
!sudo apt-get install zlib1g-dev
!sudo apt-get install libssl-dev
!sudo apt-get install openssl libgdbm-dev
!sudo apt-get install libgdbm-compat-dev
!sudo apt-get install liblzma-dev
!sudo apt-get install libreadline-dev
!sudo apt-get install libncursesw5-dev
!sudo apt-get install libffi7
!sudo apt-get install libffi-dev uuid-dev
!sudo apt-get install libffi-dev
```

Потім потрібно ініціалізувати файл `builddozer.spec` та налаштувати під наші потреби, вказавши назву додатку, версію, дозволи, бібліотеки, які використано, та багато іншого.

Після всіх налаштувань можна почати компілювати. Недолік використання Google Colab в тому, що кожного разу потрібно заново встановлювати всі бібліотеки і налаштовувати `builddozer.spec`, оскільки сеанс триває лише 24 години, після чого видаляється все, що не було збережено на Google Диску.

Після вдалої компіляції файл з'явиться в папці `bin` (рис 2.3) та його можна буде запустити на телефоні (рис 2.4).

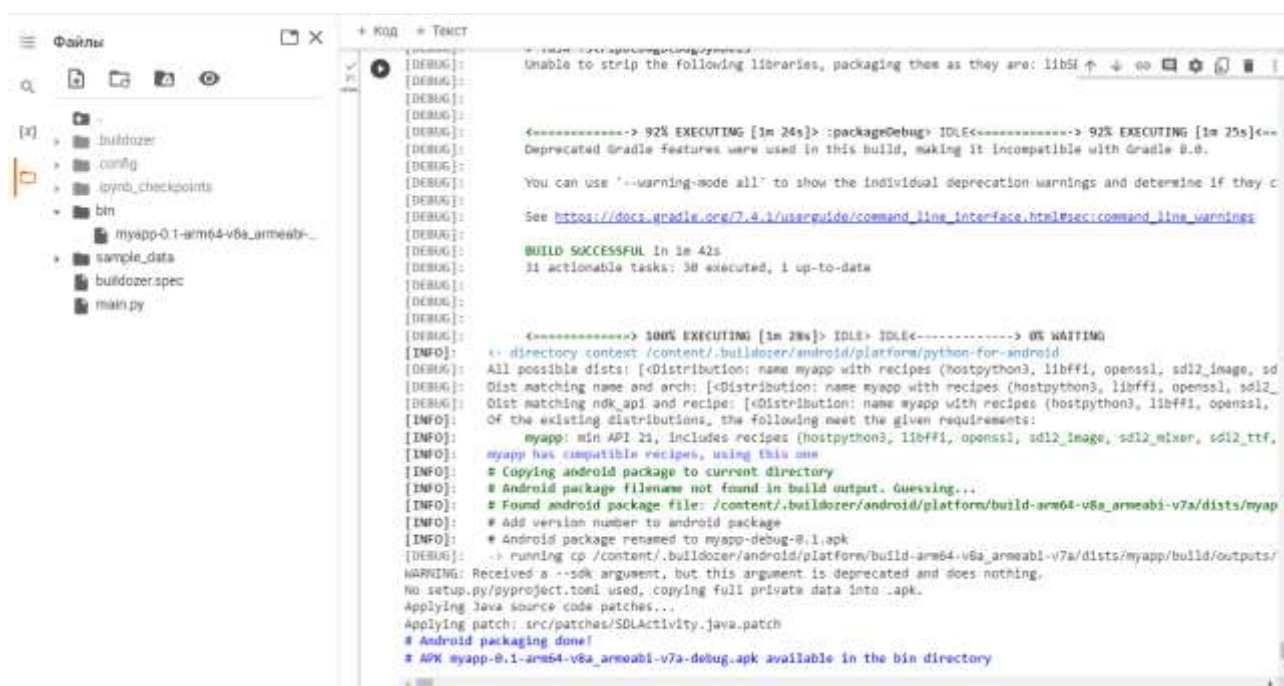


Рисунок 2.3 – Компіляція арк-файлу

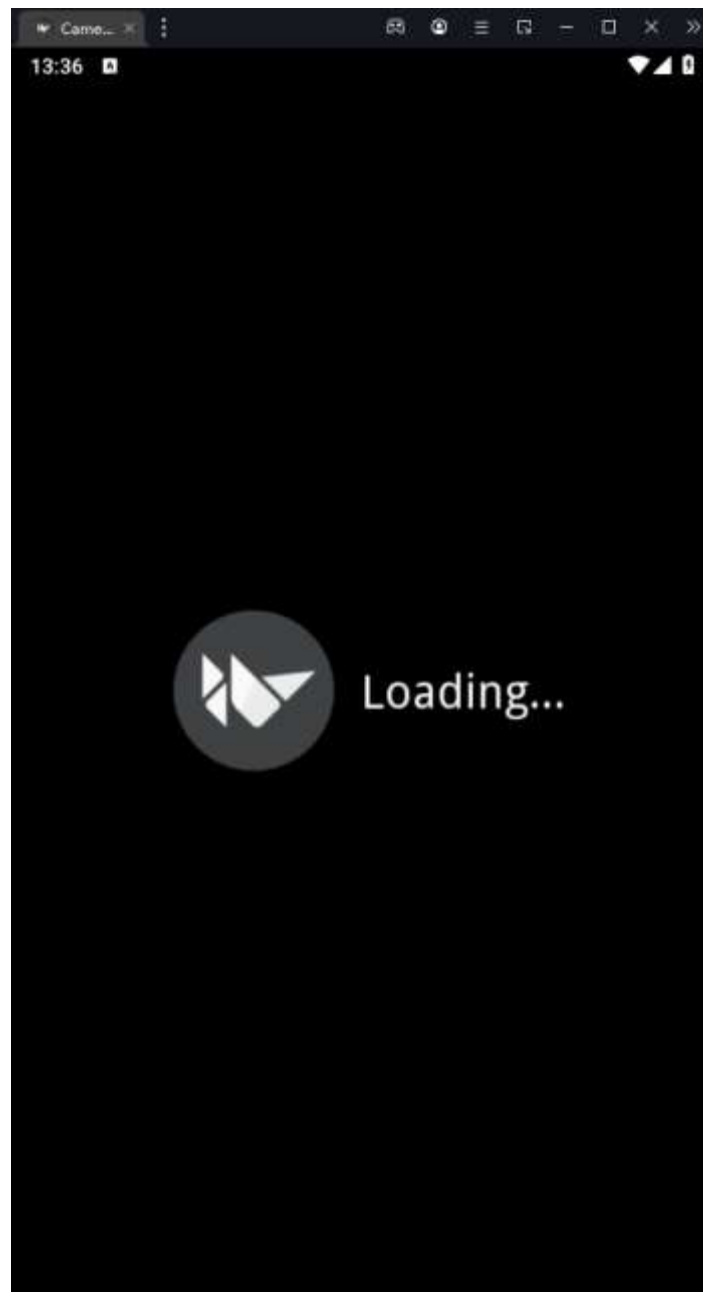


Рисунок 2.4 – Завантаження додатку

Отже, в нас є працюючий додаток для відправки зроблених фото на подальшу обробку і фільтрацію.

2.3 Розробка додатка для виокремлення об'єктів з фото

На цьому етапі створимо додаток на мові Python та бібліотеці OpenCV для знаходження об'єктів на фото. Будемо використовувати вже створену систему для визначення об'єктів yolo. Вона дозволяє створювати свої пресети штучного інтелекту для використання їх у програмі.

Спочатку розглянемо можливість знаходження людей на фото, оскільки вони мають найбільш виражений зовнішній вигляд, порівняно з рослинами.

Завантажимо до проекту 2 файли з налаштуванням пресету та вагою синапсів відповідно yolo.cfg та yolo.weights. Ці файли було згенеровано за допомогою сервісу roboflow.

Зчитуємо їх та визначимо налаштування (Лістинг 2.3).

Лістинг 2.3:

