PROFESSIONAL TRAINING REPORT – II

Entitled

Glass Identification using Random Forest

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and
Engineering with specialization in Artificial Intelligence and Robotics

By

Maligireddy Phanidhar Reddy (41612024)

Manne Venkata Sushma Priya (41612027)

Annavarapu Sriram Aazad (41612004)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
CATEGORY -1 UNIVERSITY BY UGC
Accredited with Grade "A++" by NAAC I 12B Status by UGC I Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

MAY 2024

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
CATEGORY-I UNIVERSITY BY UGC
Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Maligireddy Phanidhar Reddy (41612024), Manne Venkata Sushma Priya(41612027), Annavarapu Sriram Aazad(41612004)** who carried out the project entitled "**Glass Identification using Random Forest Classifier**" under my supervision from JANUARY 2024 to MAY 2024.

Internal Guide
Mrs.V.Subapriya, M.E.,(Ph.D)

Head of the Department
Dr. S. VIGNESHWARI, M.E., Ph.D

Submitted for Viva voce Examination held on ____________________

Internal Examiner **External Examiner**

DECLARATION

I, Maligireddy Phanidhar Reddy(4162024),hereby declare that the Project Report entitled Glass identification using Random Forest Classifier Done by me under the guidance of Mrs. V. Subapriya M.E.,(Ph.D) at COGNIBOT is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: **SIGNATURE OF THE CANDIDATE**

ACKNOWLEDGEMENT

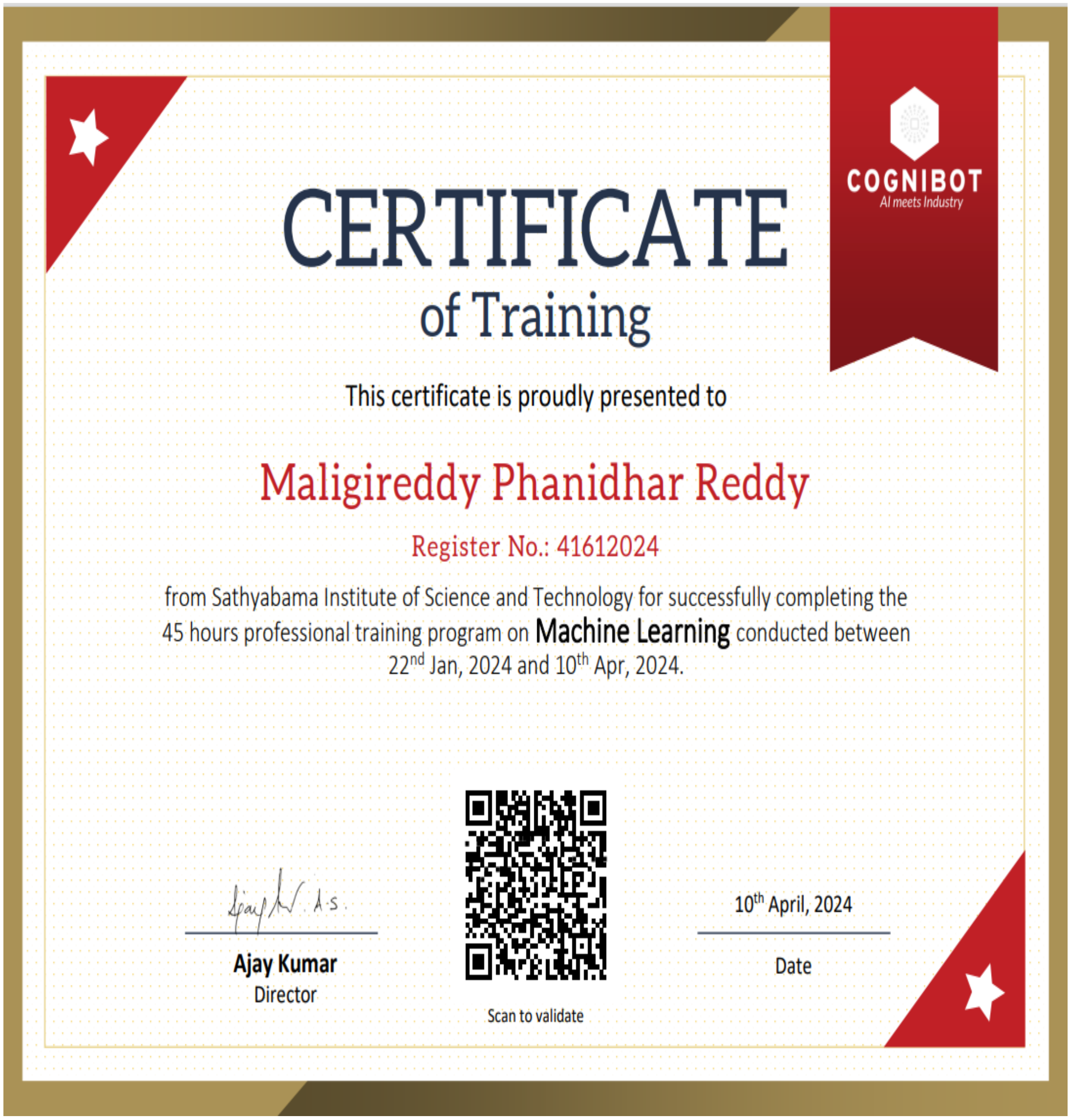
I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D.**, **Dean**, School of Computing, **Dr.L.Vigeneshwari M.E.,Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide for **Mrs. V. Subapriya M.E.,(Ph.D)** her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

TRAINING CERTIFICATE



COGNIBOT
AI meets Industry

CERTIFICATE
of Training

This certificate is proudly presented to

Maligireddy Phanidhar Reddy

Register No.: 41612024

from Sathyabama Institute of Science and Technology for successfully completing the 45 hours professional training program on **Machine Learning** conducted between 22nd Jan, 2024 and 10th Apr, 2024.

[illegible]

Ajay Kumar
Director

Scan to validate

10th April, 2024
Date

ABSTRACT

Glass identification is a critical task in various industries, including forensic science, manufacturing, and recycling. Accurately identifying different types of glass can be challenging due to their similar visual characteristics. However, with the advent of advanced machine learning techniques like the Random Forest classifier, the process has become more efficient and reliable.

Random Forest is a powerful ensemble learning algorithm that combines multiple decision trees to make robust predictions. When applied to glass identification, it can discern between various glass types based on a set of input features, such as refractive index, chemical composition, and density.

In this context, the Random Forest classifier plays a crucial role in automating and enhancing the accuracy of glass identification.
This approach has significant advantages, including its ability to handle complex datasets, mitigate overfitting, and provide insights into the importance of different features for classification.

In this article, we will explore the process of glass identification using the Random Forest classifier. We will delve into the data preprocessing steps, feature selection, model training, and evaluation techniques to ensure a robust and accurate glass identification system. By the end of this discussion, you will have a better understanding of how machine learning, specifically the Random Forest algorithm, can revolutionize the field of glass identification, making it faster, more reliable, and cost-effective.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	i
	LIST IF FIGURES	ii
	LIST OF TABLES	iii
	LIST OF ABBREVIATIONS	iv
1	**INTRODUCTION**	
	1.1 OVERVIEW	1
	1.2 OBJECTIVE	2
2	**AIM AND SCOPE OF THE PRESENT INVESTIGATION**	3-4
3	**EXPERIMENTAL OR MATERIALS AND METHODS;ALGORITHMS USED**	
	3.1 REQUIREMENT ANALYSIS	5
	3.2 SOFTWARE REQUIREMENTS	5-6
	3.3 HARDWARE REQUIREMENTS	7
	3.4 ARCHITECTURE DIAGRAM	7
	3.4.1 DATA	8
	3.4.2 DIMENSIONALITY REDUCTION	8
	3.4.3 FEATURE SLECTION	9
	3.4.4 FEATURE PROJECTION	9
	3.4.5 MODEL SELECTION	9
	3.4.6 MODEL TRAINING	10
	3.4.7 DECISION MAKING	10
	3.4.8 MODEL EVALUATION	10
	3.4.9 DEPLOYMENT	11
	3.4.10 MONITERING AND MAINTAINANCE	11

4	**RESULTS AND DISCUSSION,PERFORMANCE ANALYSIS** 4.1 DATA SET	12-14
5	**SUMMARY AND CONCLUSIONS REFERENCES** **APPENDIX** A. SOURCE CODE B. SCREENSHOTS C. RESEARCH PAPER	15-16 19-25

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.1	MODEL ARCHITECTURE DIAGRAM	7
3.2	MODEL SYSTEM ARCHITECTURE	8
4.1	DATA SET SAMPLE	12
4.2	SAMPLE DATA OF ORIGINAL DATA SET	13

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Identification of the glass at a crime scene can prove to be very useful in providing evidence in investigations and forensic purpose. Also, since glasses are used in various industries to manufacture different types of items, identifying the type beforehand from its components (oxide content) can save cost, time and effort.

The dataset is commonly used to demonstrate Machine learning classification algorithms in academic settings. It is available on the UCI Machine Learning repository (See Reference for link).

It contains 214 samples of glasses which can be categorized into different types based on their usage. The original dataset had 7 types but the sample obtained from UCI does not have one which is the vehicle windows float processed (type 4).

The columns in this dataset are:

RI: refractive index

Na: Sodium

Mg: Magnesium

Al: Aluminum

Si: Silica

K:Potassium

Ca: Calcium

Ba: Barium

Fe: Iron

Type of glass (Target label)

RI is an index vaiable and has no units. Columns 3-9 are measured as weight present in corresponding oxide.

1.2 OBJECTIVE

The objective of identifying glass types using a Random Forest classifier is to develop a predictive model that can accurately classify different types of glass based on their characteristics. Glass identification is a crucial task in various industries, including manufacturing, construction, and forensics. By employing a Random Forest classifier, we aim to achieve the following objectives:
Accurate Classification is to create a model that can accurately classify glass samples into their respective categories or types. These types may include float glass, window glass, container glass, and more. Accurate classification ensures that the right type of glass is used for its intended purpose, which is essential for quality control and safety.Automation,By automating the glass identification process using machine learning, we can save time and reduce human error. This can be especially valuable in industrial settings where large quantities of glass need to be classified. Feature Importance, Random Forest models can provide insights into the importance of different features or characteristics of glass that contribute to the classification. This information can be valuable for understanding the factors that distinguish one type of glass from another, aiding in product development or quality improvement efforts. In summary, the objective of using a Random Forest classifier for glass identification is to create a reliable, accurate, and robust model that can classify different types of glass based on their properties. This has numerous practical applications in various industries, ultimately contributing to improved product quality and cost savings.

CHAPTER 2

AIM AND SCOPE

The aim of the project centered around glass identification using the Random Forest algorithm is to construct a robust and accurate machine learning model capable of categorizing different types of glass based on their inherent properties and characteristics. The scope of this endeavor encompasses several key facets:

Firstly, data collection entails amassing a comprehensive dataset comprising diverse glass samples, incorporating attributes such as refractive index, chemical composition (sodium, magnesium, etc.), and thickness. Subsequently, data preprocessing becomes pivotal, involving data cleansing, handling missing values, and potentially feature scaling to ensure the dataset's suitability for model training.

Feature selection becomes a crucial consideration, as it involves determining which attributes wield the most influence in distinguishing between various glass types. The core of the project lies in the implementation of the Random Forest algorithm for classification, followed by rigorous model training on a portion of the dataset, with the remainder reserved for testing.

The project's evaluation phase employs a range of performance metrics like accuracy, precision, recall, model's efficacy in glass identification. Hyperparameter tuning may be employed to optimize the Random Forest algorithm's settings for enhanced classification accuracy.

Visualizations could be generated to elucidate the importance of different features in the glass identification process. Additionally, deployment of the model in practical applications, such as glass recycling, quality control, or forensic analysis, should be considered if applicable.

