DTU Electro
Department of Electro and Photonics Engineering

Semantic Mapping and Sensor Fusion for UAV Navigation

Master Thesis

Semantic Mapping and Sensor Fusion for UAV Navigation

Master Thesis
January, 2024

Student: Haoran Wei
Supervisor: Roberto Galeazzi, Peter Iwer Hoedt Karstensen

Cover photo: Vibeke Hempler, 2012
Published by: DTU, Department of Electro and Photonics Engineering, Ørsteds Plads, Bygning 348, 2800 Kgs. Lyngby Denmark
www.electro.dtu.dk

Approval

This thesis has been prepared over six months at the Section for Robotics and Automation, within the Department of Electrical and Photonics Engineering, at the Technical University of Denmark, DTU, in partial fulfilment of the requirements for the degree of Master of Science in Electrical Engineering, the Automation and Robot Technology study line.

It is assumed that the reader has a basic understanding of statistics, linear algebra, kinematics and computer vision.

Haoran Wei - s212656

Signature

14/01/2024

Date

Abstract

Localization is crucial for Unmanned Aerial Vehicles (UAVs) when conducting Simultaneous Localization and Mapping (SLAM) with a pre-existing map. In this context, the use of RGB-D cameras emerges as a potent technique, offering significant advantages in addressing the complexities of slam challenges. This research introduces a framework that exploits the detailed information available in RGB imagery for 2D instance segmentation. By integrating this segmentation with point cloud data derived from depth information, we can effectively isolate and analyze key segment points. This enables segment-based mapping and localization that utilizes both color and depth information. Our methodology demonstrates that leveraging RGB-D data in localization tasks within a prior semantic map not only accelerates the real-time processing of point clouds and local maps but also enhances the accuracy of landmark matching through improved descriptors.

Acknowledgements

I would like to extend my heartfelt gratitude to my project supervisor, Prof. Roberto Galeazzi. His profound insights and guidance have opened my eyes to the fascinating world of Unmanned Aerial Vehicles (UAVs) and the intricacies of project management. His patience and constant encouragement have been instrumental in fueling my continuous learning journey in this field, molding me into a more skilled engineer. His flexible approach to project development, coupled with a stringent attitude towards process management, has significantly enhanced my understanding of effective project execution.

I am also immensely grateful to Ph.D. Peter Iwer Hoedt Karstensen for his consistent guidance and invaluable assistance with the experimental aspects of my thesis. His anecdotes and positive energy, especially during our regular weekly meetings, have been a source of inspiration. Moreover, his anecdotes about life in Finland, from the massive snowfalls to the long winter nights, brought much-needed levity and energy to my tight project schedule, helping to alleviate stress.

My sincere thanks also go to the many individuals involved in the broader aspects of my project, whose lively and fruitful discussions have contributed greatly to my learning experience. Their advice and insights, though they were not officially part of my thesis, have been incredibly valuable.

My journey would have been significantly less joyful without the support of my friends and my girlfriend, whose companionship has been a pillar of strength. The laughter, talks, and shared experiences have not only kept me grounded but also provided much-needed moments of levity and relaxation amidst the rigors of academic life.

Lastly, I owe an immeasurable debt of gratitude to my family in China. Their relentless support and love over the past two years have been the bedrock of my ability to embark on this journey at DTU and start this remarkable chapter of my life. Their sacrifices have not gone unnoticed and have been the wind beneath my wings, allowing me to soar to new heights.

Acronyms

AI Artificial Intelligence. 1

ASTA Autonomous Systems Test Arena. 10, 11

CNN Convolutional Neural Network. vi, 8, 10, 11, 22

DTU Technical University of Denmark. 10

EKF Extended Kalman Filter. vi, 2, 8, 10, 29

GNSS Global Navigation Satellite System. 1, 11

HPC High Performance Computing. 3, 20

IMU Inertial Measurement Unit. vi, 1, 3, 8, 11

LiDAR Light Detection and Ranging. 1, 2

ML Machine Learning. 8

RF Radio Frequency. 1

RLE Run-Length Encoding. 17

SLAM Simultaneous Localization and Mapping. iii

UAV Unmanned Aerial Vehicle. 1, 2, 4

UAVs Unmanned Aerial Vehicles. iii, 1, 2, 10, 24

Contents

1 Introduction

1.1 Background and Motivation

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have seen a meteoric rise in various sectors, revolutionizing practices in fields ranging from agriculture[1], disaster management[2], surveillance[3], to logistics[4]. The flexibility, reduced operational costs, and unique aerial capabilities of UAVs have made them indispensable tools in modern technology landscapes. Their applications extend from precision agriculture, where they are used for crop monitoring and spraying, to search and rescue missions in disaster-struck areas, offering rapid, real-time information and accessibility to otherwise unreachable zones[1, 2].

A critical aspect of UAV is their ability to navigate accurately in diverse environments. Traditionally, UAVs navigation has relied on Inertial Measurement Unit (IMU), Global Navigation Satellite System (GNSS)[5], and other Radio Frequency (RF)[6, 7] technologies. However, these systems often face limitations in signal-compromised environments, such as indoors or dense urban areas. Urban environments, with their high-rise buildings, create 'urban canyons' that significantly disrupt GNSS signals. Similarly, indoor environments, dense forests, and subterranean settings like tunnels and caves present environments where GNSS signals are weak or non-existent. The reliance on these traditional navigation systems thus poses a significant limitation to the operational capacity of UAVs in these scenarios. To overcome these challenges, the integration of alternative technologies, including RGB-D cameras, optical sensors, and Light Detection and Ranging (LiDAR), has become increasingly prevalent[8, 9]. RGB-D cameras provide rich visual and depth data, allowing for enhanced spatial awareness and object recognition capabilities. LiDAR technology, known for its precision in distance measurement and its ability to operate in a variety of lighting conditions, offers an additional layer of environmental understanding[10]. LiDAR lack color information and are often more expensive than RGB-D systems. These limitations position RGB-D cameras as a viable alternative, particularly in small-scale environments like indoor environment[11].

The fusion of sensors has yielded promising results, particularly in vision-based methods employing cost-effective and versatile visual sensors, which have demonstrated substantial advantages in UAV navigation[12]. Extensive research focusing on visual localization[13], obstacle avoidance[14], and path planning[15] has further underlined the critical role of visual navigation in this field. The research[16] propose a system to estimate UAV position by fusing data from IMU and RGB-D sensor, which improve the reliability of UAV system. Another approach for autonomous UAV navigation using RGB-D data in GNSS-denied environments is presented in[17], focusing on 3D marker recognition for precise pose estimation and emphasizing algorithms optimized for real-time application with minimal computational burden.

The integration of Artificial Intelligence (AI) plays a crucial role in enhancing UAV navigation capabilities. The growing significance of AI in improving both efficiency and effectiveness of UAV is highlighted in [18]. AI enhances UAV navigation tasks by enabling more autonomous and precise operations. For instance, AI algorithms can analyze environmental data to optimize flight paths, reduce energy consumption, and ensure safety in complex scenarios. The use of AI also allows UAVs to perform intricate tasks such as

search and rescue operations[19] and agricultural perceptiopn[20] with greater accuracy and efficiency.

An advanced application of AI in UAV is semantic mapping. It leverages the full potential of RGB-D data by utilizing both color and depth information. The development of 3D semantic maps for indoor settings using RGB-D data is explored in [21], presenting methods for integrating rich semantic details into indoor small-scale maps to enhance robot navigation and interaction in complex environments. Additionally, [22] introduces a novel deep learning approach for 3D object detection, the Frustum PointNets framework, which effectively combines 2D object detection with advanced 3D deep learning techniques for object localization in large-scale scenes.

To further utilize AI in UAV navigation task, segment-based localization in a global map using 3D point clouds is an area of focus. The study in [23] delves into the robustness of this approach against environmental changes and variations in illumination. It presents a solution to localization and mapping challenges by extracting segments from 3D point clouds. The method described in [10] for 3D segment mapping using data-driven descriptors proves efficient in object localization in point clouds and precise 3D bounding box estimation, surpassing existing methods in standard benchmarks. The integration of semantic information from RGB imagery into maps generated from LiDAR systems is further investigated in [24], emphasizing the importance of semantic understanding in robotics.

1.2 Problem Statement

The primary challenge addressed in this thesis is the limitation of traditional navigation systems in UAV operations within signal-compromised or temperaly denied environments like gnss. When lacking of those sensors' aided, drift and noise from IMU will accumulate and impede the effective deployment of UAVs in critical applications where precision and reliability are paramount. In this case, RGB-D aided system where IMU we assume will always work can deal with such scenario. Fusion of visual data from rgb-d and IMU data can significantly imporve the robustness of UAV navigatio system.

1.3 Project Scope and Objectives

The scope of this project encompasses the development of advanced navigation systems for UAVs, it includes preparing dataset of landmarks, developing 2 deep learnig for 2D instance segmentation and 3D point cloud featrue extraction, mapping 3D semantic map, pose estimation in a prior map by leveraging the integration of RGB-D cameras and IMU The objectives are:

- To prepare a dataset with annotated landmarks.
- To train deep learning models for 2D instance segmentation and descriptor extraction of 3D segment.
- To build a global 3D semantic map of the test environment.
- to develop matching and localization algorithms between local and global map.
- To develop an Extended Kalman Filter (EKF) to fuse RGB-D sensor and IMU sensor for improved navigation

1.4 Methods and Tools

This section provides a summary of the comprehensive approach to UAV navigation focusing on segment-based mapping and localization in GNSS-denied environments. The

methodology encompasses various stages, from data acquisition to real-time pose estimation, leveraging advanced computer vision and sensor fusion techniques.

1.4.1 Data Collection and Preparation

- Environment Setup: Customized dataset in an indoor environment with specific categories.
- Ground Truth Acquisition: Utilizes a 3D motion capture system.
- IMU and RGB-D Data Collection: Drone Holybro X500 V2 equipped with an IMU and an Intel RealSense Depth Camera D455.
- Image Annotation: Manual annotation of RGB images using COCO Annotator[25].

1.4.2 2D Instance Segmentation

- Model: Mask R-CNN[26] with the mmdetection toolbox[27].
- Backbone: ResNet-50 with Feature Pyramid Network (FPN)[26].
- Training: Implemented on High Performance Computing (HPC) facilities.

1.4.3 Segment Generation

- Process: Combining 2D segmented data with depth information.
- Transformation: Conversion from camera to world frame using transformation matrices.

1.4.4 Descriptor Extraction

- CNN Model: Adaptation of the SegMap's CNN model[28] in TensorFlow 2.x.
- Preprocessing: Filtering and downsampling using Open3D.
- HPC Training: Training on DTU's HPC facilities.

1.4.5 Global and Local Map Generation

- Global Map: Created from point clouds in the world frame.
- Local Map: Generated in real-time using IMU data for pose estimation.

1.4.6 Real-Time Pose Estimation with IMU

- Integration: Fusion of IMU data for real-time pose estimation.
- Algorithm: Use of an for sensor fusion.

1.4.7 Matching Strategy in a Prior Map

- Approach: Matching local map segments with the global map.

1.4.8 Segment-Based Pose Estimation

- EKF: Refinement of pose estimation by fusing visual data with IMU data.

1.4.9 Evaluation and Analysis

- Performance Analysis: Evaluates the accuracy of sensor fusion algorithms.
- Challenges and Limitations: Addresses difficulties in segment-based localization and sensor fusion.

1.4.10 Tools and Software

- Software: Includes COCO Annotator, mmdetection, Open3D, TensorFlow, and ROS.
- HPC Resources: Utilization of DTU's High Performance Computing facilities.

1.4.11 Real-World data Implications

- **Application Scope:** Potential in autonomous UAV navigation in indoor or GPS-denied environments.

1.5 Novelty and contributions

The key contributions of this thesis include:

- The development of 2 datasets for landmarks including images with COCO format[29] labels and 3D segments with global positions.
- The integration of RGB-D based semantic mapping.
- The successful implementation of segment-based matching and localization, demonstrating its effectiveness in practical scenarios.
- Sensor fusion of RGB-D and IMU for enhancing UAV navigation.

2 Technical Concepts

2.1 Control and Communication of UAV

PX4 is an open-source flight control software for drones and other unmanned vehicles. It's highly configurable and supports a wide range of aircraft types. In the context of IMU data, PX4 plays a crucial role as it interfaces directly with the drone's hardware sensors to gather data such as acceleration, gyro and their bias.

QGroundControl is a ground control station software that provides full flight control and mission planning for any MAVLink-enabled drone. It offers an intuitive user interface to interact with PX4. When it comes to IMU data, QGroundControl can be used to visualize this data in real-time or to adjust settings and calibrations related to the IMU sensors on the drone.

MAVLink (Micro Air Vehicle Link) is a lightweight communication protocol between drones and a ground control station. It plays a pivotal role in the pipeline by acting as the communication bridge. MAVLink facilitates the transfer of IMU data from the PX4 flight controller to QGroundControl in real-time. It ensures that the data packets are small and efficient, crucial for maintaining robust and responsive communication, especially in dynamic flying environments.

2.2 2D Instance Segmentation

2D instance segmentation is a computer vision task that involves identifying and delineating each distinct object of interest appearing in an image. Unlike simple object detection that just locates and classifies objects (often with bounding boxes), instance segmentation goes further by precisely outlining the shape of each object. This is done at the pixel level, making it a more detailed and complex task than basic detection or classification.

2.3 High Performance Computing

HPC refers to the use of supercomputers and parallel processing techniques for running advanced applications efficiently, reliably, and quickly. This typically involves harnessing the power of multiple computers to perform complex calculations.

For the perspective of this thesis, HPC involves the use of a single GPU in a HPC cluster at the DTU, with each core and its RAM customised, along with settings for the runtime wall and the CUDA used for Tensorflow.

