UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA MAGISTRALE IN
Computer Engineering

"Dataset of Panoramic Images for People Tracking in Service Robotics"

Relatore: Prof. Emanuele Menegatti

Laureando/a: Sepideh Shamsizadeh

Correlatore: Prof. Alberto Pretto
Dott. Alberto Bacchin

ANNO ACCADEMICO 2022 –2023

Data di laurea 13/07/2023

"Women, Life, Freedom"
— Mahsa Amini

Abstract

Panoramic cameras have gained popularity in service robotics for capturing high-quality images that provide a comprehensive 360 degree view of a robot's surroundings. These cameras offer significant advantages for tracking people and facilitating people-guiding functionalities in autonomous mobile robots. However, the lack of datasets specifically designed for people tracking in panoramic videos, in particular including the accurate localization of individuals in the real world, poses a significant challenge to train and evaluate algorithms. This thesis proposes a framework that introduces an innovative approach to autolabeling panoramic videos, annotating the precise positioning of each person within the world frame, rather than solely in the image frame, without the need for human intervention. To achieve this, the framework utilizes a technique for joint panoramic camera and laser rangefinder calibration a laser rangefinder to extract a homography matrix. This matrix is then employed to fuse the people detection from 2D range data and panoramic images, enhancing the overall detection accuracy. By leveraging image detection, the proposed approach demonstrates significant improvements in the detection capabilities of 2D range data. The experimental results have provided compelling evidence supporting the effectiveness of the proposed framework. By applying this framework, a dataset of panoramic images for people tracking in service robotics can be generated. This dataset would serve as a valuable resource for training and evaluating people tracking algorithms in panoramic video settings. Additionally, the availability of such a dataset would facilitate the advancement of people-guiding functionalities in autonomous mobile robots, leading to more reliable and efficient human-robot interaction in various service scenarios.

Contents

Listing of figures

Listing of tables

Listing of acronyms

A

AR Augmented Reality

C

Calthec California Institute of Technology

CNN Convolutional Neural Network

COCO Common Objects in Context

D

DETR DEtection TRansformer

DLA Deep Layer Aggregation

DR-SPAAM Distance Robust SPatial Attention and Auto-regressive Model

DSA Dynamic Scale Adjustment

DukeMTMC Duke Multi-Target, Multi-Camera

F

FAIR Facebook AI Research

FN False Negative

FP False Positive

FPN Feature Pyramid Network

G

GNN Global Nearest Neighbor

H

HD high-definition

HOG Histograms of Oriented Gradients

I

IoU Intersection over Union

K

KITTI Karlsruhe Institute of Technology and Toyota Technological Institute

L

LiDAR Light Detection and Ranging

M

ML Mostly Lost Targets

MOT Multi Object Tracking

MOTA Multiple Object Tracking Accuracy

MRE Mean Reprojection Error

MT Mostly Tracked Targets

R

R-CNN Region-based Convolutional Neural Network

RANSAC Random Sample Consensus

re-ID Re-Identification

ResNet Residual Network

RFA Residual Feature Aggregation

ROI Region of Interest

ROS Robot Operating System

RPN Region Proposal Network

S

SIFT Scale-Invariant Feature Transform

SPP-net Spatial Pyramid Pooling Network

SSD Single Shot Detector

SURF Speeded Up Robust Feature

SVD Singular Value Decomposition

SVM Support Vector Machine

Swin Swin Transformer

T

TP True Positive

U

UKF The Unscented Kalman Filter

V

VGG Visual Geometry Group

ViT Vision Transformer

VR Virtual Reality

Y

YOLO You Only Look Once

1 Introduction

1.1 Motivation

Assistance robots, also known as social robots, are becoming increasingly popular in various fields because they can interact with humans, showing behaviors and emotions similar to humans. These robots are effective in tasks such as providing companionship, assisting with daily activities, and supporting therapy and education. For example, in healthcare settings like hospitals, clinics, and medical centers, patient-guide robots can help patients navigate the facility and find different departments or specialized areas like examination rooms or diagnostic imaging facilities.

To enhance the effectiveness of a robot's guidance capabilities, the integration of various components is essential, such as sensors (e.g., cameras, lasers) and computer vision technology. These components enable the robot to detect, track, and re-identify individuals, ensuring it can accurately locate and monitor them as they move within its surroundings. Omnidirectional cameras have emerged as a popular choice for capturing high-quality images of the robot's environment due to their ability to provide a 360 degree field of view. This comprehensive view ensures that individuals cannot escape the camera's sight, resulting in more reliable and safer tracking, thus enabling more robust people-guiding functionality in autonomous mobile robots. Nonetheless, despite the potential of omnidirectional cameras, people detection and tracking systems utilizing these cameras are still in the early stages of development, and further

research is required to improve their performance.

To enhance the performance of service-guide robots in indoor facilities, it is crucial to develop accurate and reliable people detection and tracking systems that utilize omnidirectional cameras. However, the creation of a dataset for training and evaluating such systems presents several challenges. One significant challenge is the laborious and time-consuming nature of manual labeling. Annotating the positions of people in each frame of the dataset requires significant human effort, which becomes impractical when dealing with large-scale datasets. Moreover, many available datasets solely provide people's positions in the image frames, often in the form of bounding boxes. While this information is useful for image-based tasks, it does not capture the real-world positions of individuals, which is vital for robot navigation and interaction.

To address these challenges, we propose an autolabeling framework that automates the labeling process and enables the creation of a large-scale dataset with reduced manual effort. The framework leverages computer vision techniques and laser-based detectors to automatically detect and track people in omnidirectional video footage. By accurately estimating the real-world positions of individuals and considering occlusions caused by obstacles or other people, the autolabeling framework generates annotations that reflect the actual spatial locations of people within the environment. By providing an automated labeling framework, we aim to contribute to the development of more accurate and reliable people detection and tracking systems using omnidirectional cameras. The resulting dataset will serve as a valuable resource for evaluating and training algorithms, facilitating advancements in robot navigation, human-robot interaction, and other related areas. Ultimately, our goal is to improve the capabilities of service-guide robots and enhance their effectiveness in assisting and interacting with individuals in indoor settings.

1.2 Research Challenge

The task of automatically labeling panoramic frames captured with an omnidirectional camera, showcasing the real-world positions of people in each frame, presents several significant research challenges. One key challenge involves fusing data from laser sensors with the panoramic camera to achieve more accurate and robust annotations. The fusion of these two modalities can provide complementary information, enhancing the precision of people tracking in challenging environments.

