МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Інформаційна технологія штучного сприйняття роботехнічною системою лісових умов»

за напрямом підготовки 6.050101 «Комп'ютерні науки»

Виконавець роботи: студент групи ІТ-51 Даценко Дарина Сергіївна

Кваліфікаційна робота бакалавра
захищена на засіданні ЕК
з оцінкою _____ «___» _____2019 р.

Науковий керівник _____ к.т.н., доц., Шендрик В. В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів
без відповідних посилань.

Студент _____
(підпис)

Суми-2019

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Напрям підготовки – 6.050101 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____ В. В. Шендрик
«__» _____ 2019 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Даценко Дарина Сергіївна

1 Тема роботи *Інформаційна технологія штучного сприйняття роботехнічною системою лісових умов*

керівник роботи *Шендрик Віра Вікторівна, к.т.н., доцент*

затверджені наказом по університету від «_»_____2019 р. № 0834-III

2 Строк подання студентом роботи «_»_____2019 р.

3 Вхідні дані до роботи технічне завдання, вимоги до оформлення дипломної роботи, мотодичні вказівки

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, постановка задачі, проектування системи, розробка кінцевого продукту

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, існуючі рішення, недоліки, проект SEMFIRE, мета і задачі, дипломного проекту, рішення, моделювання роботи системи, діаграма варіантів використання, архітектура системи, діаграма процесів, системні конфігурації, інструменти та фреймворки, створення навчальної вибірки, результати семантичної сегментації, оцінка точності системи, результати, порівняння з аналогами, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Визначити вимоги до інформаційної системи та виконати планування робіт	3 дні	
2	Огляд методів для комп'ютерного зору, вибір нейронної мережі	4 дні	
3	Збір та підготовка даних для тренування нейронної мережі	5 днів	
4	Тренування мережі на підготовленому наборі даних	8 днів	
5	Тестування і оцінка результатів роботи нейронної мережі	6 днів	
6	Адаптація інформаційної технології штучного сприйняття для роботи в реальному часі	10 днів	

Студент _____ Даценко Д. С.
(підпис)

Керівник роботи _____ к.т.н., доц. Шендрик В. В.
(підпис)

РЕФЕРАТ

Записка: 69 стор., 27 рис., 3 додатка, 46 джерел інформації.

Мета роботи — створення інформаційної технології для ідентифікації рослинності у лісі для запобігання пожежам.

Об'єкт дослідження — процес сприйняття сухої рослинності у лісі для виконання автоматизованих очисних робіт.

Методи дослідження — згорткова нейронна мережа для семантичної сегментації зображень.

Результати — розроблено інформаційне та програмне забезпечення системи виявлення потенційно самозаймистої рослинності у лісі. Впроваджено та оцінено систему на справжньому роботі.

У першому розділі було розглянуто сучасний стан та напрямки розвитку автономних систем та їх застосування для вирішення проблем в навколишньому середовищі. Також розглянуто сучасні підходи до вирішення задач штучного сприйняття на основі нейронних мереж.

У другому розділі було сформовано мету і задачі проекту, а також обрано засоби реалізації. Опис характеристик обраних засобів та інструментів також був представлений.

У розділі 3 було здійснено проектування і моделювання технології. Створено структурно-функціональну модель і діаграму процесів, які описують взаємодію модулів і функціональні можливості проекту.

Четвертий розділ присвячений розробці технології. Описано здійснені кроки та налаштування компонентів. Проведено підготовку набору даних і тренування мережі.

В останньому розділі було проведено оцінку результатів, проаналізовано фактори, які вплинули на показники системи та зроблено висновки.

Ключові слова: ЛІСОВІ ПОЖЕЖІ, СЕМАНТИЧНА СЕГМЕНТАЦІЯ, НЕЙРОННА МЕРЕЖА, НАВЧАЛЬНА ВИБІРКА, АВТОНОМНІ РОБОТИ, ШТУЧНЕ СПРИЙНЯТТЯ, МОБІЛЬНА РОБОТОТЕХНІКА, КОМП'ЮТЕРНИЙ ЗІР.

ЗМІСТ

ВСТУП

Згідно зі статистичними даними, представленими Європейською комісією, щорічно у Європі відбувається близько 65 000 лісових пожеж [1], які призводять до спустошення регіонів та смертельних жертв [2]. Лісові пожежі мають значний вплив на економіку, що значно перевищує просту втрату деревини як основного ресурсу. Фактично, вони призводять до відсутності можливостей відновлення лісів і, отже, до нерівномірного негативного впливу на навколишнє середовище. Лісові пожежі впливають на допоміжні галузі, такі як бджільництво та лісове господарство, що призводить до міграції з сільської місцевості.

Це означає, що існує потреба в більш ефективному способі для запобігання поширення лісових пожеж. Більшість існуючих технічних рішень базуються на використанні людино-керованих або дистанційно керованих машин. Такі роботи вимагають висококваліфікованих навичок і, часто, бувають небезпечними або недостатньо ефективними.

Отже, основною метою дипломної роботи є розроблення інформаційної технології, використовуючи комп’ютерний зір та нейронну мережу, для виявлення потенційно самозаймистої рослинності у лісі для попередження пожеж. Для досягнення цієї мети необхідно вирішити наступні задачі:

- визначити вимоги до інформаційної системи та виконати планування робіт;
- провести огляд літератури, обрати й налаштувати інструменти для вирішення поточної проблеми;
- провести збір та підготовку даних;
- розробити програмне забезпечення;
- провести тестування і оцінку результатів роботи технології;

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Одним з найефективніших заходів для запобігання лісовим пожежам є сприяння процедурам технічного обслуговування ландшафту, а саме проведення «чистки» лісу, що активно знижує накопичення захаращень шляхом регулярного обрізання, скошування, збирання та утилізації лісового сміття [4]. Висушена рослинність є основною причиною лісових пожеж, тому організації з захисту природи, благодійні організації, лісові асоціації та муніципальні установи проводять заходи з очистки лісових територій [3].

Насправді, незважаючи на ресурси, вкладені в запобігання пожежам у всьому світі, кількість пожеж продовжує значно зростати з року в рік [2]. Необхідність збереження лісогосподарських територій завдяки активному скороченню накопичень лісового сміття призводить до великих інвестицій, причому основна увага приділяється необхідним людським ресурсам [3].

Виходячи з цього виникає необхідність розробити технологічні рішення, які забезпечать безпеку праці, одночасно прискорюючи операції з очистки лісових територій.

Рисунок 1.1 – Способи очистки лісу від захаращень

1.2 Огляд літератури

Автономні системи дуже часто використовуються в різних умовах та галузях. Використання таких систем особливо важливо в разі виникнення стихійних лих, наприклад урагану або повені, або техногенних катастроф, таких як вибух Чорнобильської АЕС, коли доступ до територій може бути обмежений чи навіть небезпечний. У наступних параграфах буде розглянуто різні приклади роботизованих та автономних систем.

1.2.1 Повітряні роботи

Використання безпілотних літальних апаратів (БПЛА, англ. Unmanned aerial vehicle, скор. UAV) широко використовується в таких галузях, як реагування на надзвичайні ситуації, військові та наукові дослідження, що дає більше шансів зберегти людські життя [28].

Дрони часто застосовуються для виконання завдань з використанням тактики SWARM [29] для співпраці для прийняття рішень і збільшення обсягу інформації. Рій безпілотників може обстежити набагато більшу площу разом, тому що їхня площа огляду стає більшою.

Рисунок 1.2 – Використання дронів для огляду територій

1.2.2 Водні роботи

Водні роботи можуть проводити операції на поверхні води, під водою чи на дні водоймищ. Дані роботи застосовуються для видобутку нафти і газу, корисних копалин, підводному зборі даних, у пошукових і рятівних операціях, для моніторингу підводних змін навколишнього середовища, а також у військових та оборонних цілях.

Водні роботи можуть використовуватись у заходах з очистки водоймищ. Наприклад, на поверхні озер дуже часто виникають скупчення водоростей, які час від часу необхідно очищати. Велика кількість сміття, особливо пластику потрапляє у водойми, які виносять його в океан. Це призводить до гибелі та зараження створінь, а також забруднення океану токсинами і хімікатами.

1.2.3 Наземні роботи

Наземні роботи використовуються в промислових, військових та повсякденних задачах. Це можуть бути домашні роботи, включаючи розважальних роботів і ті, що виконують певні побутові завдання, як очищення підлоги або садівництво. Це також можуть бути безпілотні автомобілі.

Наземні машини надають високу гнучкість для встановлення різних сенсорів (камер, термодатчиків, тощо), що сприяє їх використанню на відкритих територіях. Сенсори можуть використовуватися для сприйняття навколишнього середовища, наприклад, дані з камер можуть надати здатність для класифікації зовнішнього середовища.

1.2.4 Роботи для сільського та лісового господарства

Зі зменшенням кількості людей в лісовій професії [30], постійно зростає потреба в інших методах виконання трудомістких робіт [31], таких як очищення лісів. В останні роки автоматизовані системи стали новим варіантом допомоги експертам для підтримки сільськогосподарських угідь. Для виконання завдань, які можуть бути небезпечними для звичайного лісового професіонала, можуть

використовуватись роботи. Це означає, що вони можуть бути відправлені в райони з високим рівнем небезпеки [31]. Використання роботів в геодезичних дослідженнях призвело до значного збільшення кількості даних, оскільки робот здатний виконувати кілька завдань, які могли б зайняти у команди професіоналів значно більше часу [36].

Рисунок 1.3 – Приклад використання робота в сільському господарстві

1.2.5 Робототехніка та штучний інтелект

Штучний інтелект (ШІ) - це форма обчислення, принцип роботи якої тісно пов'язаний з роботою мозку людини. ШІ може бути застосований практично у всьому, від розпізнавання осіб до розробки систем, здатних виконувати складні завдання [37], як автономне водіння. ШІ може бути використаний в наступних цілях:

- Планування маршруту

  Здатність сприйняття і аналізу середовища для планування маршруту. Прикладом цього може бути програмне забезпечення для картографування (mapping) з попередніх маршрутів, за якими людина подорожувала, з метою створення не нанесених на карту маршрутів [37].

- Аналізування

  Здатність розуміти, чому відбувається ситуація. Наприклад, може бути розроблена модель ШІ для розуміння, як кліматичні умови змінюють зростання рослин у різних реальних сценаріях [37].

- Вирішення проблем

  Можливість розглянути проблему і зрозуміти, як її вирішити. ШІ може обчислити кроки, які потім можуть бути прийняті для вирішення проблеми [37].

- Симуляція (Імітація)

  Здатність симуляції конкретної ситуації. Прикладом цього можуть служити програми краш-тестів, що використовуються автомобільними компаніями. Модель ШІ зможе дати результат, який імітує ситуацію з реального життя [37].

- Сприйняття

  Здатність зрозуміти, що відбувається в конкретній ситуації. Наприклад, це може використовуватись автомобільними компаніями для розуміння стану водія автомобіля (засипання, втома) [37].

1.2.6 Комп’ютерний зір

Комп'ютерний зір - це сфера комп'ютерних наук, яка працює над тим, щоб дозволити комп'ютерам розуміти зміст цифрових зображень. Як правило, це включає розробку методів, які намагаються відтворити можливості людського зору. Розуміння змісту цифрових зображень може включати опис зображення чи його частин. Наприклад, комп’ютерний зір може використовуватись для вирішення наступних задач: розпізнавання обличь і жестів, пошук зображень, розпізнавання ручного письма, для створення безпілотних автомобілів.

Для вирішення задач комп'ютерного зору використовуються методи починаючи від математичних функцій до технологій глибокого навчання нейронних мереж.

1.2.7 Нейронні мережі

Штучна нейронна мережа (ШНМ) є формальною назвою нейронної мережі (NN), яка є формою обчислення, що має структури і функції схожі на мозок людини [7]. Мережа складається з штучних нейронів, з'єднаних зваженими зв'язками. Цей розділ містить огляд деяких основних принципів штучних мереж. Більше інформації з цього питання може бути знайдено у [7], [16], [37], [38].

Нейронні мережі можна розглядати як класифікатори, які витягають ієрархічні характеристики з вхідних даних і навчають моделі для виконання специфічних завдань. На рис. 1.5 зображено приклад простої нейронної мережі. Параметри моделі, тобто проміжні нейрони в прихованому шарі, тренуються і навчаються за допомогою методів оптимізації, для яких повинна бути визначена функцію втрат (англ. loss function) або витрат (англ. cost function). Ця функція кодує ймовірність того, що вихід нейронної мережі буде максимально наближений до бажаного.