2.4 Mask R-CNN

Mask R-CNN (Region-based Convolutional Neural Network) is a state-of-the-art model for instance segmentation that extends the capabilities of Faster R-CNN, a model known for its efficiency in object detection.

This thesis employed the Mask R-CNN framework, using the mmdetection[27] toolbox, to conduct instance segmentation on a customized dataset. The configuration of this model is meticulously tailored to suit the specific requirements of my research.

The model utilizes the ResNet-50 architecture as its backbone combined with a Feature Pyramid Network (FPN). This setup allows for effective feature extraction at multiple scales, which is crucial for accurately detecting and segmenting objects of varying sizes.

The ResNet-50 backbone, known for its depth and robustness, helps in capturing complex features, while the FPN enhances the model's ability to handle objects at different scales, thereby improving overall segmentation accuracy.

2.4.1 Model Evaluation

Mask Loss: This metric measures how well the model is predicting the segmentation masks as compared to the ground truth during training. Lower mask loss indicates better performance in terms of the model's segmentation capabilities. From the plot, it's evident that the mask loss decreases over epochs, which suggests that the model is learning and improving its ability to segment the objects accurately.

Accuracy: This represents the model's ability to correctly classify the objects it detects. It's a crucial metric for understanding the classification performance of the model. The accuracy plot shows fluctuations, which is common in training as the model learns from different batches of data. However, the overall trend should ideally be upwards. The sharp dip seen around epoch 6 could be due to several reasons such as a bad batch of data, a need for regularization, or it could just be a random fluctuation.

Learning Rate: The learning rate dictates the size of the steps the model takes during optimization. A higher learning rate allows the model to learn faster, but it can overshoot the minimum loss. A lower learning rate ensures more precise convergence but can slow down the training. The plot shows a learning rate schedule where the learning rate increases initially and then decreases, which is a strategy to help the model converge faster while allowing fine adjustments as it gets closer to the minimum.

The mean Average Precision (mAP) is a common metric used to evaluate the performance of models in object detection tasks, including both bounding box prediction and instance segmentation. The ”bbox” refers to bounding box detection metrics, and ”segm” refers to segmentation metrics. Each of these tasks has different challenges and the mAP helps to quantify how well a model performs on each.

Bounding Box mAP (bbox mAP): This metric measures the accuracy of the predicted bounding boxes against the ground truth boxes. The average precision is calculated for each class and then averaged over all classes. The bbox mAP is particularly important for tasks where the location of the object is as important as the correct classification.

Segmentation mAP (segm mAP): This metric is used for instance segmentation tasks, where the model predicts not only the bounding box but also the pixel-wise mask of each object. The segm mAP thus measures the accuracy of the overlap between the predicted mask and the ground truth mask for each object instance.

mAP at IoU=0.50 (mAP @0.50): IoU, or Intersection over Union, is a measure of the overlap between the predicted box/mask and the ground truth. An IoU of 0.50 means that the overlap must be at least 50% for a prediction to be considered correct. This threshold is more lenient and is generally easier for models to achieve.

mAP at IoU=0.75 (mAP @0.75): A more stringent measure of performance is the mAP at an IoU of 0.75, which requires the predicted box/mask to overlap the ground truth by at least 75%. Achieving high scores at this threshold indicates a more precise model, as it must produce predictions that are very close to the ground truth.

2.5 Coordinates Frames

Transformation between different frames, such as the camera frame and the world frame, is a fundamental concept in robotics, computer vision, and related fields. This process

involves translating and rotating points from one coordinate system to another.

1. **Camera Frame (C)**:
    - Origin: Located at the camera's optical center.
    - Axes: Defined based on the camera's orientation. Typically, X_C and Y_C are in the image plane, and Z_C is perpendicular to this plane.
2. **IMU Frame (I)**:
    - Origin: Located at the center of the drone.
    - Axes: X_I point forward, Y_I to the right, and Z_I downward relative to the drone's usual flight orientation.
3. **Drone Frame (D)**:
    - Origin: Located at the center of rigid body created in motion capture system.
4. **World Frame (W)**:
    - A fixed, global coordinate system was defined by motion capture system and used as a reference frame.
    - Provide precise position and orientation of drone frame.

2.5.1 Transformation Matrices

Transformation from the camera frame to the drone frame, and then to the world frame, involves two steps:

1. **Camera to Drone (CD)**:
    - Rotation Matrix (R_{CD}) and Translation Vector ($\vec{t}_{CD}$).
    - Transformation Matrix $T_{CD} = \begin{pmatrix} R_{CD} & \vec{t}_{CD} \\ 0 & 1 \end{pmatrix}$.
2. **Drone to World (DW)**:
    - Rotation Matrix (R_{DW}) and Translation Vector ($\vec{t}_{DW}$).
    - Transformation Matrix $T_{DW} = \begin{pmatrix} R_{DW} & \vec{t}_{DW} \\ 0 & 1 \end{pmatrix}$.

Transformation from the IMU frame to the drone frame, and then to the world frame, involves two steps:

1. **IMU to Drone (ID)**:
    - Since the IMU Frame is parallel to the Camera Frame, the transformation matrix is often similar or identical to the Camera to Drone transformation.
    - Rotation Matrix (R_{ID}) and Translation Vector ($\vec{t}_{ID}$).
    - Transformation Matrix $T_{ID} = \begin{pmatrix} R_{ID} & \vec{t}_{ID} \\ 0 & 1 \end{pmatrix}$.
2. **Drone to World (DW)**:
    - Rotation Matrix (R_{DW}) and Translation Vector ($\vec{t}_{DW}$).
    - Transformation Matrix $T_{DW} = \begin{pmatrix} R_{DW} & \vec{t}_{DW} \\ 0 & 1 \end{pmatrix}$.

To find the transformation from the camera frame to the world frame, we multiply the two transformation matrices:

$$T_{CW} = T_{DW} \times T_{CD}$$

2.5.2 Homogeneous Coordinates

For a point P_C in the camera frame, its transformation to the world frame P_W is given by:

$$P_W = T_{CW} \times P_C$$

For a point P_I in the IMU frame, its transformation to the world frame P_W is given by:

$$P_W = T_{IW} \times P_I$$

2.6 Convolutional Neural Network

Overview of CNN CNN have revolutionized the field of computer vision and spatial data analysis. Their ability to automatically and adaptively learn spatial hierarchies of features from input images or three-dimensional data makes them exceptionally suitable for tasks like image and video recognition, image classification, and medical image analysis. The core idea behind CNNs involves the use of convolutional layers, which apply convolution operations to the input, passing the result to the next layer. This process results in the network's ability to learn increasingly complex features as we go deeper into the network.

TensorFlow Framework TensorFlow, developed by the Google Brain team, is a cornerstone in the field of machine learning and deep learning. It offers a comprehensive, flexible ecosystem of tools, libraries, and community resources that enables researchers to push the state-of-the-art in Machine Learning (ML), and developers to easily build and deploy ML-powered applications.

2.7 Inertial Measurement Unit

Drift Error Also known as bias drift, is a type of error in IMU sensors that results in a gradual deviation from the true measurement over time. This error is particularly prominent in gyroscopes, where it manifests as a slow change in the angular velocity reading, even when the UAV is stationary. Drift error can be caused by various factors, including changes in temperature, sensor aging, and manufacturing inconsistencies. In the context of pose estimation, drift error accumulates over time, leading to increasing inaccuracies in orientation estimates, which subsequently affect the calculated position.

Noise Error In IMU sensors, it is typically characterized by random fluctuations in the sensor readings, which arise from a variety of sources such as electronic noise, temperature variations, and sensor resolution limits. Unlike shift error, noise error is non-systematic and varies randomly over time. This type of error primarily impacts the precision of the measurements. Noise in accelerometer and gyroscope data can lead to inaccuracies in derived quantities like velocity and position, especially when integrating these measurements over time.

2.8 Extended Kalman Filter

This thesis present the formulation of an Extended Kalman Filter (EKF) designed for state estimation in drone applications. The EKF is a nonlinear version of the classic Kalman Filter that linearizes about an estimate of the current mean and covariance. In our approach, the state vector is represented as $[x, y, z, \text{rx}, \text{ry}, \text{rz}, \text{vx}, \text{vy}, \text{vz}]$, where x, y, z are the position coordinates, $\text{rx}, \text{ry}, \text{rz}$ could be orientation in Euler angles or a rotation vector format, and $\text{vx}, \text{vy}, \text{vz}$ are the velocity components.

2.8.1 Initialization

The filter is initialized with the state covariance matrix P, process noise covariance matrix Q, and measurement noise covariance matrix R. The matrices Q and R are initialized based on estimated variances for position, orientation, and velocity. The state vector is initialized as a zero vector of length 9, representing an initial guess at the true state.

2.8.2 Prediction Step

The prediction step involves updating the state vector using Inertial Measurement Unit (IMU) data, which includes acceleration and gyroscope readings. A bias correction is applied to both accelerometer and gyroscope data. The gravity vector is calculated and subtracted from the accelerometer readings to isolate the drone's acceleration. The drone's orientation is updated by integrating the gyroscope data. The rotation matrix from the IMU to the drone frame and from the drone frame to the world frame are used to transform the IMU readings to the world frame. The velocity and position states are updated by integrating the acceleration.

The state covariance matrix P is also updated in this step using the state transition matrix F, which is derived from the Jacobian of the state transition with respect to the state vector.

2.8.3 Update Step

In the update step, the filter incorporates visual measurements to refine the state estimates. The measurement update corrects the state estimate using a measurement residual, the Kalman gain, and the measurement matrix H. The state covariance matrix P is also updated to reflect the new estimate's uncertainty.

2.8.4 Mathematical Formulation

The state prediction and update equations in the EKF framework are given as follows:

Prediction:

$$\begin{aligned}\hat{x}_{k|k-1} &= f(\hat{x}_{k-1|k-1}, u_k)\\ P_{k|k-1} &= F_k P_{k-1|k-1} F_k^T + Q_k\end{aligned}$$

Update:

$$\begin{aligned}y_k &= z_k - H_k \hat{x}_{k|k-1}\\ S_k &= H_k P_{k|k-1} H_k^T + R_k\\ K_k &= P_{k|k-1} H_k^T S_k^{-1}\\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k y_k\\ P_{k|k} &= (I - K_k H_k) P_{k|k-1}\end{aligned}$$

Here, f represents the state transition function, u_k is the control input (IMU data), F_k is the state transition matrix, Q_k is the process noise covariance, z_k is the measurement, H_k is the measurement matrix, R_k is the measurement noise covariance, and K_k is the Kalman gain.

3 Approach

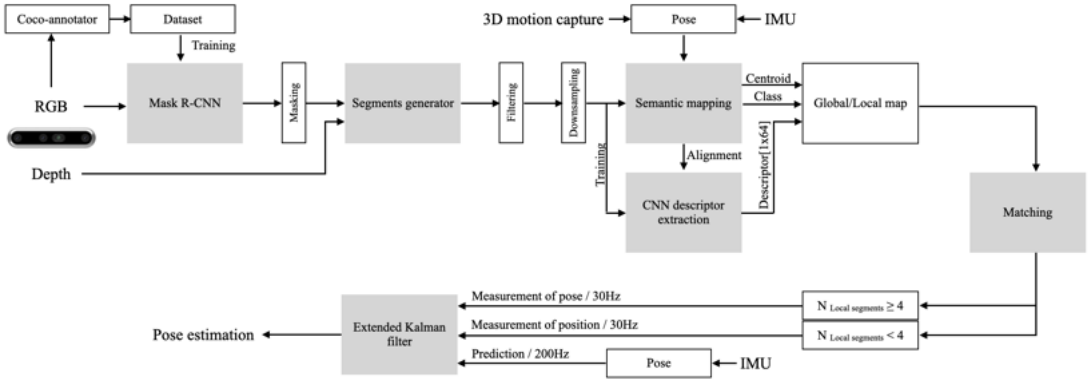

Figure 3.1: **Segment-Based Mapping and Localization Pipeline Overview.** This pipeline starts with 2D segmentation using COCO Annotator[25] for training a Mask R-CNN model. Segments from point clouds, merged with depth and instance masks, are positioned using motion capture or IMU data. These segments train a CNN to produce segment descriptors, which then are used to build global and local maps. Matching between maps at 30Hz aids in pose estimation, supplementing the main pose prediction from IMU data at 200Hz in an EKF. The process enables real-time pose estimation and trajectory plotting.

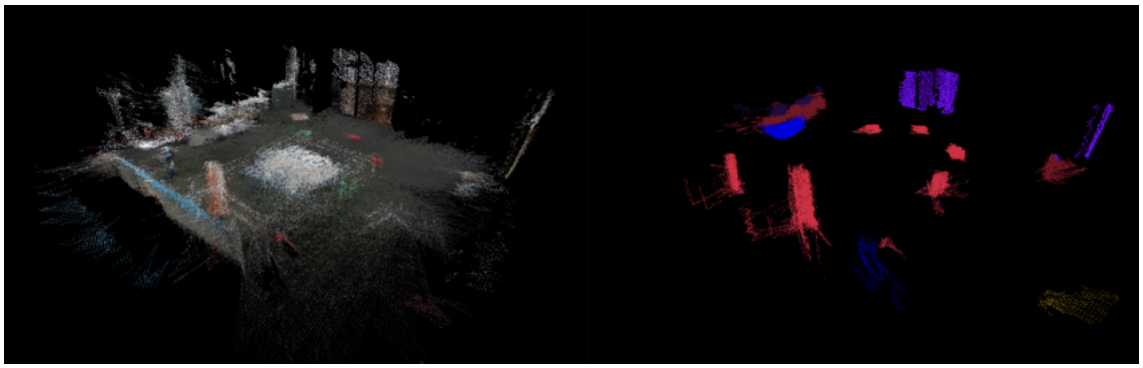

Figure 3.2: **Semantic Mapping from RGB-D Data.** This figure presents a semantic map covering an area of 10m x 10m, derived from data collected in Autonomous Systems Test Arena (ASTA) at Technical University of Denmark (DTU), a large indoor infrastructure. The left image is created from the original RGB-D point cloud data, combined with global positioning obtained from the motion capture system in ASTA. The right image is the result of augmenting semantic segments extracted using the methodology outlined in Fig.3.9. Additionally, noise reduction techniques have been applied to enhance the clarity and accuracy of the semantic map.