Another challenge pertains to effectively handling noise and uncertainties in detection and

tracking algorithms. The panoramic nature of the camera introduces distortions and occlusions that can impact the accuracy of people detection and tracking. Robust algorithms need to be developed to handle these challenges, considering factors such as partial occlusions, variations in lighting conditions, and crowded scenes.

Addressing these research challenges requires the development of novel methodologies and algorithms that combine data from panoramic cameras and laser sensors, effectively handle noise and uncertainties, and incorporate contextual information. By overcoming these challenges, we can create a tool that automates the labeling process for panoramic images, facilitating the creation of large-scale datasets for evaluating and training panoramic people tracking systems. This, in turn, will contribute to the advancement of research in the field, leading to more accurate and reliable algorithms for real-world applications.

1.3 Contributions

This thesis makes significant contributions to the field of autolabeling frameworks and dataset creation for people tracking in panoramic images. The research encompasses several key contributions.

The first contribution involves gathering data for people tracking using an omnidirectional camera and a laser rangefinder. By combining these two technologies, we can leverage a comprehensive view of the surroundings can be captured to increase the performance. To do so, we built upon an existing calibration technique, extending it to the usage with panoramic images. This calibration process extracts a homography matrix, enabling the accurate transformation of points from the robot plane to the image plane. Consequently, we were able to filter the noisy laser detection using a visual people detector, achieving higher accuracy in detection.

To ensure precise and continuous tracking of people, a tracking algorithm must be applied. Therefore, as a second contribution, we develop a data association algorithm to map the detection in meaningful tracks, taking into account the challenges introduced by a 360 degree field of view.

Finally, we provide an initial validation of the effectiveness of the proposed framework by collecting a small dataset of indoor panoramic videos with a mobile robot equipped with laser rangefinders. The evaluation phase highlights the potentialities of the proposed ideas and suggests many useful developing direction to further improve the proposed framework.

1.4 Chapter Outline

This thesis investigates the utilization of omnidirectional cameras for people detection and tracking in indoor environments. The chapters are structured as follows:

Chapter 2: Background and Related Work

Chapter 2 provides a comprehensive overview of existing approaches and methodologies in the field of people detection and tracking using omnidirectional cameras. We review relevant literature, highlight key studies, and discuss the gaps and limitations in the current research. This chapter sets the foundation for the development and evaluation of person detection and tracking systems in indoor environments.

Chapter 3: Methodology

In Chapter 3, we present the methodology employed in this research. We detail the technical aspects, procedures, and tools used for creating the automatic annotation tool.

Chapter 4: Evaluation and Experimental Results

Chapter 4 delves into the evaluation metrics, and experimental results. We explain the criteria and measures employed to determine the accuracy, reliability, and efficiency of the algorithms. Additionally, we present the experimental results obtained from applying the developed methodology to the dataset of panoramic images.

Chapter 5: Conclusion and Future Works

Chapter 5 serves as the concluding chapter of the thesis. We summarize the key findings, discuss the implications of the research, and provide a comprehensive conclusion. Additionally, we highlight the limitations of the study and suggest future research directions to further advance the field of people detection and tracking using omnidirectional cameras in indoor environments.

Finally, we will conclude this introduction chapter, which has provided a clear overview of the motivation, research challenges, and contributions of this master's thesis. By addressing the need for accurate people detection and tracking systems using omnidirectional cameras in indoor environments, and by developing an innovative method, we aim to create a dataset of panoramic images for people tracking labeling the precise positioning of each person within the world frame.

2

Background and Related Work

In this chapter, we will first discuss the motivation behind creating a new dataset for tracking people using panoramic video and labeling their positions in the world frame. We will compare existing people tracking datasets and highlight the limitations that have inspired us to develop a new dataset that addresses these shortcomings. Next, we will provide a comprehensive definition of video labeling and explore different methods for labeling objects and people in videos. We will discuss manual annotation techniques, semi-automatic approaches, and the use of deep learning algorithms for automated video labeling. Additionally, we will examine popular frameworks and algorithms for object and people detection, highlighting their strengths and limitations in various tracking scenarios.

Following that, we will delve into the topic of people tracking. We will explain the concept of tracking individuals over time in a video sequence and discuss different approaches and algorithms used in people tracking. This will include methods based on Kalman filters, particle filters, deep learning-based trackers, and graph-based approaches. We will explore their advantages, challenges, and suitability for different tracking scenarios. Subsequently, we will discuss camera and laser calibration techniques. These techniques are crucial for aligning the coordinate systems of cameras and laser sensors, enabling accurate fusion of data from both modalities. We will explain the importance of calibration in the context of people tracking and discuss popular calibration methods used in the field. Finally, we will provide an overview of omnidirectional people-tracking frameworks [17], which form the focus of the approach developed in this thesis. We will explain how these frameworks, capturing a wide panoramic view,

offer distinct advantages for tracking individuals in complex environments. We will highlight the benefits of using omnidirectional cameras and discuss how these frameworks contribute to improved tracking accuracy and reliability.

2.1 Datasets for Real-World People Tracking: Position in Image Frame

People tracking datasets are curated collections of video sequences or images that are specifically created or annotated to facilitate the development and evaluation of algorithms and systems for tracking individuals in various scenarios. These datasets typically provide ground truth annotations, such as bounding boxes or pixel-level labels, that indicate the location and identity of people in the frames. They serve as benchmarks for evaluating the performance of different tracking methods and enable researchers to compare and advance the state-of-the-art in people tracking.

2.1.1 COCO Dataset

The Common Objects in Context (COCO) dataset [18] is a widely used benchmark dataset for object detection, segmentation, and captioning tasks in computer vision research. It was created to provide a diverse and challenging set of images that represent common objects found in everyday scenes. The COCO dataset contains over 200,000 labeled images, making it one of the largest and most comprehensive datasets for object recognition tasks. These images cover a wide range of categories, including people, animals, vehicles, household items, and more. Each image is labeled with object annotations that indicate the presence, location, and category of various objects within the image [18]. The object annotations in the COCO dataset are detailed and precise, providing bounding boxes that tightly enclose each object. Additionally, the dataset includes segmentation masks for object instances, enabling more fine-grained analysis and pixel-level understanding. This makes the COCO dataset particularly suitable for tasks like instance segmentation, where the goal is to identify and delineate individual object instances within an image. In addition to object annotations, the COCO dataset also includes captions for a subset of the images. These captions describe the content of the image in natural language, providing textual descriptions that can be used for tasks like image captioning or multimodal learning [18].