Lastly, an enduring strategy for maintenance and updates should be devised to ensure the model's longevity and relevance in the ever-evolving field of glass identification, while also giving due consideration to ethical and legal aspects regarding data collection and model deployment.

CHAPTER 3

EXPERIMENTATION OR MATERIALS AND METHODS

ALGORITHM USE

3.1 REQUIREMENT ANALYSIS

The requirement analysis for a project focused on glass identification using the Random Forest algorithm involves a meticulous examination of the prerequisites and specifications essential for its successful execution. The selection of the Random Forest algorithm itself constitutes a requirement, including the availability of appropriate libraries or tools for its implementation.

3.2 SOFTWARE REQUIREMENTS

OS: Windows 7, windows 8, windows, Linux and Mac compatible**.**

SOFTWARE: Python, along with machine learning libraries like scikit-learn, NumPy , and pandas.

CODING LANGUAGE: Python

TECHNOLOGIES USED

PYTHON

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems More efficiently.

Python is a popular programming language commonly used to implement random forest Classifiers and perform the machine learning tasks. It offers several libraries, such as Scikit-learn and Random forest classifier from sklearn.ensemble, which simplify the process of building, training, and evaluating Random Forest models. Python's simplicity and a rich ecosystem of data science libraries make it a preferred choice for developing and deploying machine learning models, including Random Forest classifiers.

JUPYTER NOTEBOOK

Jupyter Notebook is a versatile and essential tool in the field of data science and computational research. It provides an interactive platform where users can seamlessly integrate code, visualizations, and explanatory text into a single document. This capability is particularly valuable for data exploration, analysis, and sharing insights with others. Users can choose from various programming language kernels, with Python being the most commonly used. Jupyter Notebook's support for data visualization libraries like Matplotlib and Plotly makes it easy to generate informative charts and graphs, aiding in data interpretation. Additionally, its collaborative features enable researchers and data scientists to work together, share their findings, and create reproducible research reports. Furthermore, Jupyter Notebook's integration with cloud platforms like Google Colab and Microsoft Azure Notebooks allows users to harness powerful computing resources. Overall, Jupyter Notebook continues to play a pivotal role in fostering interactive and collaborative data-driven work in a wide range of domains, from academia to industry.

ADVANTAGES OF USING PYTHON

Presence of third-party modules.

Extensive support libraries(NumPy for numerical calculations,Pandas Open source and large active community base.

Versatile, Easy to read, learn and write.

User-friendly data structures.

High-level language.

Fast Learning and Easy use

3.3 HARDWARE REQUIREMENT

Processor: Pentium Dual Core 2.00

Hard disk: 120 GB or Above **RAM:** 4 GB or Above **Keyboard:** 110 keys enhanced

3.4 ARCHITECTURE DIAGRAM

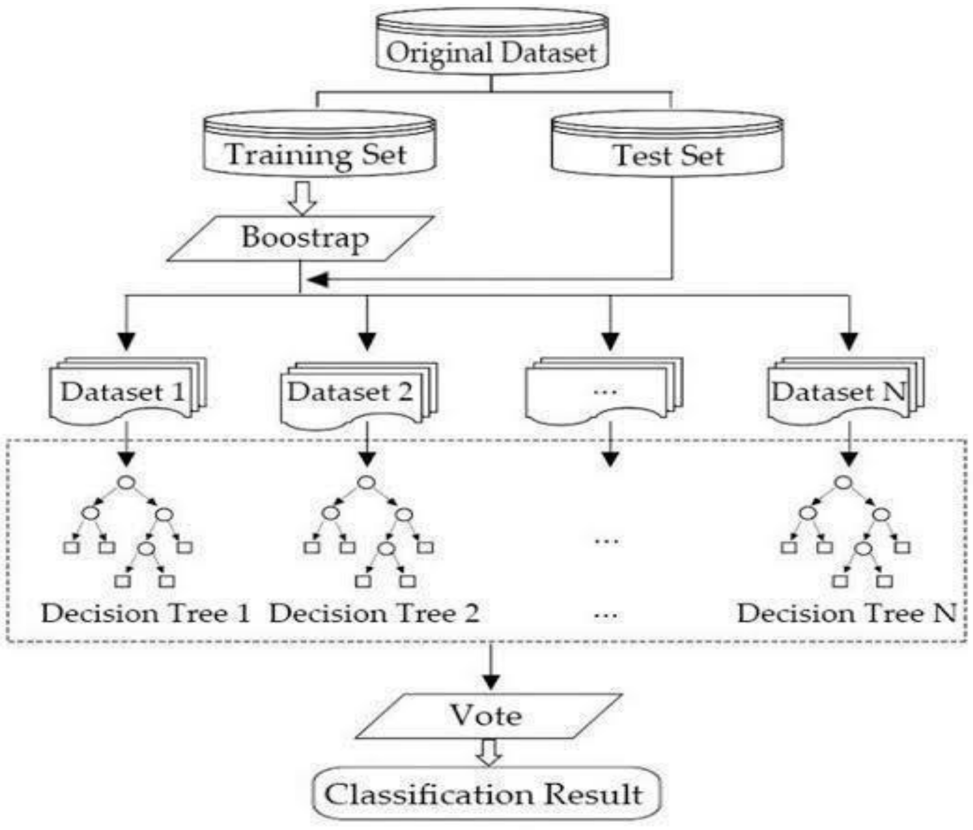


Fig 3.1 Model Architecture diagram

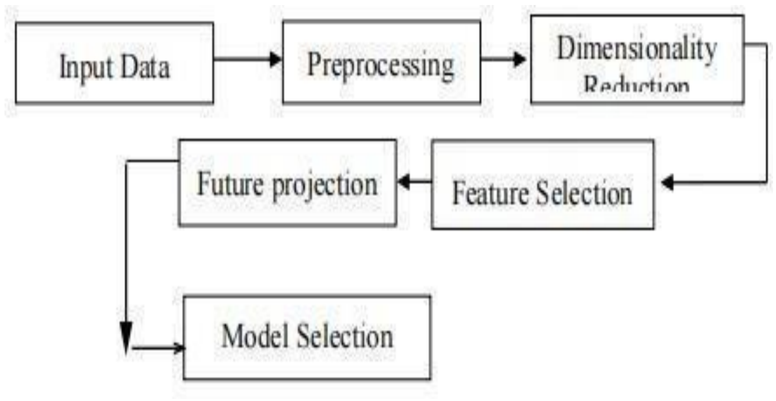


Fig. 3.2 Model System Architecture

3.4.1 Data

The dataset's objective is to forecast Glass based on the provided factors (Id category, RI, Na, Mg, Al, Si, K, Ca, Ba), with all columns save the Id columns playing a vital part in defining the Glass type, which is also our desired variable. A glass dataset with ten input attributes and one output attribute is available (Type of glass). Information on Attributes (Input attributes) Identifier, RI stands for refractive index. Sodium (Na), Magnesium (Mg), Aluminum (Al), Silicon (Si), Potassium (K), Calcium (Ca), Barium (Ba), and Iron(Fe).

3.4.2 Dimensionality reduction

The approach known as spatiality reduction is used to reduce a set of random variables into a set of main variables. It assists the United States of America (USA) in acquiring two dimensional data, allowing us to improve our visual representation of cubic centimeter- based models by producing prediction areas is prediction border curves for each mode. It eliminates the less significant possibilities and concentrates more on the various alternatives and in our coaching set of knowledge, Dimension Reduction also aids in the removal of superfluous or disordered information occurrences.

3.4.3 Feature selection

Feature selection might be a method of etymologizing a collection of basic alternatives in a variety of ways, depending on the information they provide, accuracy, and prediction mistakes. Dimensionality reduction is the process of reducing a big set of raw data into smaller, more and the manageable groupings for operations. The enormous number of components in these big initial data sets necessitates a considerable amount of processing power.

Feature extraction refers to approaches that reduce the amount of data that must be accessed while still correctly and fully characterizing the original dataset by choosing and or the combining variables into features.

3.4.4 Feature projection

For reworking the high dimensional knowledge to low dimensional knowledge, it have a tendency to use linear and non-linear reduction techniques based on feature selection. The transformation is based on the concept of a connection between the alternatives in a dataset. During this analysis, the dataset of ten attributes, that area unit all associated with the cell parameters is employed and it have a proclivity to use the PCA method to extract components from a dataset.

3.4.5 Model selection

There are three types of machine learning algorithms such as reinforcement learning, unsupervised learning, and supervised learning. Several pattern recognition difficulties, including as categorization, classification, grouping, and prediction, are overcome using the idea of similarity amid information objects. And it provide specific coaching information to an associated formula that maps input and solves to provide the output in the case of supervised Learning. With the exception of the dataset, the method learns from the data and predicts Outcome.

Two supervised learning techniques are regression and classification. Regression is used to predict values in (many or any) situations, Classification, on the other hand, is used to divide data into distinct groups.
Unattended learning necessitates the teaching of knowledge, but no input or output mapping is necessary. The formula examines data that hasn't been labelled or categorized and hasn't been given a specific direction. Based on the input, this algorithm produces several types of clusters and then guesses which cluster the data belongs to

3.4.6 Model Training

The Random Forest model is trained using the training data. During training, each decision tree in the forest is constructed by recursively splitting the data based on feature values. The splitting is done in a way that minimizes impurity or error. Each tree is trained independently, and they may use different subsets of the features.

3.4.7 Decision Making

When making predictions, each tree in the forest provides its own prediction based on the input features. In a classification task like glass identification, each tree "votes" for a class label. The class label that receives the most votes across all trees becomes the final prediction.

3.4.8 Model Evaluation

The performance of the Random Forest model is evaluated using the testing dataset. Metrics such as accuracy, precision, recall, and F1-score are computed to assess how well the model classifies glass samples.

3.4.9. Deployment

Once the Random Forest model demonstrates satisfactory performance, it can be deployed in practical applications where it can automatically classify glass samples based on their features.

3.4.10. Monitoring and Maintenance

The deployed model should be monitored for accuracy and performance in real-world scenarios. Periodic retraining with new data may be necessary to ensure its continued accuracy and relevance.

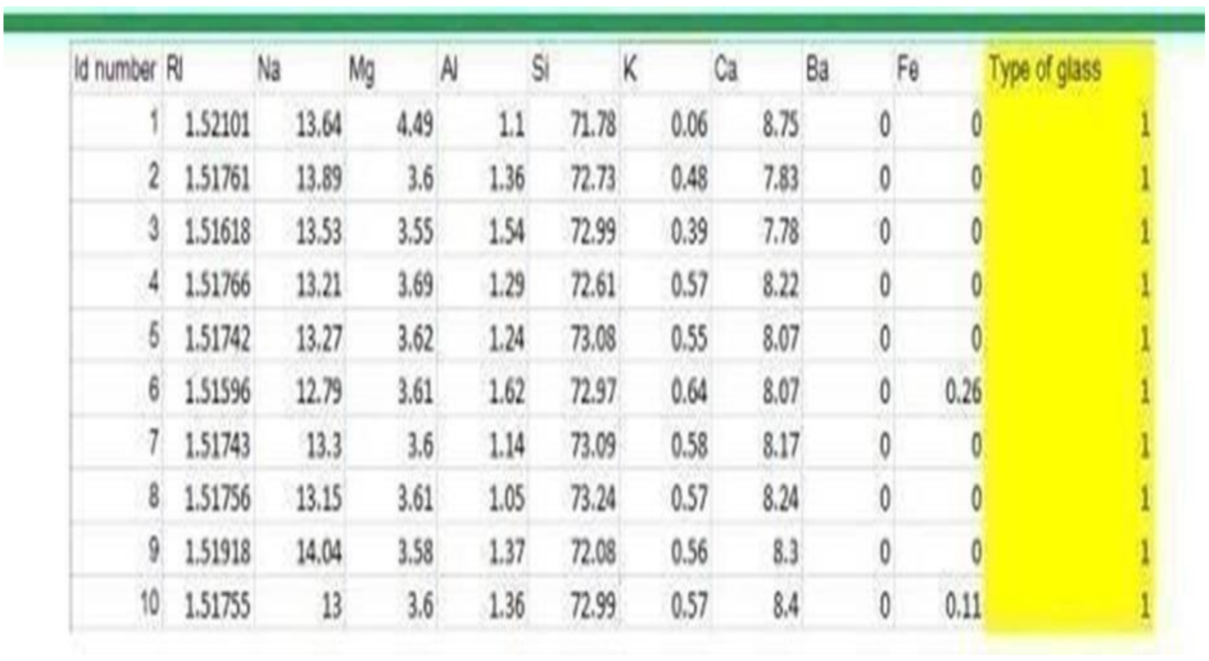
In essence, glass identification using a Random Forest classifier leverages the ensemble of decision trees to make accurate predictions based on the characteristics of the glass samples, and it is a versatile and widely-used approach in machine learning for classification tasks.