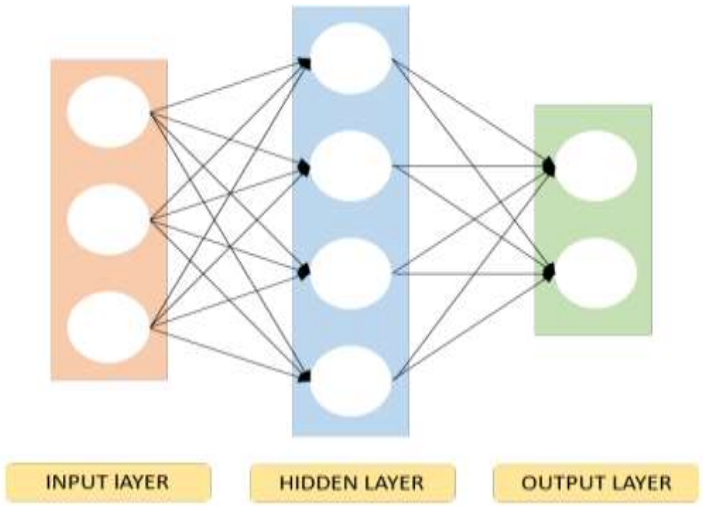


Рисунок 1.5 – Візуалізація 3-шарової архітектури нейронної мережі

Персептрони - це алгоритми, які використовуються для створення бінарних класифікаторів. Бінарні класифікатори є функцією, здатною вирішувати, якому з двох класів належить вхідний потік, представлений вектором чисел. Ці функції повністю залежать від використовуваного методу, згідно з яким персептрони були навчені.

Згорткова нейронна мережа (англ. ConvNets або CNNs) є однією з основних технологій для виконання завдач з розпізнавання та класифікації зображень. ЗНМ використовують фільтри згортки, які представлені локально зв'язаними нейронами, для виконання складних операцій. Мережа складається з вхідного, прихованого і вихідного шарів. Прихований шар може складатись з декількох шарів, що використовують математичні рівняння залежно до обраного фреймворку. На рис. 1.6 показана візуалізація однієї такої згорткової нейронної мережі.

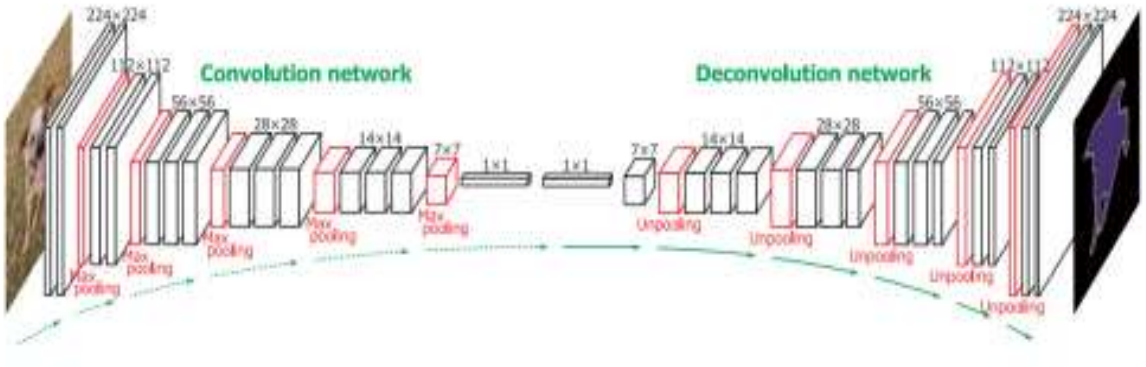


Рисунок 1.6 – Приклад архітектури згорткової нейронної мережі.

Серед найбільш відомих архітектур ЗНМ - LeNet [39], AlexNet [40], VGGNet [41], GoogleNet [43], ResNet [42]. Дослідження, розроблене Олексієм Крижевським, Іллою Сацкевером і Джеффом Хінтоном, зробило використання згорткових нейронних мереж популярним у комп'ютерному зорі [40].

1.2.8 Семантична сегментація

Уміння інтерпретувати сцену є важливою здатністю для робота, який повинен взаємодіяти з його оточенням. Знання того, що знаходиться перед роботом, є актуальним для навігації, маніпулювання або планування.

Семантична сегментація позначає кожен піксель зображення етикеткою класу і таким чином надає роботові детальну семантичну мапу оточення. У даній роботі класи об'єктів відповідають лісовому сміттю і фону. У багатокласовому середовищі класи можуть бути згруповані в кущі, дерева, листя тощо.

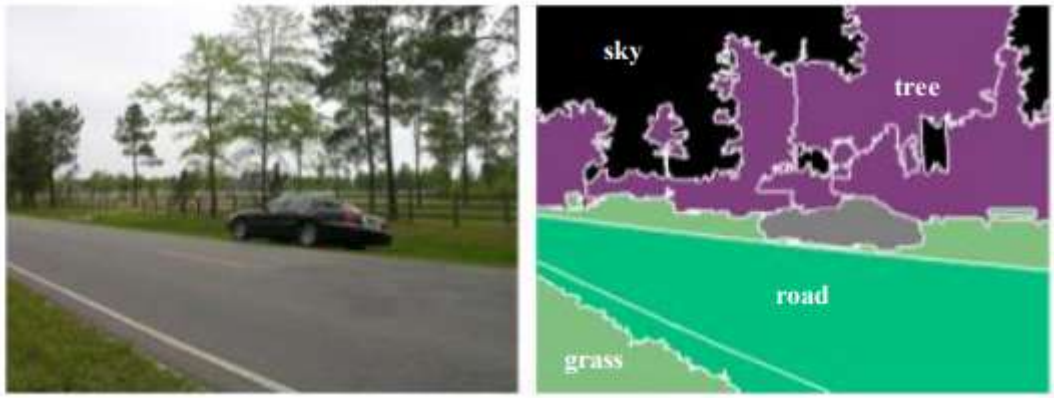


Рисунок 1.7 - Зразок семантичної сегментації зображення

Фреймворк Bonnet [19] це ЗНМ для виконання задачі семантичної сегментації. Bonnet заснований на ERFNet [20], InceptionV3 [21], і MobilenetsV2 [22], а також на передчасних вагах на чотирьох різних наборах даних. Забезпечує модульний підхід для спрощення навчання ЗНМ незалежно від використовуваного набору даних і передбачуваного завдання. Даний фреймворк має інструмент розгортання системи на реальній робототехнічній платформі. Інтерфейс навчання реалізований на Python з використанням технології TensorFlow, а інтерфейс розгортання представлений бібліотекою C++, яка може бути легко інтегрована в існуючу кодову базу з можливістю використання ROS.

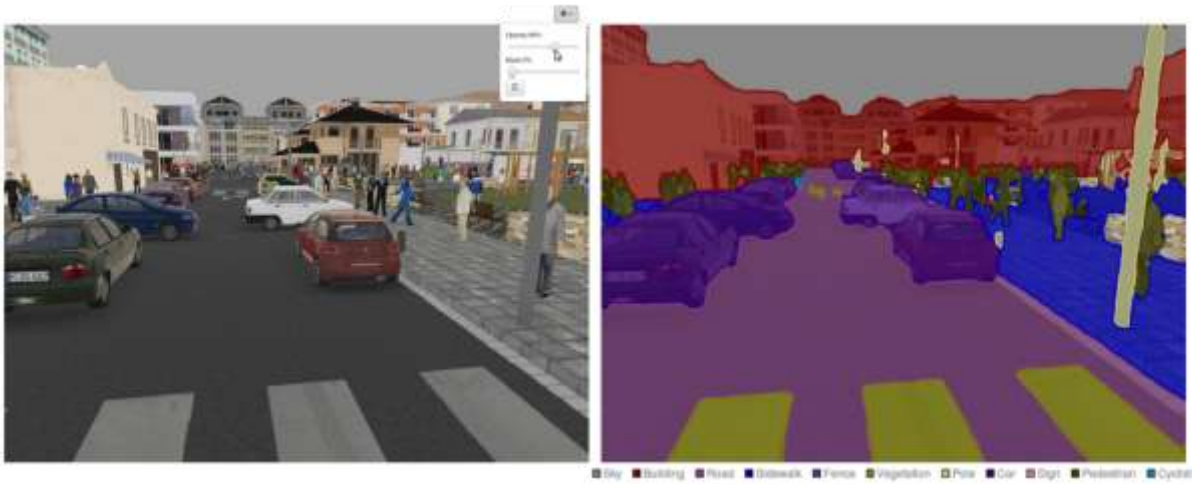


Рисунок 1.8 – Приклад результату отриманого з мережі Bonnet

1.3 Проект SEMFIRE

Проект SEMFIRE [27] є проектом, спрямованим на впровадження мережі роботів, що допомагають зменшити кількість пожежонебезпечних матеріалів у лісах. Зменшення кількості сухої рослинності у лісовому середовищі означає, що шанси утворення пожеж менші. Лісові пожежі мають великий вплив на місцеві райони, включаючи, але не обмежуючись, значне скорочення туризму, що призводить до збільшення місцевого безробіття. Роботи, що використовуються в SEMFIRE [27], працюють разом для виявлення та ліквідації районів з високим рівнем ризику шляхом сприйняття ландшафту та видалення зон з високим ризиком. У цьому процесі використовуються два роботи. Робот-розвідник - великий робот-мульчер [27] і дрон для спостереження з висоти. Робота розвідника полягає в дослідженні ділянок лісових пожеж. Дрон досягає цього, використовуючи тактику SWARM, яка передбачає використання декількох роботів-скаутів для створення карти навколишніх областей для розвідника, щоб працювати автономно або напівавтономно з використанням людського оператора. Метою робота-розвідника є видалення пожежонебезпечного матеріалу, наприклад, висушеної рослинності, це досягається з використанням штучного сприйняття, що ідентифікує регіони та їх вилучення [27].

Рисунок 1.4 – Автономний робот для очистки лісу

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Мета та задачі

Головною метою є розроблення інформаційної технології для виявлення потенційно самозаймистої рослинності у лісі для попередження пожеж. Це має бути автоматизоване рішення з використанням мобільної роботизованої платформи і/або смартфону чи ноутбуку

Для досягнення мети потрібно виконати наступні задачі. По-перше, необхідно обрати інструмент для збору даних. Це може бути камера чи інший сенсор, який буде надавати інформацію про оточення. По-друге, повинен бути обраний засіб, за допомогою якого це оточення буде розпізнаватись. Також, потрібно визначитись з технологією, яка буде використана для зв’язку з роботом.

Більш детально задачі проекту описані і технічному завданні, яке було розміщене у додатку А.

2.2 Вибір засобів реалізації

Для вирішення задачі розпізнавання рослинності у лісі, буде використовуватись штучний інтелект, за допомогою навчання згорткової нейронної мережі (ЗНМ). Моделі ЗНМ залежать від набору даних, на якому вони навчаються, що означає, що використовуваний набір даних повинен бути зроблений спеціально для вирішення конкретної задачі. Згідно з цілями проекту завдання моделі ЗНМ полягає у визначенні особливостей зображення лісу. Це означає, що зібрані дані повинні бути зображеннями різної рослинності, яка повинна бути визначена, як та, що може мати високий ризик для спричинення лісових пожеж.

В наступних підрозділах приведений опис основних компонентів та програмних засобів, які використовуються для реалізації проекту.

2.2.1 Гусеничний навантажувач Bobcat T190

Гусеничний навантажувач Bobcat T190 (рис.3) був обраний за низкою причин, головним чином з них:

• Можливість закріплення різних інструментів, а саме механічного мульчеру, який необхідний для виконання завдання подрібнення сухої рослинності;

• Ця конкретна модель повністю дозволяє використовувати електронні механізми для розробки віддалених та автономних процедур управління;

• Це добре відома, добре підтримувана машина з доступними спеціалістами для технічного обслуговування.

Рисунок 2.1 – Гусеничний навантажувач Bobcat

2.2.2 Камера Intel Realsense D345

Intel RealSense D345 є базовим сенсором в даному проекті, яка буде закріплена на корпус Bobcat машини, описаної вище. Дана камера має два датчики глибини, RGB і інфрачервоний датчики. Використання цієї камери надає можливість отримання потоку RGB кадрів достатньої роздільної здатності. Також використання Intel Realsense є доцільним завдяки його невеликим розмірам.