The COCO dataset has become a standard benchmark for evaluating the performance of object detection, segmentation, and captioning algorithms. It has spurred significant advancements in computer vision research and has facilitated the development of state-of-the-art models in these areas. The availability of such a large and diverse dataset has allowed researchers to train models that generalize well across different object categories and image contexts. Researchers and practitioners often use the COCO dataset to train and evaluate their models, comparing their results to those of other algorithms on a common and standardized dataset. This ensures fair and reliable comparisons of different approaches and promotes the advancement of the field. Overall, the COCO dataset has played a crucial role in advancing object recognition and understanding in computer vision and has become an invaluable resource for the research community [18].

2.1.2 MOT Challenge Datasets

The Multi Object Tracking (MOT) Challenge datasets are widely recognized benchmarks for evaluating multi-object tracking algorithms. The MOT Challenge has released several datasets over the years, including MOT15, MOT16, MOT17, and MOT20. These datasets are commonly used in the research community for assessing the performance of different tracking methods. Here is an overview of the MOT Challenge datasets:

- **MOT15:** The MOT15 [19] dataset consists of 22 sequences recorded from different camera views in outdoor scenes. It includes a wide range of challenges such as occlusions, crowded scenarios, and complex motion patterns. The annotations provide bounding boxes and identity labels for pedestrians.
- **MOT16:** Building upon the MOT15 dataset, the MOT16 [20] dataset offers additional sequences and introduces more challenging scenarios. It includes a total of 14 training sequences and 11 test sequences, providing annotations for pedestrians.
- **MOT17:** The MOT17 [20] dataset is an extension of the MOT16 dataset. It features improved annotations, including manual annotation refinement for better accuracy. The dataset covers a diverse set of challenges, including occlusions, scale variations, and crowded scenes.
- **MOT20:** The MOT20 [21] dataset is an updated version of the MOT17 dataset, primarily focusing on pedestrian tracking. It contains a collection of challenging video sequences recorded from different camera views, with high-density pedestrian scenarios. The dataset includes annotations for bounding boxes, visibility, and identity information.

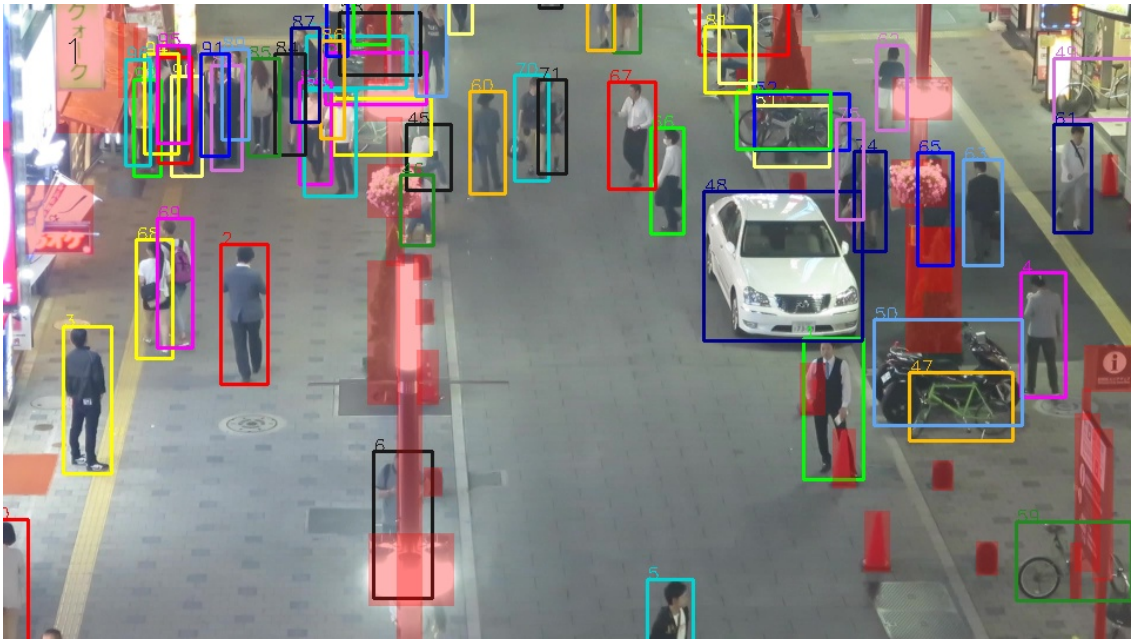

Figure 2.1: Sample data for MOT20 dataset [1].

These MOT Challenge datasets are valuable resources for benchmarking and comparing different multi-object tracking algorithms. They enable researchers to assess the performance of their methods on real-world video sequences with varying levels of difficulty. The datasets are carefully annotated, providing ground truth information about pedestrian positions and identities, a sample of it illustrated in Figure 2.1, allowing for comprehensive evaluation and analysis of tracking algorithms. Researchers often use the MOT Challenge datasets to demonstrate the effectiveness of their tracking approaches and to compare their results against state-of-the-art methods. The datasets serve as a common platform for evaluating the advancements in multi-object tracking algorithms, fostering healthy competition, and driving progress in the field [1].

2.1.3 CrowdHuman Dataset

The CrowdHuman [2] Dataset is a comprehensive dataset specifically designed for pedestrian analysis in crowded scenes. It focuses on human-centric analysis, providing annotations for pedestrian detection, segmentation, and keypoint estimation. The dataset contains a large number of high-resolution images captured in various crowded scenarios, such as shopping malls, concerts, and public events. It offers a wide range of challenges, including occlusions,

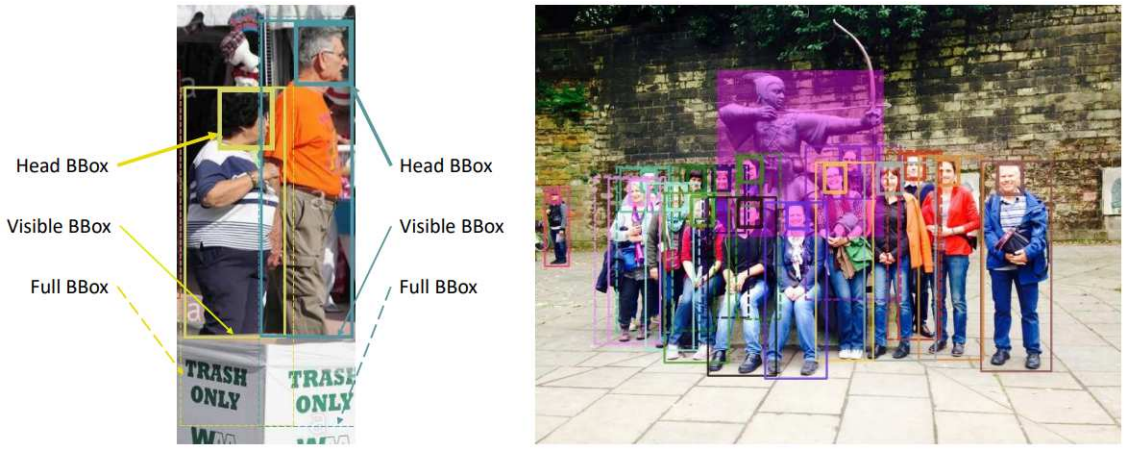

Figure 2.2: An illustrative example of Crowdhuman's three kinds of annotations: Head Bounding-Box, Visible Bounding-Box, and Full Bounding-Box [2].