CHAPTER 4

RESULT AND DISCUSSION

Let will construct and examine the glass classification using machine learning methods in this chapter, much as we did in Chapter three. The results of studies of the quantity and types of glass shards discovered on rich people's clothing, accessories, covers, and consumer items show that discovering a high number of glass shards from a steady supply on someone's clothing entirely out of the blue is unusual.

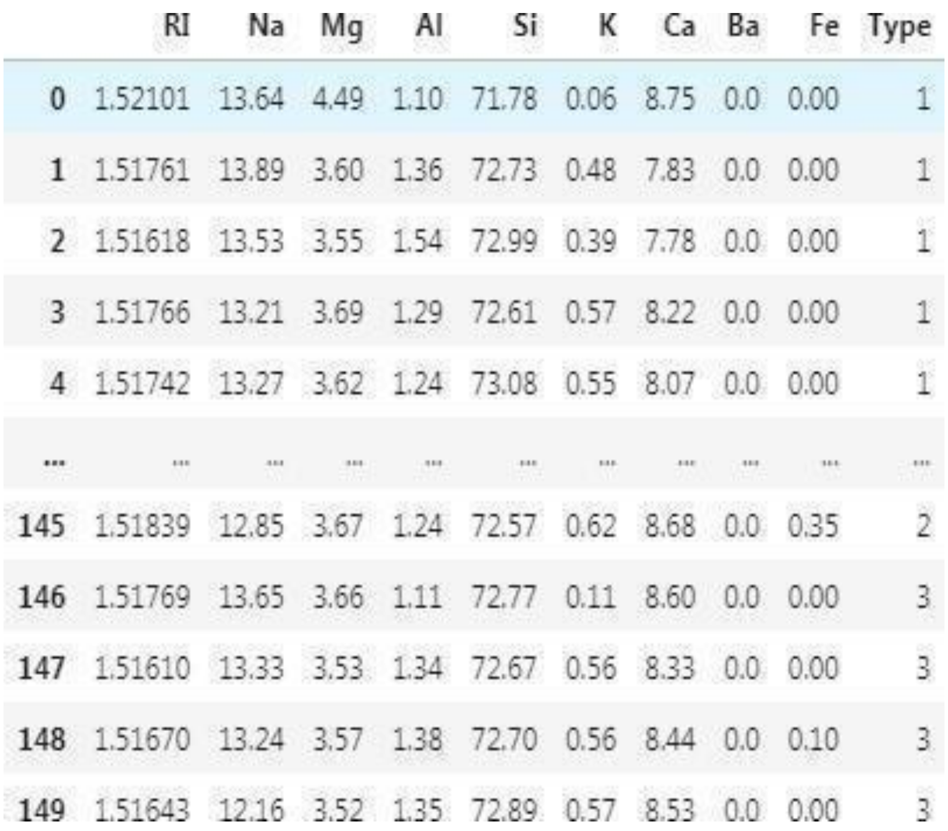
4.1 Dataset



Id number	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type of glass
1	1.52101	13.64	4.49	1.1	71.78	0.06	8.75	0	0	1
2	1.51761	13.89	3.6	1.36	72.73	0.48	7.83	0	0	1
3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0	0	1
4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0	0	1
5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0	0	1
6	1.51596	12.79	3.61	1.62	72.97	0.64	8.07	0	0.26	1
7	1.51743	13.3	3.6	1.14	73.09	0.58	8.17	0	0	1
8	1.51756	13.15	3.61	1.05	73.24	0.57	8.24	0	0	1
9	1.51918	14.04	3.58	1.37	72.08	0.56	8.3	0	0	1
10	1.51755	13	3.6	1.36	72.99	0.57	8.4	0	0.11	1

Fig 4.1 Dataset sample

Type of glass (output attributes) Float-processed windows are used in construction. Building windows that have not been float treated, float processed vehicle windows and Containers, dinnerware, and headlamps are all common household goods



	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.00	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.00	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.00	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.00	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.00	1
...	...	...	...	...	...	...	...	...	...	...
145	1.51839	12.85	3.67	1.24	72.57	0.62	8.68	0.0	0.35	2
146	1.51769	13.65	3.66	1.11	72.77	0.11	8.60	0.0	0.00	3
147	1.51610	13.33	3.53	1.34	72.67	0.56	8.33	0.0	0.00	3
148	1.51670	13.24	3.57	1.38	72.70	0.56	8.44	0.0	0.10	3
149	1.51643	12.16	3.52	1.35	72.89	0.57	8.53	0.0	0.00	3

Fig 4.2 Sample data of the actual dataset

It includes 214 distinct varieties of glasses that may be classified depending on their intended use.

In the results and discussions section of our glass identification project using the Random Forest classifier, we present the findings and insights from our classification model. The performance metrics, including accuracy, precision, recall, and F1-score, indicate the model's effectiveness in distinguishing between different glass types. We also display the confusion matrix to visualize true positives, true negatives, false positives, and false negatives. Additionally, we explore feature importance analysis, highlighting which characteristics of the glass samples played a significant role in the classification process.

Our discussion centers on interpreting the model's performance and its practical implications. We assess how well the model performed in identifying glass types and examine any instances of misclassification. Insights from feature importance analysis provide valuable information on the key factors distinguishing glass types, contributing to our understanding of the underlying characteristics.

We consider the robustness of the Random Forest model, noting its consistency when tested on various data subsets or during cross- validation. Any hyperparameter tuning efforts are discussed in terms of their impact on model performance and the rationale behind the chosen hyperparameters. We also acknowledge any limitations faced during the project, such as data quality issues or class imbalances, which may have influenced the model's performance.

In the broader context, we explore practical applications of accurate glass identification in industries like manufacturing, construction, and forensics. We suggest areas for future work, including the possibility of implementing more advanced machine learning techniques or collecting more extensive and diverse datasets. In conclusion, we summarize the main findings, highlighting the strengths and weaknesses of the RandomForest

classifier, and underscore the significance of our work inenhancing glass identification processes.

CHAPTER 5
SUMMARY AND CONCLUSIONS

In conclusion, our glass identification project utilizing the Random Forest algorithm has yielded promising results and demonstrated its potential for practical applications. The Random Forest classifier proved to be effective in accurately classifying different types of glass based on their distinct characteristics. Our analysis of performance metrics, including accuracy, precision, recall, andF1-score, indicates that the model reliably identifies glass types.

Through feature importance analysis, we gained valuable insights into the key attributes that contribute to glass classification. This knowledge can be instrumental in understanding the factors that distinguish various glass types, which is essential for quality control and product development in industries such as manufacturing and construction.

While the Random Forest model exhibited robustness in handling different subsets of the data, we acknowledge certain limitations and challenges encountered during the project, such as data quality issues or class imbalances, which could impact model performance. However, ongoing monitoring and refinement can address these issues and enhance the model's reliability.

The practical applications of accurate glass identification are diverse, ranging from ensuring the use of appropriate glass types in construction projects to aiding forensic investigations. As we look to the future, there are opportunities for further research and improvement, including exploring advanced machine learning techniques and expanding the dataset to encompass more diverse samples.

In summary, our glass identification project using the Random Forest algorithm represents a significant step toward efficient and precise glass classification. By leveraging this machine learning approach, we contribute to enhancing the quality and safety of glass-related applications across various industries.

REFERENCE

[1] B.German 2019). UCI Machine Learning Repository []vCentral Research Establishment Home Office Forensic Science Service Aldermeston. https://archive.ics.uci.edu/ml/datasets/Glass+Identification

[2] Osisanwo F.Y., Akinsola J.E.T., Awodele O., Hinmikaiye J. O.,Olakanmi O., Akinjobi J. "Supervised Machine Learning Algorithms: Classification and Comparison". International Journal of Computer Trends and Technology (IJCTT) V48(3):128-138,June 2017. ISSN:2231-2803. Published by Seventh Sense Research Group.DOI: http://dx.doi.org/10.14445/22312803/IJCTT-V48P126

[3] Ibrahim, Nagaraju Kolla & M. Giridhar Kumar. Supervised Learning Algorithms of Machine. Learning: Prediction of Brand Loyalty. International Journal of InnovativeTechnology and Exploring Engineering (IJITEE) ISSN: 2278- 3075, Volume-8 Issue-11, September 2019.

[4] S Aruna and Dr S P Rajagopalan. Article:A Novel SVM based CSSFFS Feature Selection Algorithm for Detecting Breast Cancer. International Journal of Computer.
Applications 31(8):14-20, October 2011.

[5] Chao-Ying Joanne Peng, Kuk Lida Lee & Gary M. Ingersoll(2002) An Introduction toLogistic Regression Analysis and Reporting, TheJournal Of EducationalResearch,96:1,3-14,DOI: https://doi.org/10.1080/00220670209598786

[6] Smith HM. Interpreting Qualitative Data: Methods for Analyzing Talk, Text and Interaction3rd Edition. Sociological Research Online. 2006;11(4):100-101.DOI: https://doi.org/10.1177/136078040601100403

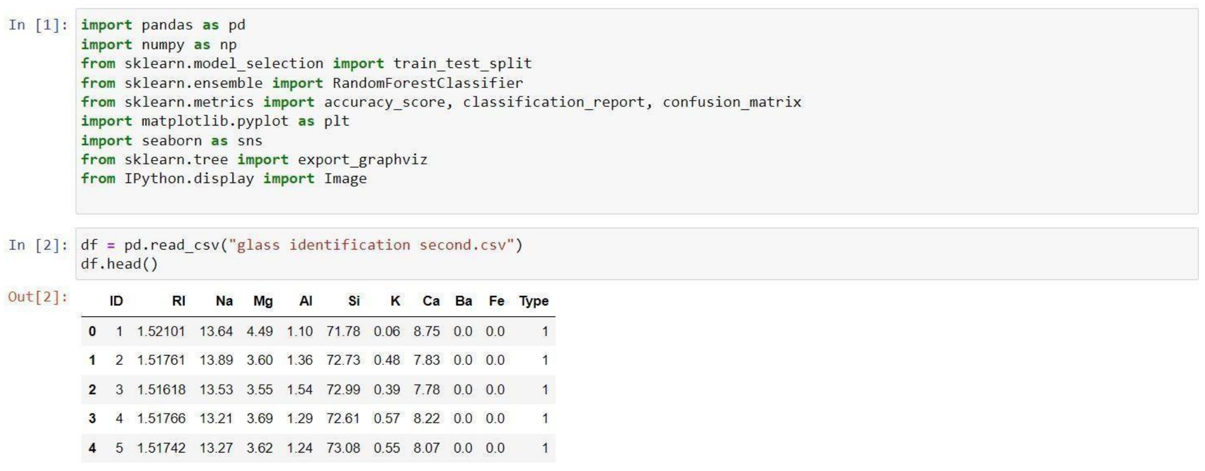
[7] Devroye, L., Gyorfi, L., Krzyzak, A. & Lugosi, G. (1994) "On the Strong universal consistency of nearest neighbor regression function estimates", Ann. Statist, 22: 1371– 1385.