```
net = cv2.dnn.readNetFromDarknet("Resources\yolo.cfg",  
                                "Resources\yolo.weights")  
  
layer_names = net.getLayerNames()  
out_layers_indexes = net.getUnconnectedOutLayers()  
out_layers = [layer_names[index - 1] for index in out_layers_indexes]
```

Потім зчитуємо зображення(рис.2.5) для обробки і відправимо його на фільтрацію (Лістинг 2.4).



Рисунок 2.5 – Зображення для тесту

Лістинг 2.4:

```
image = cv2.imread(img_path)
image = apply_yolo_object_detection(image)
```

Обробимо зображення та визначимо об'єкти(Лістинг 2.5):

Лістинг 2.5:

```
blob = cv2.dnn.blobFromImage(image_to_process, 1 / 255, (608, 608),
                              (0, 0, 0), swapRB=True, crop=False)
net.setInput(blob)
outs = net.forward(out_layers)
for out in outs:
    for obj in out:
        scores = obj[5:]
        class_index = np.argmax(scores)
        class_score = scores[class_index]
        if class_score > 0:
            center_x = int(obj[0] * width)
```

```

center_y = int(obj[1] * height)
obj_width = int(obj[2] * width)
obj_height = int(obj[3] * height)
box = [center_x - obj_width // 2, center_y - obj_height // 2,
       obj_width, obj_height]
sizes.append(obj_height)
boxes.append(box)
class_indexes.append(class_index)
class_scores.append(float(class_score))

```

Далі, з розрахунку, що середня висота людини 180 см, відобразимо яких людей знайшла програма та їх пораховані висоту та ширину (Лістинг 2.6) (рис 2.6).

Лістинг 2.6:

```

chosen_boxes = cv2.dnn.NMSBoxes(boxes, class_scores, 0.0, 0.4)
for box_index in chosen_boxes:
    box = boxes[box_index]
    class_index = class_indexes[box_index]
    if classes[class_index] in classes_to_look_for:
        objects_count += 1
        image_to_process = draw_object_bounding_box(image_to_process,
                                                    class_index, box, biggest)
final_image = draw_object_count(image_to_process, objects_count)

```

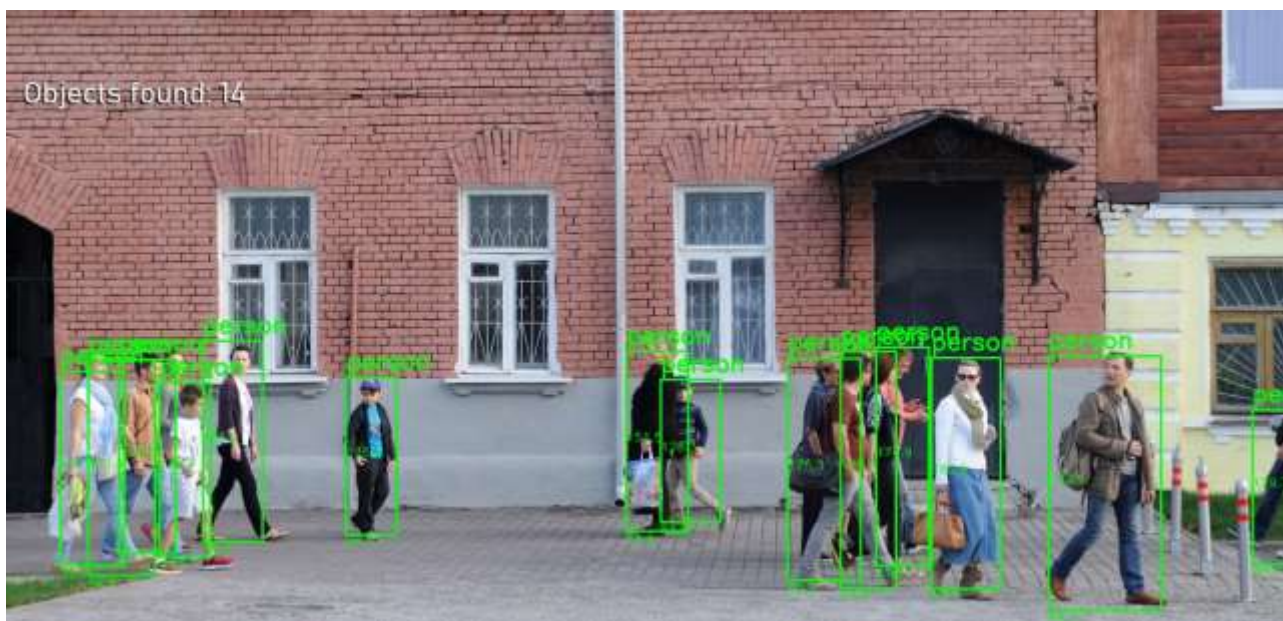


Рисунок 2.6 – Результат роботи програми

Для розрахунку характеристик треба знайти коефіцієнт зміни проєкції відносно віддаленості від фокусу камери за формулою (2.1).

$$k = \sqrt{\left| \frac{H}{2} - h \right|} \quad (2.1)$$

де H – висоти зображення;

h – відстань від найнижчого пікселя до стоп людини.

Для перевірки на соняшниках було прибрано функцію відображення розмірів, для того щоб було зрозуміло, яку кількість соняшників може виокремити програма (рис 2.7).

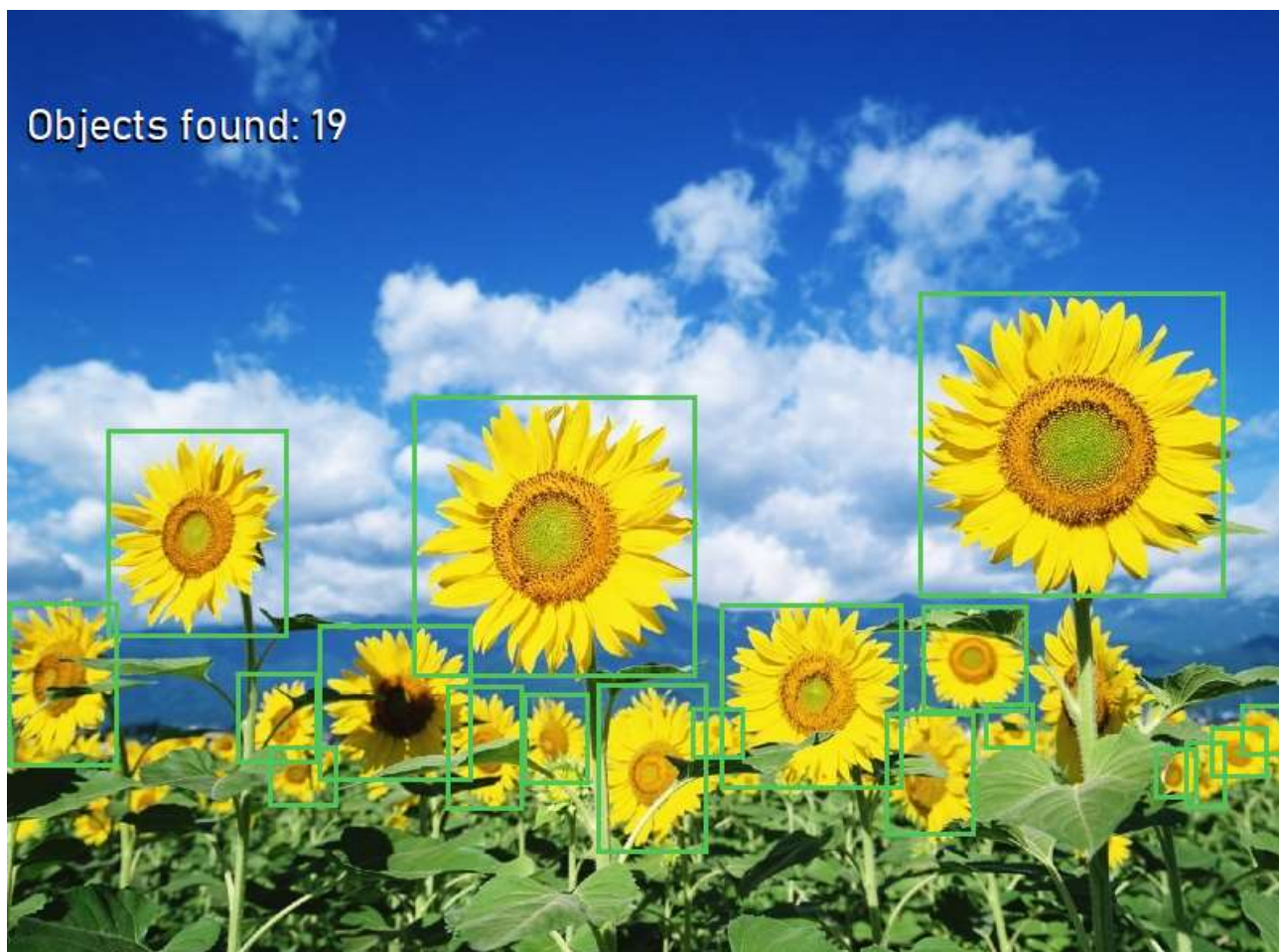


Рисунок 2.7 – Знаходження соняшників на зображенні

2.4 Створення програми для створення 3D моделі

Спочатку за допомогою Blender створимо шаблон з середніми характеристиками соняшника (рис 2.8).



Рисунок 2.8 – Шаблон соняшника

Далі напишемо програму для автоматичної модифікації соняшника (Лістинг 2.7). Для використання бібліотеки `aspose` потрібно конвертувати файл `.obj` в файл `.fbx`.

Лістинг 2.7