3.1 Environment of Data Caption

Datasets are crucial for training and testing deep learning models, significantly impacting the performance of UAVs navigation systems. This thesis focuses on the importance of

image datasets for training 2D instance segmentation models and the role of segments or point clouds for training 3D CNN descriptor extraction models. These are vital for achieving accurate matching in a pre-existing map. Additionally, IMU data and accurate position ground truth are essential for evaluating the model's performance in localization tasks, especially when GNSS signals are unavailable and IMU data might introduce trajectory shift errors.

A notable challenge is that most existing 3D datasets are associated with LiDAR sensors, which have characteristics differing from 3D point clouds generated by RGB-D sensors. This disparity can affect final testing outcomes. Furthermore, most RGB-D datasets are based on indoor scenarios, which are not directly applicable to UAVs that require testing in medium or large-scale outdoor environments.

Given these challenges, creating a customized dataset for this thesis is both necessary and beneficial. Before data collection, it is essential to define the categories of objects to be included. Utilizing the resources available at ASTA test facility, seven categories have been identified: Box, Building, Computer, Table, Chair, People, and Other. The 'People' category is specifically designed to detect dynamic elements, which are not considered valid instances for localization and are filtered out during segment generation. The instances from these categories are distributed over an area measuring 10m x 10m, as illustrated in Fig. 3.3.

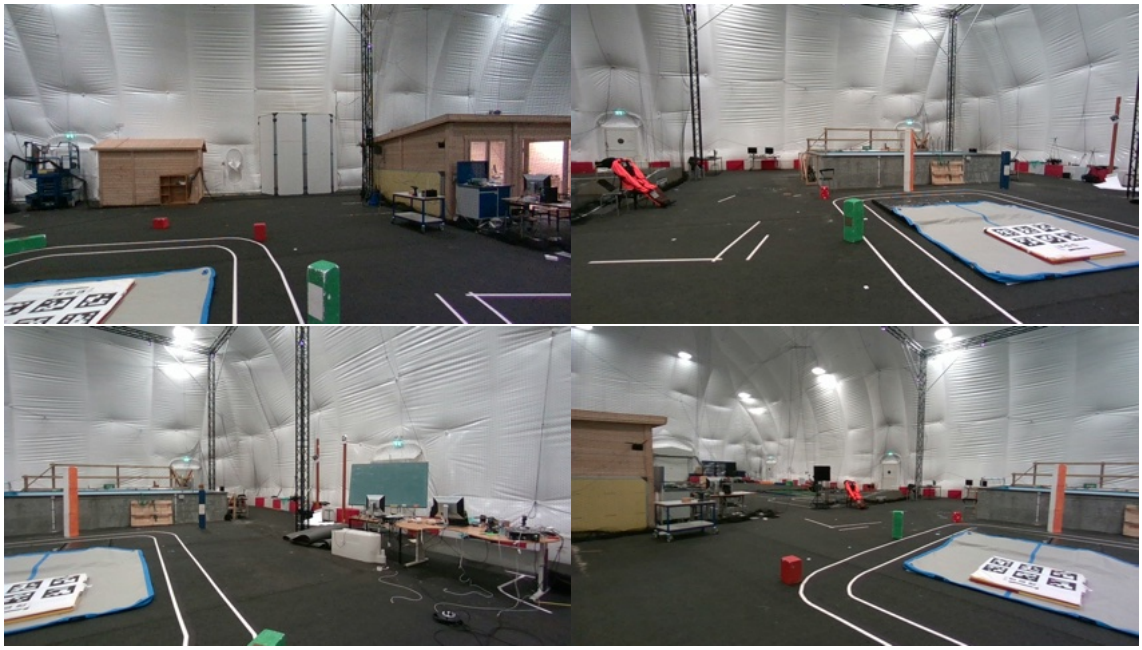

Figure 3.3: Setting of the Environment.

3.2 Methods and Tools Used for Data Capture

For data collection, the drone Holybro X500 V2, equipped with the Holybro Pixhawk 4 Autopilot Flight Controller, LattePanda 3 Delta, and Intel RealSense Depth Camera D455, is utilized.

3.2.1 Ground Truth

The 3D motion capture system at ASTA, covering a 16m x 12m area with a camera height of 8m, provides the drone's pose in the world frame (Fig. A.3). This system, featuring 16

Prime 17W cameras, boasts an accuracy of <5mm and repeatability of <1mm. To enable system detection of the drone, five reflective markers are attached, avoiding symmetrical placements. The drone, adorned with markers (Fig. A.4), is maneuvered in the space, allowing marker detection and selection as a rigid body in the Motive software. The data streaming pipeline from the motion capture system is pre-configured. Users must connect to the 'asta_optitrack_5G' local network and run the ROS client to initiate data streaming.

To remotely operate the Ubuntu 20.04 system running on the microcomputer Lattepanda aboard the drone, an OpenSSH server is employed. ROS Noetic is used to launch the ROS client in the VRPN workspace. The 'rosbag' command records topics containing information about the drone's pose and timestamps in the world frame at a rate of 100Hz. For clarity, this thesis will refer to this as the 'drone frame' to distinguish it from the IMU frame and RGB-D camera frame.

3.2.2 Image Query

RGB-D data is captured using the Python wrapper for the Intel RealSense camera, the 'pyrealsense2' package[30]. Camera parameters are saved as a '.json' file. RGB and depth images are captured at a resolution of 640x480, with a frequency of 5Hz for training and mapping data, and 30Hz for real-world testing and evaluation. All imagery is stored on the microcomputer onboard the drone.

3.2.3 IMU Data

MAVLink is employed to establish communication between the drone and QGroundControl on the microcomputer. Flight logs, reflecting PX4 firmware 1.3 behavior, including IMU messages, are saved in '.ulg' files. For efficient data processing, these IMU data are converted into CSV files using the Python package 'pyulog'[31].

3.3 Annotation and Split of Dataset

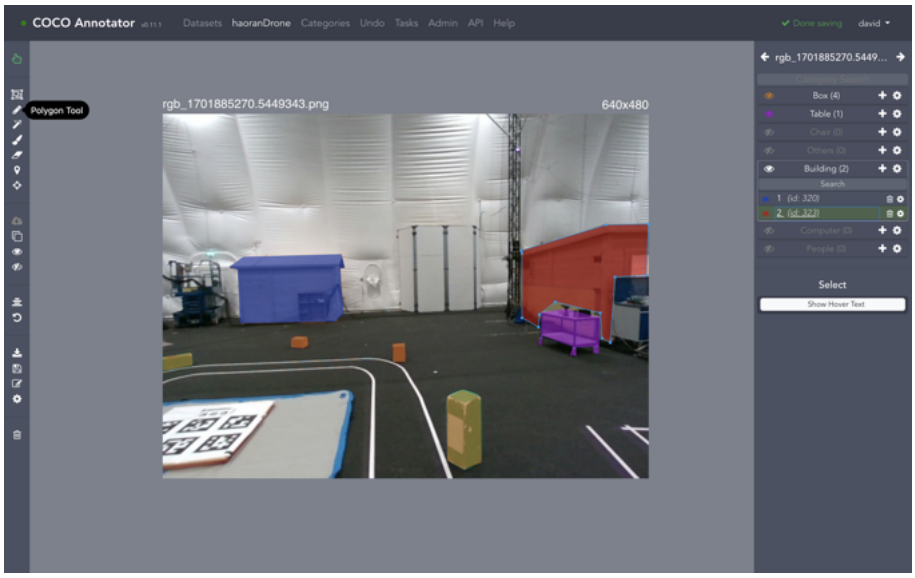

Figure 3.4: Illustration of Instance Segmentation Labeling.

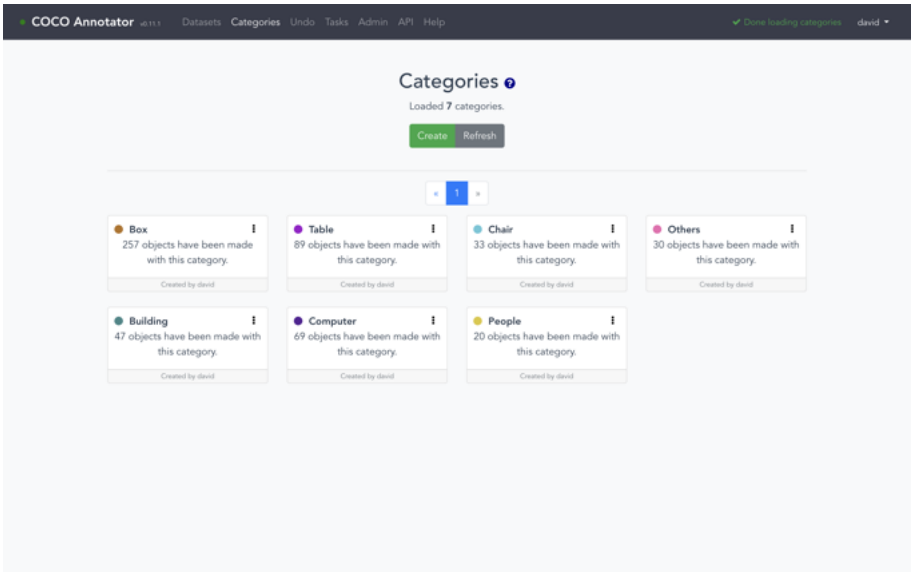

Figure 3.5: Distribution of Categories in the Dataset.

The preparation of a dataset is a key step in training a model to achieve optimal performance. For this project, a customized dataset comprising seven categories was developed. Initially, the dataset included raw images without segmentation labels. To facilitate customized annotation and meet training objectives, the COCO Annotator tool was employed for manual segmentation of raw RGB images in the COCO format. Figure 3.4 illustrates the process of creating polygons for segmentation. These polygons were drawn around objects falling within the seven predefined categories, allowing for the generation of annotations in the COCO format.

Subsequently, the dataset was enriched with these annotations, resulting in a total of 128 labeled images encompassing 545 labeled objects, as depicted in Figure 3.5. This annotated dataset forms the foundation for training the deep learning models for instance segmentation, ensuring that the models are tailored to the specific requirements of the project.

3.4 2D Instance Segmentation

This section introduces the 2D instance segmentation method, including model setup, GPU setup, and model evaluation.

3.4.1 Mask R-CNN Settings

The training configuration adopts a 1x schedule, a standard yet effective training strategy in mmdetection, which balances training time and model performance. The optimizer used is SGD (Stochastic Gradient Descent) with a learning rate of 0.02 and a weight decay of 0.0005, parameters chosen to optimize the learning process with small dataset. The model initializes with pre-trained weights from a Mask R-CNN model trained on the COCO dataset, which provides a solid foundation for learning the specific features of your custom dataset. It's notable that the structure of the pre-trained model is different from mine, but it still is meaningful to load only the same parts.

My dataset, annotated in the COCO format, includes a variety of classes like 'Box', 'Table', 'Chair', 'Others', 'Building', 'Computer', and 'People'. This diverse range of classes indicates a comprehensive approach to object detection and segmentation, allowing the model to learn and differentiate between all defined object types.

For the Region Proposal Network (RPN) and Region of Interest (ROI) heads, the default configuration have configured specific parameters such as anchor generation ratios and scales, bbox coding, and ROI extraction strategies. These settings are crucial for determining how the model proposes candidate object regions and how it processes these regions for accurate classification and segmentation.

The model employs predefined loss functions for bounding box prediction (L1Loss) and mask prediction (CrossEntropyLoss), ensuring precise segmentation of objects. Additionally, the classifier is configured to differentiate among 7 distinct classes, aligning with the diverse nature of the dataset.

Data augmentation and preprocessing are integral parts of the training pipeline. The model employs strategies like random resizing and flipping, enhancing the robustness of the model by exposing it to varied representations of input data.

For evaluation, the model's performance is assessed using standard COCO metrics for both bounding box accuracy and segmentation quality. This approach ensures a comprehensive evaluation of the model's effectiveness in both detecting objects and accurately segmenting them.

Regarding the settings for the training environment, such as CUDA settings for GPU acceleration and distributed training configurations, are essential for efficient training, especially when dealing with complex datasets. The model also incorporates visualization and logging mechanisms, facilitating effective monitoring and debugging during the training process. All outputs and intermediate results are organized and stored in a specified working directory, ensuring a structured approach to model development and evaluation.

3.4.2 High Performance Computing (HPC) Settings

For the instance segmentation task, This thesis utilized the HPC environment to train a Mask R-CNN model. A script was written to submit a job to the HPC facilities at DTU, using the LSF Resource Manager/Scheduler for job management. This script specifies various parameters for job execution, including queue type, job name, number of cores, GPU usage, memory requirements, and walltime limit.

In DTU's HPC environment, job scripts are essential for running applications in batch mode, minimizing human intervention. The script begins with the standard shebang line `#!/bin/sh`, followed by `#BSUB` commands as instructions to the LSF system. For example, `-q gpuv100` defines the queue type, `-J seg_torchrun_new_config` sets the job name, and `-n 4` requests four cores.

A single GPU is allocated exclusively with `-gpu "num=1:mode=exclusive_process"`. Memory requirements are specified as `-R "rusage[mem=3GB]"`, allocating 3GB of memory per core. The job's maximum duration is set with `-W 24:00`, limiting it to 24 hours.

The script then sets up the environment and executes commands. The line `conda activate forpred` activates a Python environment with necessary dependencies. Finally, `torchrun` initiates the Mask R-CNN model training.