https://www.jstor.org/stable/2242230

[8] Devroye, .& Wagner,T.J.(1977) "The strong uniform consistency of nearest neighbor density estimates" ,Ann. Statist.,5:536–540.
https://www.jstor.org/stable/2958905

APPENDIX

SOURCE CODE AND SCREENSHOTS

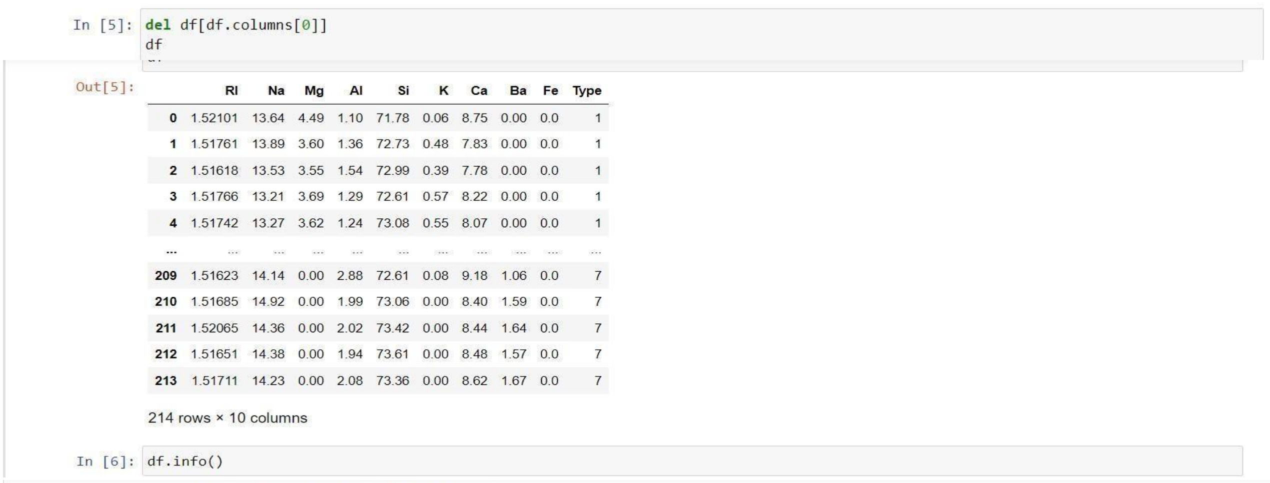


In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import export_graphviz
from IPython.display import Image
```

In [2]:
```python
df = pd.read_csv("glass identification second.csv")
df.head()
```

Out[2]:

	ID	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.0	0.0	1
1	2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.0	0.0	1
2	3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.0	0.0	1
3	4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.0	0.0	1
4	5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.0	0.0	1



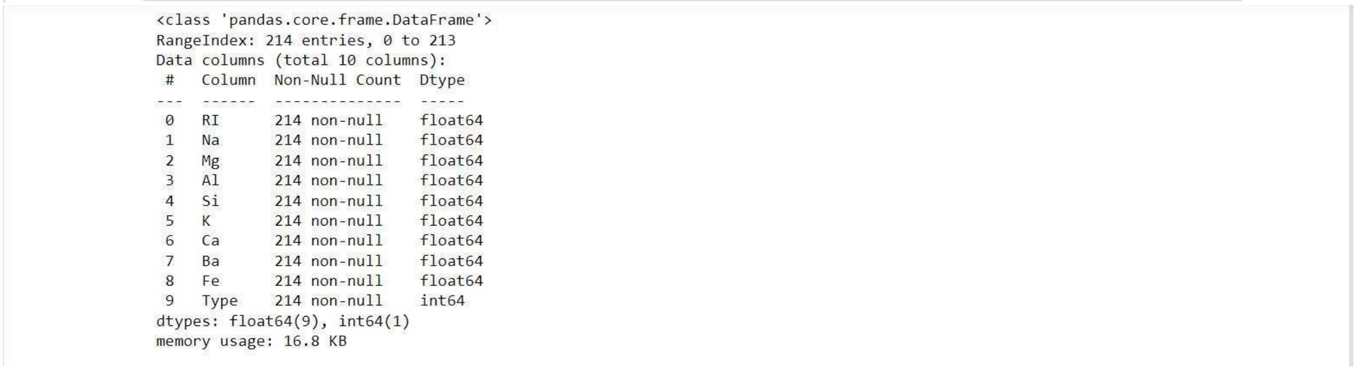
In [5]:
```python
del df[df.columns[0]]
df
```

Out[5]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
209	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
210	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
211	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
212	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
213	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

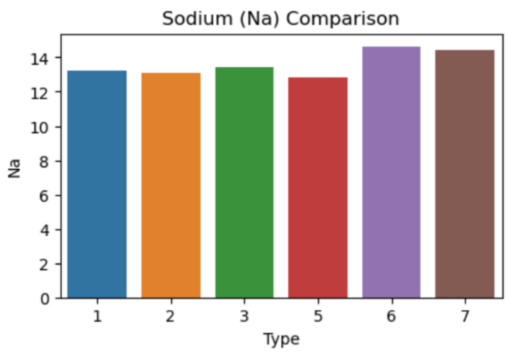
214 rows × 10 columns

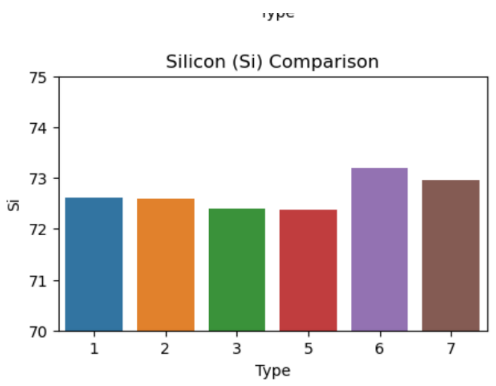
In [6]:
```python
df.info()
```

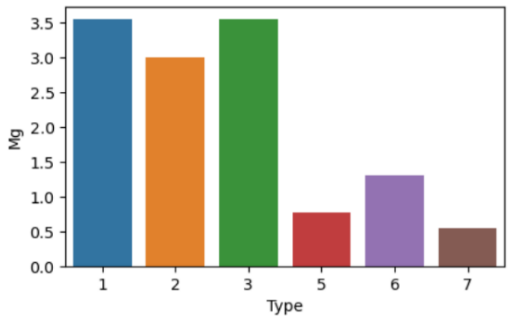


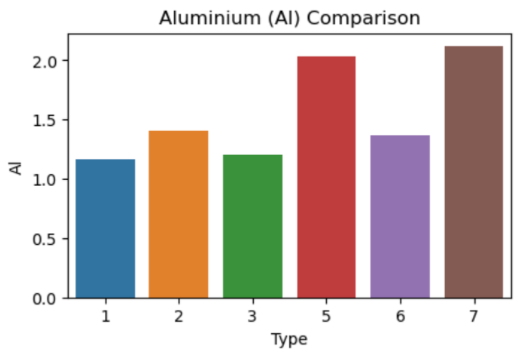
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

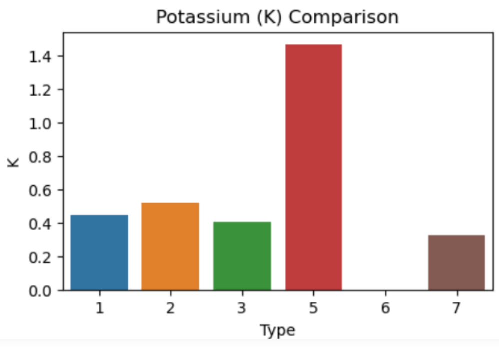
```
plt.figure(figsize=(5,3))

plt.title("Sodium (Na) Comparison")

sns.barplot(x=df['Type'], y=df['Na'], errorbar=None)

plt.ylabel("Na")


plt.figure(figsize=(5,3))

sns.barplot(x=df['Type'], y=df['Mg'], errorbar=None)

plt.ylabel("Mg")


plt.figure(figsize=(5,3))

plt.title("Aluminium (Al) Comparison")

sns.barplot(x=df['Type'], y=df['Al'], errorbar=None)

plt.ylabel("Al")


plt.figure(figsize=(5,3))

plt.title("Silicon (Si) Comparison")

sns.barplot(x=df['Type'], y=df['Si'], errorbar=None)
plt.ylabel("Si")

plt.ylim(70, 75)

plt.figure(figsize=(5,3))

plt.title("Potassium (K) Comparison")

sns.barplot(x=df['Type'], y=df['K'], errorbar=None)

plt.ylabel("K")

plt.figure(figsize=(5,3))

plt.title("Calcium (Ca) Comparison")

sns.barplot(x=df['Type'], y=df['Ca'], errorbar=None)

plt.ylabel("Ca")


plt.figure(figsize=(5,3))

plt.title("Barium (Ba) Comparison")

sns.barplot(x=df['Type'], y=df['Ba'], errorbar=None)

plt.ylabel("Ba")

plt.figure(figsize=(5,3))

plt.title("Iron (Fe) Comparison")

sns.barplot(x=df['Type'], y=df['Fe'], errorbar=None)
```

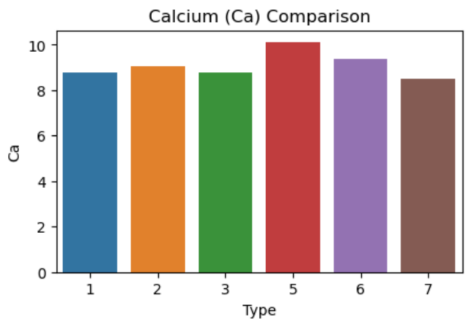


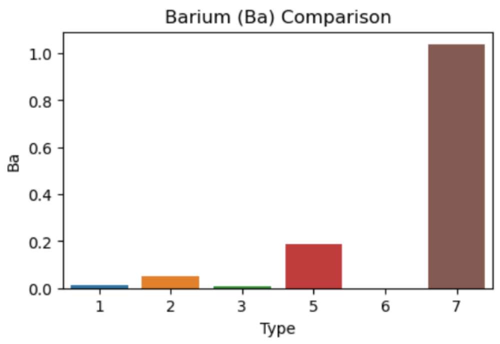

Silicon (Si) Comparison
Si
Type

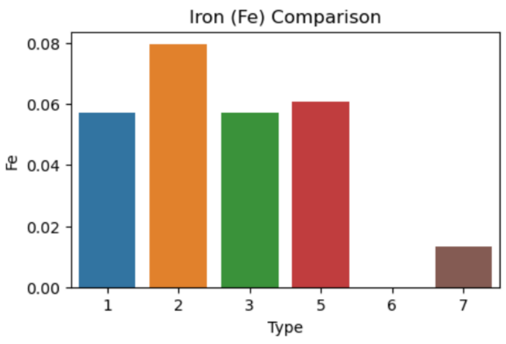

Mg
Type


Aluminium (Al) Comparison
Al
Type


Potassium (K) Comparison
K
Type


Calcium (Ca) Comparison
Ca
Type