scale variations, and different viewing angles, making it suitable for developing and evaluating algorithms for real-world pedestrian tracking. With detailed annotations as depicted in Figure 2.2 for bounding boxes, segmentation masks, and keypoints, the CrowdHuman Dataset enables researchers to tackle complex tasks related to people tracking and behavior analysis in crowded environments.

2.1.4 CityPersons Dataset

The CityPersons [22] dataset is a widely used benchmark dataset for pedestrian detection in urban street scenes. It was introduced to address the limitations of existing pedestrian datasets by providing more challenging scenarios encountered in real-world urban environments. The dataset contains high-resolution images captured from diverse urban scenes, including busy streets, crowded squares, and public spaces. It encompasses a wide range of pedestrian sizes, orientations, and occlusions, making it suitable for evaluating the performance of pedestrian detection algorithms under realistic conditions.

One important aspect of the CityPersons dataset is its comprehensive annotation. Each image is densely annotated with bounding boxes around pedestrian instances, providing precise localization information. The annotations include detailed attributes such as occlusion level, body orientation, and visibility, allowing for fine-grained analysis of pedestrian detection algorithms. The dataset consists of over 5,000 training images and 5,000 testing images, with a total of more than 35,000 annotated pedestrian instances. It also provides a set of evaluation metrics,

such as log-average miss rate and average precision, to quantify the performance of pedestrian detectors accurately. The CityPersons dataset has become a standard benchmark in the field, facilitating the development and evaluation of advanced pedestrian detection algorithms for urban environments [22].

2.1.5 Caltech Pedestrian Dataset

The California Institute of Technology (Calthec) [23] Pedestrian Dataset is a renowned benchmark dataset extensively utilized for pedestrian detection research in computer vision. It was collected by the Calthec Vision Lab and consists of approximately 10 hours of high-resolution video footage captured from a moving vehicle in urban environments. The dataset provides a challenging and realistic setting for pedestrian detection algorithms, with complex scenarios including occlusions, varying viewpoints, and different weather and lighting conditions. It has been instrumental in advancing research on pedestrian detection and has been a benchmark for evaluating the performance of algorithms in real-world situations.

The dataset contains a total of about 250,000 frames, with approximately 350,000 annotated pedestrian instances. The annotations provide bounding box coordinates for each detected pedestrian in the video frames. The dataset also includes detailed information such as occlusion levels, truncation levels, and pedestrian visibility. Additionally, it provides a training and testing split, allowing researchers to develop and evaluate their pedestrian detection algorithms effectively. The Calthec Pedestrian Dataset has been widely used in computer vision research and has contributed to the development of state-of-the-art pedestrian detection algorithms [23].

2.1.6 Comparison of People Tracking Datasets

When comparing people tracking datasets, several factors can be considered:

- **Scenarios and Environments:** Different datasets cover a wide range of scenarios, including outdoor environments, urban streets, surveillance settings, and more. Researchers should select datasets that align with the target application or research focus.

- **Annotations:** The level of annotations varies across datasets. Some datasets provide only bounding box annotations, while others offer pixel-level labels or even more detailed annotations, such as keypoint locations or identity information. The choice depends on the specific tracking task and the desired level of detail.

- **Challenges:** Datasets may include specific challenges, such as occlusions, scale variations, crowded scenes, or fast motion. Evaluating algorithms on datasets with diverse challenges helps assess their robustness and generalization capabilities.
- **Dataset Size:** The size of the dataset, in terms of the number of video sequences or images, is an important consideration. Larger datasets provide a more comprehensive evaluation and allow for more reliable performance analysis.
- **Community Adoption:** The popularity and widespread use of a dataset within the research community can also be a factor to consider. Datasets that have been extensively used and have established benchmarks often receive more attention and comparison from researchers.

2.2 Datasets for Real-World Pedestrian Tracking: Position in World Frame

When it comes to real-world people tracking, having datasets that provide the position of individuals in the world frame is crucial. These datasets play a vital role in the development and evaluation of algorithms for various applications such as autonomous driving, surveillance, and human-computer interaction. By offering accurate and precise information about the position of people in the world coordinate system, these datasets enable researchers to train and test tracking algorithms in realistic and complex scenarios. With access to such datasets, researchers can advance the field of people tracking by developing robust solutions that can handle occlusions, crowd scenarios, and varying environmental conditions. The availability of datasets with position information in the world frame contributes to the progress of computer vision and tracking research, helping to create safer and more efficient systems in real-world settings.

2.2.1 KITTI Dataset

The Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [24] Tracking Dataset is a widely used computer vision dataset specifically designed for evaluating multi-object tracking algorithms in autonomous driving scenarios. It consists of a large collection of real-world video sequences captured from a car-mounted sensor platform in urban environments. The dataset provides a comprehensive benchmark for tracking objects such as cars,

pedestrians, and cyclists, and it includes complex scenarios with occlusions, crowded scenes, and varying lighting conditions. The KITTI Tracking Dataset has played a crucial role in advancing research and development in the field of autonomous driving and has become a standard benchmark for evaluating tracking algorithms.

The dataset contains a total of 21 video sequences, with about 12 minutes of footage captured at a frame rate of 10 frames per second. The video sequences are divided into different categories, including urban, residential, and road scenes, providing a diverse set of scenarios for evaluation. For each frame, the dataset provides accurate 2D and 3D bounding box annotations for all tracked objects, along with additional information such as object velocities and tracking IDs. The KITTI Tracking Dataset has been extensively used for training and evaluating multi-object tracking algorithms, enabling researchers to develop robust solutions for tracking objects in real-world driving scenarios [24].