3.4.3 HPC Running Summary

After completing the HPC job script, the resource usage report was sent by email. It shows the CPU usage was 52244.18 seconds, indicating that the 24-hour time wall was not exceeded. Maximum memory usage peaked at 7599 MB and averaged 6132.80 MB, while total memory requests were 12288.00 MB, suggesting that the task was within the memory allocation.

3.4.4 Model Evaluation

Model was evaluated by introducing mask loss, accuracy, and learning rate in this section.

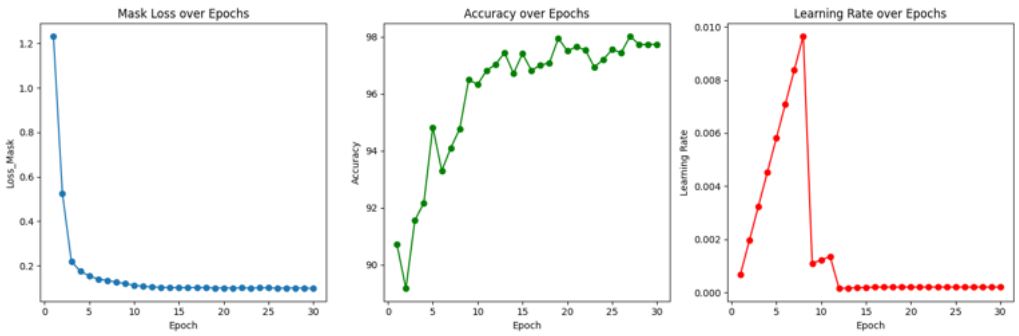

Figure 3.6: Mask Loss, Accuracy and Learning Rate over Epochs.

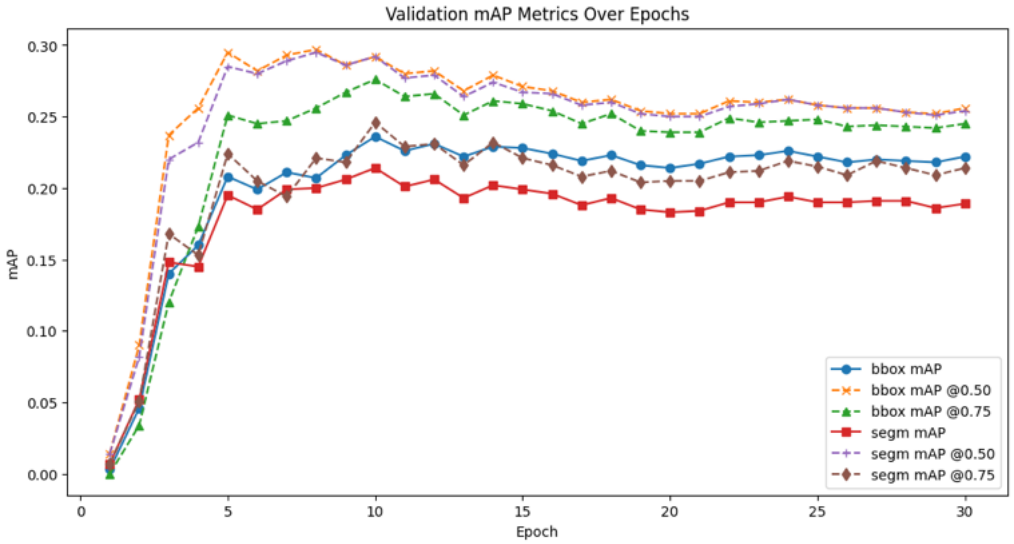

Figure 3.7: BBox and segm mAP over Epochs.

Fig.3.7 shows the model's performance in terms of mAP metrics across validation epochs. The analysis of this plot reveals several insights:

General Trend: There is an upward trend in both bounding box (bbox) and segmentation (segm) mAP values across epochs, suggesting that the model is learning and improving its predictive accuracy over time.

Fluctuations: Some fluctuations in mAP values are observed, which is normal in the training process. These could be due to the model learning new patterns, overfitting on certain data points, or adjustments in the learning rate. Despite these fluctuations, the general trend should be upward.

IoU Thresholds: The mAP values at IoU=0.50 are consistently higher than at IoU=0.75. This is expected as a lower IoU threshold is less stringent. The improvement in mAP at IoU=0.75 suggests that the model is becoming more precise in its predictions.

Comparison Between bbox and segm mAP: Both bbox and segm mAP are crucial for evaluating the model's performance. Typically, bbox mAP is higher than segm mAP as

segmentation is a more complex task. If the segm mAP is close to the bbox mAP, it indicates effective segmentation capability.

Late Epochs Performance: Towards the later epochs, a plateau or a slight decrease in certain mAP metrics is observed. This could indicate overfitting or that the model is reaching its performance limit with the current architecture and hyperparameters.

Segmentation Performance: The segm mAP shows significant improvement, which is noteworthy given the complexity of segmentation tasks. Comparable performance in both bbox and segm mAP suggests proficiency in detection and segmentation.

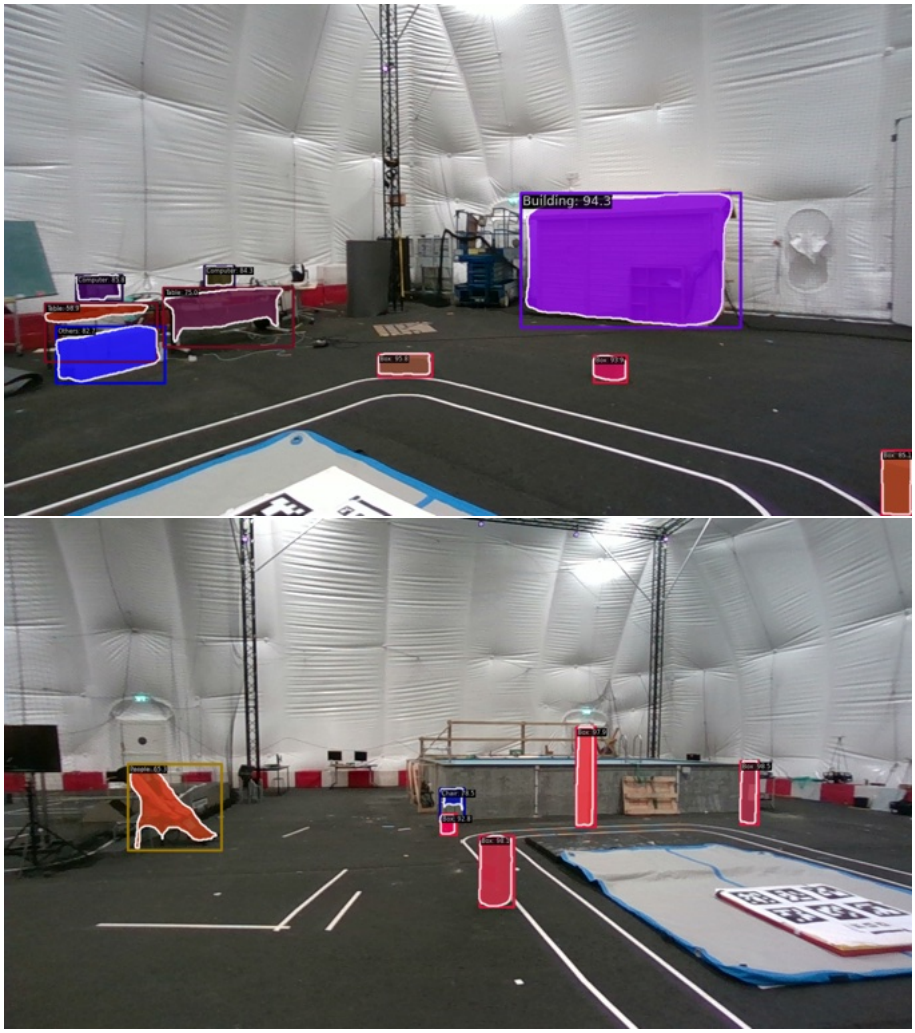

Figure 3.8: Visualization of Mask R-CNN Instance Segmentation.

Figure 3.8 presents examples of the Mask R-CNN model's instance segmentation. The model effectively segments and classifies objects within a middle-scale environment. Detected instances are enclosed by bounding boxes, and segmentation masks accurately outline object contours. The model confidently classifies various objects, such as people and boxes, and closely adheres to object boundaries. This precision is crucial for applications requiring exact object localization and shape delineation.

The provided visualizations show that the Mask R-CNN model is not only adept at detecting objects but also excels in accurately segmenting them, an essential aspect for prac-

tical applications in diverse environments. The high level of detail in the segmentation masks underscores the model's sophistication and its suitability for complex real-world applications.

3.5 Segment Generator

This section outlines the method used to generate segments in the form of colored point clouds. These segments are derived from instance segmentation and point clouds, which are in turn generated from corresponding depth images and camera parameters. Figure 3.9 visualizes this pipeline.

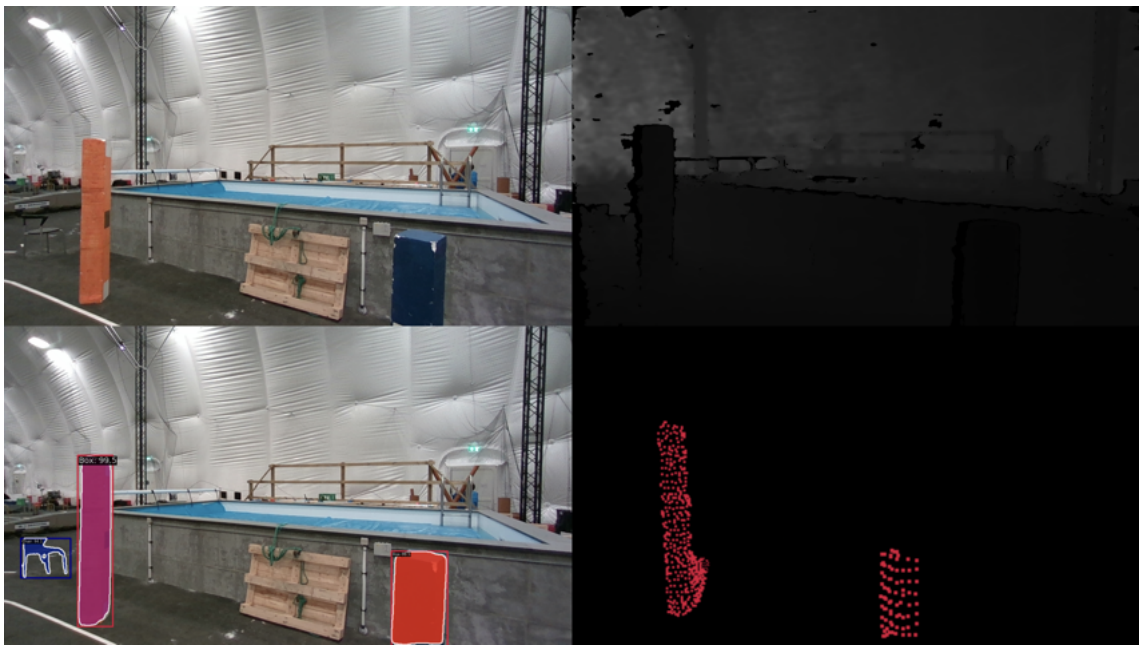

Figure 3.9: **Visualization of Segments Extraction Process.** The procedure initiates with RGB-D data acquisition. A trained Mask R-CNN model then performs 2D instance segmentation. For each identified instance, a mask and label are generated, leading to the creation of distinct segments within the camera frame. These segments are subsequently color-coded based on a palette of seven unique labels.

Employing the best-performing Mask R-CNN model from the HPC, instance segmentation is executed on RGB images, with predictions such as masks, scores, and labels saved in a .json file. Notably, Run-Length Encoding (RLE), a form of lossless data compression, is utilized to encode masks for efficient storage. Pycocotools, part of the COCO API[32], is then used to decode these masks from RLE format.

The framework designed for segment extraction leverages the rich data obtained from RGB image instance segmentation to generate colored segments for each identified instance. Figure 3.10 presents the algorithmic flow of this process.

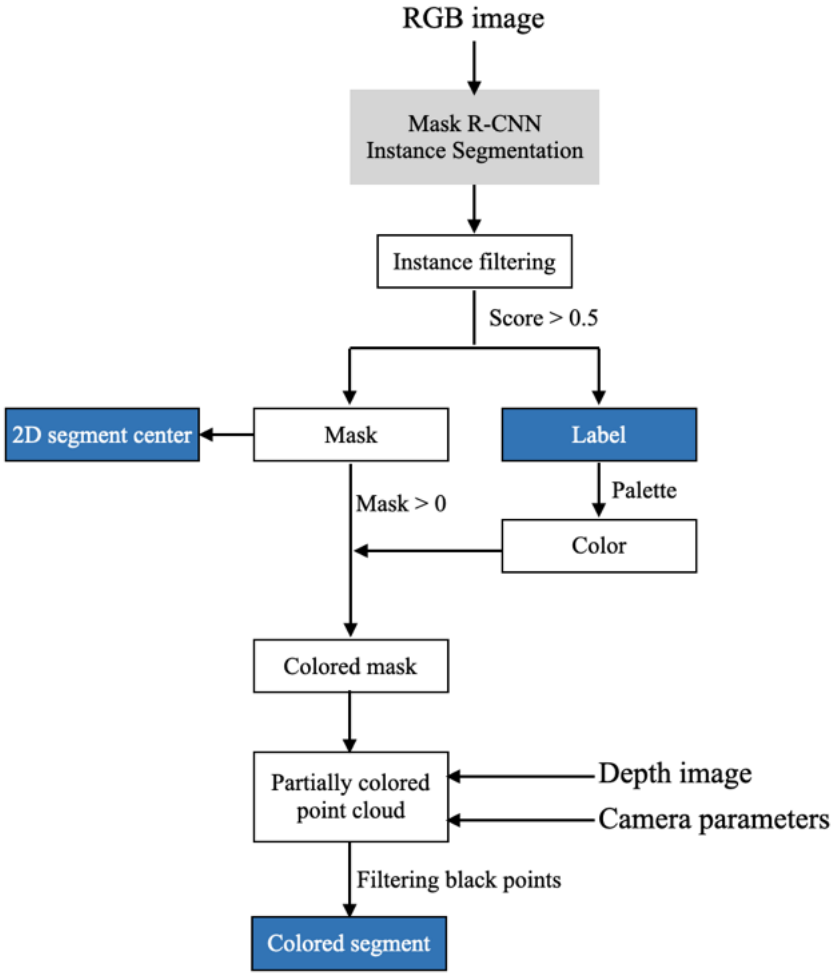

Figure 3.10: Algorithmic Flow of Segment Extraction.