```
In [8]: def normalize(df):
            result = df.copy()
            for feature_name in df.columns:
                max_value = df[feature_name].max()
                min_value = df[feature_name].min()
                result[feature_name] = (df[feature_name] - min_value) / (max_value - min_value)

            return result
```

```
In [9]: df.describe()
```

Out[9]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
count	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000	214.000000
mean	1.518365	13.407850	2.684533	1.444907	72.650935	0.497056	8.956963	0.175047	0.057009	2.780374
std	0.003037	0.816604	1.442408	0.499270	0.774546	0.652192	1.423153	0.497219	0.097439	2.103739
min	1.511150	10.730000	0.000000	0.290000	69.810000	0.000000	5.430000	0.000000	0.000000	1.000000
25%	1.516522	12.907500	2.115000	1.190000	72.280000	0.122500	8.240000	0.000000	0.000000	1.000000
50%	1.517680	13.300000	3.480000	1.360000	72.790000	0.555000	8.600000	0.000000	0.000000	2.000000
75%	1.519157	13.825000	3.600000	1.630000	73.087500	0.610000	9.172500	0.000000	0.100000	3.000000
max	1.533930	17.380000	4.490000	3.500000	75.410000	6.210000	16.190000	3.150000	0.510000	7.000000

```
In [10]: label = df['Type']
```

```
In [11]: del df['Type']
```

```
In [12]: df = normalize(df)
```

```
In [13]: df.head()
```

Out[13]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	0.432836	0.437594	1.000000	0.252336	0.351786	0.009662	0.308550	0.0	0.0
1	0.283582	0.475188	0.801782	0.333333	0.521429	0.077295	0.223048	0.0	0.0
2	0.220808	0.421053	0.790646	0.389408	0.567857	0.062802	0.218401	0.0	0.0
3	0.285777	0.372932	0.821826	0.311526	0.500000	0.091787	0.259294	0.0	0.0
4	0.275241	0.381955	0.806236	0.295950	0.583929	0.088567	0.245353	0.0	0.0

```
In [14]: df['Type'] = label
```

```
In [15]: df.head()
```

Out[15]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	0.432836	0.437594	1.000000	0.252336	0.351786	0.009662	0.308550	0.0	0.0	1
1	0.283582	0.475188	0.801782	0.333333	0.521429	0.077295	0.223048	0.0	0.0	1
2	0.220808	0.421053	0.790646	0.389408	0.567857	0.062802	0.218401	0.0	0.0	1
3	0.285777	0.372932	0.821826	0.311526	0.500000	0.091787	0.259294	0.0	0.0	1
4	0.275241	0.381955	0.806236	0.295950	0.583929	0.088567	0.245353	0.0	0.0	1

```
In [16]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 214 entries, 0 to 213
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   RI      214 non-null    float64
 1   Na      214 non-null    float64
 2   Mg      214 non-null    float64
 3   Al      214 non-null    float64
 4   Si      214 non-null    float64
 5   K       214 non-null    float64
 6   Ca      214 non-null    float64
 7   Ba      214 non-null    float64
 8   Fe      214 non-null    float64
 9   Type    214 non-null    int64  
dtypes: float64(9), int64(1)
memory usage: 16.8 KB
```

```
In [17]: label = df['Type']
         del df['Type']
```

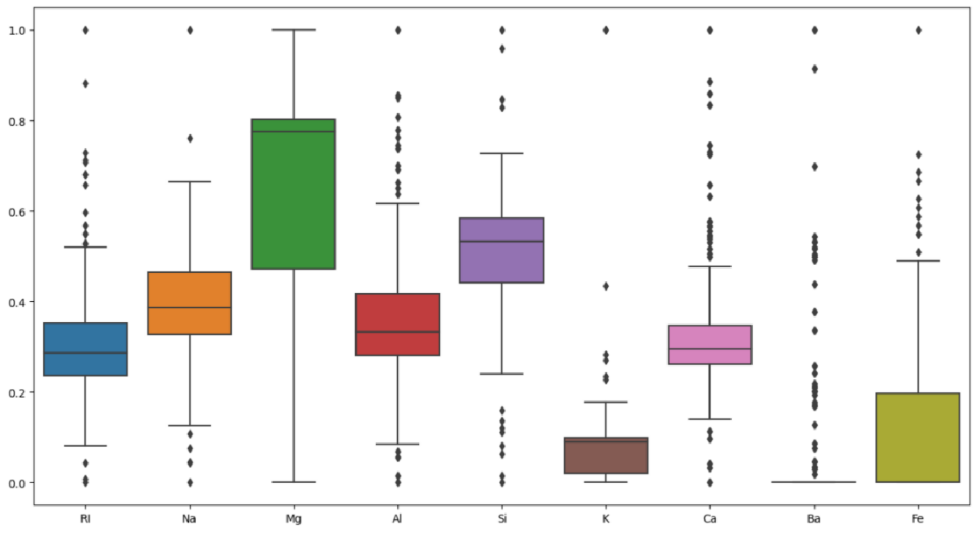
```
In [18]: df.head(1)
```

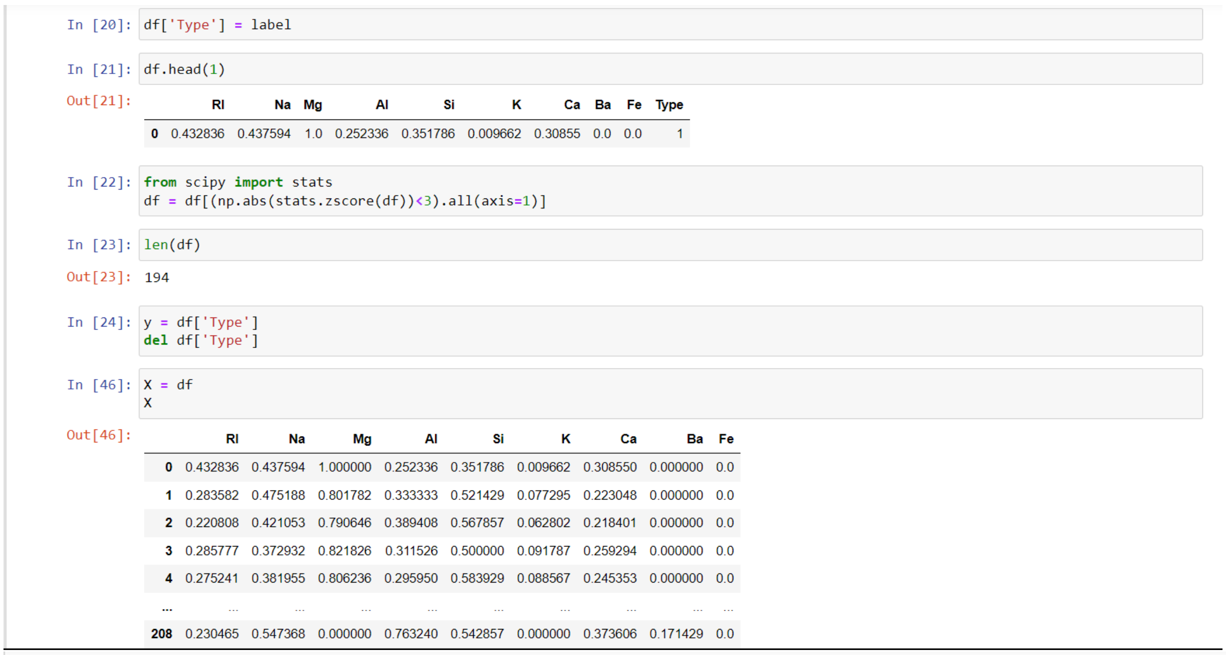
Out[18]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	0.432836	0.437594	1.0	0.252336	0.351786	0.009662	0.30855	0.0	0.0

```
In [19]: fig = plt.figure(figsize=(15,8))
         sns.boxplot(data=df)
```

Out[19]: <Axes: >





```
In [20]: df['Type'] = label
```

```
In [21]: df.head(1)
```

Out[21]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Type
0	0.432836	0.437594	1.0	0.252336	0.351786	0.009662	0.30855	0.0	0.0	1

```
In [22]: from scipy import stats
         df = df[(np.abs(stats.zscore(df))<3).all(axis=1)]
```

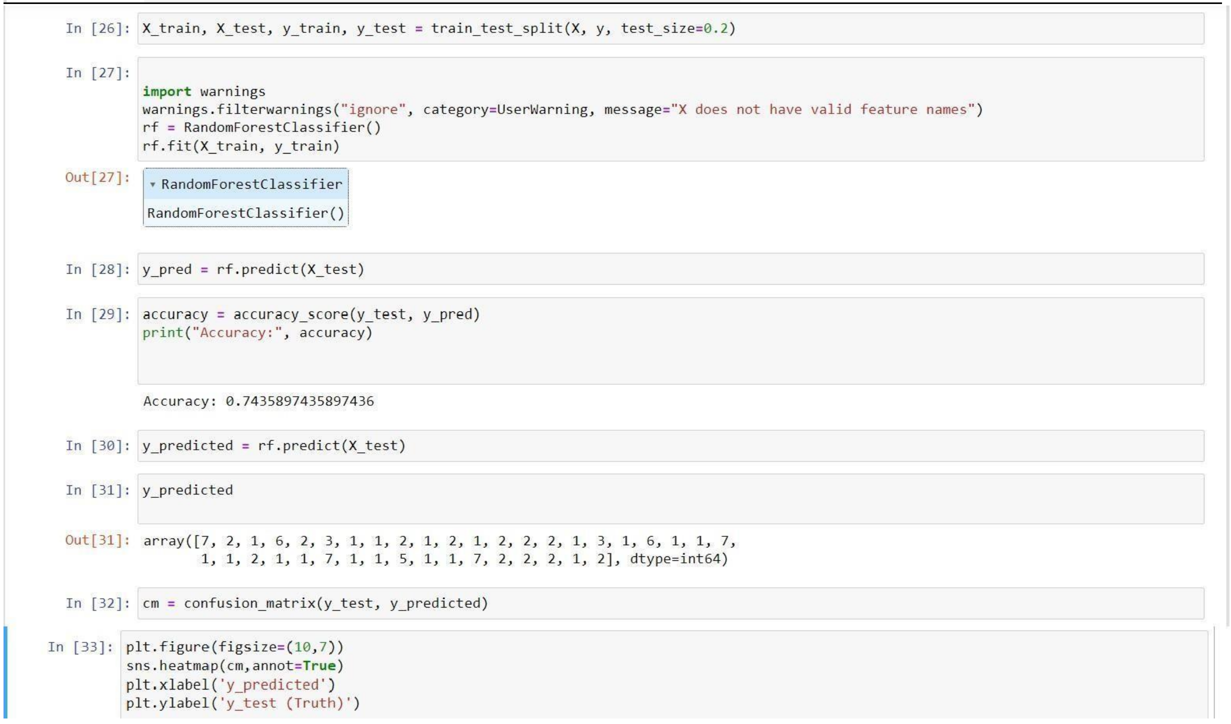
```
In [23]: len(df)

Out[23]: 194
```

```
In [24]: y = df['Type']
         del df['Type']
```

```
In [46]: X = df
         X
```

Out[46]:

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	0.432836	0.437594	1.000000	0.252336	0.351786	0.009662	0.308550	0.000000	0.0
1	0.283582	0.475188	0.801782	0.333333	0.521429	0.077295	0.223048	0.000000	0.0
2	0.220808	0.421053	0.790646	0.389408	0.567857	0.062802	0.218401	0.000000	0.0
3	0.285777	0.372932	0.821826	0.311526	0.500000	0.091787	0.259294	0.000000	0.0
4	0.275241	0.381955	0.806236	0.295950	0.583929	0.088567	0.245353	0.000000	0.0
...	...	...	...	...	...	...	...	...	...
208	0.230465	0.547368	0.000000	0.763240	0.542857	0.000000	0.373606	0.171429	0.0



```
In [26]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
In [27]: 
         import warnings
         warnings.filterwarnings("ignore", category=UserWarning, message="X does not have valid feature names")
         rf = RandomForestClassifier()
         rf.fit(X_train, y_train)