2.2.2 DukeMTMC Dataset

Duke Multi-Target, Multi-Camera (DukeMTMC) [25] is a prominent dataset designed for multi-camera tracking in outdoor environments. It offers a comprehensive collection of high-resolution videos captured by eight cameras strategically placed in the Duke University campus area. The dataset provides annotations in the form of bounding boxes and unique IDs for pedestrians, enabling the tracking of individuals across multiple camera views. DukeMTMC encompasses diverse scenarios, including crowded areas, occlusions, and varying lighting conditions, presenting challenges for tracking algorithms. One notable feature of DukeMTMC is its emphasis on identity preservation, ensuring consistent IDs for pedestrians across different cameras. This property makes it suitable for evaluating algorithms that aim to maintain accurate and consistent identities of individuals in multi-camera tracking scenarios. The dataset has gained widespread adoption in computer vision and tracking communities, serving as a benchmark for the development and evaluation of multi-camera tracking algorithms.

By utilizing the DukeMTMC dataset, researchers have the opportunity to advance the field of multi-camera tracking by developing and evaluating algorithms that can handle real-world outdoor scenarios. The dataset's realistic settings and rich annotations facilitate the exploration of various tracking challenges, such as crowded scenes and occlusions. The evaluation protocols provided with the dataset allow for fair comparisons against state-of-the-art methods. Overall, DukeMTMC plays a crucial role in the advancement of multi-camera tracking research, fostering the development of robust tracking systems that can operate effectively in complex

outdoor environmentsthe [25].

2.2.3 Waymo Open Dataset

The Waymo [26] Open Dataset is a comprehensive dataset provided by Waymo, a leading autonomous driving technology company. It is designed to facilitate research and development in the field of autonomous driving. The dataset contains a vast collection of high-resolution sensor data, including Light Detection and Ranging (LiDAR) point clouds, high-definition (HD) images, and calibrated sensor data from Waymo self-driving cars. It covers various urban and suburban environments, capturing diverse real-world driving scenarios. The Waymo Open Dataset offers a valuable resource for training and evaluating algorithms in areas such as perception, tracking, and autonomous vehicle control.

The dataset provides rich annotations for a wide range of objects, including pedestrians, cyclists, vehicles, and more. The annotations include accurate 3D bounding box information, allowing for precise object tracking and localization. In addition, the dataset includes data collected under different weather and lighting conditions, making it highly versatile for algorithm development and testing. The Waymo Open Dataset has gained popularity within the research community and has been influential in advancing the state-of-the-art in autonomous driving systems [26].

2.2.4 nuScenes Dataset

The nuScenes [27] Dataset is a large-scale dataset created by nuTonomy [1]. It is designed to support research and development in the field of autonomous driving. The dataset contains a diverse collection of sensor data, including high-resolution images, LiDAR point clouds, radar data, and ego vehicle information. It covers urban driving scenarios and includes data captured in various weather and lighting conditions. The nuScenes Dataset provides a valuable resource for training and evaluating algorithms in areas such as perception, mapping, motion planning, and control for autonomous vehicles.

One notable aspect of the nuScenes Dataset is its comprehensive annotations. It provides detailed 3D bounding box annotations for a wide range of objects, including pedestrians, cyclists, vehicles, and more. The dataset also includes annotations for drivable areas, lanes, and

[1] nuTonomy was a self-driving car startup company that was founded in 2013 by Karl Iagnemma and Emilio Frazzoli, both researchers from the Massachusetts Institute of Technology (MIT).

traffic lights. These annotations enable researchers to develop and evaluate algorithms for various tasks, such as object detection, tracking, and scene understanding. The nuScenes Dataset has gained significant attention within the autonomous driving research community and has been instrumental in advancing the state-of-the-art in autonomous driving technology [27].

2.2.5 ApolloScape

ApolloScape [28] is a large-scale dataset created by the Apollo autonomous driving platform, developed by Baidu. It aims to provide researchers and developers with a comprehensive dataset for advancing the field of autonomous driving. The dataset includes diverse sensor data such as high-resolution images, LiDAR point clouds, and calibrated sensor data captured from different weather and lighting conditions. ApolloScape offers a wide range of scenarios, including urban, suburban, and highway driving, and covers various challenging situations to enable the development and evaluation of robust autonomous driving algorithms. One notable feature of ApolloScape is its detailed annotations. The dataset provides annotations for a variety of objects, including pedestrians, vehicles, cyclists, and traffic signs. The annotations include pixel-level semantic segmentation, instance-level semantic segmentation, and 3D bounding box information. These annotations allow researchers to train and test algorithms for tasks such as object detection, tracking, and scene understanding. ApolloScape has become an important resource in the autonomous driving research community, contributing to advancements in perception, planning, and control systems [28].

Datasets for real-world people tracking with position information in the world frame are particularly valuable when working with omnidirectional cameras. Omnidirectional cameras capture a 360 degree field of view, allowing for a more comprehensive understanding of the surrounding environment. However, analyzing data from such cameras poses challenges in terms of accurately estimating the positions of objects, including people, in the real world. Having datasets specifically tailored for real-world people tracking with positions in the world frame helps address these challenges. By providing precise spatial information, these datasets enable researchers to develop algorithms and models that can accurately track people in omnidirectional camera footage. They facilitate the training and evaluation of algorithms that account for the unique characteristics of omnidirectional camera data, such as distortion and perspective variations.

Moreover, datasets with position information in the world frame for omnidirectional cam-

eras contribute to the advancement of various applications, such as augmented reality, virtual reality, and autonomous navigation systems. They allow researchers to develop robust solutions for these applications by providing a realistic representation of the real world and enabling accurate localization and tracking of people. Overall, datasets with position information in the world frame for omnidirectional cameras play a crucial role in advancing research and development in the field of real-world people tracking with omnidirectional vision, facilitating the creation of more accurate and reliable systems in a variety of domains.

2.3 Dataset of Panoramic Images People Tracking

The dataset of panoramic images people tracking refers to a collection of data specifically designed for the task of tracking people in panoramic images. It is a dataset that focuses on understanding and analyzing the movement and behavior of individuals in immersive and wide-angle visual environments. Panoramic images typically provide a wide field of view, capturing a 360 degree view of the surroundings. This dataset is created by collecting such panoramic images from various sources, such as specialized cameras or sensor systems capable of capturing a full panoramic view.