The process filters out instances with a confidence score below 0.5, ensuring only those with a high likelihood of accurate label prediction are retained. A dictionary maps each label to a specific color, which is then applied to pixels where the mask is positive. Additionally, the 2D segment center and label are recorded for later use.

The next step involves combining the colored mask with the depth image and camera parameters, including depth camera intrinsics and scale, to generate a colored point cloud. Points that do not correspond to any segment (i.e., black points) are removed, leaving only the points with the designated color for each segment. This filtered point cloud represents the final output, serving as a colored segment for each individual instance.

This process not only ensures accurate segmentation but also adds a layer of interpretability by color-coding each segment according to its category. The innovative framework effectively utilizes the information from instance segmentation to enrich the point cloud data, providing a more comprehensive and visually intuitive understanding of the scene and the objects within it.

The segments generated through this method are integral for further analysis and applications where precise object localization and characterization are essential. By combining

the depth information from the point clouds with the detailed segmentation from the Mask R-CNN model, this approach effectively bridges the gap between 2D image analysis and 3D spatial representation.

3.6 Descriptor Extraction

This module focuses on the extraction of descriptors from segments aligned in the world frame. Utilizing eigenvalues, the process generates descriptors of size [1x64] for matching in a pre-existing map after the training process. The CNN model employed for training is adapted from the SegMap codebase[10, 33, 23], which is documented in the segmappy branch[28].

3.6.1 Model Setting

SegMap's CNN Model The original CNN model from the SegMap project was based on TensorFlow 1.x, which necessitated explicit session management and a more complex coding style. Adapting to TensorFlow 2.x entailed significant modifications to simplify the code and ensure compatibility.

The model begins with an input layer tailored for 3D data, followed by several convolutional layers with ReLU activation for feature extraction. Interspersed with these are max pooling layers, which reduce data dimensionality and enhance feature detection efficiency. Batch normalization and dropout techniques are incorporated to promote training stability and mitigate overfitting. Following a flattening step, the model integrates additional scales data into dense layers for more profound feature analysis. Furthermore, the model incorporates a reconstruction network with deconvolutional layers, characteristic of autoencoder structures. The architecture's optimization uses softmax cross-entropy for classification and reconstruction loss, with the Adam optimizer overseeing the learning process.

3.6.2 Prepocessing of Dataset

Filtering and Downsampling Given segments from segment generator and transformed from camera frame to world frame, Open3D, a versatile library used for 3D data processing is used to filter out point cloud segments with number of points fewer than a threshold and then apply a downsampling method to ensure the number of points in each segment falls within a desired range.

In the approach using Open3D, point cloud segments are first evaluated to ensure they contain a sufficient number of points, specifically more than 900. This step is crucial for maintaining a level of detail adequate for further processing. Segments that don't meet this threshold are excluded from the analysis.

For those segments that do qualify, a voxel-based downsampling method is applied. This technique consolidates points within cubic voxels of a predetermined size (0.05 in this case), effectively reducing the point cloud's density. This downsampling not only enhances computational efficiency by decreasing the number of points but also helps in mitigating noise, thereby refining the quality of the data.

While the goal is to retain point counts in the range of 100 to 5000, this specific range control isn't directly addressed in the snippet. However, it can be managed by adjusting the voxel size parameter, which influences the resultant point density post-downsampling. Such adjustments ensure that the final point cloud segments are neither too sparse nor excessively dense, making them suitable for accurate and efficient 3D analysis.

Saving of All segments, positions, timestamps, labels and 2D centers A structured approach is designed to save various types of information related to point cloud segments

into CSV files. This is achieved through the `save_all_info` function, which orchestrates the recording of point cloud data, timestamps, segments' center positions in world frame, labels, and 2D centers, each into separate CSV files.

Within `save_all_info`, specific sub-functions are dedicated to handling distinct data types. The `write_to_segments` function stores the spatial coordinates of each point in the point cloud along with associated image and segment identifiers. Timestamps are logged through the `write_to_timestamps` function, linking each time point to the corresponding image and segment. The `write_to_positions` function captures the 3D center positions of segments in the world frame. Additionally, segment labels are stored using the `write_to_labels` function, facilitating future matching algorithms. These segments' center positions are crucial as they provide the real-world spatial context of the point cloud data.

Additionally, the code includes functionality to record the 2D centers of these segments in image coordinates. This data, representing the segments' locations in pixel coordinates, is critical for image-based analysis and calculations.

Storing both the 3D positions in the world frame and the 2D pixel coordinates in the image is a strategic step in preparing for the Perspective-n-Point (PnP) problem. The combination of these two types of data enables the accurate calculation of the drone's pose by correlating points in the image frame with their counterparts in the world frame, a key aspect of spatial orientation and navigation tasks in drone operations.

By ensuring that each piece of information – from spatial coordinates to temporal and categorical data – is meticulously recorded and organized, the `save_all_info` function sets the stage for advanced processing and analysis, including dataset preparation for CNN model and the crucial task of computing the drone's pose using PnP algorithms in the subsequent part of this thesis.

3.6.3 HPC Settings

This section elaborate on the utilization of HPC for training a CNN model, following the earlier application for Mask R-CNN model training.

Job Submission Script: My script, tailored for DTU's HPC facility, orchestrates a training job leveraging the LSF Resource Manager/Scheduler. It commences with a shebang `#!/bin/sh` and employs `#BSUB` directives to delineate job specifications. These include queue assignment `#BSUB -q gpuv100` for GPU usage, job naming `#BSUB -J train_cnn`, core allocation `#BSUB -n 4` for four cores, and exclusive GPU access. Memory requirements are explicitly stated, requesting 4GB per core and specifying host and GPU characteristics to optimize performance. A walltime limit of 18 hours is set, alongside notification and output file configurations.

The script's latter part focuses on the environment setup, activating a Conda environment and loading modules like CUDA, cuDNN, and TensorRT, essential for GPU-accelerated computation in neural networks. The execution of the training script

Integration of CUDA for TensorFlow 2.x: The HPC environment's configuration for TensorFlow 2.x involves specific CUDA settings. As per DTU HPC's guidelines[34], configuring TensorFlow to leverage CUDA enables efficient GPU usage, vital for accelerating deep learning computations. This setup ensures TensorFlow's compatibility with the available GPU resources, maximizing computational efficiency and significantly reducing training time for the CNN model.

3.6.4 HPC Running Summary

The job log reflects successful execution, indicating start and completion times. The resource summary reveals a CPU time usage of 20799.21 seconds and memory consumption peaking at 2847 MB, against a larger requested quota. The job's runtime and turnaround time fit well within the allocated limits.

3.6.5 Model Evaluation

CNN model was evaluated by reconstruction loss and training accuracy.

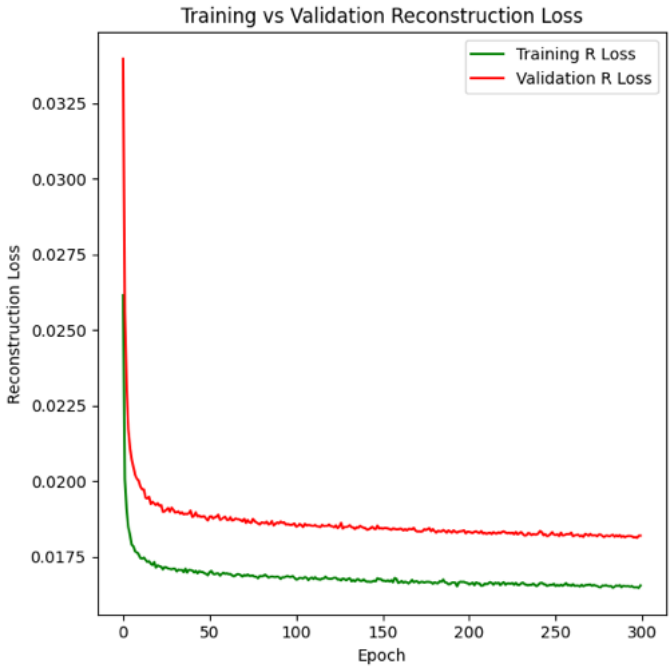

Figure 3.11: Training vs Validation Reconstruction Loss

For a deep learning model to be robust and to mitigate the risk of overfitting, training it on a large and diverse dataset is crucial. In this thesis, the dataset for training is augmented with a substantial dataset derived from the KITTI odometry dataset[35], which has been manually annotated by SegMap researchers[10]. This approach ensures a rich and varied training environment for the model.

The effectiveness of the training process is evident from the significant decrease in reconstruction loss observed in both the training and validation phases, as shown in Figure 3.11. This reduction in loss indicates that the model is learning effectively from the data and improving its ability to reconstruct the input. Notably, the training accuracy achieved is approximately 50%. While this might appear modest, it is significant in the context of the complex tasks the model is performing. This level of accuracy, in conjunction with the observed loss reduction, suggests that the model is training effectively and achieving a good balance between learning from the training data and generalizing to new, unseen data.

It's important to note that training accuracy alone is not the sole indicator of a model's effectiveness. The decrease in reconstruction loss is a strong indicator that the model is capturing the underlying patterns in the data, which is critical for tasks such as descriptor extraction and instance segmentation. These results suggest that the model is well-suited for these tasks and has the potential to perform effectively in practical applications.

3.7 Global/Local Map Generator

This section will explain the process of creating a comprehensive 3D map by transforming point cloud data from the camera frame to the world frame. This transformation is integral to the generation of both global and local maps, each with its unique methodology and data integration process.

Global Map Generation: The transformation from the drone to the world frame is precisely determined by the motion capture system. This high level of accuracy allows for the effective alignment of each segmented point cloud in camera frame with its corresponding pose in the world frame.

These segments are initially identified in the camera frame, utilizing a 2D instance segmentation model. This model not only detects the segments but also provides critical data such as the 2D image pixel center and the associated label for each segment. Additionally, a high-dimensional descriptor generated from a CNN model enriches the segment data, contributing to a more detailed and informative global map.

The following plots in Fig. 3.12 and Fig. 3.13 shows trajectories when capturing the data at ASTA. Notably, the second columns of both 3D and 2D show trajectories that used for generation of global map.

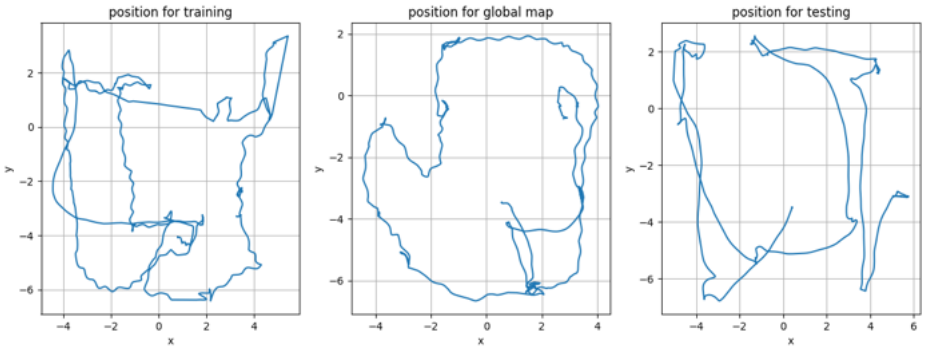

Figure 3.12: 2D Trajectories of Data Capture

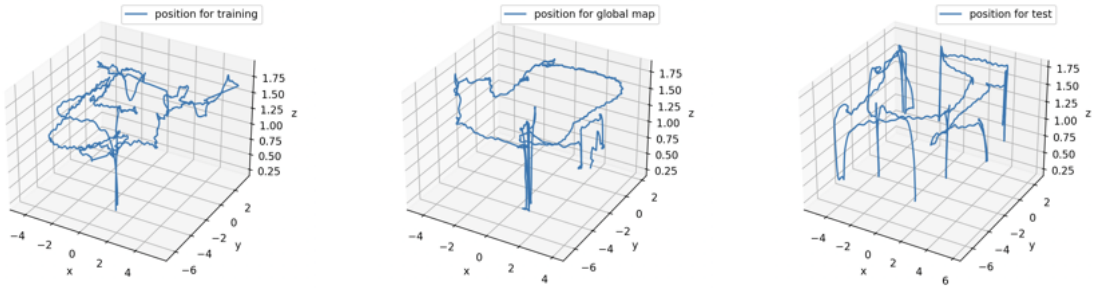

Figure 3.13: 3D Trajectories of Data Capture

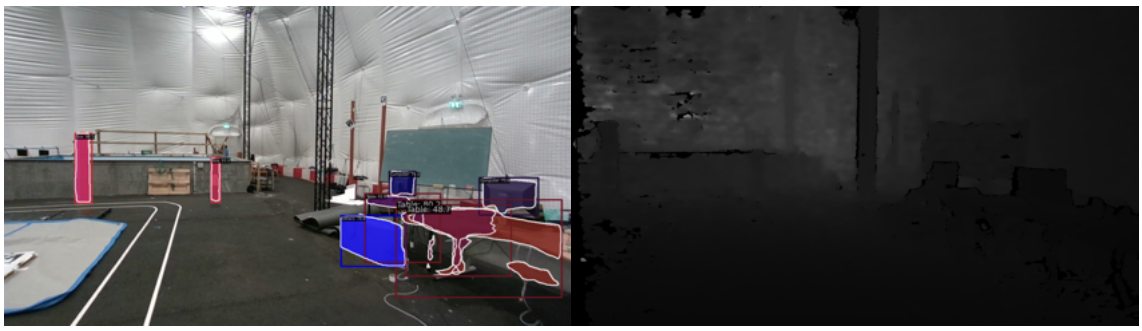

Figure 3.14: 2D segmentation and Depth image

Local Map Generation: In contrast, the local map generation follows a different protocol. Each segment in the camera frame is first transformed into the drone frame. The drone's pose, estimated by the IMU, is then used to calculate the transformation matrix from the drone frame to the world frame. This step is crucial for estimating the 3D centroid of each segment in the world frame. The local map segments, like the global map, retain information about their 2D image pixel center and label, which is vital for subsequent matching processes and the integration of local map data into the global map framework.