```
scene = Scene()
scene.open("Input\\sunflower_template.fbx")
img_path = input("Path to image(recapcha): ")
image = cv2.imread(img_path)
camera = Camera()
scene.root_node.create_child_node("camera", camera)
camera.ParentNode.Transform.Translation = Vector3(10, 10, 10)
camera.LookAt = Vector3.Origin()
opt = ImageRenderOptions()
opt.AssetDirectories.Add(image.GetDataDir() + "textures")
opt.EnableShadows = True
```

```
scene.Render(opt)  
img_name = repr(img_path).split('\\r')[-1].split('.')[0]  
scene.save("Output\\sunflower_{}.fbx".format(img_name))
```

В результаті роботи програми отримаємо файл .fbx, який потім можна конвертувати назад в .obj для збереження.

В якості вхідних даних подається зображення, або кілька зображень для більш точного результату (рис 2.9).



Рисунок 2.9 – Вхідні дані

Після його обробки буде створено нову 3D модель (рис 2.10). Програма визначить колір стебла та змінить загальний колір шаблону, після чого спробує побудувати 3D модель головки соняшника.



Рисунок 2.10 – Результат обробки 3D шаблону соняшника

Потім потрібно проаналізувати, які дані ми згенерували, а які отримали за допомогою додатку. Для цього порівняємо шаблон та отриману 3D фігуру (Лістинг 2.8), всі однакові по формі елементи виділимо синім кольором (рис 2.11).

Лістинг 2.8