Рисунок 2.2 – Intel Realsense D345 камера

2.2.3 Фреймворк Robot Operating System

Robot Operating System (ROS) є основним способом запуску програмного забезпечення на роботі. ROS надає сервіси, які розробник може використовувати для керування роботом і збору даних. Основні відомості про те, як використовувати ROS, можна знайти в [45]. Вузли (nodes) ROS використовуються для керування роботом. За допомогою вузлів можна керувати фізичними компонентами, такими як колеса і камера, або відправляти повідомлення (messages) на робота для ініціювання сценаріїв. Тему (topic) ROS можна використовувати для зв'язку з роботом. Існують різні типи тем, які розробник може опублікувати (publish) або підписатись (subscribe) на них. Це дозволяє опубліковувати і дистанційно отримувати вихідні дані з різних скриптів. Повідомлення ROS можуть бути простих типів даних, як рядки (string) або цілі числа (int), а також більш складних типів, на зразок зображення (Image) різних каналів. Розробник може також створювати власні користувацькі класи для надсилання персоналізованих типів даних.

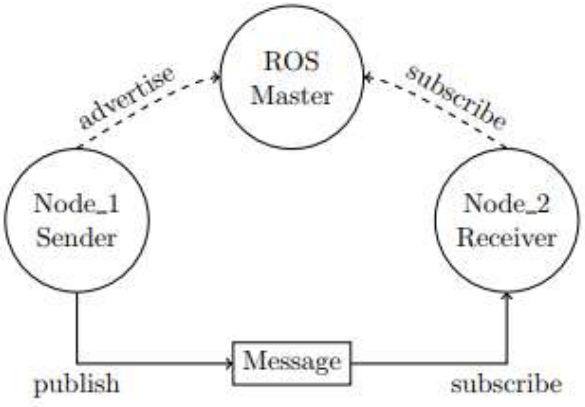


Рисунок 2.3 – Схема роботи системи ROS

2.2.4 Інструмент Labelme

Семантична сегментація виконується нейронною мережею, яка вимагає позначених мітками класів зображень. Для досягнення цієї мети був використаний інструмент Labelme [44]. Labelme - це інструмент, який дозволяє створювати полігони, кола і точки для встановлення меж класів на зображенні. Приклад процесу маркування наведено на рис. 2.4.

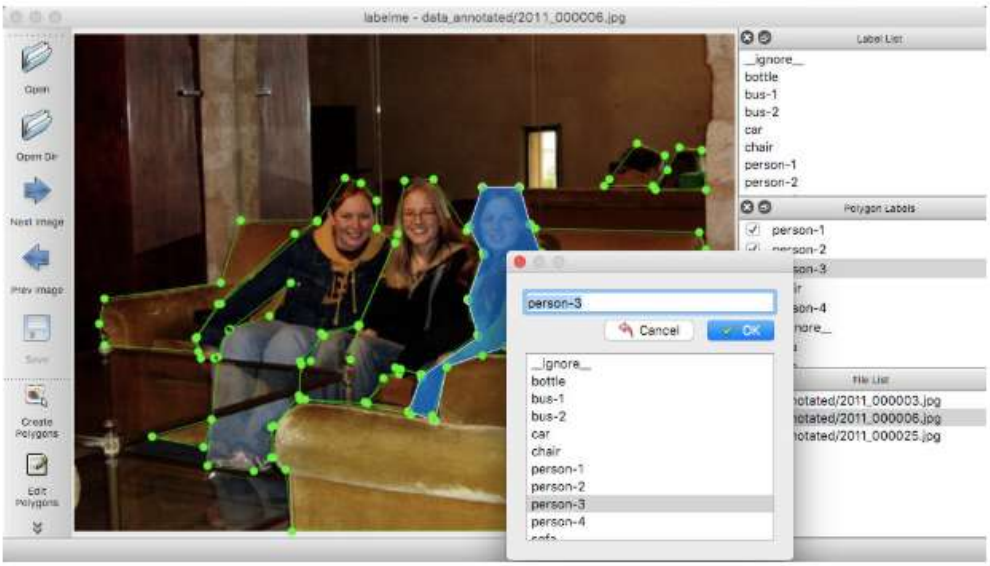


Рисунок 2.4 – Приклад використання Labelme

2.2.5 Бібліотека OpenCV

OpenCV - це бібліотека комп'ютерного бачення для візуальних обчислень і обробки зображень. В даному проекті OpenCV використовується для маніпуляцій над зображеннями на вході і виході системи, тобто проводиться попередня і пост обробка зображень [46].

2.2.6 Фреймворк Bonnet

Bonnet - це фреймворк для семантичної сегментації RGB зображень, який має конвеєр навчання розроблений на Python і бібліотеку розгортання на C++. Бібліотеку розгортання можна використовувати автономно або як вузол ROS.

Архітектура Bonnet (див. рис. 2.5) заснована на ERFNet [20], InceptionV3 [21], і MobilenetsV2 [22]. Кодова база дозволяє навчати модель на декількох GPU.

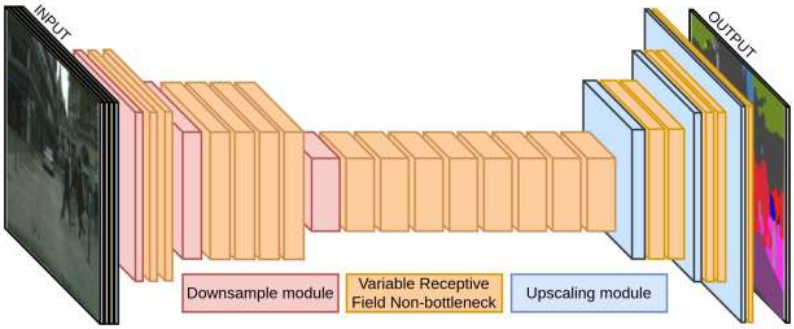


Рисунок 2.5 – Візуалізація шарів ЗНМ Bonnet

Використання Bonnet розділено на два етапи. По-перше, навчання моделей для виведення піксельно-точних семантичних класів з конкретного набору даних через Python інтерфейс, який здатний отримати доступ до TensorFlow API для навчання нейронної мережі. По-друге, розгортання моделі в реальній робототехнічній платформі за допомогою C++ інтерфейсу, що дозволяє користувачеві зробити висновок з роботи підготовленої моделі в існуючій програмі C ++, або в системі з підтримкою ROS. На рис. 2.6 показаний модульний опис фреймворку від прикладного до апаратного рівня, який детальніше буде пояснено в наступних розділах.

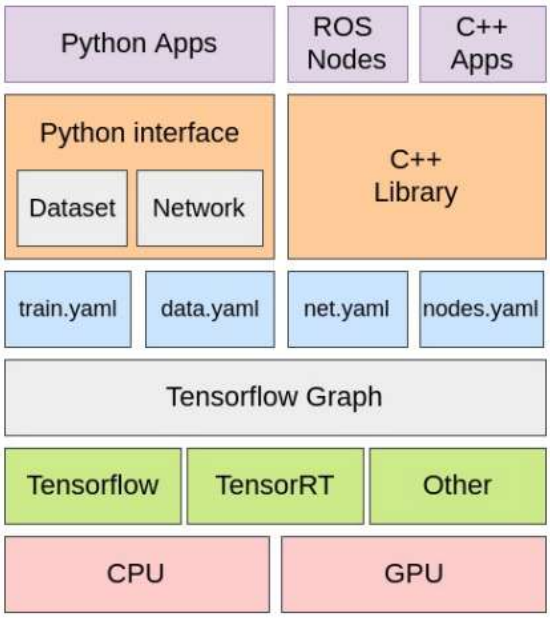


Рисунок 2.6 – Структура Bonnet фреймворку

Нижче наведено приклад роботи Bonnet фреймворку для використання сегментації в різних сценаріях (рис. 2.7).

Рисунок 2.7 – Приклади семантичної сегментації за допомогою Bonnet

Як видно з рис. 2.7, технологія семантичної сегментації може використовуватись для високоточної піксельної класифікації об’єктів чи регіонів інтересу.

3 ПРОЕКТУВАННЯ СИСТЕМИ

3.1 Структурно-функціональне моделювання

Функціональна модель необхідна для відображення взаємозв’язків між модулями системи у вигляді ієрархії взаємопов'язаних діаграм. Для створення функціональної моделі було використано програмний продукт Erwin Process Modeler.

Контекстну діаграму процесу сегментації зображень представлено на рис. 3.1.

Рисунок 3.1 – Контекстна діаграма

Дана діаграма побудована з блоків діяльності, що відображають взаємодію між функціями та ресурсами, які потрібні для забезпечення діяльності системи.

Входом до функції «Класифікація зображення» є вхідне зображення.

На виході буде ROS-повідомлення.

Механізмами є персональний комп’ютер та зв'язок з системою ROS (Robot Operating System).

На першому рівні деталізації моделі (рис. 3.2) головна батьківська діаграма декомпозується на наступні блоки:

- попередня обробка зображення;
- класифікація зображення;

– конструювання ROS-повідомлення.

Рисунок 3.2 – Діаграма декомпозиції IDEF0

Вхідними стрілками до діяльності «Попередня обробка зображення» є «Вхідне зображення; вихідними – «Оброблене зображення»; стрілками контролю – «Вимоги проекту SEMFIRE» та «Навчальна вибірка»; стрілками механізмів – «ROS».

Вхідними стрілками до діяльності «Класифікація зображення» є «Оброблене зображення»; стрілками контролю – «Вимоги проекту SEMFIRE»; стрілками механізмів – «ПК».

Вхідними стрілками до діяльності «Конструювання ROS-повідомлення» є «Сегментоване зображення»; вихідними – «ROS-повідомлення»; стрілками контролю – «Вимоги проекту SEMFIRE» ; стрілками механізмів – «ROS».

Наведена нижче схема (рис. 3.3) зображує взаємодію між експертом навчання, контролером камери, системою візуалізації, системою прийняття рішень і інформаційною технологією.

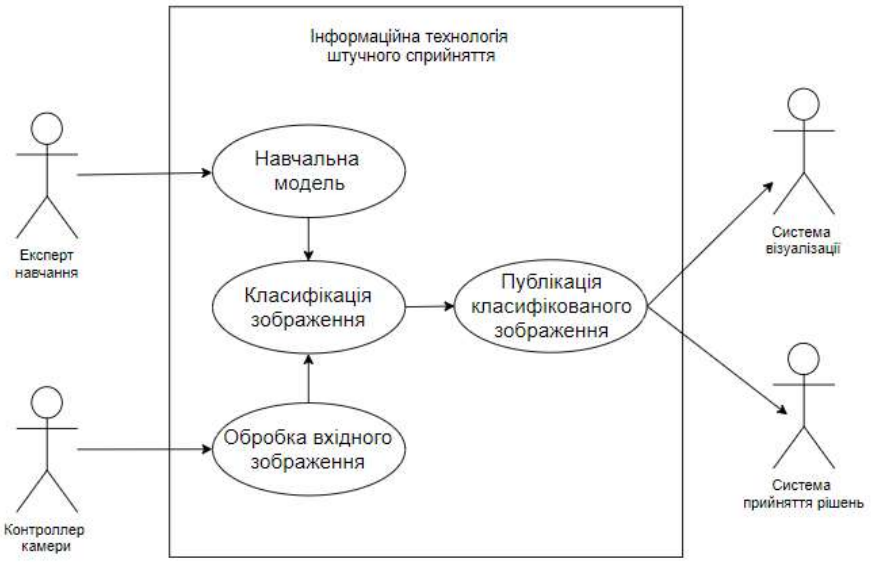


Рисунок 3.3 – Use Case діаграма

3.2 Моделювання системи

Для візуального сприйняття систему можна розділити на три підсистеми, такі як створення і підготовка набору даних (dataset), налаштування нейронної мережі для сегментації зображень (CNN preparation) і отримання фінальних результатів (Final Results) (рис. 3.4).