The primary objective of this dataset is to provide researchers and developers with a resource to train and evaluate algorithms and models for person tracking in panoramic images. Person tracking involves detecting and following individuals as they move within the panoramic scene. It can include tasks like person detection, tracking, Re-Identification (re-ID), pose estimation, and activity recognition. The dataset may include various annotations and labels to facilitate the development of tracking algorithms. These annotations can consist of bounding boxes around people, unique identifiers or labels for individuals, and temporal information about their movements. The dataset may also cover a diverse range of scenarios and environments, including indoor and outdoor settings, crowded areas, public spaces, and specific applications like surveillance or virtual reality. The diversity of the dataset enables the evaluation of tracking algorithms in different challenging conditions

2.3.1 CVIP360 Dataset

The CVIP360 dataset [3] is a freely available dataset designed for testing tracking and distancing algorithms in omnidirectional videos. It consists of 16 real videos captured using a Garmin VIRB 360 omnidirectional device, with a resolution of 3840 × 2160 pixels and a frame rate of

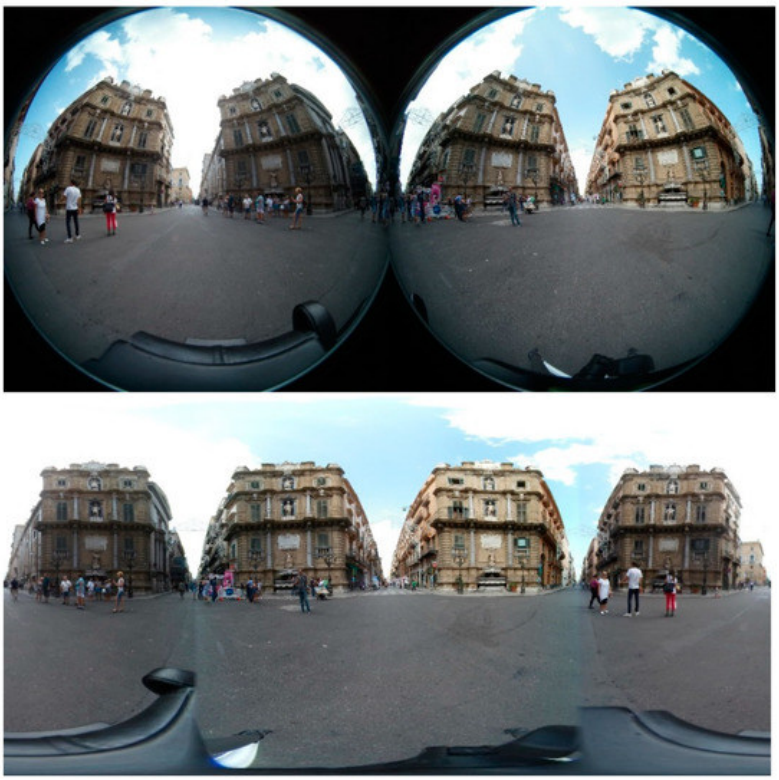

Figure 2.3: Sample data of CVIP360 Dataset [3].

25 frames per second. The dataset includes 10 indoor videos and 6 outdoor videos to cover different lighting and environmental conditions. The dataset contains a total of 17,000 equirectangular frames, which are images representing a 360 degree view of the scene, a sample data illustrated in Figure 2.3. Additionally, it includes over 50,000 annotated bounding boxes of pedestrians. The annotations were performed using two strategies[3]: Pedestrian Location Annotation: Manual labeling was done to mark the locations of pedestrians in the equirectangular frames using bounding boxes. This information helps in tracking individuals in the videos. Distance Annotation: Each pixel in the equirectangular frames was assigned a "real" distance to the camera's location. To achieve this, markers were placed on the ground at fixed and known distances from the camera before capturing the videos. The marker positions were used to establish the relationship between pixel coordinates in the equirectangular frames and the corresponding distances. These distance annotations enable distance estimation and distancing algorithms.

For outdoor videos, markers were placed at various distances and angles to cover the entire

360 degree view. In the case of indoor videos, markers were placed along four directions to account for distortions caused by misalignments. The relationship between pixel coordinates and distances was estimated using piecewise linear interpolation. The CVIP360 dataset is the first of its kind to provide omnidirectional videos of indoor and outdoor scenes with depth information and annotations of moving pedestrians. It serves as a valuable resource for developing and evaluating algorithms related to distancing, tracking, and video surveillance applications in omnidirectional video environments [3].

2.3.2 JRDB-Pose Dataset

The JRDB-Pose dataset [4] is a comprehensive collection of annotated panoramic frames for human pose estimation and tracking. It is derived from the JRDB dataset, comprising 54 sequences captured in both indoor and outdoor locations within a university campus. The dataset contains 57,687 panoramic frames, each with annotations for 636,000 pose instances and 11 million labeled keypoints. These annotations cover 17 keypoints for each body pose,as shown in Figure 2.4, including head, eyes, neck, shoulders, elbows, wrists, hips, knees, and ankles. Notably, JRDB-Pose introduces occlusion labels for each keypoint, providing valuable information for understanding and improving pose estimation under occlusion. The annotations in JRDB-Pose are temporally consistent, allowing for reliable tracking of individuals even during periods of full occlusion. The dataset includes tracking IDs that align with existing 2D and 3D bounding box annotations from JRDB, ensuring consistency across different annotation types. Additionally, the dataset merges annotations from five camera views of the 360 degree cylindrical video stream into a single panoramic image. However, it is important to note that not all visible individuals are labeled, especially those situated far away from the robot [4]. JRDB-Pose offers a diverse range of pose scales and distributions, varying across different scenes and reflecting a wide range of human motions and activities. This variation provides researchers with a realistic dataset to develop and evaluate pose estimation algorithms for different scale scenarios. The dataset also integrates with annotations from the JRDB and JRDB-Act datasets, making it a valuable resource for multi-task learning in human detection/-tracking, pose estimation/tracking, individual action recognition, social group analysis, and social activity detection. Furthermore, the dataset's occlusion labels enable researchers to study and improve pose estimation methods under challenging occlusion conditions, as it includes a large number of labeled occluded and invisible joints [4].

Figure 2.4: Sample of the annotations in JRDB-Pose [4].

2.4 Introduction to Video Labeling Techniques

Video labeling plays a crucial role in people's detection and tracking tasks within computer vision. It involves assigning labels or annotations to individuals, actions, events, or regions of interest in video data. Accurate labeling is essential for developing and evaluating effective people detection and tracking algorithms. Various video labeling techniques are available, including manual annotation, semi-automatic annotation, and fully automatic annotation [29].

2.4.1 Manual Annotation

Manual annotation is a traditional method for labeling videos, where human annotators carefully analyze the video frames and manually assign labels to individuals or regions of interest. This process requires annotators to identify and track people's positions and movements throughout the video. Manual annotation ensures high-quality and accurate labels, serving as ground truth for training and evaluating people detection and tracking algorithms. However, this approach is time-consuming and relies heavily on the expertise and training of the annotators [29, 30].