```
template = Scene()  
obj1 = Scene()  
template.open("Input\\sunflower_template.fbx")  
obj1.open(img_path)  
for x in template.size.x:
```

```

for y in template.size.y:
    for z in template.size.z:
        if template.to_vector()[x, y, z] - obj1.to_vector()[x, y, z] == 0:
            obj1.to_vector()[x, y, z].diffuse_color = Vector3(Color.blue)
img_name = repr(img_path).split('\\r')[-1].split('.')[0]
obj1.save("Output\\sunflower_{ }_blue.fbx".format(img_name))

```

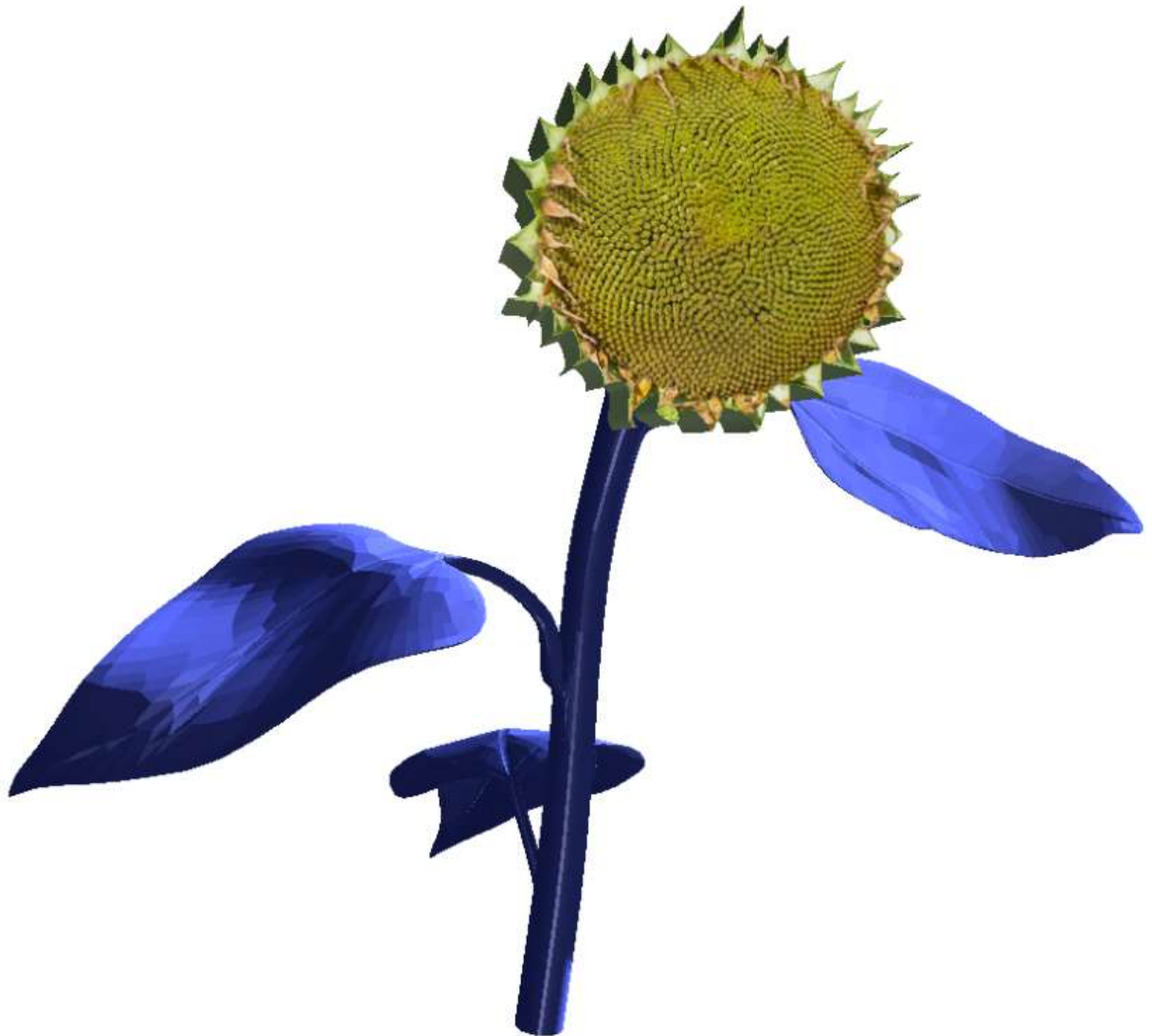


Рисунок 2.11 – Результат порівняння шаблону з отриманим об'єктом

Отже, після виконання всіх функцій програми ми отримаємо каталог 3D об'єктів отриманих при фотографуванні соняшника, а також базу даних наповнену деякими характеристиками, такими як колір та розмір.

Цифрова модель дозволяє відтворити реалістичне зображення рослини з усіма її деталями. Це допомагає наочно представити форму, структуру, листя, квіти та інші частини рослини. Така візуалізація може бути використана для навчальних цілей, наукових досліджень, виробничого процесу або просто задоволення цікавості.

2.5 Підсумки зробленої роботи

Для досягнення мети роботи – створення 3D моделі рослин, було зроблено наступні дії:

- проаналізовано існуючі методи та засоби 3D моделювання, розглянуто програмне забезпечення для роботи з 3D моделями – Reality Capture та Cognex VisionPro;
- розроблено етапи та побудовано функціональну схему для 3D моделювання рослини;
- за наданими даними Інститута олійних культур НААН України створено базу для навчання;
- за допомогою сервісу Roboflow створено модель для розпізнавання рослини;
- за допомогою бібліотеки Kivy створено мобільний застосунок для 3D моделювання;
- за допомогою бібліотеки Aspose створено остаточну 3D моделі рослини.

ВИСНОВОК

У цій дипломній роботі досліджувався процес побудови цифрової моделі рослини від розробки етапів 3D моделювання до створення мобільного додатку для спрощення побудови моделі. Було доведено практичну цінність проведеної роботи та проаналізовано вихідні дані програм.

Метою роботи було побудова 3D моделі рослини, використовуючи методи Deep Learning та різні допоміжні сервіси, які було перераховано в цій роботі.

Під час створення цифрової моделі рослини були розглянуті різні підходи та методи, які дозволяють зобразити реалістичну та деталізовану модель рослини. Були проаналізовані переваги та обмеження традиційних методів вивчення рослин і показано, що цифрові моделі розв'язують ці проблеми, надаючи нові можливості для дослідження та аналізу рослинного світу.

На основі проведених досліджень було розроблено етапи створення цифрової моделі рослини, які включають збір вхідних даних, аналіз даних, створення програм для обробки даних і побудови 3D моделі, а також її модифікації. Кожен з цих етапів відіграє важливу роль у формуванні якісної та реалістичної цифрової моделі рослини.

Основними перевагами створення цифрових моделей рослин є можливість детального вивчення морфології та структури рослини, аналіз впливу зовнішніх факторів на ріст та розвиток, прогнозування поведінки рослини в різних умовах та використання моделі в промисловому виробництві та дизайні.