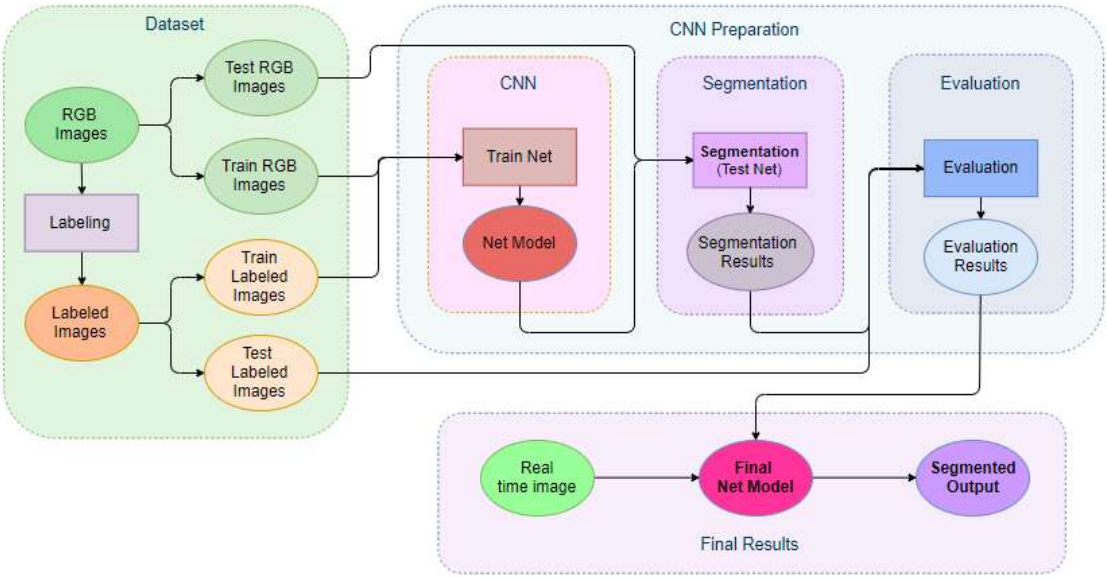


Рисунок 3.4 – Діаграма інформаційних потоків

3.2.1 Опис системи

Зображення у вигляді потоку кадрів з Intel Realsense камери поступає на image subscriber системи ROS. Далі відбувається попередня обробка зображення, наприклад, зміна роздільної здатності та контрасту, і передача до нейронної мережі. На наступних етапах відбувається пост обробка вихідного зображення, його публікація у систему ROS і на робота. Візуальна схема системи представлена на рис. 3.5.

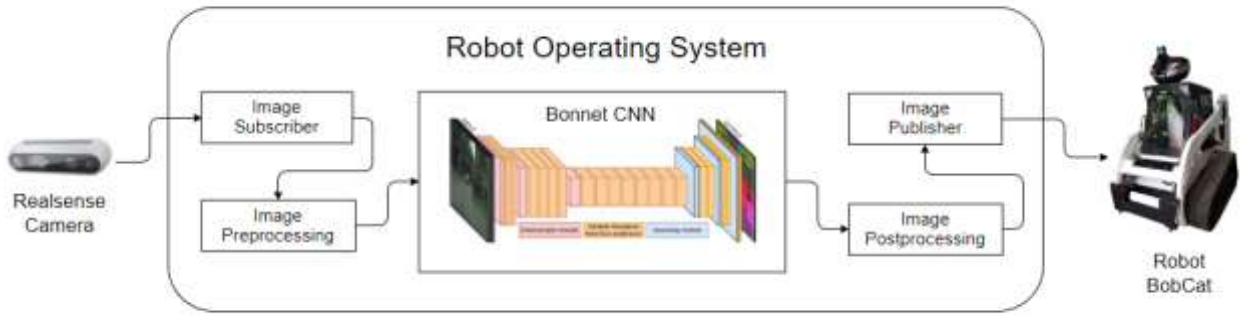


Рисунок 3.5 – Схема архітектури системи

Вихідне зображення поступає на робота, який локалізує класифіковані регіони інтересу і приймає рішення щодо видалення рослинності.

4 РОЗРОБКА КІНЦЕВОГО ПРОДУКТУ

4.1 Налаштування робочого середовища

Перша частина у процесі розробки проекту полягала у створенні і налаштуванні робочого середовища. Нижче описані кроки, які необхідно здійснити.

4.1.1 Налаштування Ubuntu OS і ROS

Для створення автономних систем для роботів зазвичай використовується Ubuntu. Тому у якості операційної системи було вирішено використовувати стабільну версію Ubuntu 16.04. На Ubuntu 16.04 необхідно встановити версію ROS Kinetic. Після завершення установки наступний крок полягає в оновленні змінних середовища на ОС. Це важливий крок, оскільки він надає операційній системі шлях до пакетів і команд ROS. Встановлення робочого простору catkin. Робоче середовище catkin - це каталог, який дозволяє користувачам створювати або змінювати існуючі пакунки (packages). Catkin містить папки build, source папку та development. Останнім процесом є пошук bash файлу після використання команди catkin make. Більш детальну інформацію про встановлення та налаштування ROS можна знайти у [45]. ROS поставляється з версією OpenCV3 і python; і так як Bonnet потребує наявність OpenCV, то додатково встановлювати цю бібліотеку не потрібно.

4.1.2 Налаштування фреймворку Bonnet

Наступним кроком було налаштування фреймворку Bonnet з підтримкою GPU. Bonnet може працювати і без підтримки GPU; однак запуск Bonnet на графічному процесорі CUDA GPU робить його значно швидшим. Підтримка GPU вимагає встановлення CUDA та CuDNN; також машина повинна мати NVIDIA

GPU, в даному проекті було використано NVIDIA GeForce GTX1050 GPU і встановлено відповідний драйвер nvidia-410.

Для роботи фреймворку також було встановлено:

- Tensorflow 1.12
- CUDA9
- CuDNN7

На рис. 4.1 зображено візуальну схему необхідних компонентів для роботи фреймворку Bonnet.

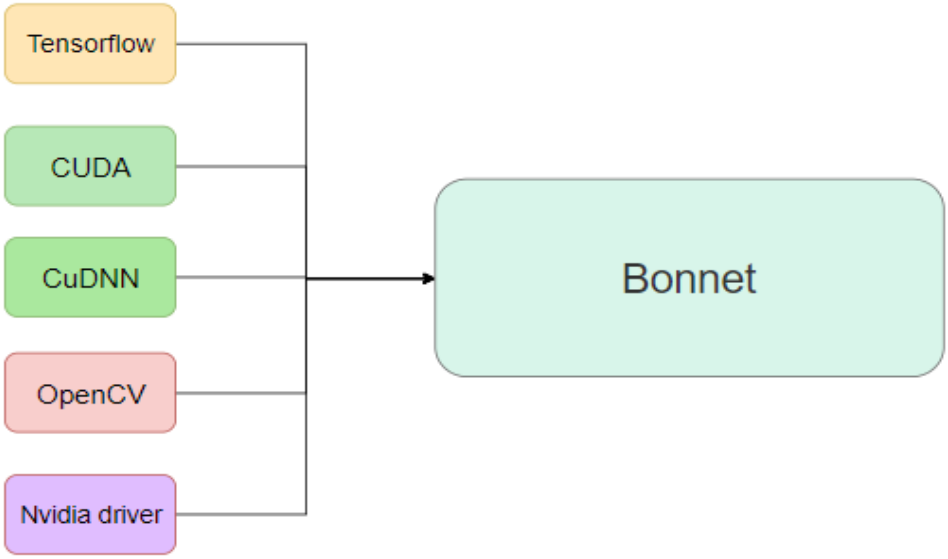
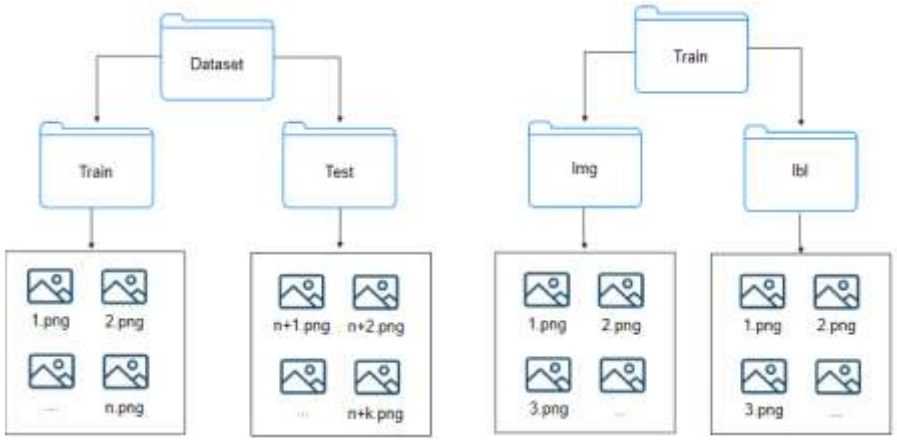


Рисунок 4.1 – Візуальна схема залежностей для Bonnet

4.2 Створення набору даних і тренування мережі

4.2.1 Організація тек

Набір даних із зображеннями лісу має бути розділений на дві частини: тренування і тест, як правило, у пропорції 80/20 відповідно (рис. 4.2а). Для подальшого використання набору даних у нейронній мережі, зображення повинні бути відсортовані в img (rgb зображення) та lbl (анотації) папки (рис. 4.2б).



(а) (б)

Рисунок 4.2 – Організація тек

4.2.2 Створення датасету

Для створення набору даних було використано Intel Realsense камеру, що була підключена до ноутбуку через USB 3.0 кабель. Збирання даних (рис. 4.3) зайняло декілька днів, так як необхідно було переконатися, що максимальну кількість змінних середовища враховано для підвищення точності класифікації рослинності у лісі.

(а) Придатне для використання зображення

(б) Зміна перспективи з рухом робота

(в) Розмите зображення, так як робот переміщається в лісі

(г) Відмінності в освітленні, звичайно, будуть виникати

Рисунок 4.3 – Зображення з датасету

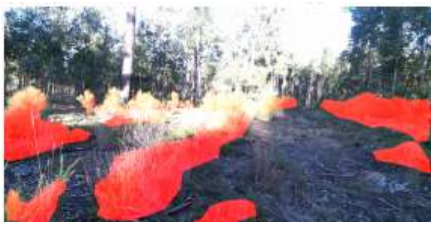
4.2.3 Анотація даних

Після цього дані анотувались за допомогою програми Labelme. У рамках цілей даного проекту було позначено набір даних з 200 зображень.

Рисунок 4.4 – Приклад нанесення мітки на зображення

Результати процесу маркування зберігаються у файлі json, які потім необхідно конвертувати у PNG чи JPEG формат для використання нейронною мережею. За допомогою створеного python скрипту зображення конвертується у бажаний формат з необхідною роздільною здатністю з метою зменшення кількості пікселей для обробки мережею, і, як наслідок, зменшення кількості часу на тренування моделі. Також для візуального перегляду анотацій було створено програму для накладання маркувань і оригінального RGB зображення (рис. 4.5).

(а) оригінальне зображення (б) анотоване зображення



(в) накладення маркування на оригінал

Рисунок 4.5 – Приклад анотованих даних

4.2.4 Налаштування Bonnet і тренування моделі

Для налаштування мережі необхідно змінити конфігурації файлів "data.yaml", "net.yaml" і "train.yaml", які відповідають за параметри мережі та набору даних (датасету).

У файлі "data.yaml" необхідно вказати наступні основні параметри:

- name - відповідає за назву датасету, яка в контексті даного проекту буде “forest”;
- data_dir – шлях до папки з зображеннями і мітками лісу (папки img, lbl);
- label_map - назва класів на зображенні, було вказано два класи: “fuel” і “background”;
- label_remap – вказується кількість класів і відповідне число;
- color_map – колір кожного класу з відповідним числом, вказаним в параметрі label_remap;
- img_prop – розміри зображення і кількість каналів.

У файлі net.yaml вказуються параметри нейронної мережі:

- name – назва датасету (“forest”);
- n_k_lyr – кількість фільтрів у кожному шарі;
- train_lyr – список шарів для тренування.

Файл train.yaml відповідає за налаштування нейронної мережі для тренування. Основні параметри:

- max_epochs - кількість епох тренування;
- batch_size – залежить від кількості зображень та потужності GPU;
- gpus – кількість використовуваних GPU;
- save_imgs – зберігати зображення в процесі навчання моделі чи ні.

Після налаштування конфігурацій та організації тек датасету з зображеннями та мітками запускається програма для тренування моделі мережі:

```
$ ./cnn_train.py -d data.yaml -n net.yaml -t train.yaml -l /path/log/
```

Де параметри означають наступне:

- *"cnn_train.py"* містить код, необхідний для завантаження конфігураційних файлів і взаємодії з набором даних і мережевими класами, для того, щоб навчити систему.
- Можна вказати log теку, в якій тренована модель, tensorboard log і деякі прогнозовані зображення будуть збережені під час тренування.