Estimating the Transformation Matrix with IMU Data: IMU data plays a pivotal role in estimating the drone's current pose in the world frame, particularly for the local map. The IMU's sensors, including accelerometers and gyroscopes, provide raw data on the drone's linear acceleration and angular velocity. By integrating this data over time, an estimation of the drone's velocity, orientation, and position relative to its initial state is achieved. This transformation matrix provided by the IMU data is instrumental in estimating the pose of segments in the world frame.

Limitations in Local Map Realization: Although the ideal scenario for local map generation would involve detecting segments across a wide range of distances, as illustrated in Figures 3.15, the current setup using a single RGB-D camera faces limitations. Specifically, the camera's range restricts detection to nearer segments, typically in darker shades of blue or red, as shown in Figure 3.16. This limitation highlights the need for equipment capable of longer-range perception for more comprehensive local mapping.

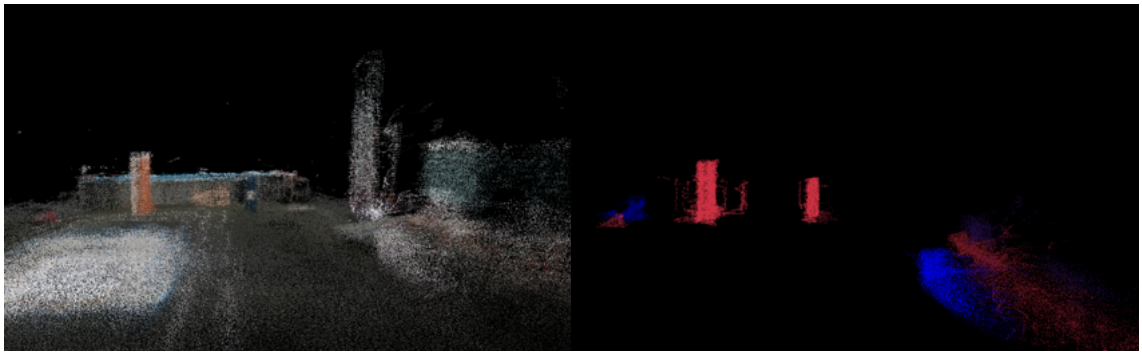

Figure 3.15: Visualization of Frame in the Global Real Map and Semantic Map

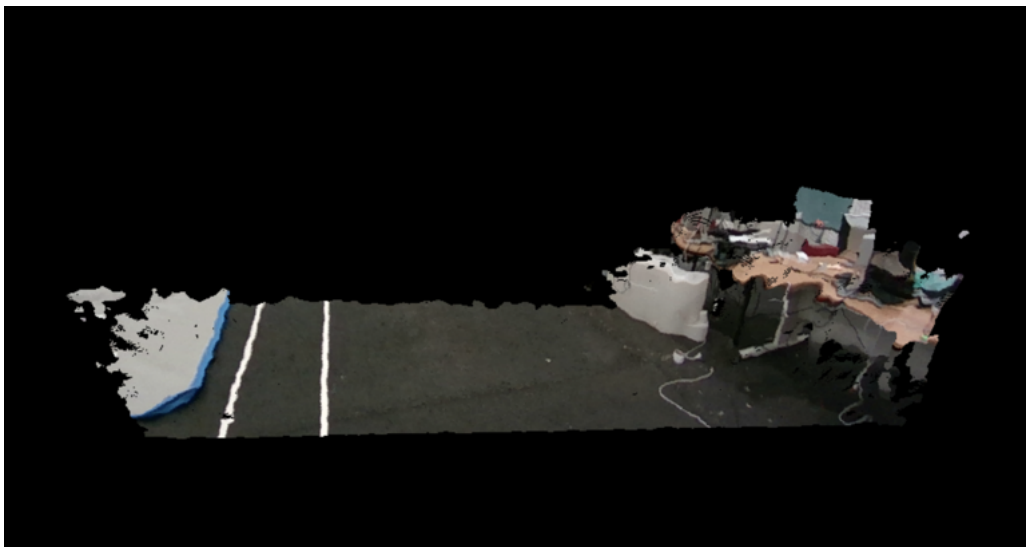

Figure 3.16: Local Point Cloud with RGB Information.

Visualization of Segment Transformations: To provide a clear understanding of the transformation process, visualizations of local segments in both the camera frame and the world frame are presented. Left picture in Fig. 3.17 shows the segments as they appear in the camera frame, whereas right picture in Fig. **??** displays these same segments transformed into the world frame. These visualizations underscore the transformation process, highlighting the transition of data from the drone's perspective to a global context, which is essential for accurate map generation and spatial analysis.

Figure 3.17: Local Segment in Camera Frame and World Frame.

3.8 Methodology for Real-Time Pose Estimation with IMU

Pose estimation plays a pivotal role in the field of UAVs, enabling critical capabilities such as autonomous navigation, obstacle avoidance, and dynamic interaction with the environment. The real-time estimation of the UAV's position and orientation, referred to as pose, is fundamental for effective operational control and autonomy.

3.8.1 Challenges in Real-Time Pose Estimation

Real-time pose estimation in dynamic environments presents several challenges. These include the need for high accuracy and low latency in the face of limited computational resources, environmental variability, and the inherent noise in sensor data.

3.8.2 Overview of the Approach

This section presents the approach to real-time pose estimation for UAVs, leveraging data from Inertial Measurement Units (IMUs). The method focuses on estimating the drone's

position (x, y, z) and orientation (roll, pitch, yaw) in real-time, using sensor fusion and integration techniques. IMUs measure linear accelerations and angular velocities. The accelerometers measure specific force (acceleration) along three orthogonal axes, while gyroscopes measure angular velocity around these axes. These measurements are the primary inputs for estimating the UAV's pose.

Data Acquisition and Preprocessing IMUs measure linear accelerations and angular velocities. The accelerometers measure specific force (acceleration) along three orthogonal axes, while gyroscopes measure angular velocity around these axes. These measurements are the primary inputs for estimating the UAV's pose. Preprocessing steps are crucial in enhancing the accuracy of this data. These steps include:

- Bias Removal: Both accelerometers and gyroscopes exhibit biases that can significantly affect measurement accuracy. These biases are subtracted from the raw measurements to enhance data fidelity.
- Noise Filtering: IMU data is susceptible to noise, often necessitating the application of filtering techniques to obtain cleaner signals.

Transformation and Integration Techniques The transformation from the IMU's coordinate system to the drone's coordinate system is achieved using transformation matrix:

$$T_{ID} = \begin{pmatrix} R_{ID} & \vec{t}_{ID} \\ 0 & 1 \end{pmatrix}$$

.

Here translation vector is emply and this matrix accounts for the orientation differences between the IMU and the drone's frame of reference. The transformation matrix from drone frame to world frame is:

$$T_{DW} = \begin{pmatrix} R_{DW} & \vec{t}_{DW} \\ 0 & 1 \end{pmatrix}$$

Unlike the T_{ID} which is predefined by user and has fix value, T_{DW} needs to be calculated by using gyro to update orientation and the R_{DW} and use the updated R_{DW} to update velocity and position by applying the ration matrix to acceleration.

Orientation Estimation The orientation of the UAV is estimated by integrating the angular velocity measurements from the gyroscopes. The integration process involves computing the incremental changes in orientation over time, described by the following equation:

$$\Delta\text{orientation} = \text{gyro} \times dt \tag{3.1}$$

where gyro is the angular velocity vector and dt is the time interval. The drone's orientation in terms of roll, pitch, and yaw is updated incrementally using these calculations.

Position Estimation The position of the UAV is estimated by integrating the acceleration data to compute velocity and subsequently position. The accelerations, after being corrected for gravitational effects, are integrated to obtain velocity:

$$\text{velocity}_t = \text{velocity}_{t-1} + \text{acceleration} \times dt \tag{3.2}$$

The velocity is then integrated to calculate the position:

$$\text{position}_t = \text{position}_{t-1} + \text{velocity}_t \times dt + 0.5 \times \text{acceleration} \times dt^2 \tag{3.3}$$

This method provides a comprehensive framework for real-time pose estimation using IMU data, which is essential for the dynamic and autonomous operation of UAVs.

3.8.3 Mathematical Formulation of IMU-Based Pose Estimation

The pose estimation algorithm integrates data from the Inertial Measurement Unit (IMU) to compute the orientation and position of the UAV in the world frame. The process involves several transformation and integration steps, abstractly formulated as follows:

Coordinate Frame Definitions

- f_I: The coordinate frame of the IMU.
- f_D: The coordinate frame of the drone.
- f_W: The world coordinate frame.

Transformation Matrix and Bias Correction

The transformation from the IMU frame to the drone frame is defined by the rotation matrix R_{ID}:

$$R_{ID} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \tag{3.4}$$

Bias is removed from the IMU measurements as follows:

$$gyro_{bias} = \begin{bmatrix} 0.0020178240054855253 \\ 0.0017592206368581425 \\ 0.0017427009552286577 \end{bmatrix}, \tag{3.5}$$

$$acc_{bias} = \begin{bmatrix} 0.28478655780245 \\ -0.024718655019298263 \\ -9.828971270850001 \end{bmatrix}, \tag{3.6}$$

$$acc_{corrected} = acc - acc_{bias}, \tag{3.7}$$

$$gyro_{corrected} = gyro - gyro_{bias}. \tag{3.8}$$

Orientation and Position Estimation

The orientation estimation involves transforming the gyroscopic data from the IMU frame to the drone frame, and then to the world frame, followed by integration over time:

$$gyro_W = R_{DW} \cdot R_{ID} \cdot gyro_{corrected}, \tag{3.9}$$

$$\Delta_{orientation} = gyro_W \times dt, \tag{3.10}$$

$$orientation_t = orientation_{t-1} + \Delta_{orientation}. \tag{3.11}$$

For position estimation, the acceleration data is similarly transformed and integrated:

$$acc_W = R_{DW} \cdot R_{ID} \cdot acc_{corrected}, \tag{3.12}$$

$$\Delta_v = acc_W \times dt, \tag{3.13}$$

$$v_t = v_{t-1} + \Delta_v, \tag{3.14}$$

$$\Delta_{position} = v_{t-1} \times dt + \frac{1}{2} acc_D \times dt^2, \tag{3.15}$$

$$position_t = position_{t-1} + \Delta_{position} \tag{3.16}$$

where v, t and dt are velocity, current time and sampling time.

3.8.4 Validation and Applications

The accuracy and reliability of the pose estimation method are validated through various experiments and simulations. In IMU-based pose estimation for UAVs, two significant sources of error that affect accuracy and reliability are drift error and noise error[36].

Effective mitigation of drift and noise errors is crucial in IMU-based pose estimation systems. Approaches for addressing these errors include advanced sensor calibration techniques, implementation of filtering algorithms such as the Kalman filter, and sensor fusion methodologies that integrate data from vision-based pose estimation to enhance overall system accuracy and robustness.

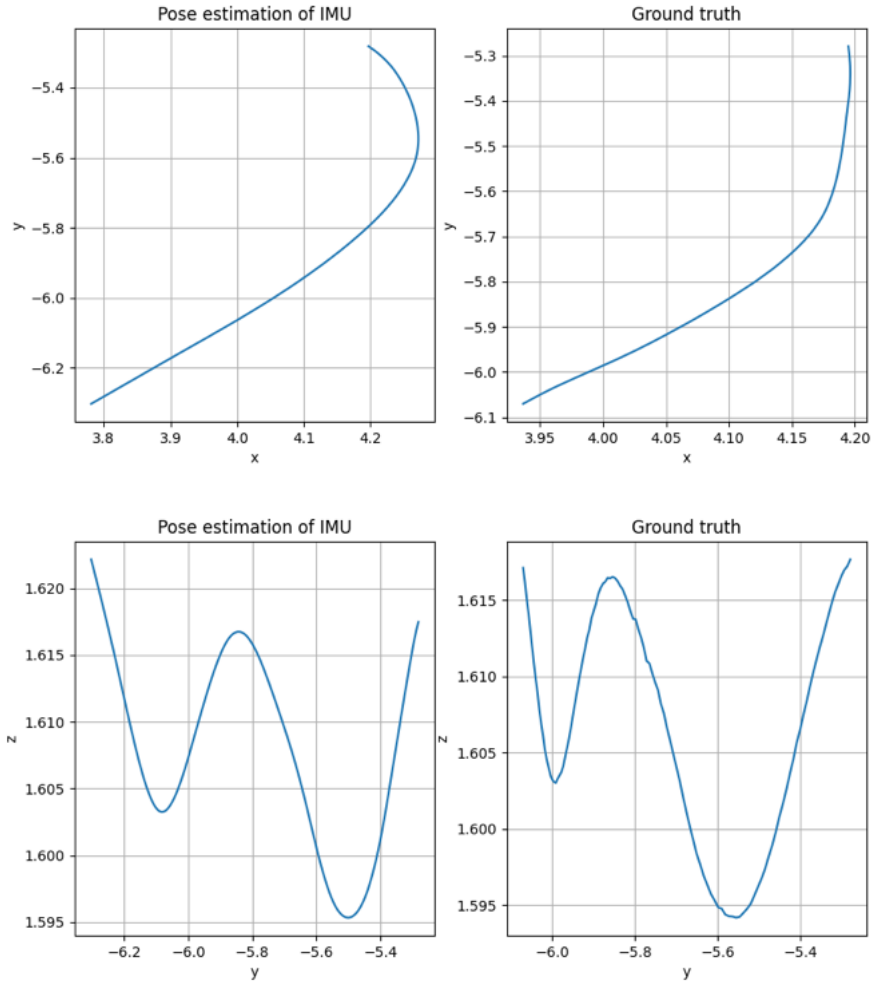

Figure 3.18: IMU-based Pose Estimation in 2D.