Результатом створення цифрової моделі рослини є можливість візуалізувати рослинний об'єкт з усіма його деталями, проводити детальний аналіз та вивчення рослинних характеристик, симулювати та прогнозувати поведінку рослини, а також використовувати модель у виробництві та дизайні.

Важливим етапом створення 3D-моделі рослини є збір вхідних даних, який може включати фотографії, вимірювання, біологічні дослідження тощо. Ці дані забезпечують основу для точної реконструкції моделі.

Використання 3D-моделей рослин має широкі перспективи застосування. Вона може бути використана в ботаніці, екології, сільському господарстві, ландшафтному дизайні, виробництві анімації та багатьох інших галузях.

Отже, створення цифрової моделі рослини є потужним інструментом для вивчення та дослідження рослинного світу, що дозволяє отримати нові знання та розширити можливості застосування рослин у різних галузях.

Програму можна використовувати в сільському господарстві, наукових дослідженнях та ландшафтному дизайні.

ПЕРЕЛІК ПОСИЛАНЬ

1. **Синяк, В. О.** Цифрове моделювання у науці та технологіях [Текст] / В. О. Синяк, Ю. А. Горінський, М. О. Чабан. – К.: НТУУ «КПІ», 2016. – 192 с.
2. **Шинкаренко, В. І.** 3D-моделювання та віртуальна реальність [Текст] / В. І. Шинкаренко. – К.: ДІА, 2015. – 248 с.
3. **Жукова, І.О.** Інтерактивна віртуальна реальність: комп'ютерні системи і технології [Текст] / І.О. Жукова, С. М. Жуков. – К.: КНЕУ, 2012. – 227 с.
4. **Климов, Ю.П.** Графічне моделювання в архітектурі [Текст] / Ю. П. Климов, В. В. Доценко, Л. С. Карпова. – К.: Будівельник, 2010. – 256 с.
5. **Горбачук, В.В.** Комп'ютерна графіка та віртуальна реальність [Текст] / В. В. Горбачук, О. В. Горбачук, В. П. Мартиненко, Є. В. Сокурєнко – К.: Видавничий дім «Професіонал», 2009. – 479 с.
6. Вікі/Програмування – Що таке мова програмування Python [Електронний ресурс] – 2023. – Режим доступу: <https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovaniya-python/>
7. Welcome to Kivy [Електронний ресурс] – 2023. – Режим доступу: <https://kivy.org/doc/stable>
8. Brief overview of Ubuntu: features of the system and its releases [Електронний ресурс] – 2021. – Режим доступу: <https://timeweb.com/ru/community/articles/obzor-ubuntu>
9. Google Colab або Google Colaboratory: що це таке [Електронний ресурс] – 2023. – Режим доступу: <https://www.hwlibre.com/uk/google-colaboratory/>
10. Open Source Computer Vision [Електронний ресурс] – 2023. – Режим доступу: <https://docs.opencv.org/4.7.0/d1/dfb/intro.html>
11. Everything you need to build and deploy computer vision models [Електронний ресурс] – 2023. – Режим доступу: <https://roboflow.com/>

12. 3D in Python – Creating and Reading a 3D Model Scene in Python [Електронний ресурс] – 2023. – Режим доступу: <https://blog.aspose.com/ru/3d/3d-in-python/>

13. Комп'ютерний зір, віртуальна реальність та обробка зображень – частина штучного інтелекту [Електронний ресурс] – 2021. – Режим доступу: https://dut.edu.ua/ua/news-1-576-9030-kompyuterniy-zir-virtualna-realnist-ta-obrobka-zobrazhen-%E2%80%93-chastina-shtuchnogo-intelektu_kafedra-shtuchnogo-intelektu

14. Що таке фотограмметрія, як вона допомагає зберегти українську спадщину і чим корисна для геймдеву [Електронний ресурс] – 2023. – Режим доступу: <https://gamedev.dou.ua/blogs/skeiron-developer-about-photogrammetry/>

ДОДАТОК А.
Програмні коди