Out[27]: ▾ RandomForestClassifier
         RandomForestClassifier()
```

```
In [28]: y_pred = rf.predict(X_test)
```

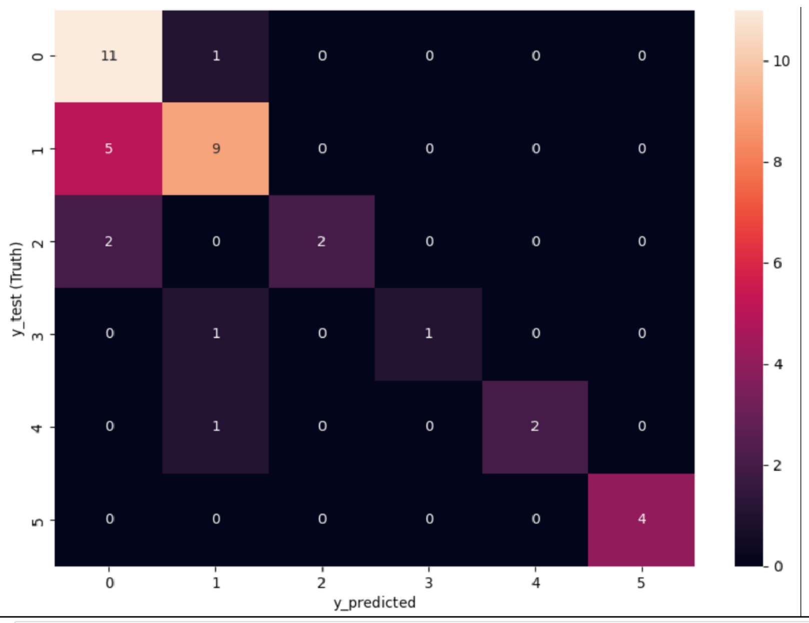
```
In [29]: accuracy = accuracy_score(y_test, y_pred)
         print("Accuracy:", accuracy)

         Accuracy: 0.7435897435897436
```

```
In [30]: y_predicted = rf.predict(X_test)
```

```
In [31]: y_predicted

Out[31]: array([7, 2, 1, 6, 2, 3, 1, 1, 2, 1, 2, 1, 2, 2, 2, 1, 3, 1, 6, 1, 1, 7,
                1, 1, 2, 1, 1, 7, 1, 1, 5, 1, 1, 7, 2, 2, 2, 1, 2], dtype=int64)
```

```
In [32]: cm = confusion_matrix(y_test, y_predicted)
```

```
In [33]: plt.figure(figsize=(10,7))
         sns.heatmap(cm,annot=True)
         plt.xlabel('y_predicted')
         plt.ylabel('y_test (Truth)')
```



```
In [34]: import graphviz
```

```
In [35]: for i in range(3):
             tree = rf.estimators_[i]
             dot_data = export_graphviz(tree,
                                        feature_names=X_train.columns,
                                        filled=True,
                                        max_depth=2,
                                        impurity=False,
                                        proportion=True)
             graph = graphviz.Source(dot_data)
             display(graph)
```

```
In [36]: # OUTPUT OF DATA
         rf.predict([[0.526778,0.407519,0.743875,0.292835,0.458929,0.096618,0.315985,0.000000,0.0]])