Figure 3.18 demonstrates that IMU-based pose estimation performs effectively over short intervals, such as those slightly exceeding one second. This short-term accuracy underscores the importance of precise calibration and bias correction in the estimation process. However, for real-time applications requiring continuous pose estimation, the limitations of IMU data become more apparent.

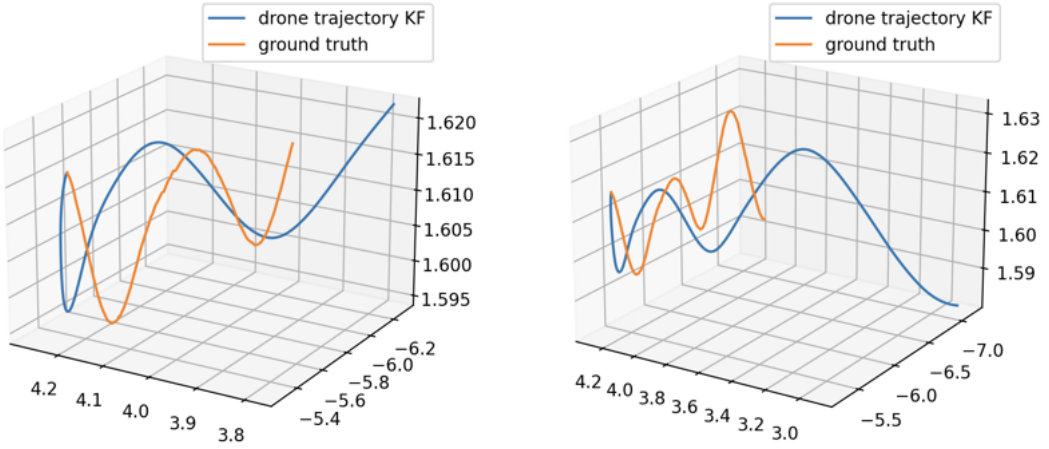

Figure 3.19: IMU-based Pose Estimation in 3D.

As depicted in Figure 3.19, while IMU data can provide a reasonable estimation in the short term, it is prone to accumulating errors over time. In scenarios requiring ongoing, real-time pose estimation, these errors can become significant. Shift error and noise error, in particular, tend to accumulate, leading to a level of inaccuracy that may be unacceptable for precise navigation tasks. This accumulation of errors is a critical challenge in utilizing IMU for long-term tracking and necessitates the consideration of additional data sources or correction methodologies to maintain accuracy in pose estimation over extended periods.

These findings highlight the inherent trade-offs in using IMU data for pose estimation: while useful for short-duration estimations due to its immediacy and relative simplicity, it lacks the long-term reliability required for sustained and precise navigation tasks. Therefore, for applications demanding continuous and accurate pose tracking, such as in advanced drone navigation or autonomous vehicle systems, it becomes necessary to either integrate additional sensing modalities or develop sophisticated algorithms to compensate for the IMU's limitations. This could involve the use of visual data or other sensor fusion techniques to correct and refine the pose estimates over time, ensuring sustained accuracy and reliability in real-world applications.

3.9 Matching Strategy in a Prior Map

Accurate localization in GPS-denied environments is a significant challenge in autonomous navigation, necessitating advanced techniques in computer vision and sensor fusion. This section discusses the methodologies and algorithms employed for matching sensor-derived data with a pre-established map.

The proposed method combines gyroscopic and accelerometric data from IMUs with visual segment analysis. This involves extracting features from environmental landmarks, with a focus on segment-based visual data processing. Segments, which include descriptors, centroids, and labels, are extracted from the visual data and then matched against a prior map containing similar information for global segments.

The matching strategy encompasses several steps:

IMU Data Processing: The raw IMU data is processed to estimate real-time pose, addressing biases and noise typically associated with these sensors.

Visual Data Segmentation: Concurrently, visual data is segmented to obtain descriptors, labels, and estimated 3D centroids, along with 2D centers of segments.

Data Fusion: The IMU and visual data streams are fused, marking entries with labels like `'IMU'` or `'Visual'` to manage the differing streaming rates of the sensors. High-frequency IMU data is used for pose estimation and stored in an EKF. When lower-frequency (30Hz) visual data arrives, the estimated pose serves as a reference for generating the local map.

Local Map Analysis: The local map, along with descriptors, estimated 3D centroids, and labels, is analyzed using the KD-Trees algorithm for efficient nearest-neighbor searches. This step involves finding the top five matches within the prior map and retrieving the corresponding K-nearest neighbor (KNN) distances.

Distance Calculation: Euclidean distances between the pairs of centroids (both from the local and prior maps) are calculated.

Combined Distance Evaluation: A combined distance metric is computed, incorporating the KNN distance, Euclidean distance, and a score reflecting the label consistency. The segment with the smallest combined distance is identified as the match in the prior map.

This intricate process of matching segments in the local map to those in the prior map is pivotal for segment-based pose estimation. By accurately identifying corresponding segments, the system can effectively determine the device's location relative to known landmarks in the prior map. This approach is especially crucial in environments where traditional GPS-based localization is unreliable or unavailable.

The integration of IMU and visual data, along with the sophisticated use of algorithms like KD-Trees for nearest-neighbor searches, exemplifies the advanced level of sensor fusion and data analysis required in modern autonomous navigation systems. This matching strategy not only enhances the accuracy of localization but also contributes to the overall robustness and reliability of the navigation system in dynamic and challenging environments.

The subsequent section will delve into how these matched segments are utilized for segment-based pose estimation, further advancing the capability of the system to navigate.

3.10 Segment-based Pose Estimation

Pose estimation, the process of determining the position and orientation of a device within a space, is crucial for various applications in robotics and autonomous navigation. Through the implementation of an Extended Kalman Filter, uncertainties in IMU sensor data is deal with segment-base measurement in this section.

The EKF is designed to fuse IMU data with visual information for real-time pose estimation. It maintains a state vector, potentially encompassing position, orientation, and velocity of the drone. It is configured with specific variances for these parameters, reflecting the characteristics and sensor noise of the sensor in use.

The EKF operates by predicting the next state of the system based on the current state and control inputs derived from the IMU data. It then integrates the visual data, involving the position and possible orientation information extracted from the segmented visual data. This integration is achieved through a measurement update step, where the filter combines the predicted state with the new measurements to produce an updated estimate of the drone's pose.

Result of visual pose estimation will differ when the number of valid segments in visual data is lower than 4, which means there are not enough segments to measure pose by utilizing PnP method. In this scenario, only position will be measured and used to updates state in EKF. When there are more than 4 segments detected, pose can be estimated by using PnP method, and more segments detected, more accurate the estimation would be.

The mathematical backbone of the EKF involves formulating both a state transition model and a measurement model. The state transition model predicts the system's future state, considering the dynamics of movement and sensor inputs. The measurement model, on the other hand, relates the actual sensor measurements to the predicted state. Through a series of update and prediction steps, the EKF continuously refines its estimate of the device's pose, minimizing the overall uncertainty by balancing the predictions with new sensor inputs. This process is crucial for achieving accurate and reliable pose estimation in dynamic environments.

3.11 Evaluation of Sensor Fusion

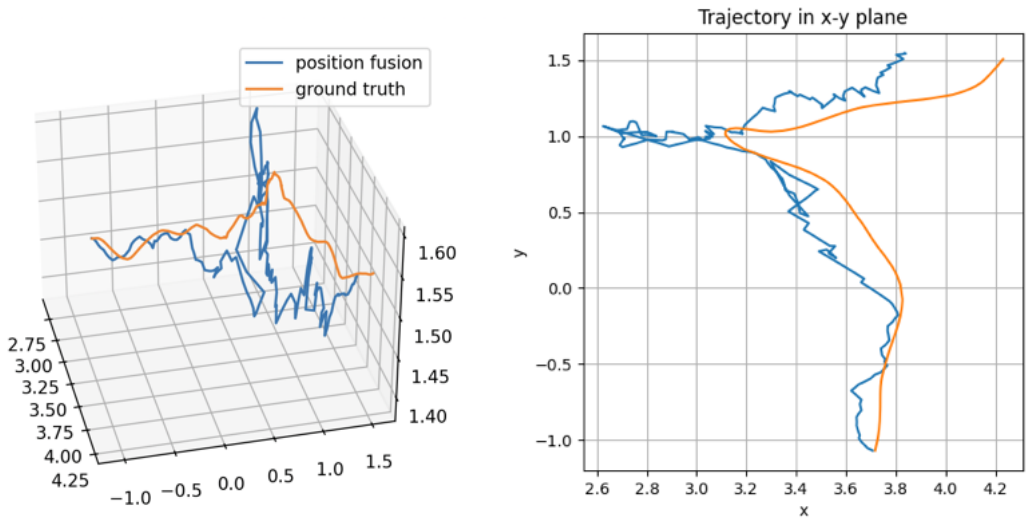

Figure 3.20: Comparison in 3D and Comparison in x-y Plane.

Figure 3.21: Visualization of Moving in 10 Seconds.

Figure 3.22: **Visual Marks of x, y or z Axis.** The red lines indicate update in EKF and its rate is 30Hz, while the IMU streaming for EKF prediction is 200Hz (IMU marks are not plotted.)

The Fig.3.20 shows the trajectory of the drone in the x-y plane. There are two lines: the estimated trajectory (blue) and the ground truth (orange). The alignment between the estimated trajectory and the ground truth in the x-y plane exhibits the system's sophisticated

capability to adapt and correct its path dynamically. The instances where the estimated trajectory deviates from the ground truth can be attributed to the challenging nature of segment-based localization, which relies heavily on the quality and consistency of visual data. Yet, the algorithm's ability to re-align with the ground truth path suggests a resilient matching strategy that can self-correct in response to new data inputs, a hallmark of intelligent systems.

Figure 3.23: Error in x, y, z and Postion

In Fig.3.22 and Fig. 3.23, we examine the estimated and actual positions over time along the x, y, and z axes, including an analysis of their respective errors. The results display a commendable level of accuracy in the y and z-axis estimations, as indicated by the minimal fluctuation in these dimensions. This highlights the system's effectiveness in estimating positions along these axes.

However, a notable observation is made in the x-axis estimation. From the midpoint of the observed period to its end, there is a relative bias evident in the x-axis. This deviation can be attributed to the inherent characteristics of the 3D segments used in the pose estimation process. Specifically, the centroid of these segments, crucial for pose estimation, is influenced by the varying observation angles when segments are matched in the global map.

This variance in observation angles can lead to discrepancies in the x-axis estimation, as the relative position of the centroid shifts depending on the drone's perspective. This effect underscores the complexity of segment-based localization and the challenges posed by the multi-dimensional nature of the task. Despite this, the overall consistency and accuracy in the y and z-axis estimations are promising, indicating that the system has a reliable performance in these aspects. The observed bias in the x-axis offers valuable insights into the limitations and areas for potential refinement in the algorithm, contributing to the ongoing development of more robust pose estimation methods.

Despite these small deviations, the overall consistency in the trajectory patterns suggests that the system is capable of stable performance under various conditions. This level of predictability is very important in real-time systems. Being able to foresee and correct errors is crucial for enhancing reliability during operations.

4 Conclusion and Future Work

The thesis presents a practical approach to real-time pose estimation by combining IMU data with visual information. Using an Extended Kalman Filter as a central component for sensor fusion reflects a creative blend of traditional estimation methods with contemporary sensor data to address the pressing need for reliable localization in environments where GNSS is not available.

The algorithm's capacity to correct its path and align with verified trajectories is a testament to its potential. It indicates a system that's designed to be aware and adaptive, qualities that are essential in dynamic environments where conditions change rapidly.

Emphasizing segment-based mapping and localization demonstrates an appreciation for the finer details of environmental interaction, which is often missed by more general point-based systems. This detail-oriented approach could lead to a deeper understanding of various terrains and settings, a valuable trait for precision-critical applications.

The methods and tools used here suggest a system that can be scaled and tailored to different needs. The versatility of this system opens up possibilities ranging from navigating indoor spaces to maneuvering through cluttered outdoor environments, and beyond.

This work signals the broader impact that thoughtfully designed autonomous systems might have. The strategies and techniques developed could potentially be adapted for diverse uses, including autonomous vehicles and sophisticated navigational aids.

Looking ahead, further refining the sensor calibration process, improving data fusion methods, and considering the inclusion of additional types of sensors are practical steps that could be taken. These efforts, alongside making the system more computationally efficient, will be critical for preparing the system for real-world application where dependability is crucial. Additionally, tapping into the capabilities of deep learning for more effective feature extraction and adding layers of semantic understanding might offer new ways to interpret and interact with different environments.

For real-world applications, the system's robustness and adaptability should be evaluated through comprehensive testing across a range of scenarios. These tests will provide the solid grounding needed for future exploration and potential deployment.