```
main.py
import cv2
import numpy as np
from aspose.threed import Scene
from aspose.threed.entities import Camera
from aspose.threed.render import ImageRenderOptions
from aspose.threed.utilities import Vector3
from aspose.pydrawing import Color
from art import tprint

def get_new_3d_model(img_path):
    scene = Scene()

    scene.open("Input\\sunflower_template.fbx")

    image = cv2.imread(img_path)

    camera = Camera()
    scene.root_node.create_child_node("camera", camera)
    camera.ParentNode.Transform.Translation = Vector3(10, 10, 10)
    camera.LookAt = Vector3.Origin

    opt = ImageRenderOptions()
    opt.AssetDirectories.Add(image.GetDataDir() + "textures")
    opt.EnableShadows = True
    scene.Render(opt)
```

```
img_name = repr(img_path).split('\r')[-1].split('.')[0]
scene.save("Output\\sunflower_{}.fbx".format(img_name))
```

```
def compare_template_and_new_model(img_path):
    template = Scene()
    obj1 = Scene()
    template.open("Input\\sunflower_template.fbx")
    obj1.open(img_path)

    for x in template.size.x:
        for y in template.size.y:
            for z in template.size.z:
                if template.to_vector()[x, y, z] - obj1.to_vector()[x, y, z] == 0:
                    obj1.to_vector()[x, y, z].diffuse_color = Vector3(Color.blue)

    img_name = repr(img_path).split('\r')[-1].split('.')[0]
    obj1.save("Output\\sunflower_{}_blue.fbx".format(img_name))
```

```
def apply_yolo_object_detection(image_to_process):
    height, width, _ = image_to_process.shape
    blob = cv2.dnn.blobFromImage(image_to_process, 1 / 255, (608, 608),
                                  (0, 0, 0), swapRB=True, crop=False)
    net.setInput(blob)
    outs = net.forward(out_layers)
    class_indexes, class_scores, boxes, sizes = ([[] for i in range(4)])
    objects_count = 0
```

```

for out in outs:
    for obj in out:
        scores = obj[5:]
        class_index = np.argmax(scores)
        class_score = scores[class_index]
        if class_score > 0:
            center_x = int(obj[0] * width)
            center_y = int(obj[1] * height)
            obj_width = int(obj[2] * width)
            obj_height = int(obj[3] * height)
            box = [center_x - obj_width // 2, center_y - obj_height // 2,
                   obj_width, obj_height]
            sizes.append(obj_height)
            boxes.append(box)
            class_indexes.append(class_index)
            class_scores.append(float(class_score))
biggest = max(sizes) / 180
chosen_boxes = cv2.dnn.NMSBoxes(boxes, class_scores, 0.0, 0.4)
for box_index in chosen_boxes:
    box = boxes[box_index]
    class_index = class_indexes[box_index]
    if classes[class_index] in classes_to_look_for:
        objects_count += 1
        image_to_process = draw_object_bounding_box(image_to_process,
                                                    class_index, box, biggest)

final_image = draw_object_count(image_to_process, objects_count)
return final_image

```