2.4.2 Semi-Automatic Annotation

Semi-automatic annotation combines the advantages of manual and automatic approaches by leveraging automated techniques to assist human annotators. These techniques provide tools or algorithms that aid in the labeling process, reducing the annotation time. For people detection and tracking, one example is the "scribble-based" annotation approach. Annotators provide scribbles on individuals of interest, and the annotations are propagated to other frames of the video. This reduces the manual effort required for labeling while maintaining label accuracy [29, 30].

2.4.3 Fully Automatic Annotation

Fully automatic annotation aims to remove human intervention from the labeling process entirely. Computer algorithms automatically detect and label people in video frames without human interaction. Deep learning-based approaches have gained popularity for fully automatic video labeling. These approaches use neural network models trained on large datasets to learn visual features and temporal dynamics, enabling them to detect and track people in videos. However, fully automatic annotation methods may encounter challenges in accurately labeling people due to occlusions, clutter, and motion blur in video data [29, 30].

While manual annotation provides accurate labels, it is time-consuming. Semi-automatic annotation reduces the annotation time but still requires human involvement. Fully automatic annotation offers speed but may not achieve the same level of accuracy as manual or semi-automatic methods, particularly in complex video scenes involving people. To address the research challenge of precise people detection and tracking in panoramic videos, alternative approaches are necessary. The focus of this thesis is to develop innovative techniques that can accurately label panoramic video frames with the real-world positions of people, overcoming the limitations of existing video labeling methods.

2.5 Object Detection

Object detection is a crucial task in computer vision that involves finding specific objects in digital images. This task requires developing computer models and techniques that can accurately locate and classify objects within an image. The effectiveness of these models is measured by their accuracy in identifying objects and their speed in processing images. Object detection

serves as the foundation for various other computer vision tasks, including instance segmentation (dividing an image into regions for each object), image captioning (generating descriptions of objects in an image), and object tracking (following objects as they move). In recent years, the field of object detection has seen significant advancements, thanks to deep learning techniques. These techniques have revolutionized the way objects are detected in images and have led to remarkable progress in the field. Object detection finds applications in various real-world scenarios, such as autonomous driving, robot vision, and video surveillance. These applications greatly benefit from the ability to detect and track objects in real time, allowing them to respond promptly and accurately to changes in their surroundings [31]. As technology continues to evolve, object detection methods are expected to become even more powerful and efficient, enabling a wide range of practical applications in different domains.

2.6 People Detection

Detecting people in images is a challenging task in computer vision due to the diverse ways in which people can look, move, and appear in images. Factors such as articulation, viewpoint, and appearance introduce significant variations. However, the ability to detect and track people is crucial for applications such as robotics, image and video organization, surveillance, and automotive safety. Extensive research has been conducted on visual people detection, resulting in continuous advancements and the development of innovative techniques [32]. People detection is a subfield of object detection, which focuses on locating and identifying objects in digital images or video frames. People detection presents unique challenges due to the wide variability in human appearance, pose, clothing, lighting conditions, and occlusions. Researchers have developed various approaches to address these challenges. Let's explore some of the popular methods for people detection:

2.6.1 Haar Cascades

Haar Cascades, introduced by Viola and Jones [33], is a widely used method for people detection. It relies on a set of pre-trained classifiers to detect the presence of people in an image. These classifiers are created using Haar-like features, which are simple rectangular patterns used to identify edges, lines, and corners in the image. The Haar-like features are combined to form a classifier, which is trained using a machine learning algorithm like Adaboost [34]. One of the advantages of Haar Cascades is its real-time performance, making it suitable for applica-

tions that require fast processing. However, it may struggle with detecting people in complex or cluttered scenes and may be sensitive to variations in lighting conditions.

2.6.2 Histograms of Oriented Gradients

Histograms of Oriented Gradients (HOG) [35] is another popular method for people detection. It involves calculating histograms that capture the direction of gradients in the image and using them to train a classifier specifically designed for detecting people. The HOG features are effective in capturing the shape and texture of the human body, enabling the detection of people even in crowded scenes. HOG-based approaches perform well in different lighting conditions and handle occlusions to some extent. However, they may struggle with detecting people in unusual poses or when there is significant overlap between individuals.

2.6.3 Template Matching

Template matching is a straightforward approach for people detection. It involves comparing a template image of a person with the image being analyzed to determine if there is a match. This method can be effective for detecting people in controlled environments where there is little variation in pose, lighting conditions, and appearance. However, it is less robust to changes in these factors, making it less suitable for more complex or dynamic scenes[36].

2.6.4 Motion Detection

Motion detection is another approach for people detection, particularly in surveillance applications. It involves identifying regions in the image that are in motion and determining if they correspond to a person. This method is often used when the camera is stationary, and people within the scene are moving. While motion detection can be useful for detecting people, it may also detect other moving objects or produce false positives due to environmental factors such as wind-blown foliage or changes in lighting conditions [37].

2.6.5 Deep Learning-based Detectors

Deep Learning, a subfield of Machine Learning, has significantly advanced the performance of people detection algorithms. Deep Learning-based detectors involve training artificial neural networks, particularly Convolutional Neural Networks (CNNs) [38], on large datasets of labeled images to learn the features that are most relevant for detecting people. Once trained, the

network can be applied to detect people in new images. Popular Deep Learning architectures for people detection include Faster Region-based Convolutional Neural Network (R-CNN), You Only Look Once (YOLO), and Single Shot Detector (SSD). These architectures differ in their approach to object detection, but they all utilize CNNs to extract features from the image and make predictions about the location and size of people. Deep Learning-based detectors have shown impressive performance, achieving high accuracy in detecting people in various scenes. However, they often require substantial computational resources and training data, and their performance can be affected by challenges such as occlusions, cluttered backgrounds, varying lighting conditions, and changes in pose and appearance.

While these techniques have made significant progress in people detection, it is important to consider their pros and cons. Haar Cascades and HOG-based methods offer real-time performance and robustness to different lighting conditions but may struggle with complex scenes and unusual poses. Template matching is simple and effective in controlled environments but lacks robustness to changes in factors like lighting and appearance. Motion detection is useful for surveillance applications but may produce false positives and detect other moving objects. Deep Learning-based detectors have achieved high accuracy but require substantial computational resources and training data, and their performance can be impacted by various challenges. Recognizing these limitations motivates further research to develop more robust and efficient people detection methods.

2.7 CNN-based Detectors

In the past decade, deep learning-based techniques have made significant advancements in computer vision. These techniques leverage deep CNNs, which are highly effective in extracting useful information from images. For the task of object detection, CNNs has become the standard method in the research literature. Object detection involves recognizing and localizing objects, and state-of-the-art models can be categorized into two types: two-stage and one-stage detectors [38] shown in Figure 2.5.