Після тренування моделі буде сформовано файл моделі у форматі `acc` і `iou`. Ця модель буде використовуватись для сементичної сегментації зображень. В папці `image` зберігаються одне або багато зображень для тестування. Для запуску тестування моделі використовуємо `cnn_use.py` :

```
$ ./cnn_use.py -l /path/log/ -p /path/pretrained -i /path/image
```

4.2.5 Інтеграція з ROS

Для комунікації з роботом в проект було інтегровано систему ROS (Robot Operating System). Це було реалізовано шляхом створення повідомлення типу даних `sensor_msgs/Image.msg`, яке потім за допомогою ROS опубліковувача (publisher) надсилається в систему ROS. Для робота було створено підписник (subscriber), який збирає повідомлення з зображеннями.

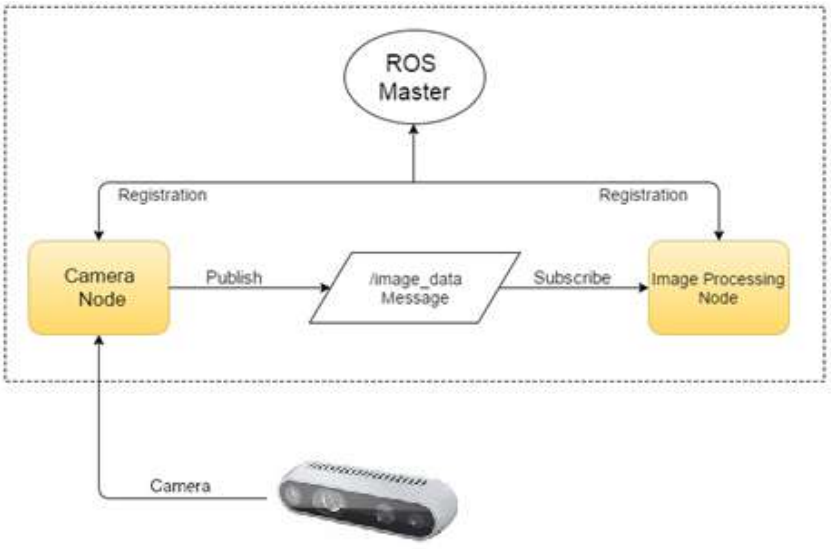


Рисунок 4.6 – Візуалізація ROS процесів

5 РЕЗУЛЬТАТИ СЕМАНТИЧНОЇ СЕГМЕНТАЦІЇ

Після виконання кроків описаних у попередньому розділі були отримані результати (рис. 5.1). Червоним кольором виділено регіони з рослинністю, яка має бути видалена роботом.

Рисунок 5.1 – Класифікація зображень лісу

5.1 Продуктивність обробки зображення

Метою тесту продуктивності є вимірювання часу, необхідного для виведення міток класу одного зображення з роздільною здатністю 705 x 360 пікселів без урахування читання і запису файлів. На графічному процесорі Nvidia GeForce 1050x потрібно 0.1 секунди, щоб визначити мітки класу на одному зображенні. Якщо запустити систему на CPU, то процес класифікації одного зображення займає приблизно 9 секунд. Час виконання на CPU може бути зменшено до приблизно 3.8 секунди в середньому, якщо паралельно запускаються п'ять процесів.

5.2 Оцінка результатів

Якісна оцінка результатів використання мережі Bonnet для сегментації пікселів показана в табл. 1 і табл. 2. Метрика, що використовується в процедурі оцінювання - це середній перетин через об'єднання (англ. mean Intersection over Union), яка є загальноприйнятою метрикою, що використовується при оцінці продуктивності сегментації зображення [21]. IOU обчислюється за формулою (5.1).

$$mIoU = \frac{1}{C} \sum_{i=1}^{C} \frac{\text{TruePos}_j}{\text{TruePos}_j + \text{FalsePos}_j + \text{FalseNeg}_j} \tag{5.1}$$

Пояснення змінних формули (5.1):

- **C** – кількість класів;
- **TruePos** – кількість **правильно** класифікованих пікселей, які **відносяться** до класу;
- **FalsePos** – кількість **неправильно** класифікованих пікселей, які **не відносяться** до класу;
- **FalseNeg** – кількість **неправильно** класифікованих пікселей, які **відносяться** до класу.

Всього було протестовано 500 різних зображень лісу. Результати роботи класифікації зображень представлено в табл. 1. Найкращий результат у 93% був отриманий з вибірки зображень з роздільною здатністю 470 на 240 пікселів, та кількістю епох тренування – 10000, так як кількість епох прямо пропорційно впливає на результат.

У випадку тренування на зображеннях з більшою роздільною здатністю більша кількість пікселів, що була неправильно анотована погіршує результат класифікації. Також необхідно врахувати, що навчальна вибірка була обмежена кількістю 200 зображень, тобто у разі збільшення кількості якісних даних можливо досягти кращих результатів. Одним з факторів, який впливає на

результати є те, що при створенні навчальної вибірки людина може помилятись з тим, які ділянки повинні бути ідентифіковані, як потенційно сприятливі для початку пожеж.

Таблиця 1 – Результати семантичної сегментації зображень

Кількість епох	Роздільна здатність	Точність
1000	705 x 360	0.51
5000	705 x 360	0.68
10000	705 x 360	0.89
1000	1410 x 720	0.59
5000	940 x 480	0.74
10000	470 x 240	0.93

Отже, як видно з табл. 1, поставлена мета і задача дипломної роботи були виконані, і в подальшому може використовуватись роботизованою системою для очистки лісових територій.

ВИСНОВКИ

У цій роботі представлено проблему лісових пожеж та описано їх негативний вплив на довкілля і суспільство. Проведено опис сучасних методів вирішення проблеми та обрано кращий варіант, що вплине на зменшення кількості пожеж у лісі.

Для сприяння вирішенню проблеми було запропоновано використання робота, розробленого в проекті SEMFIRE [27]. Для того, щоб робот розумів, які ділянки необхідно очистити було застосовано класифікацію лісових зображень. За допомогою даного підходу було виявлено регіони з рослинністю, що може загорітись.

Методом класифікації зображень обрано семантичну сегментацію, яка надає можливість класифікувати кожний піксель зображення. Для цього було використано фреймворк Bonnet. Конфігурації і навчання нейронної мережі проводились з використанням виключно RGB даних. Навчальна вибірка була сформована з зображень лісових умов, з попереднім виділенням регіонів інтересу висушеної рослинності у лісі. Для цього було здійснено декілька походів до лісу з метою збору якісних даних за допомогою Realsense камери.

Після навчання і тестування моделі нейронної мережі було проведено оцінку результатів. Найкращий отриманий результат класифікації рослинності, що повинна бути знищена, – це 93 %. Згодом, було впроваджено та ретельно оцінено систему на справжньому роботі. Результати свідчать про те, що система виконала поставлену задачу і може працювати на рівні близько 10 Гц, що є достатнім для оперативного проведення очисних робіт в лісах.

Для покращення результатів можливо підготовити більший за обсягом та якісніший набір навчальних даних. розроблено інформаційне та програмне забезпечення системи виявлення потенційно самозаймистої рослинності у лісі

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. J. San-Miguel-Ayanz, E. Schulte, G. Schmuck, A. Camia, P.Strobl, G. Liberta, C. Giovando, R. Boca, F. Sedano, P.Kempeneers, and D. McInerney, Comprehensive monitoringof wildres in Europe: the European Forest Fire InformationSystem (EFFIS), European Commission, Joint Research Cen-tre Italy, 2012.

2. F. Moreira, and G. Pe'er Agricultural policy can reducewildres. Science, 359(6379), 2018, pp. 1001.

3. C. Ribeiro, S. Valente, C. Coelho, and E. Figueiredo, A lookat forest res in Portugal: technical, institutional, and socialperceptions. Scandinavian Journal of Forest Research, 30(4),2015, 317-325.

4. F. C. Dennis, Fire-resistant landscaping. Colorado State Uni-versity Cooperative Extensio, 1999.

5. C. Slappendel, I. Laird, I. Kawachi, S. Marshall, and C. Cryer,Factors aecting work-related injury among forestry workers: A review. Journal of safety research, 24(1), 1993, 19-32.

6. L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A.L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint, abs/1606.00915, 2016.

7. Нейронні мережі - шлях до глибинного навчання [Електронний ресурс]. – Режим доступу : URL : **https://codeguida.com/post/739**.

8. Andrej Karpathy. Convolutional Neural Networks for Visual Recognition (Course Notes) [Електронний ресурс] – Режим доступу : URL : http://cs231n.github.io/.

9. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV), 2015.

10. Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: arXiv preprint arXiv,

abs/1602.07554, 2015.

11. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint, abs/1409.1556, 2014.

12. V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI), 2017.

13. O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. arXiv preprint, abs/1505.04597, 2015.

14. K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2016.

15. H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. arXiv preprint, abs/1612.01105, 2016.

16. S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV), 2015.

17. Paszke, A. Chaurasia, S. Kim, E. Culurciello ENet. A Deep Neural Network Architecture for Real-Time Semantic Segmentation, arXiv preprint , abs / 1606.02147, 2016

18. P. Ghamisil, M. Couceiro, F. Martins, J. Benediktsson. Multilevel image segmentation based on fractional-order Darwinian particle swarm optimization. IEEE Transactions on Geoscience and Remote sensing, 52 (5), 2382-2394, 2014.

19. Фреймворк Bonnet [Електронний ресурс]. – Режим доступу : URL : https://github.com/PRBonn/bonnet

20. Eduardo Romera , Jose M. Alvarez , Luis M. Bergasa, Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. IEEE Trans. on Intelligent Transportation Systems (ITS)], 19(1):263– 272, 2018.

21. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision, arXiv preprint, 2015 abs/1704.08545, 2017.

22. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks]. arXiv preprint, 2018.

23. Мова програмування Python [Електронний ресурс]. – Режим доступу : URL : https://uk.wikipedia.org/wiki/Python

24. Мова програмування C++ [Електронний ресурс]. – Режим доступу : URL : https://uk.wikipedia.org/wiki/C%2B%2B

25. Robot Operating System: [Електронний ресурс] // Вікіпедія – вільна енциклопедія. – Режим доступу : URL : https://en.wikipedia.org/wiki/Robot_Operating_System

26. Ralph D Nyland. Decline in forestry education enrollment The decline in forestry education enrollment-Some observations and opinions La disminuci´on de la matr´ıcula en educaci´on forestal-algunas observaciones y opiniones. Tech. rep. 2. 2008, pp. 105–108. url: https://scielo.conicyt. cl/pdf/bosque/v29n2/Art01.pdf.

27. Micael S Couceiro, Joao F Ferreira, and Rui P Rocha. SEMFIRE : Towards a new generation of forestry maintenance multi-robot systems SEMFIRE : Towards a new generation of forestry maintenance multi-robot systems. 2019.

28. Marzena Polka, Szymon Ptak, Lukasz Kuziora, and Aneta Kuczynska. “The Use of Unmanned Aerial Vehicles by Urban Search and Rescue Groups”. In: Drones - Applications September (2018). doi: 10 . 5772 / intechopen.73320.

29. Mauro S Innocente and Paolo Grasso. Swarms of autonomous drones selforganised to fight the spread of wildfires. Tech. rep. 2018. url: http : //ceur-ws.org.

30. Ralph D Nyland. Decline in forestry education enrollment The decline in forestry education enrollment-Some observations and opinions La disminuci´on de la matr´ıcula en educaci´on forestal-algunas observaciones y opiniones. Tech. rep. 2. 2008, pp. 105–108. url: https://scielo.conicyt. cl/pdf/bosque/v29n2/Art01.pdf.

31. Richard Parker, Karen Bayne, and Peter W Clinton. “Robotics in forestry”. In: April (2016).

32. Pranay Agrawal and Bishakh Bhattacharya. “AQUATIC MULTI-ROBOT SYSTEM FOR LAKE CLEANING”.

33. Janine Rybka. Lake Erie: The Costs and Benefits of Clean Water. url: https://www.cuyahogaswcd.org/blog/2018/08/07/lake-erie-thecosts-and-benefits-of-clean-water (visited on 04/25/2019).