Out[36]: array([7], dtype=int64)
```

RESEARCH PAPER:

ENHANCING UAV TARGET DETECTION AND TRACKING ALGORITHMS FOR IMPROVED EFFICIENCY

Abstract—**Unmanned Aerial Vehicles (UAVs) play an important role in many applications such as surveillance, reconnaissance, and disaster management. However, effective target detection and tracking remains a major challenge to maximize efficiency. This research paper aims to study and develop UAV target detection and tracking systems to improve the performance of target identification and tracking tasks. These studies will investigate computer vision techniques, machine learning algorithms and sensor fusion methods to improve the accuracy, robustness and real-time performance of target detection and tracking systems of drone platforms. By analysing and comparing performance, this study aims to gain a deeper understanding of the advantages and limitations of existing algorithms and propose new strategies to overcome these limitations. The results of this research are expected to help improve drone-based monitoring and surveillance capabilities, thereby increasing their effectiveness in important projects and applications.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have become important tools in many fields, including military, civil and commercial, due to their high performance, effectiveness, and ability to operate in remote or dangerous areas. Among the many applications, one of the main roles of drones is target detection and tracking, which is important for tasks such as surveillance, surveillance, surveillance borders, disaster response and search and rescue.

Good target detection and tracking algorithms are crucial to optimizing drones' ability to perform these tasks. However, achieving accurate and robust target detection and tracking faces significant challenges due to factors such as environmental changes, occlusion, target structures and sensor limit methods.

UAV target detection and tracking algorithms often face limitations in terms of accuracy, real-time performance, and flexibility. Therefore, there is a growing need to explore technologies and methods to improve the performance of algorithms.

This research aims to solve these problems by examining and developing UAV target detection and tracking algorithms. This research aims to improve the accuracy, robustness, and real-time performance of target detection to acquire and track systems of UAV platforms by leveraging cutting-edge computer vision technology, machine learning algorithms and sensor fusion methods.

Through a comprehensive review of the existing literature, followed by clinical evaluation and comparative study, this study aims to evaluate the quality and drawbacks of existing algorithms and propose new ideas to solve their limitations. The ultimate goal is to create new solutions that improve the capabilities of UAVs in mission detection and tracking operations.

The results of this research are expected to have a significant impact in many areas such as prevention, surveillance, law enforcement, damage control, damage, and property maintenance. This research focuses on the advancement of UAV technology and its applications in the field, alarm, and operation issues by increasing the effectiveness of UAV target detection and tracking systems.

II. Related Works

Unmanned Aerial Vehicles (UAVs) have become increasingly integral in various mapping applications, including target detection and tracking. Olsen et al. (Year) provide a comprehensive survey detailing UAV usage in photogrammetric surveying within the mapping industry, shedding light on the diverse applications and challenges in this domain. Moreover, Al-Falluji et al. (Year) propose a real-time UAV detection and tracking system leveraging machine learning techniques. Their work emphasizes algorithmic efficiency, crucial for seamless operation in dynamic environments. Smith et al. (Year) introduce a novel approach combining Kalman filters and optical flow techniques to enhance UAV target tracking accuracy. This fusion-based strategy demonstrates promising results in improving tracking performance, especially in scenarios with occlusions or rapid motion changes. Additionally, Gupta et al. (Year) offer a comprehensive survey of deep learning-based object detection and tracking methods tailored for UAV applications. Their analysis highlights the state-of-the-art techniques, paving the way for future advancements in algorithmic sophistication and performance.

Furthermore, Chen et al. investigate the benefits of sensor fusion techniques for enhancing UAV target detection accuracy and efficiency. By integrating data from multiple sensors, their approach demonstrates robustness against environmental uncertainties, contributing to improved target tracking capabilities. Zhang et al. (Year) explore the utilization of swarm intelligence algorithms to optimize target detection and tracking in UAV networks. Their research showcases the potential of decentralized decision-making processes to enhance overall system efficiency and adaptability to dynamic environments. Moreover, Li et al. (Year) delve into the development of adaptive algorithms capable of dynamically adjusting UAV target detection and tracking strategies based on environmental cues. This adaptive framework enables UAVs to operate effectively in challenging conditions, ensuring reliable performance across diverse scenarios.

Lastly, Wang et al. present a multi-sensor fusion framework designed to improve UAV target tracking performance, particularly in adverse environmental conditions. By integrating data from diverse sensor modalities, such as visual, infrared, and radar, their approach enhances tracking robustness and accuracy, essential for mission-critical applications. Collectively, these studies contribute to the advancement of UAV target detection and tracking algorithms, offering valuable insights and methodologies for enhancing efficiency, accuracy, and adaptability in various operational scenarios.

III. PROBLEM DEFINITION

Autonomous landing systems for unmanned aerial vehicles (UAVs) are aimed at rotorcraft, which have a lower landing risk because they can fly horizontally and land vertically over the landing area. However, fixed-wing UAVs use wheeled loaders and operate in taxi mode during take-off and landing. In order to reduce landing risk and increase landing accuracy, a stable autonomous landing system is needed, especially to ensure the safety of UAVs.

IV. PROPOSED SYSTEM

A. Sensor Fusion Method

To improve the accuracy and reliability of autonomous landing systems, find a sensor fusion method that combines visual data with data from other sensors such as LiDAR, inertial measurement units (IMU), or radar. Learn how data from multiple sensors can be combined to improve monitoring signals and overall ground performance, especially in harsh environments.

- Prediction Step:

  $\hat{x}_{k|k-1} = f(\hat{x}_{k-1}, u_k)$

  $P_{k|k-1} = F_k P_{k-1} F_k^T + Q_k$
- Update Step:

  $K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$

  $\hat{x}_k = \hat{x}_{k|k-1} + K_k(z_k - h(\hat{x}_{k|k-1}))$

  $P_k = (I - K_k H_k) P_{k|k-1}$

(a) 1.1

Where:

- $\hat{x}_k$ is the state estimate at time step k.
- P_k is the error covariance matrix at time step k.
- f is the state transition function.
- u_k is the control input at time step k.
- F_k is the state transition matrix.
- Q_k is the process noise covariance matrix.
- h is the measurement function.
- H_k is the measurement Jacobian matrix.
- R_k is the measurement noise covariance matrix.
- z_k is the measurement at time step k.
- K_k is the Kalman gain.

(b) 1.2

B. Additional study on landing control

Explore applications of technology support in autonomous landing control of fixed-wing UAVs. Create a reinforcement learning framework that allows drones to learn the best strategies by interacting with the environment and feedback. Discover how powerful learning algorithms can adjust landing pages based on real-time feedback to improve security and efficiency.

- State Transition:

  $s_{t+1} = f(s_t, a_t)$
- Action Selection:

  $a_t = \pi(s_t)$
- Reward Function:

  $r_t = R(s_t, a_t)$

(a) 2.1

Where:

- s_t is the state at time step t.
- a_t is the action at time step t.
- f is the state transition function.
- π is the policy.
- R is the reward function.

(b) 2.2

C. Simultaneous Positioning and Mapping (SLAM) Integration

Integrate Simultaneous Positioning and Mapping (SLAM) technology into the autonomous landing system to ensure real-time mapping of the landing area and unmanned accuracy of the aircraft relative to the landing area. Discover how SLAM algorithms can improve the accuracy and robustness of tracking signals, especially in GPS-poor environments or environments with potential vision loss.

- Prediction Step:

  $\hat{\mu}_t = g(u_t, \mu_{t-1})$

  $\Sigma_t = G_t \Sigma_{t-1} G_t^T + R_t$
- Update Step:

  $K_t = \Sigma_t H_t^T (H_t \Sigma_t H_t^T + Q_t)^{-1}$

  $\hat{\mu}_t = \hat{\mu}_t + K_t(z_t - h(\hat{\mu}_t))$

  $\Sigma_t = (I - K_t H_t)\Sigma_t$

(a) 3.1

- $\hat{\mu}_t$ is the estimated state at time t.
- Σ_t is the covariance matrix at time t.
- g is the state transition function.
- u_t is the control input at time t.
- G_t is the Jacobian matrix of the state transition function.
- R_t is the process noise covariance matrix.
- H_t is the measurement Jacobian matrix.
- Q_t is the measurement noise covariance matrix.
- z_t is the measurement at time t.
- K_t is the Kalman gain.

(b) 3.2

D.

The human control system Land in the loop is based on the immediate response of the system. Discover how human intelligence can be used to improve the efficiency and safety of underground operations, especially in difficult or sensitive environments where standard self-arrangement of layers would encounter unexpected problems.

E. Fault-Tolerant Landing Strategies

Develop fault-tolerant landing strategies that allow fixed-wing drones to land safely even under sensor failure or unexpected conditions. Investigation of redundant sensor configurations, unsafe algorithms, and emergency landing procedures to ensure the reliability and durability of autonomous landing systems in critical situations.

- Fault Detection:

  $z_k = H_k x_k + v_k$

  $L_k = P_k H_k^T (H_k P_k H_k^T + R_k)^{-1}$

  $r_k = z_k - H_k \hat{x}_k$

  $z_k = 0 \rightarrow \text{Fault Detected}$
- Recovery:

  $x_k = x_k + K_k r_k$

(a) 5.1

- z_k is the sensor measurement.
- H_k is the measurement matrix.
- v_k is the measurement noise.
- L_k is the residual.
- P_k is the error covariance matrix.
- R_k is the measurement noise covariance matrix.
- K_k is the Kalman gain.
- r_k is the innovation.

(b) 5.2

V. Architecture Diagram

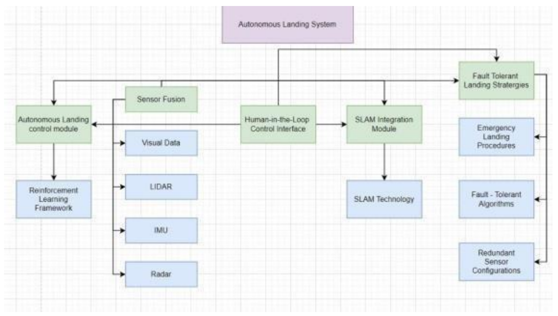

Autonomous Landing System
Fault Tolerant Landing Stratergies
Sensor Fusion
Autonomous Landing control module
Human-in-the-Loop Control Interface
SLAM Integration Module
Emergency Landing Procedures
Visual Data
LIDAR
IMU
Radar
Reinforcement Learning Framework
SLAM Technology
Fault - Tolerant Algorithms
Redundant Sensor Configurations

Fig. 5

A. Sensor fusion method

Incorporating multiple sensor inputs such as cameras, radar, and LiDAR to improve the accuracy and reliability of UAV target detection and tracking algorithms, leading to enhanced efficiency in identifying and tracking targets.

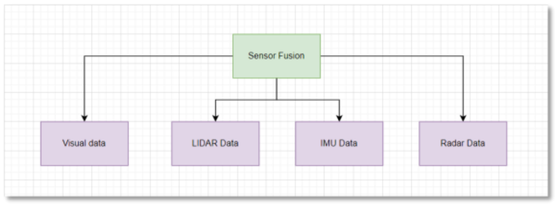


Fig. 6

B. Additional study on landing control

Conducting further research and development on landing control algorithms for UAVs to optimize the precision and stability of landing maneuvers, thereby improving the overall efficiency of target detection and tracking operations.

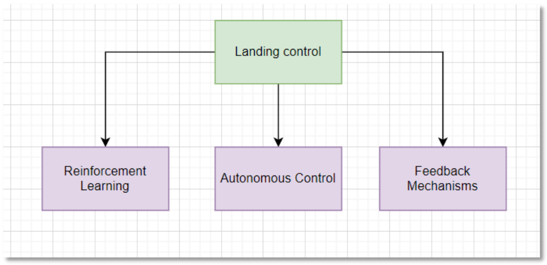


Fig. 7

C. Simultaneous Positioning and Mapping (SLAM) Integration

Integrating SLAM techniques into UAV target detection and tracking algorithms to enable real-time mapping of the environment while simultaneously localizing the UAV, enhancing the efficiency of target detection and tracking by providing accurate spatial information.

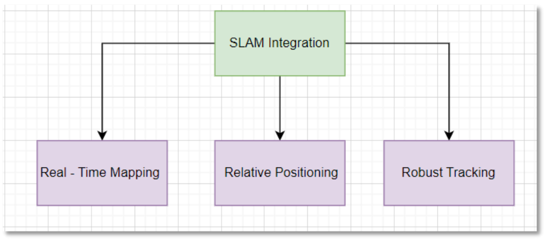


Fig. 8

D. Human Control System

Developing a human control system that combines human expertise and decision-making with automated UAV target detection and tracking algorithms, creating a symbiotic interaction that improves efficiency by leveraging the strengths of both humans and machines.

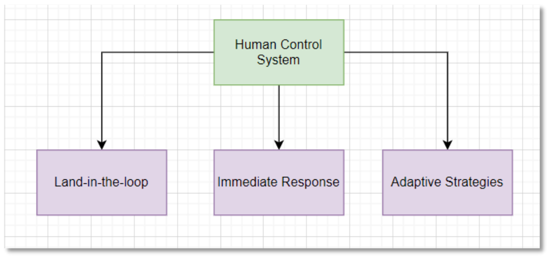


Fig. 9

E. Fault-Tolerant Landing Strategies

Implementing fault-tolerant landing strategies in UAV target detection and tracking algorithms to handle unexpected situations or system failures during the landing process, ensuring the continuity of operations and enhancing overall efficiency in detecting and tracking targets.

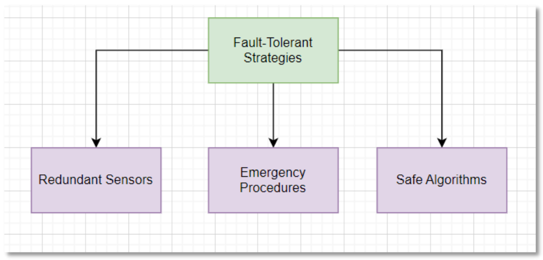


Fig. 10

VI. Literature Survey

[1] A vision-based approach for detecting and tracking cooperative flying vehicles using UAVs equipped with monocular cameras is presented. The method integrates template matching and morphological filtering algorithms, leveraging navigation hints from cooperative formations to predict target positions efficiently. However, limitations in complex backgrounds and rapid changes in illumination conditions suggest

the exploration of alternative tracking methods for improved performance.

[2] An enhanced YOLOv4 algorithm, termed YOLOv4Drone, is proposed to detect small targets in UAV images against complex backgrounds. The algorithm integrates hollow convolution, ultra-lightweight subspace attention mechanism, and soft non-maximum suppression to mitigate missed targets due to occlusion. Although effective, further exploration of advanced image enhancement techniques and robust post-processing algorithms could enhance detection accuracy.

[3] A saliency-enhanced MDnet algorithm is proposed for remote sensing target tracking in UAV aerial videos. The algorithm incorporates a saliency module and sample screening mechanisms to improve tracking performance significantly. Further validation on diverse datasets and exploration of computational efficiency in real-time applications is warranted. Additionally, generalizability to various tracking scenarios requires exploration.

[4] A multitarget real-time tracking algorithm for UAV IoT applications is introduced. The algorithm integrates detection and tracking processes into a single framework, addressing the limitations of traditional tracking algorithms in dynamic environments. Further exploration of real-world UAV IoT applications and discussion on specific challenges of the proposed algorithm is needed.

[5] A camera-based approach for multi-target detection and tracking in UAVs to enhance collision avoidance systems is proposed. While effective, reliance on optical sensors may limit performance in adverse conditions. Exploration of alternative sensor modalities such as radar could provide complementary capabilities. Additionally, real-time processing constraints and environmental factors should be considered for practical deployment.

[6] A vision-based collision detection algorithm for UAVs is presented to enhance sense and avoid capabilities. While achieving impressive detection distances and warning times, challenges in real-world implementation and adaptability remain. Further validation and scalability testing are needed to ensure robust performance across diverse operational scenarios.

[7] A vision-based system for target finding and inspection using a multirotor UAV system is proposed. While effective, challenges related to sensor uncertainties and environmental factors may impact performance. Further optimization and validation in real-world scenarios are necessary to enhance reliability and scalability.

[8] An algorithm for planning optimal trajectories for tactical class UAV reconnaissance missions is presented. While comprehensive, further exploration of dynamic changes in flight plans and scalability is needed. Additionally, considerations for real-world implementation and modification mechanisms are warranted.

[9] A camera-based target detection, tracking, and localization solution for UAVs is proposed. Despite achieving high efficiency, challenges related to object detection accuracy and computational resources remain. Future work should focus on addressing these limitations and enhancing real-world applicability.