```

def draw_object_bounding_box(image_to_process, index, box, biggest):
    height, wid, _ = image_to_process.shape
    x, y, w, h = box
    start = (x, y)
    end = (x + w, y + h)
    color = (0, 255, 0)
    width = 2
    final_image = cv2.rectangle(image_to_process, start, end, color, width)

    start = (x, y - 10)
    font_size = 1
    font = cv2.FONT_HERSHEY_SIMPLEX
    text = classes[index]
    final_image = cv2.putText(final_image, text, start, font,
                              font_size, color, width, cv2.LINE_AA)

    length = (height - h - y)**(1/2) if height - h - y > 0 else -(h + y - height)**(1/2)
    start = (x, y + h + 15)
    font_size = .5
    width = 1
    text = str(round(w / biggest + length, 1))
    final_image = cv2.putText(final_image, text, start, font,
                              font_size, color, width, cv2.LINE_AA)

    start = (x, y + h // 2)
    text = str(round(h / biggest + length * wid / height, 1))
    final_image = cv2.putText(final_image, text, start, font,
                              font_size, color, width, cv2.LINE_AA)

    return final_image

```

```

def draw_object_count(image_to_process, objects_count):
    start = (10, 120)
    font_size = 1.5
    font = cv2.FONT_HERSHEY_SIMPLEX
    width = 3
    text = "Objects found: " + str(objects_count)

    white_color = (255, 255, 255)
    black_outline_color = (0, 0, 0)
    final_image = cv2.putText(image_to_process, text, start, font, font_size,
                               black_outline_color, width * 3, cv2.LINE_AA)
    final_image = cv2.putText(final_image, text, start, font, font_size,
                               white_color, width, cv2.LINE_AA)

    return final_image

def start_image_object_detection(img_path):
    try:
        image = cv2.imread(img_path)
        image = apply_yolo_object_detection(image)
        cv2.imshow("Image", image)
        if cv2.waitKey(0):
            cv2.destroyAllWindows()
    except KeyboardInterrupt:
        pass

```

```

if __name__ == '__main__':
    net = cv2.dnn.readNetFromDarknet("Resources\yolo.cfg",
    "Resources\yolo.weights")
    layer_names = net.getLayerNames()
    out_layers_indexes = net.getUnconnectedOutLayers()
    out_layers = [layer_names[index - 1] for index in out_layers_indexes]

    with open("Resources/coco.names.txt") as file:
        classes = file.read().split("\n")

    image = input("Path to image(recapcha): ")
    look_for = input("What we are looking for: ").split(',')
    list_look_for = []
    for look in look_for:
        list_look_for.append(look.strip())
    classes_to_look_for = list_look_for

    start_image_object_detection(image)

```

Camera.py

```

from kivy.app import App
from kivy.uix.floatlayout import FloatLayout
from kivy.uix.button import Button
from kivy.uix.camera import Camera
import time

```

```

class CameraClick(FloatLayout):

```

```

def __init__(self, **kwargs):
    super(CameraClick, self).__init__(**kwargs)

    self.cam = Camera(index = 0, play=True, size_hint=(1, .8),
pos_hint={'x':0, 'y':.18})
    self.add_widget(self.cam)

    self.butSave = Button(text="Зберегти", size_hint=(0.3, .15),
pos_hint={'x':.35, 'y':.01}, font_size=40)
    self.butSave.bind(on_press=self.capture)
    self.add_widget(self.butSave)

def capture(self, instance):
    camera = self.cam
    timestr = time.strftime("%Y%m%d_%H%M%S")
    camera.export_to_png("IMG_{}.png".format(timestr))

class main(App):
    def build(self):
        return CameraClick()

if __name__ == '__main__':
    main().run()

```

Налаштування:

!pip install buildozer

!pip install cython==0.29.19

```
!sudo apt-get install python3-pip
!sudo apt-get install build-essential
!sudo apt-get install git
!sudo apt-get install python3
!sudo apt-get install python3-dev
!sudo apt-get install ffmpeg
!sudo apt-get install libsdl2-dev
!sudo apt-get install libsdl2-image-dev
!sudo apt-get install libsdl2-mixer-dev
!sudo apt-get install libsdl2-ttf-dev
!sudo apt-get install libportmidi-dev
!sudo apt-get install libswscale-dev
!sudo apt-get install libavformat-dev
!sudo apt-get install libavcodec-dev
!sudo apt-get install zlib1g-dev
!sudo apt-get install libgstreamer1.0
!sudo apt-get install gstreamer1.0-plugins-base
!sudo apt-get install gstreamer1.0-plugins-good
!sudo apt-get install build-essential
!sudo apt-get install libsqlite3-dev
!sudo apt-get install sqlite3
!sudo apt-get install bzip2
!sudo apt-get install libbz2-dev
!sudo apt-get install zlib1g-dev
!sudo apt-get install libssl-dev
!sudo apt-get install openssl libgdbm-dev
!sudo apt-get install libgdbm-compat-dev
!sudo apt-get install liblzma-dev
!sudo apt-get install libreadline-dev
```

```
!sudo apt-get install libncursesw5-dev  
!sudo apt-get install libffi7  
!sudo apt-get install libffi-dev uuid-dev  
!sudo apt-get install libffi-dev
```