34. Aleks Buczkowski. How accurate is your drone survey? Everything you need to know. 2017. url: http : / / geoawesomeness . com / accurate - drone-survey-everything-need-know/ (visited on 10/26/2018).

35. Kiana Ehsani, Hessam Bagherinezhad, Joseph Redmon, Roozbeh Mottaghi, and Ali Farhadi. Who Let The Dogs Out? Modeling Dog Behavior From Visual Data. Tech. rep. arXiv: 1803 . 10827v1. url: https : //pjreddie.com/media/files/papers/1803.10827.pdf.

36. M.S. Essers and T.H.J. Vaneker. “Developing Concepts for Improved Efficiency of Robot Work Preparation”. In: Procedia CIRP 7 (2013), pp. 515– 520. issn: 2212-8271. doi: 10.1016/J.PROCIR.2013.06.025. url: https: //www.sciencedirect.com/science/article/pii/S2212827113002941.

37. Chethan Kumar GN. Artificial Intelligence: Definition, Types, Examples, Technologies. 2018. url: https://medium.com/@chethankumargn/artificialintelligence-definition-types-examples-technologies-962ea75c7b9b (visited on 04/16/2019).

38. Uk-ras White Papers. “Artifical Intelligence and Machine Learning Artificial Intelligence and Robotics”. In: (2017).

39. Freund, Y., Schapire, R.E.: A Desicion-Theoretic Generalization of On-line Learning and an Application to Boosting. Journal of Computer and System Sciences 55, 119–139 (1997)

40. Taigman, Y., Yang, M., Ranzato, M.A., Wolf, L.: Deepface: Closing the Gap to Humanlevel Performance in Face Verification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colombus, Ohio, USA, 1701–1708 (2014)

41. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-scale Image Recognition. Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA, 1–14 (2015)

42. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image

Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 770–778 (2016)

43. Szegedy, C., et al.: Going Deeper with Convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, Massachusetts, USA, 1–9 (2015).

44. B. C. Russell, A. Torralba, K. P. Murphy, W. T. Freeman, LabelMe: a database and web-based tool for image annotation. MIT AI Lab Memo AIM-2005-025, September, 2005. PDF

45. William D. Smart, Brian Gerkey, Morgan Quigley. Programming Robots with ROS. — O'Reilly Media, Inc., 2015. — ISBN 9781449323899.

46. Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue. pp. 40:40–40:56. doi:10.1145/2181796.2206309.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи «Інформаційна технологія штучного сприйняття роботехнічною системою лісових умов»

Суми 2019

1 Призначення й мета створення інформаційної системи

1.1 Призначення інформаційної системи

Інформаційна система повинна виявляти самозаймисті речовини у лісових умовах для їх подальшої ліквідації автономними роботизованими комплексами.

1.2 Мета створення інформаційної системи

Використання інформаційної системи в роботизованих комплексах для боротьби з лісовими пожежами.

1.3 Цільова аудиторія

У цільовій аудиторії інформаційної системи можна виділити наступні групи:

1. Міські та обласні муніципальні органи.
2. Комунальні організації.
3. Приватні підприємства пов’язані з лісогосподарчим сектором.
4. Інші зацікавлені сторони.

2 Вимоги до інформаційної системи

1.4 Вимоги до інформаційної системи в цілому

1.4.1 Вимоги до структури й функціонування інформаційної системи

Інформаційна система повинна бути реалізована у вигляді модулю для обміну інформацією та даними з використанням ROS (Robot Operating System) фреймворку. Модуль повинен складатися із взаємозалежних вузлів із чітко розділеними функціями.

1.4.2 Вимоги до персоналу

Для підтримки й експлуатації системи персонал повинен мати загальні навички роботи з персональним комп'ютером та операційною системою Linux.

1.4.3 Вимоги до збереження інформації

У системі повинен бути передбачений механізм резервного копіювання структури й умісту бази даних. Резервне копіювання графічного вмісту повинне здійснюватися автоматично.

2.3 Вимоги до видів забезпечення

1.1.1 Вимоги до інформаційного забезпечення

Реалізація системи відбувається з використанням:

- ROS
- Linux OS

3 Склад і зміст робіт зі створення системи

Докладний опис етапів роботи зі створення системи наведено в табл. А.1.

Таблиця А.1 – Етапи створення системи

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Визначити вимоги до інформаційної системи та виконати планування робіт	3 дні
2	Огляд методів для комп'ютерного зору, вибір нейронної мережі	4 дні
3	Збір та підготовка даних для тренування нейронної мережі	5 днів
4	Тренування мережі на підготовленому наборі даних	8 днів
5	Тестування і оцінка результатів роботи нейронної мережі	6 днів
6	Адаптація інформаційної технології штучного сприйняття для роботи в реальному часі	10 днів
	Загальна тривалість робіт (з урахуванням резервного строку на налагодження й виправлення помилок) і строк закінчення проекту	36 днів

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

Планування змісту робіт

Структурна декомпозиція робіт (work breakdown structure, WBS) - це графічне подання згрупованих елементів проекту у вигляді пакетів робіт, які ієрархічно пов'язані з продуктом проекту. Така структура необхідна для забезпечення ефективного управління проектом, визначення і структурування переліку робіт, створення структури звітності та розуміння задач виконавця. На рисунку А.1 представлена структурна декомпозиція робіт проекту.

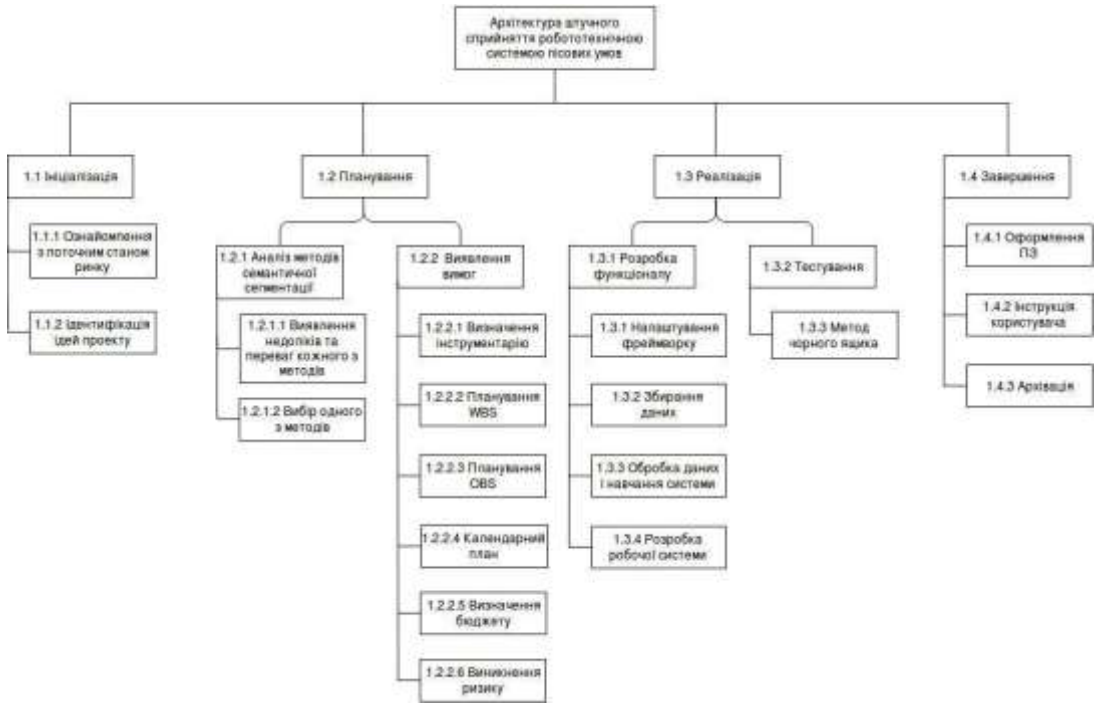


Рисунок А.1 – WBS структура проекту

Планування структури виконавців

Наступним кроком розробки структури проекту є визначення організаційної структури (OBS) проекту.

Організаційна структура проекту (OBS) – є графічним відображенням учасників проекту (фізичних та юридичних осіб) та їхніх відповідальних осіб, залучених до реалізації проекту. На верхньому рівні OBS проекту знаходиться керівник та команда управління проектом; на наступному рівні – виконавці. Останнім рівнем OBS-структури є відповідальні особи виконавців. Це не обов’язково повинні бути керівники, а ті співробітники, яким доручено безпосередньо організовувати і відповідати перед виконавцем за виконання конкретного елемента WBS-структури.

OBS структура представлена на рисунку А.2.

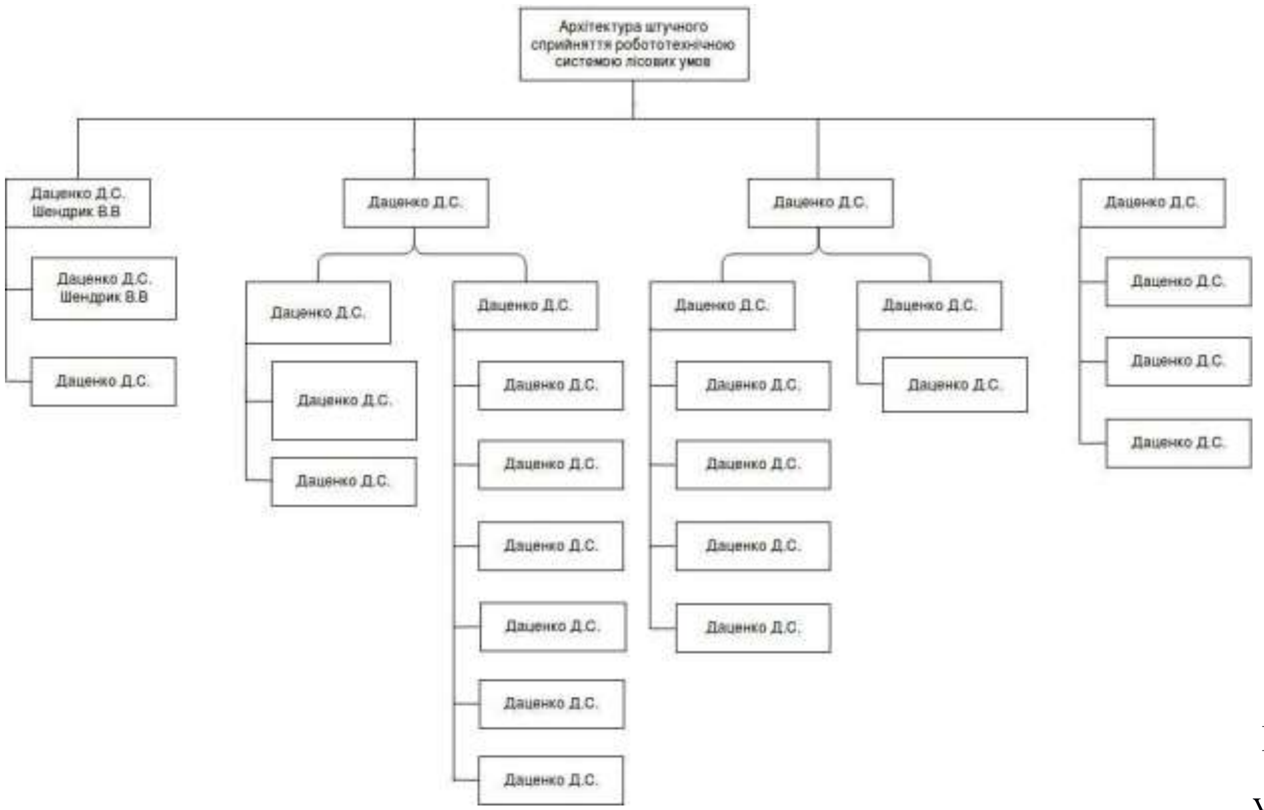


Рисунок А.2 – OBS структура

Побудова матриці відповідальності

Матриця відповідальності (Responsibility Assignment Matrix) забезпечує опис і узгодження структури відповідальності за виконання пакетів . Вона являє собою форму опису розподілу за реалізацію робіт із зазначенням ролі кожного з виконавців.

На рисунку А.3 показано матрицю відповідальності проекту.