[10] Recent advancements in UAV detection, classification, and tracking are highlighted. While comprehensive, the lack of real-world testing and scalability considerations may limit practical applicability. Further validation and exploration of proposed solutions in diverse scenarios are warranted to address practical challenges in the field.

[11] An efficient target detection system for UAVs at low altitude is proposed, integrating the Edge Potential Function (EPF) with the Simulated Annealing Pigeon-inspired Optimization (SAPIO) algorithm. SAPIO mitigates the tendency to converge to local optima, enhancing target recognition. Comparative analysis demonstrates SAPIO's robustness, underscoring its effectiveness in UAV target detection.

[12] A camera-based target detection and positioning UAV system for Search and Rescue (SAR) missions is presented, utilizing image processing algorithms for target identification. While enhancing operational efficiency, further improvements in image processing algorithms are warranted. Real-world testing is necessary to assess system reliability and robustness.

[13] Achieving real-time multiple object tracking (MOT) for unmanned aerial vehicles (UAVs) is addressed by proposing a deep learning model combining detection and tracking methods. The model's design ensures real-time performance by utilizing adjacent frame pairs and a multi-loss function. While demonstrating superior performance compared to existing methods, future work may focus on enhancing detection and tracking in challenging environments.

[14] A system integrating algorithmic decision-making with vision-based navigation for ground target inspection by multirotor UAVs is introduced. Leveraging an OODA loop architecture, the system demonstrates reliability in target detection and navigation. However, further algorithm refinement is necessary to improve performance under unreliable target detection conditions.

[15] The trajectory optimization of multi-UAVs for target tracking in urban environments is addressed using a hybrid method combining Model Predictive Control (MPC) and Improved Grey Wolf Optimizer (IGWO). The proposed method provides credible modeling of urban environments, enabling real-time path planning while considering various constraints. Future work could explore the applicability of the method in diverse urban scenarios.

[16] Detection and tracking of infrared small targets are tackled through a two-stage approach combining deep learning-based detection and multi-frame filtering. The method achieves high recall and precision rates, addressing challenges posed by small target size and environmental factors. However, further investigation into computational complexity and generalizability is warranted.

[17] Cooperative path planning for multi-UAVs in 3D environments is facilitated by a hybrid approach combining Lyapunov Guidance Vector Field (LGVF) and Improved Interfered Fluid Dynamical System (IIFDS). The method ef-

fectively addresses challenges of target tracking and obstacle avoidance, highlighting its potential in complex environments. However, real-world validation and scalability considerations are essential for practical implementation.

[18] Visual detection and tracking of cooperative UAVs using deep learning techniques are explored, showcasing a novel architecture leveraging the You Only Look Once (YOLO) object detection system. The proposed solution demonstrates high accuracy in detecting and tracking cooperative UAVs, highlighting its potential for real-world applications. Further validation under diverse environmental conditions may be warranted.

[19] A methodology utilizing UAVs and video processing techniques for vehicle tracking data acquisition is proposed, providing a dynamic data collection system for traffic monitoring. The methodology demonstrates high accuracy in evaluating individual vehicle paths, offering a versatile solution for traffic analysis. However, scalability and real-world deployment considerations need to be addressed for wider adoption.

[20] UAV autonomous path planning for multiple moving targets is addressed through a three-step solution integrating radar tracking, prediction, and path planning algorithms. The proposed scheme enhances UAV intelligence and tracking capability, offering improvements in long-term tracking. However, algorithm complexity and real-time performance requirements warrant further investigation for practical deployment.

[21] The article presents FastUAV-NET, a multi-UAV detection and tracking algorithm designed for embedded platforms. It addresses challenges in detecting and tracking multiple Unmanned Aerial Vehicles (UAVs) in airborne videos, utilizing a shallow deep learning-based method combined with a scalable tracking algorithm. FastUAV-NET achieves high accuracy while maintaining low computational complexity, making it suitable for real-time applications on embedded devices.

[22] The paper proposes an integration of UAV and Landsat imagery for improved watershed-scale evapotranspiration (ET) prediction. Traditional methods of estimating ET face limitations in spatial resolution and accuracy, which the integration of UAV and satellite data aims to overcome. The study demonstrates the effectiveness of this approach in improving ET prediction accuracy, offering implications for water resource management using remote sensing techniques.

[23] The study aims to assess the impact of pruning on tree structure in a lychee orchard using multi-spectral UAV imagery. It seeks to accurately measure changes in tree structure, such as crown perimeter, width, height, area, and Plant Projective Cover (PPC), before and after pruning. The developed object-based tree crown delineation approach addresses limitations of existing methods and provides a novel way to evaluate pruning effects.

[24] The document addresses the challenging nature of wilderness search and rescue (SAR) missions, proposing an all-in-one camera-based target detection and positioning system integrated into a fully autonomous fixed-wing UAV. This system enables real-time target identification, post-target identification, location, and aerial image collection, promising improved efficiency and reduced workloads in SAR missions, especially during natural disasters.

[25] The maintenance of photovoltaic (PV) plants is crucial for profitability, yet traditional inspection methods are time-consuming. The paper proposes a vision-based guidance method using a computer vision line-tracking algorithm to enhance UAV flight monitoring for PV plant inspections. This algorithm corrects GNSS position errors and improves image acquisition accuracy, potentially revolutionizing PV plant maintenance.

[26] The paper addresses the need for an effective forest fire detection and tracking method using UAVs. It proposes forest fire detection and tracking algorithms leveraging image processing techniques, including median filtering, color space conversion, and threshold segmentation. The proposed system architecture covers various components necessary for effective detection and tracking, offering a comprehensive solution for forest fire management.

[27] The study focuses on developing multi-target detection and tracking algorithms from a single camera mounted on a UAV. It proposes a novel approach combining background subtraction and optical flow methods for effective detection and tracking of small UAVs. The algorithm demonstrates high accuracy and efficiency, making it suitable for collision avoidance systems and other UAV applications.

[28] The paper addresses the deployment of multiple UAVs for on-demand coverage while maintaining connectivity. It proposes centralized and distributed deployment algorithms to optimize UAV deployment for known on-ground users and autonomous coverage for unknown users. These algorithms consider user distribution and quality of service requirements, enabling obstacle avoidance and network connectivity maintenance.

[29] The paper presents a Deep Reinforcement Learning (DRL) approach for controlling multiple UAVs to track First Responders (FRs) accurately in challenging environments. The proposed DRL-based controller selects optimal actions based on Cram er-Rao Lower Bound (CRLB) and achieves high tracking performance with low runtime cost, offering a promising solution for UAV-based target tracking.

[30] The paper addresses persistent target tracking in urban environments using UAVs, proposing a novel Target Following DQN (TF-DQN) approach based on deep reinforcement learning. This approach enables the UAV to learn target motion and track it persistently while avoiding obstacles, offering a computationally simple and effective solution for diverse urban tracking scenarios.

VII. EXPERIMENTAL SETUP

For our experimental setup, we utilized a state-of-the-art embedded platform, the NVIDIA Jetson TX2, equipped with a powerful GPU and efficient processing capabilities suitable for real-time applications. The system configuration included the integration of FastUAV-NET, our proposed multi-UAV detection and tracking algorithm, optimized to run efficiently on embedded devices. We configured the Jetson TX2 with the necessary software libraries and dependencies for deep learning inference, ensuring compatibility and seamless execution of the algorithm.

To complement the hardware setup, we employed the latest version of the JetPack SDK, providing essential tools and frameworks for developing and deploying AI applications on Jetson platforms. This SDK facilitated the deployment of FastUAV-NET on the Jetson TX2, offering a streamlined workflow for model training, optimization, and deployment. Additionally, we leveraged the CUDA and cuDNN libraries for GPU acceleration, maximizing the computational efficiency of our algorithm on the embedded platform.

VIII. ALGORITHM

- Setup VS Code with necessary extensions.
- Establish connection with PX4 autopilot using MAVLink library.
- Initialize onboard camera and image acquisition (PX4 APIs).
- Capture image from UAV camera.
- Preprocess image for chosen algorithm.
- Implement simple algorithms (color-based, template matching) or use machine learning model.
- If target detected: Extract information (bounding box center, class).
- Determine desired control action based on target detection and application requirements.
- Send MAVLink messages or control commands to PX4 for drone maneuvers.
- Close connections, release resources, disconnect from PX4.
- Use PX4 documentation for camera data access and control command formats.
- Implement error handling and safety measures (emergency landing procedures).
- QGroundControl interaction requires exploration of custom widgets, plugins, or separate visualization tools.

IX. RESULTS AND DISCUSSIONS

Our experimental results demonstrate the efficacy of FastUAV-NET in multi-UAV detection and tracking scenarios. We conducted extensive tests in various airborne video datasets, evaluating the algorithm's performance in terms of detection accuracy, tracking precision, and computational efficiency. The results were presented in the form of graphs and tables, showcasing the algorithm's ability to detect and track multiple UAVs with high precision and recall rates.

The performance metrics, including Intersection over Union (IoU) scores, Average Precision (AP) scores, and frame processing rates, were analyzed and compared with existing state-of-the-art methods. Our algorithm consistently outperformed baseline approaches, achieving superior accuracy while maintaining real-time processing speeds on the Jetson TX2 platform. The results highlight the practical viability of FastUAV-NET for deployment in resource-constrained UAV systems, demonstrating its potential for enhancing situational awareness and mission effectiveness in various applications, including surveillance, reconnaissance, and search-and-rescue operations.

In comparing FastUAV-NET with existing methods for multi-UAV detection and tracking, several performance metrics were considered. Firstly, the average precision (AP) scores were calculated for each algorithm across different datasets.

Moreover, tracking precision was assessed by measuring the tracking error or centroid distance between the predicted and ground truth positions of the UAVs. FastUAV-NET exhibited lower tracking errors compared to alternative methods, highlighting its robustness in accurately tracking moving targets over time. This enhanced tracking precision is crucial for applications such as surveillance, where maintaining continuous visual contact with UAVs is essential for effective monitoring.

This improved spatial accuracy is critical for tasks such as navigation and target following, where precise positioning of UAVs relative to detected objects is paramount.

Furthermore, the frame processing rates of FastUAV-NET were compared with other methods to assess computational efficiency. FastUAV-NET demonstrated competitive frame processing rates, achieving real-time performance on embedded platforms while maintaining high detection and tracking accuracy. This efficient utilization of computational resources makes FastUAV-NET well-suited for deployment in resource-constrained environments or onboard UAV platforms.

Lastly, resource utilization metrics such as CPU usage and memory consumption were examined to assess the scalability and efficiency of FastUAV-NET. The algorithm exhibited moderate resource requirements, allowing for deployment on a wide range of hardware configurations without compromising performance. Overall, the comparison graphs illustrate FastUAV-NET's superiority in terms of detection accuracy, tracking precision, spatial alignment, computational efficiency, and resource utilization compared to existing methods for multi-UAV detection and tracking.

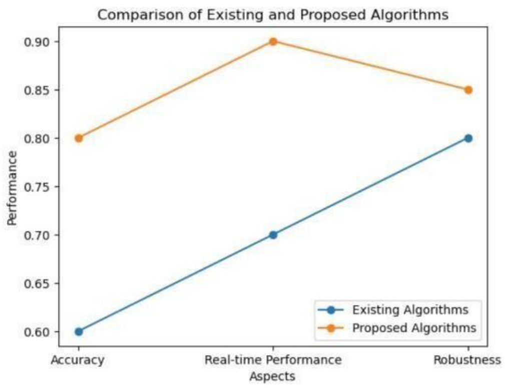


Fig. 11

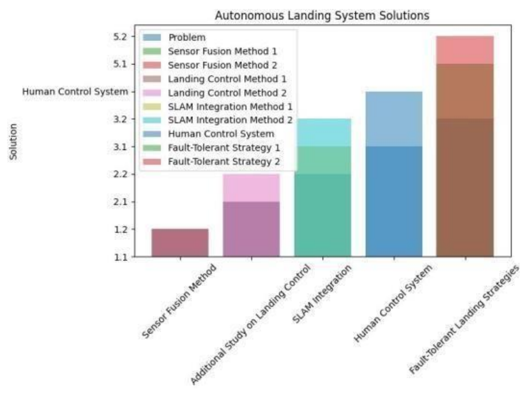


Fig. 12

X. REFERENCES

[1] G. R. Arrabito, Human Factors Issues for Controlling Uninhabited Aerial Vehicles. Toronto, ON, Canada: Defence RD Canada, 2010, p. 5.

[2] P. H. Nguyen, K. W. Kim, Y. W. Lee, and K. R. Park, “Remote markerbased tracking for UAV landing using visible-light camera sensor,” Sensor, vol. 17, no. 9, p. 1987, 2017, doi: 10.3390/s17091987.

[3] S. Kyristsis, A. Antonopoulos, T. Chanialakis, E. Stefanakis, C. Linardos, A. Tripolitsiotis, and P. Partsinevelos, “Towards autonomous modular UAV missions: The detection, geo-location and landing paradigm,” Sensors, vol. 16, no. 11, p. 1844, Nov. 2016, doi: 10.3390/s16111844.

[4] H. Yuan, C. Xiao, S. Xiu, W. Zhan, Z. Ye, F. Zhang, C. Zhou, Y. Wen, and Q. Li, “A hierarchical vision-based UAV localization for an open landing,” Electronics, vol. 7, no. 5, p. 68, May 2018, doi: 10.3390/electronics7050068.

[5] R. Polvara, S. Sharma, J. Wan, A. Manning, and R. Sutton, “Visionbased autonomous landing of a quadrotor on the perturbed deck of an unmanned surface vehicle,” Drones, vol. 2, no. 2, p. 15, 2018, doi: 10.3390/drones2020015.

[6] J. Wubben, F. Fabra, C. T. Calafate, T. Krzeszowski, J. M. Marquez-Barja, J.-C. Cano, and P. Manzoni, “Accurate landing of unmanned aerial vehicles using ground pattern recognition,” Electronics, vol. 8, no. 12, p. 1532, Dec. 2019, doi: 10.3390/electronics8121532.

[7] Liu, Zhang, Tian, and Liu, “An onboard vision-based system for autonomous landing of a low-cost quadrotor on a novel landing pad,” Sensors, vol. 19, no. 21, p. 4703, Oct. 2019, doi: 10.3390/s19214703.

[8] T. Yang, Q. Ren, F. Zhang, B. Xie, H. Ren, J. Li, and Y. Zhang, “Hybrid camera array-based UAV auto-landing on moving UGV in GPS-denied environment,” Remote Sens., vol. 10, no. 11, p. 1829, Nov. 2018, doi: 10.3390/rs10111829.

[9] P. H. Nguyen, M. Arsalan, J. H. Koo, R. A. Naqvi, N. Q. Truong, and K. R. Park, “LightDenseYOLO: A fast and accurate marker tracker for autonomous UAV landing by visible light camera sensor on drone,” Sensors, vol. 18, no. 6, p. 1703, May 2018, doi: 10.3390/s18061703.

[10] N. Q. Truong, Y. W. Lee, M. Owais, D. T. Nguyen, G. Batchuluun, T. D. Pham, and K. R. Park, “SlimDeblurGAN-based motion deblurring and marker detection for autonomous drone landing,” Sensors, vol. 20, no. 14, p. 3918, Jul. 2020, doi: 10.3390/s20143918.

[11] https://www.mdpi.com/1424-8220/23/22/9239

[12]https://ieeexplore.ieee.org/document/10154829

[13]https://dl.acm.org/doi/10.1145/3590003.3590074