Despite the challenges, such as sensor noise and the limited computing resources, the resilience of the approach is noteworthy. The ability to adapt and maintain a level of stability in the face of such obstacles highlights the practical value of the research.

In sum, this thesis makes a constructive contribution to the domain of GNSS localization and autonomous navigation. The smart combination of IMU data with visual cues, supported by machine learning insights, marks a step towards creating autonomous systems that can operate without GPS. The groundwork laid here is ripe for future advancements that could significantly enhance UAV navigation and even influence the wider field of autonomous system design. The methodologies, while there's room for improvement, set the stage for ongoing exploration in this dynamic area of study.

Bibliography

[1] Dimosthenis C. Tsouros, Stamatia Bibi, and Panagiotis G. Sarigiannidis. “A Review on UAV-Based Applications for Precision Agriculture”. In: *Information* 10.11 (2019). ISSN: 2078-2489. DOI: 10.3390/info10110349. URL: https://www.mdpi.com/2078-2489/10/11/349.

[2] Iván Maza et al. “Experimental Results in Multi-UAV Coordination for Disaster Management and Civil Security Applications”. In: *J Intell Robot Syst* 61.1-4 (2011), pp. 563–585.

[3] Naser Hossein Motlagh, Miloud Bagaa, and Tarik Taleb. “UAV-Based IoT Platform: A Crowd Surveillance Use Case”. In: *IEEE Communications Magazine* 55.2 (2017), pp. 128–134. DOI: 10.1109/MCOM.2017.1600587CM.

[4] Byung Duk Song, Kyungsu Park, and Jonghoe Kim. “Persistent UAV delivery logistics: MILP formulation and efficient heuristic”. In: *Computers Industrial Engineering* 120 (2018), pp. 418–428. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2018.05.013. URL: https://www.sciencedirect.com/science/article/pii/S0360835218302146.

[5] Chang-Sun Yoo Chang-Sun Yoo and Iee-Ki Ahn Iee-Ki Ahn. “Low cost GPS/INS sensor fusion system for UAV navigation”. In: *Digital Avionics Systems Conference, 2003. DASC ’03. The 22nd*. Vol. 2. 2003, 8.A.1-8.1–9 vol.2. DOI: 10.1109/DASC.2003.1245891.

[6] Kexin Guo et al. “Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments”. In: *International Journal of Micro Air Vehicles* 9.3 (2017), pp. 169–186.

[7] Janis Tiemann, Florian Schweikowski, and Christian Wietfeld. “Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments”. In: *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. 2015, pp. 1–7. DOI: 10.1109/IPIN.2015.7346960.

[8] G Balamurugan, J Valarmathi, and V P S Naidu. “Survey on UAV navigation in GPS denied environments”. In: *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*. 2016, pp. 198–204. DOI: 10.1109/SCOPES.2016.7955787.

[9] Yingxiu Chang et al. “A review of UAV autonomous navigation in GPS-denied environments”. In: *Robotics and Autonomous Systems* 170 (2023), p. 104533. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2023.104533. URL: https://www.sciencedirect.com/science/article/pii/S0921889023001720.

[10] Renaud Dubé et al. “SegMap: 3D Segment Mapping using Data-Driven Descriptors”. In: *Robotics: Science and Systems XIV*. RSS2018. Robotics: Science and Systems Foundation, June 2018. DOI: 10.15607/rss.2018.xiv.003. URL: http://dx.doi.org/10.15607/RSS.2018.XIV.003.

[11] Milton C.P. Santos et al. “Indoor low-cost localization system for controlling aerial robots”. In: *Control Engineering Practice* 61 (2017), pp. 93–111. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2017.01.011. URL: https://www.sciencedirect.com/science/article/pii/S0967066117300114.

[12] Gui-Song Xia Yuncheng Lu Zhucun Xue and Liangpei Zhang. “A survey on vision-based UAV navigation”. In: *Geo-spatial Information Science* 21.1 (2018), pp. 21–32. DOI: 10.1080/10095020.2017.1420509. URL: https://doi.org/10.1080/10095020.2017.1420509.

[13] Andy Couturier and Moulay A. Akhloufi. “A review on absolute visual localization for UAV”. In: *Robotics and Autonomous Systems* 135 (2021), p. 103666. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2020.103666. URL: https://www.sciencedirect.com/science/article/pii/S0921889020305066.

[14] Wilbert G. Aguilar, Verónica P. Casaliglla, and José L. Pólit. “Obstacle Avoidance Based-Visual Navigation for Micro Aerial Vehicles”. In: *Electronics* 6.1 (2017). ISSN: 2079-9292. DOI: 10.3390/electronics6010010. URL: https://www.mdpi.com/2079-9292/6/1/10.

[15] Luitpold Babel. “Flight path planning for unmanned aerial vehicles with landmark-based visual navigation”. In: *Robotics and Autonomous Systems* 62.2 (2014), pp. 142–150. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2013.11.004. URL: https://www.sciencedirect.com/science/article/pii/S0921889013002212.

[16] Milton C. P. Santos et al. “Estimating and controlling UAV position using RGB-D/IMU data fusion with decentralized information/Kalman filter”. In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. 2015, pp. 232–239. DOI: 10.1109/ICIT.2015.7125104.

[17] Amedeo Rodi Vetrella et al. “RGB-D camera-based quadrotor navigation in GPS-denied and low light environments using known 3D markers”. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2015, pp. 185–192. DOI: 10.1109/ICUAS.2015.7152290.

[18] Sifat Rezwan and Wooyeol Choi. “Artificial Intelligence Approaches for UAV Navigation: Recent Advances and Future Challenges”. In: *IEEE Access* 10 (2022), pp. 26320–26339. DOI: 10.1109/ACCESS.2022.3157626.

[19] Chunxue Wu et al. “UAV Autonomous Target Search Based on Deep Reinforcement Learning in Complex Disaster Scene”. In: *IEEE Access* 7 (2019), pp. 117227–117245. DOI: 10.1109/ACCESS.2019.2933002.

[20] Jinya Su et al. “AI meets UAVs: A survey on AI empowered UAV perception systems for precision agriculture”. In: *Neurocomputing* 518 (2023), pp. 242–270. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.11.020. URL: https://www.sciencedirect.com/science/article/pii/S0925231222013996.

[21] Zhe Zhao and Xiaoping Chen. “Building 3D semantic maps for mobile robots using RGB-D camera”. In: *Intelligent Service Robotics* 9 (2016), pp. 297–309. URL: https://api.semanticscholar.org/CorpusID:3701230.

[22] Charles R. Qi et al. *Frustum PointNets for 3D Object Detection from RGB-D Data*. 2018. arXiv: 1711.08488 [cs.CV].

[23] Renaud Dube et al. “SegMatch: Segment Based Place Recognition in 3D Point Clouds”. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[24] Andrei Cramariuc et al. *SemSegMap- 3D Segment-Based Semantic Localization*. 2021. arXiv: 2107.14715 [cs.RO].

[25] Justin Brooks. *COCO Annotator*. https://github.com/jsbroks/coco-annotator/. 2019.

[26] Kaiming He et al. *Mask R-CNN*. 2018. arXiv: 1703.06870 [cs.CV].

[27] Kai Chen et al. “MMDetection: Open MMLab Detection Toolbox and Benchmark”. In: *arXiv preprint arXiv:1906.07155* (2019).

[28] Renaud Dubé et al. *SegMap*. https://github.com/ethz-asl/segmap/tree/master/segmappy. Accessed: September 10, 2023. 2023.

[29] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[30] Intel RealSense Developers. *librealsense Python Wrappers*. https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python. [Online; accessed September 1, 2023]. 2023.
[31] PX4 Developers. *pyulog*. https://github.com/PX4/pyulog. [Online; accessed December 9, 2023]. 2023.
[32] COCO API. *COCO API*. https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools. Accessed: December 10, 2023. 2023.
[33] Renaud Dubé et al. "SegMap: Segment-based mapping and localization using data-driven descriptors". In: *The International Journal of Robotics Research* 39.2-3 (2020), pp. 339–355. DOI: 10.1177/0278364919863090.
[34] Technical University of Denmark. *HPC - Technical University of Denmark: CUDA Setting for TensorFlow 2.x*. https://www.hpc.dtu.dk/?page_id=3755. Accessed: 2023-12-23. 2023.
[35] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.
[36] Mary B. Alatise and Gerhard P. Hancke. "Pose Estimation of a Mobile Robot Based on Fusion of IMU Data and Vision Data Using an Extended Kalman Filter". In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: 10.3390/s17102164. URL: https://www.mdpi.com/1424-8220/17/10/2164.

A Title

A.1 More Trajectory Estimation

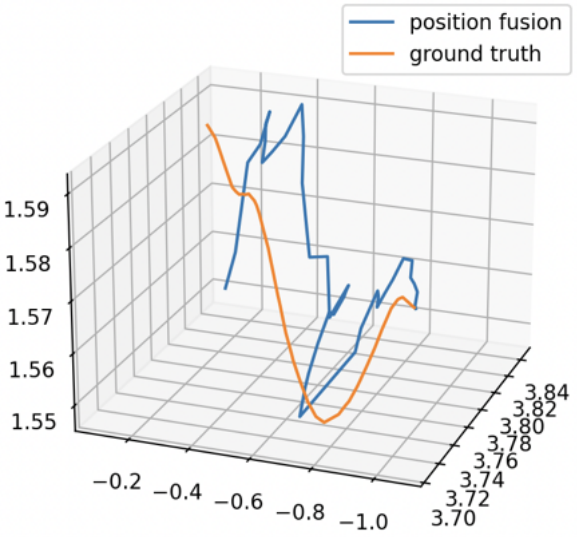

Figure A.1: Trajectory of Sensor Fusion

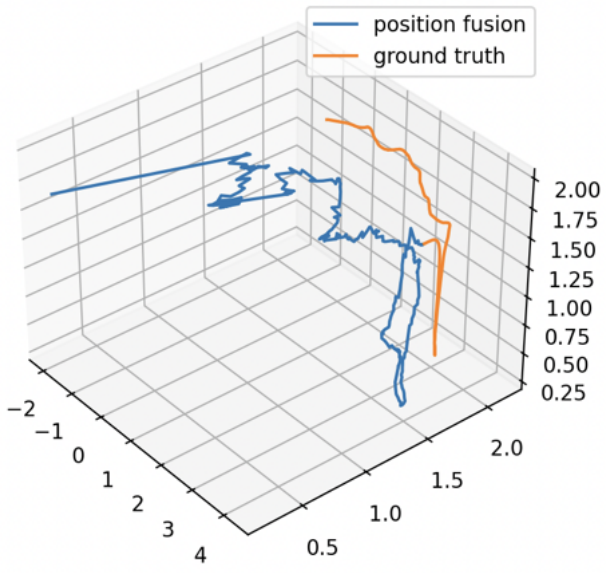

Figure A.2: Trajectory of Sensor Fusion

A.2 3D Motion Capture System

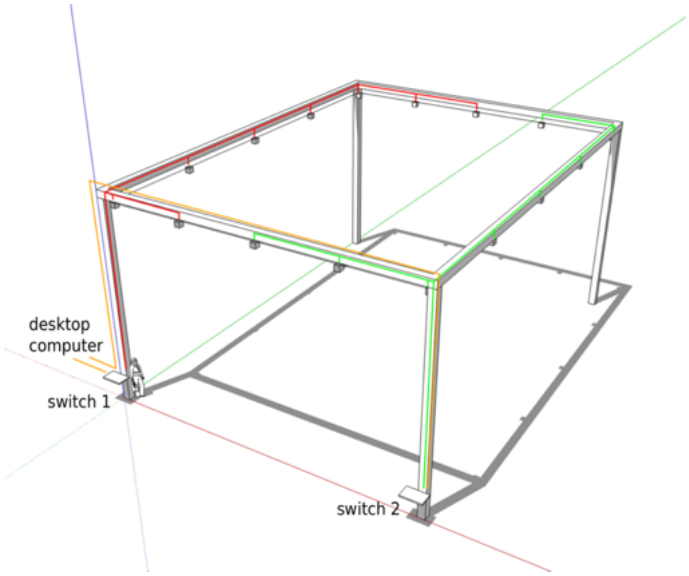

Figure A.3: 3D Motion Capture System.

Figure A.4: Markers for System Detection.

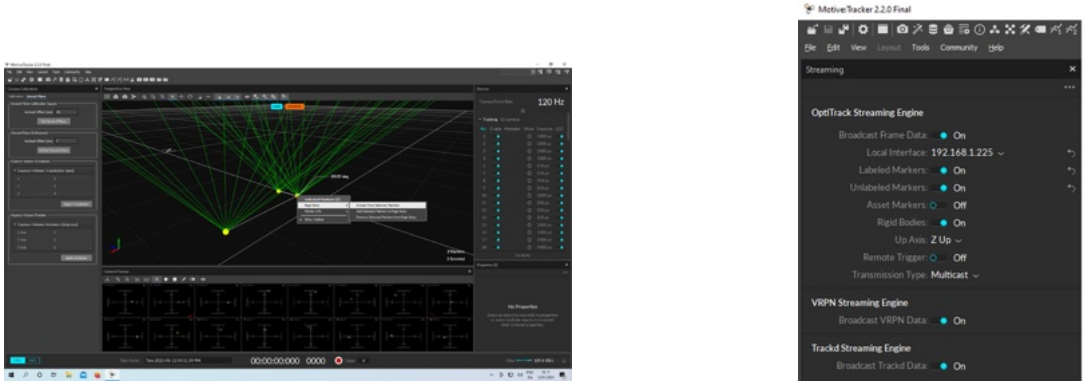

Figure A.5: Creating Rigid Body and Enabling Steaming

Technical
University of
Denmark

Ørsteds Plads, Bygning 348
2800 Kgs. Lyngby
Tlf. 4525 3800

www.electro.dtu.dk