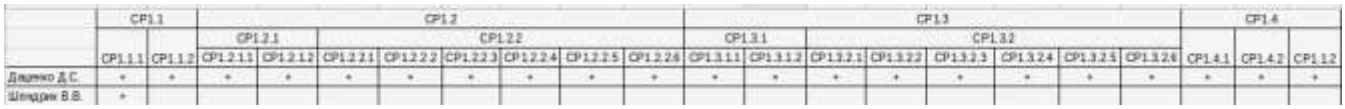


Рисунок А.3 – Матриця відповідальності

Розробка PDM мережі

PDM мережа була побудована за допомогою надбудови програми GanttProject. Ця надбудова має назву Pert діаграма. PERT призначений для дуже масштабних, одноразових, складних, нерутінний проектів. Метод має на увазі наявність невизначеності, даючи можливість розробити робочий графік проекту без точного знання деталей і необхідного часу для всіх його складових. PERT був розроблений головним чином для спрощення планування на папері і складання графіків великих і складних проектів. Метод особливо націлений на аналіз часу, який потрібен для виконання кожної окремої задачі, а також визначення мінімального необхідного часу для виконання всього проекту. Мережу зображено (рис. А.4-А.6).

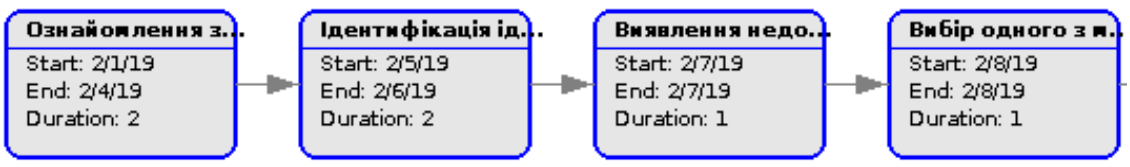


Рисунок А.4 – PDM-мережа

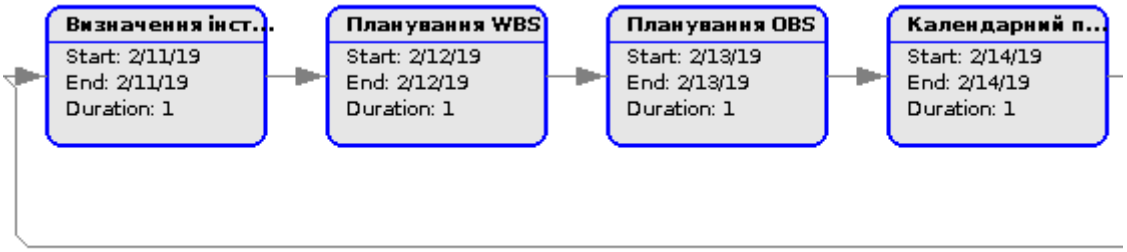


Рисунок А.5 – Продовження PDM-мережі

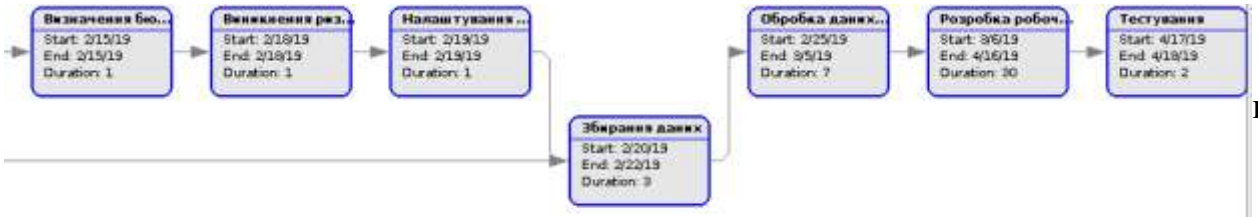


Рисунок А.6 – Кінець PDM-мережі

Побудова календарного графіку виконання ІТ-проекту

Діаграма Ганта - це візуальний спосіб відображення запланованих завдань. Горизонтальні графіки широко використовуються для планування проектів будь-яких розмірів в різних галузях і сферах. Це зручний спосіб показати, яка робота планується до виконання в певний день і час. Дана діаграма також допомагає командам і менеджерам проектів контролювати дати початку і закінчення будь-якого проекту. Все в одному просторі. Діаграму Ганта наведено на рисунку А.7.

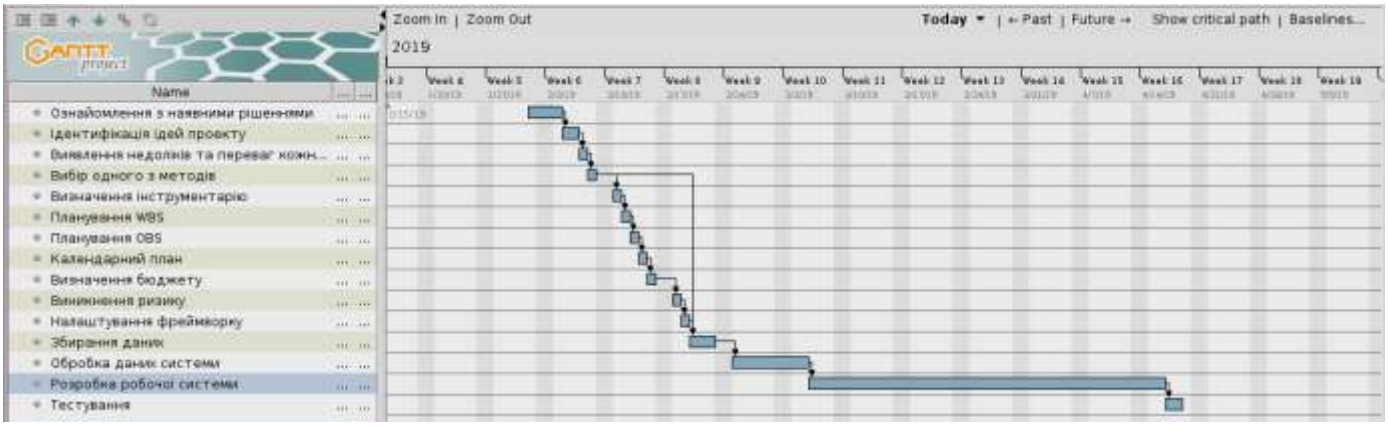


Рисунок А.7 – Календарний графік робіт

Управління ризиками проекту

Ризик – це імовірнісна подія, яка у випадку своєї появи негативно або позитивно вплине на проект.

Причинами невизначеності може бути недостатність інформації, наявність елементу випадковості, наявність протидії, невизначений час на виконання задачі, невизначена вартість або невизначені роботи, які необхідно виконувати.

Ризики класифікують наступним чином:

1. зовнішні непередбачувані ризики
2. проектні ризики нетехнічного характеру
3. техніко-технологічні ризики
4. правові ризики
5. зовнішні непередбачувані, але не до кінця визначені

Для того, щоб відобразити рівень ризику, прийнятий для конкретної особи, застосовують поняття «толерантність до ризику». Виділяють три рівні толерантності. При низькому рівні толерантності ризикові проекти не розглядаються. Середній рівень характеризується тим, що ризики аналізуються та розраховуються і тільки тоді приймається рішення чи йти на ризик. При високому рівні толерантності приймаються будь-які ризикові проекти.

Процес управління ризиками включає наступні етапи:

1. ідентифікація
2. процес оцінювання ризиків, який включає в себе якісний, кількісний аналіз.
3. Заходи реагування на ризики
4. Моніторинг заходів і ризиків

Ідентифікація ризиків – це виявлення ризиків, здатних вплинути на проект і документальне оформлення їх характеристик. Ризики, які змогли виявити і проаналізувати, називаються відомими ризиками, а ризики, ймовірність і вплив яких не змогли оцінити, - невідомими.

В даному випадку на першому етапі, в процесі виявлення ризиків можна виділити ряд ризиків.

Зовнішні непередбачувані ризики:

- природні катастрофи: повені; землетруси; шторми; кліматичні катаклізми тощо;
- неочікувані зовнішні ефекти: екологічні; соціальні;
- зриви: у створенні необхідної інфраструктури; через банкрутство підрядників; у фінансуванні: через помилки у визначенні цілей проекту; через неочікувані політичні зміни.

Зовнішні передбачувані (проте не визначені) ризики:

- економічний ризик у зв'язку із: зміною вимог споживачів; економічними змінами; посиленням конкуренції; втратою позицій на ринку;
- операційні ризики: порушення безпеки; відступ від цілей проекту; неможливість підтримання робочого стану елементів проекту; неприпустимий екологічний вплив; негативні соціальні післядії; зміни валютних курсів, нерозраховувана інфляція.

Внутрішні ризики. Внутрішні організаційні ризики:

- зриви планів робіт через: недостачу робочої сили; нестачу часу; помилки проектування; помилки планування; недоліки координації робіт; зміни керівництва; конфлікти та саботаж; зміну можливостей замовника проекту; недостатнє управління.
- перевитрати коштів через: зриви планів робіт; невірну стратегію; некваліфікований персонал; переплати за роботу/матеріали; неузгодження частин проекту; невірний кошторис.

Внутрішні технічні ризики зміни технології:

- зміни технології;
- специфічні ризики технології, що закладаються до проекту;
- помилки в проектно-кошторисній документації.

Інші ризики:

- прямі втрати майна: по транспортних спорах; обладнання і т.д.;
- непрямі збитки: перестановка обладнання;
- неотримання орендного прибутку; порушення запланованого ритму діяльності; збільшення необхідного фінансування;

Формування бюджету проекту

Останнім етапом планування проекту – є етап розподіл бюджету даного проекту. Були визначені особи, які брали участь в даному проекті та між якими необхідно розподілити бюджет:

- розробник;
- керівник.

У табл. Б.1 приведений бюджет на витрачення заробітної плати.

Таблиця Б.1 – Розподілення бюджету

Посада	За 1 год/грн	Робочих годин	Сума
Розробник	240	140	33600
Керівник	270	5	1350

Бюджет заробітної плати складає *34 950 грн*.

Робототехнічна система штучного сприйняття лісових умов

Даценко Д.С., *студентка*, Шендрик В.В*., доцент*
Сумський державний університет, м. Суми

Згідно зі статистичними даними, представленими Європейською комісією, щорічно у Європі відбувається близько 65 000 пожеж. Лісові пожежі призводять до втрати деревини, погіршують якість та швидкість лісовідновлення, впливають на такі галузі як бджільництво та лісове фермерство, що в цілому шкодить навколишньому середовищу. Необхідність зберігання лісових ділянок чистими, активно знижуючи захаращення, вимагає значних інвестицій та залучання людських ресурсів. Умови праці в прибиранні лісу суворі та небезпечні. З цієї причини вкрай важливо розробити технологічні рішення, які б забезпечили безпеку праці та прискорили операції.

Мета проекту – створити систему для зменшення накопичень пожежонебезпечного лісового сміття, тим самим надаючи допомогу в процедурах технічного обслуговування залісненого ландшафту. Більшість існуючих технічних рішень базуються на використанні людино-керованих або дистанційно керованих машин. У даному проекті пропонується створення автономної системи виявлення зон інтересу й відповідного реагування.

При функціонуванні штучної системи сприйняття необхідно вирішувати такі основні проблеми: неоднорідність лісу, що ускладнює класифікацію об’єктів середовища; наявність перешкод і нерівномірність ландшафту. Ієрархічна схема класифікації буде розроблена, використовуючи дані, отримані з сенсорів (Kinect та Intel Realsense) для генерації класифікованих зображень. Для формування навчальної та контрольної вибірки використовувалися зображення з основним RGB каналом. Основна задача класифікації - це виявити класи абсолютного знищення (наприклад, суха біомаса) та класи збереження (наприклад, людські життя, важливі природні об’єкти). Для аналізу середовища було обрано метод семантичної сегментації ідентифікації об'єктів і регіонів інтересу. Після виявлення області інтересу з лісовим сміттям система повинна прийняти рішення

відносно наступної операції – подальшого патрулювання чи очистки захаращення. Інтеграція детальних та геоприв'язаних даних сприятиме просторово-чіткому і дуже описовому визначенню класу регіону та прийняттю відповідного рішення використовуючи систему ROS (Robot Operating System). Таким чином буде розроблено програмне рішення для робота, призначеного для автономного виконання завдань з обслуговування залісненого ландшафту.

ДОДАТОК В

Лістинг файлу конфігурацій data.yaml

```
    name: "forest"
    data_dir: "/home/Daryna/segmentation/forest"
    buff: True    # if this is true we buffer buff_n images in a
fifo
    buff_nr: 500         # number of images to keep in fifo
(prefetch batch) <-should be bigger than batch size to make sense
    img_prop:
      width: 768
      height: 384
      depth: 3            # number of channels in original image
    force_resize: True   # if dataset contains images of different
size, it should be True
    force_remap: True
    label_map:
      0: 'crap'
      76: 'vegetation'
      255: 'tree'
    label_remap:          # for softmax (it must be an index of the
onehot array)
      0: 0
      76: 1
      255: 2
    color_map: # bgr
      0: [0,0,0]
      76: [0,0,255]
      255: [255,255,255]
```

Лістинг файлу конфігурацій train.yaml

```
    # train cfg file
    max_epochs: 7000
    loss: "log" # log (1/ln(fc+e)) or median_freq (median_fc/fc)
    gamma: 2            # gamma for focal loss
    lr: 0.0001          # learning rate
    decay1: 0.9         # decay for first order momentum
    decay2: 0.999       # decay for second order momentum
    epsilon: 0.00000001 # epsilon for adam
    w_decay: 0.000001   # weight decay
    lr_decay: 1.1        # decay learning rate every x epochs =
1,10
    lr_rate: 1       # decay to 1/x every epoch = 1.01,2
    acc_report_epochs: 10 # every x steps, report accuracy
    batch_size: 4      # batch size 3, 4 works, but 5 and more -
too much
    gpus: 1             # number of gpus to use
    save_imgs: True     # False doesn't save anything, True saves
some
                        # sample images (one per batch of the last
calculated batch)
                        # in log folder
    summary: False      # verbose summary
    summary_freq: 50    # steps for summary
    grads: "speed"      # speed, mem, or tf, for speed optimized,
memory optimized, or vanilla tensorflow
    ignore_crap: False   # last class in forest is crap, therefore
we want to ignore it in the metrics
```

Лістинг файлу конфігурацій net.yaml

```
    name: "bonnet"
    dropout: 0.1
    bn_decay: 0.99
    n_k_lyr: # contains the amount of filters of each layer (non-
bt-doesn't need it)
      - 16  # block 1 3x3 downsample
      - 48  # block 2 3x3 downsample
      - 80  # block 3 3x3 downsample
      - 48  # block 1 upsample
      - 32  # block 2 upsample
      - 16  # block 3 upsample
    train_lyr:          # boolean list of layers to train, starting
by the first conv
      - True  # block 1 downsample
      - True  # block 2 downsample
      - True  # block 3 downsample
      - True  # godeep
      - True  # psp
      - True  # block 1 upsample
      - True  # block 2 upsample
      - True  # block 3 upsample
      - True  # linear classifier
```

Лістинг програми accuracy.py

```
#! /bin/env/python3
from __future__ import print_function, division
import argparse
import numpy as np
import os
import os.path as osp
import warnings
import base64
import cv2

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('img_folder')
    parser.add_argument('-l',dest='lbl_folder')
    args = parser.parse_args()

    img_folder = args.img_folder
    lbl_folder = args.lbl_folder

    #read all files in directory
    filenames = sorted(os.listdir(img_folder))

    print(os.getcwd())

    tp_rates = []
    accuracy_rates = []

    for i, file in enumerate(filenames):

        print(file)

        result_img = cv2.imread(osp.join(img_folder, file))
        result_img = cv2.resize(result_img,(768,384))
```

```
            label_img = cv2.imread(osp.join(lbl_folder, file))
            label_img = cv2.resize(label_img,(768,384))

            #--- take the absolute difference of the images ---
            res = cv2.absdiff(result_img, label_img)
            overlap_img = cv2.add(result_img, label_img)

            #print(res.size)

            #--- convert the result to integer type ---
            res = res.astype(np.uint8)

            true_positives = 0
            true_negatives = 0
            label_positives = 0
            for x in range(result_img.shape[0]):
                for y in range(result_img.shape[1]):
                    if not np.array_equal(label_img[x, y],
[0,0,0]):
                        label_positives += 1
                        if np.array_equal(label_img[x, y],
result_img[x, y]):
                            true_positives += 1
                    if np.array_equal(overlap_img[x, y], [0,0,0]):
                        true_negatives += 1

            tp_rate = true_positives/label_positives
            tp_rates.append(tp_rate)

            accuracy_rate = (true_positives + true_negatives) /
(768*384)
            accuracy_rates.append(accuracy_rate)
```

```
            p = 100.0 - (np.count_nonzero(res) * 100.0)/ (768*384)
#res.size
            #print(np.count_nonzero(res))
            #print(768*384)
            #percentage.append(p)
            print("Unit accuracy = " + str(accuracy_rate))
            print("True positive rate = {}".format(tp_rate))

        print("***************RESULTS***************")
        print("Mean accuracy = " + str(np.mean(accuracy_rates)))
        print("std accuracy = " + str(np.std(accuracy_rates)))
        print("Mean true positive rate = " +
str(np.mean(tp_rates)))
        print("std true positive rate = " + str(np.std(tp_rates)))

    if __name__ == '__main__':
        main()
```

Лістинг програми json_to_images.py

```
import argparse
import base64
import json
import os
import os.path as osp
import warnings
import numpy as np
import cv2

import PIL.Image
import yaml

from labelme import utils
from os import walk

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument('json_folder')
    parser.add_argument('-o', '--out', default=None)
    args = parser.parse_args()

    json_folder = args.json_folder

    if args.out is None:
        out_dir = osp.basename(json_folder).replace('.', '_')
        out_dir = osp.join(osp.dirname(json_folder), out_dir)
    else:
        out_dir = args.out
    if not osp.exists(out_dir):
        os.mkdir(out_dir)

    #read all json files in directory
    filenames = sorted(os.listdir(json_folder))
```

```
        print(os.getcwd())

        for i, file in enumerate(filenames):

            print (file)

            data = json.load(open(osp.join(json_folder, file)))

            if data['imageData']:
                imageData = data['imageData']
            else:
                imagePath = os.path.join(os.path.dirname(file),
data['imagePath'])
                with open(imagePath, 'rb') as f:
                    imageData = f.read()
                    imageData =
base64.b64encode(imageData).decode('utf-8')
            img = utils.img_b64_to_arr(imageData)

            label_name_to_value = {'_background_': 0}
            for shape in sorted(data['shapes'], key=lambda x:
x['label']):
                label_name = shape['label']
                if label_name in label_name_to_value:
                    label_value = label_name_to_value[label_name]
                else:
                    label_value = len(label_name_to_value)
                    label_name_to_value[label_name] = label_value
            lbl = utils.shapes_to_label(img.shape, data['shapes'],
label_name_to_value)

            label_names = [None] *
(max(label_name_to_value.values()) + 1)
            for name, value in label_name_to_value.items():
                label_names[value] = name
```

```
            plus = 0

            st = 'img/{}.png'.format(i+plus)
            PIL.Image.fromarray(img).save(osp.join(out_dir, st))

            utils.lblsave(osp.join(out_dir,
'lbl/{}.png'.format(i+plus)), lbl)
            print('Saved to: %s' % out_dir)

            im = cv2.imread(osp.join(out_dir,
'lbl/{}.png'.format(i+plus)))
            im[np.where((im == [0,0,128]).all(axis = 2))] =
[0,0,255]
            im[np.where((im == [0,128,0]).all(axis = 2))] =
[255,255,255]
            im[np.where((im == [0,128,128]).all(axis = 2))] =
[255,255,255]

            cv2.imwrite(osp.join(out_dir,
'lbl/{}.png'.format(i+plus)), im)


    if __name__ == '__main__':
        main()
```

Лістинг програми overlap.py

```
from __future__ import print_function
import argparse
import numpy as np
import os
import os.path as osp
import warnings
import base64
import cv2

def main():
     parser = argparse.ArgumentParser()
     parser.add_argument('img_folder')
     parser.add_argument('-l',dest='lbl_folder')
     parser.add_argument('-o', '--out', default=None)
     args = parser.parse_args()

     img_folder = args.img_Afolder
     lbl_folder = args.lbl_folder

     if args.out is None:
          out_dir = osp.join(osp.dirname(img_folder),
'overlay_imgs')
     else:
          out_dir = args.out

     if not osp.exists(out_dir):
          os.mkdir(out_dir)

     #read all files in directory
     filenames = sorted(os.listdir(img_folder))

     print(os.getcwd())
```

```
    for i, file in enumerate(filenames):

        print(file)

        img = cv2.imread(osp.join(img_folder, file))
        img = cv2.resize(img,(768,384))

        lbl = cv2.imread(osp.join(lbl_folder, file))
        lbl = cv2.resize(lbl,(768,384))

        #transparent = cv2.add(0.5, lbl)
        output = cv2.add(lbl, img)

        cv2.imwrite(osp.join(out_dir, file), output)

if __name__ == '__main__':
    main()
```

Лістинг програми image_publisher_subscriber.cpp

```
#include "ros/ros.h"

#include "sensor_msgs/CameraInfo.h"
#include "sensor_msgs/Image.h"

// General Defines
#define NAME "rgbd_throttle"

// Default Param Defines
#define RATE 0.2

// Input Defines
#define RGB_INFO_IN   "/camera/color/camera_info"
#define RGB_RECT_IN   "/camera/color/image_raw"
#define DEPTH_INFO_IN "/camera/depth/camera_info"
#define DEPTH_RECT_IN "/camera/depth/image_rect_raw"
#define BUFFER_IN 1

// Output Defines
#define RGB_INFO_OUT   "rgb/info_out"
#define RGB_RECT_OUT   "rgb/rect_out"
#define DEPTH_INFO_OUT "depth/info_out"
#define DEPTH_RECT_OUT "depth/rect_out"
#define BUFFER_OUT 1

// Global Variables
ros::Publisher pub_rgb_info;
ros::Publisher pub_rgb_rect;
ros::Publisher pub_depth_info;
ros::Publisher pub_depth_rect;

double rate;
double secs;
```

```
double last_sent;

sensor_msgs::CameraInfo rgb_info;
sensor_msgs::Image      rgb_rect;
sensor_msgs::CameraInfo depth_info;
sensor_msgs::Image      depth_rect;

// Callbacks

void callback_rgb_info(const sensor_msgs::CameraInfo& data)
{
    rgb_info = data;
}

void callback_rgb_rect(const sensor_msgs::Image& data)
{
    rgb_rect = data;
}

void callback_depth_info(const sensor_msgs::CameraInfo& data)
{
    depth_info = data;
}

void callback_depth_rect(const sensor_msgs::Image& data)
{
    depth_rect = data;

    if (data.header.stamp.toSec() >= last_sent + secs)
    {
        pub_rgb_info.publish(rgb_info);
        pub_rgb_rect.publish(rgb_rect);
        pub_depth_info.publish(depth_info);
        pub_depth_rect.publish(depth_rect);
        last_sent = data.header.stamp.toSec();
```

```
    }
}

// Main

int main(int argc, char **argv)
{
    ros::init(argc, argv, NAME);
    ros::NodeHandle n;

    ROS_INFO("Initializing node '%s' ...", NAME);

    // Read Params
    n.param<double>("rate", rate, RATE);
    secs = 1.0 / rate;
    ROS_INFO("Time between frames: %f seconds.", secs);

    // Initialize Variables
    last_sent = 0;

    // Advertise Publishers
    pub_rgb_info   =
n.advertise<sensor_msgs::CameraInfo>(RGB_INFO_OUT,   BUFFER_OUT);
    pub_rgb_rect   = n.advertise<sensor_msgs::Image
>(RGB_RECT_OUT,   BUFFER_OUT);
    pub_depth_info =
n.advertise<sensor_msgs::CameraInfo>(DEPTH_INFO_OUT, BUFFER_OUT);
    pub_depth_rect = n.advertise<sensor_msgs::Image
>(DEPTH_RECT_OUT, BUFFER_OUT);

    ros::Subscriber sub_rgb_info   = n.subscribe(RGB_INFO_IN,
BUFFER_IN, callback_rgb_info);
    ros::Subscriber sub_rgb_rect   = n.subscribe(RGB_RECT_IN,
BUFFER_IN, callback_rgb_rect);
```

```
    ros::Subscriber sub_depth_info = n.subscribe(DEPTH_INFO_IN,
BUFFER_IN, callback_depth_info);
    ros::Subscriber sub_depth_rect = n.subscribe(DEPTH_RECT_IN,
BUFFER_IN, callback_depth_rect);

    ros::spin();
    return 0;
}
```