2.7.1 Two-stage detectors

Two-stage detectors are a class of object detection models that have been widely used in the field of computer vision. These detectors follow a specific approach to tackle the object detection problem by dividing the process into two distinct stages: region proposal and classification. In

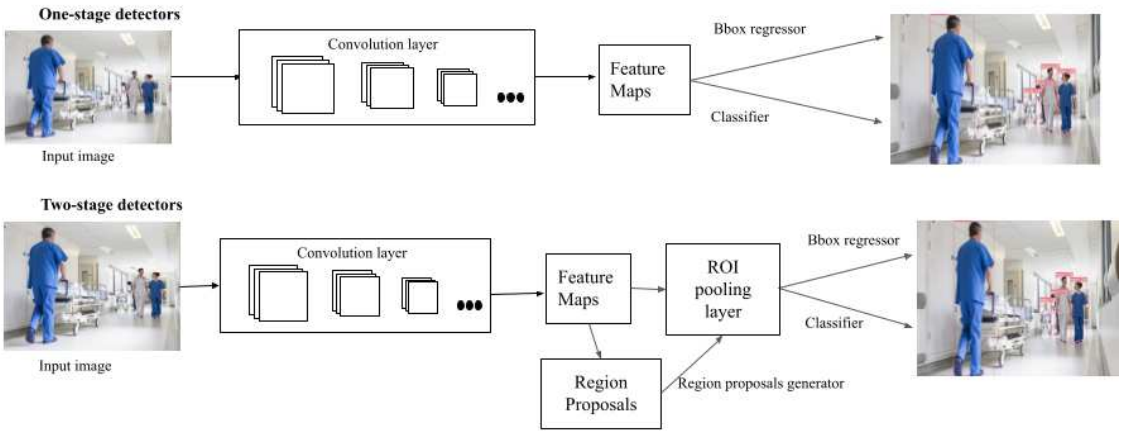

Figure 2.5: Two-stage and One-stage detectors

the region proposal stage, the model generates potential object regions by utilizing reference boxes known as anchors. These anchors represent predefined bounding boxes of various sizes and aspect ratios. By sliding these anchors across the input image, the model aims to identify regions that have a high probability of containing objects. This process helps in narrowing down the search space and focusing on areas of interest [39].

Once the potential regions are proposed, the model proceeds to the classification stage. In this stage, each proposed region is classified into different object categories. The model analyzes the features within these regions and assigns a class label based on the learned representation of each object category. Additionally, the localization of the proposed regions is refined by adjusting the bounding box coordinates to more accurately enclose the objects. The two-stage approach provides a systematic framework for object detection, allowing for more accurate localization and classification of objects in images. By separating the region proposal and classification tasks, these models can efficiently process large datasets and handle complex scenes. However, it is worth noting that the effectiveness and computational cost of two-stage detectors heavily depend on factors such as the choice of anchor design, network architecture, and hyperparameter configurations [39].

R-CNN

R-CNN [5] was a groundbreaking paper that introduced a new approach to object detection using CNNs. It demonstrated the potential of CNNs to significantly improve detection performance. The key idea behind R-CNN was to transform the object detection problem into a two-step process shown in Figure 2.6: region proposal and classification. In the region pro-

posal stage, a mean-subtracted image was passed through a region proposal module, which used a technique called Selective Search [40] to identify potential object regions. These regions were then warped and processed by a CNNs, such as AlexNet [41], to extract feature vectors. Each region's features were then fed into trained class-specific Support Vector Machines (SVMs) to obtain confidence scores. After obtaining the confidence scores, non-maximum suppression was applied to select the most relevant regions based on their overlap and class. Once the class was determined, a trained bounding box regressor predicted the precise location of the object by estimating the center coordinates, width, and height of the bounding box. Although R-CNN brought significant advancements to object detection, it had its drawbacks. Training R-CNN was a complex and time-consuming process. It involved pre-training the CNN with a large clas-

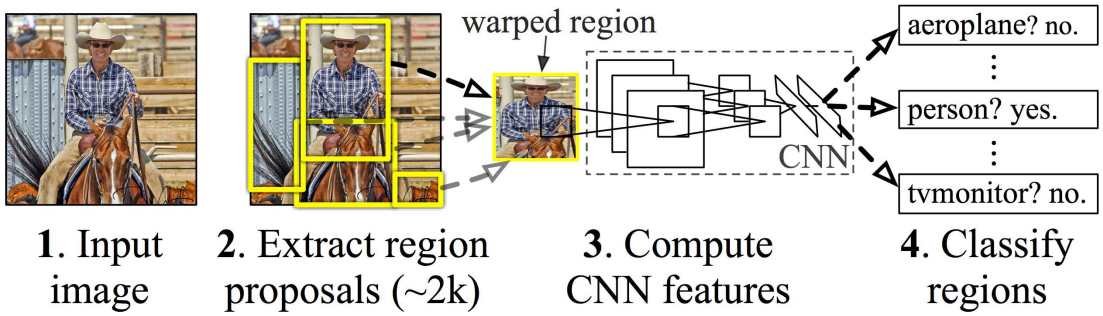

Figure 2.6: R-CNN architecture [5].

sification dataset and then fine-tuning it for detection using domain-specific images. Each class required its own SVM and bounding box regressor. Training R-CNN on small datasets took days, and the algorithm was computationally expensive, taking around 47 seconds per image. It also required substantial storage space due to the large number of proposed regions. Despite these limitations, R-CNN marked a turning point in the field of object detection and inspired further research and development [42].

SPP-net

Also referred to as Spatial Pyramid Pooling Network Spatial Pyramid Pooling Network (SPP-net) [6] is a unique CNN architecture, as depicted in Figure 2.7. Its purpose is to overcome the limitations of traditional CNNs that can only process inputs of fixed sizes. SPP-net introduces a specialized layer called the spatial pyramid pooling layer, which enables the network to handle inputs of varying sizes and generate feature vectors of fixed lengths. This is achieved by dividing the input into sub-regions at multiple scales and independently pooling the features

within each sub-region. Consequently, SPP-net captures detailed spatial information and can effectively adapt to images of different sizes. The advantages of SPP-net are substantial. Firstly, it offers flexibility by efficiently processing images of diverse sizes and aspect ratios, making it suitable for a wide range of datasets. Secondly, the spatial pyramid pooling layer allows the network to capture spatial information at various scales, enabling it to learn meaningful features regardless of the object's size or position in the input image. Similar to the R-CNN model,

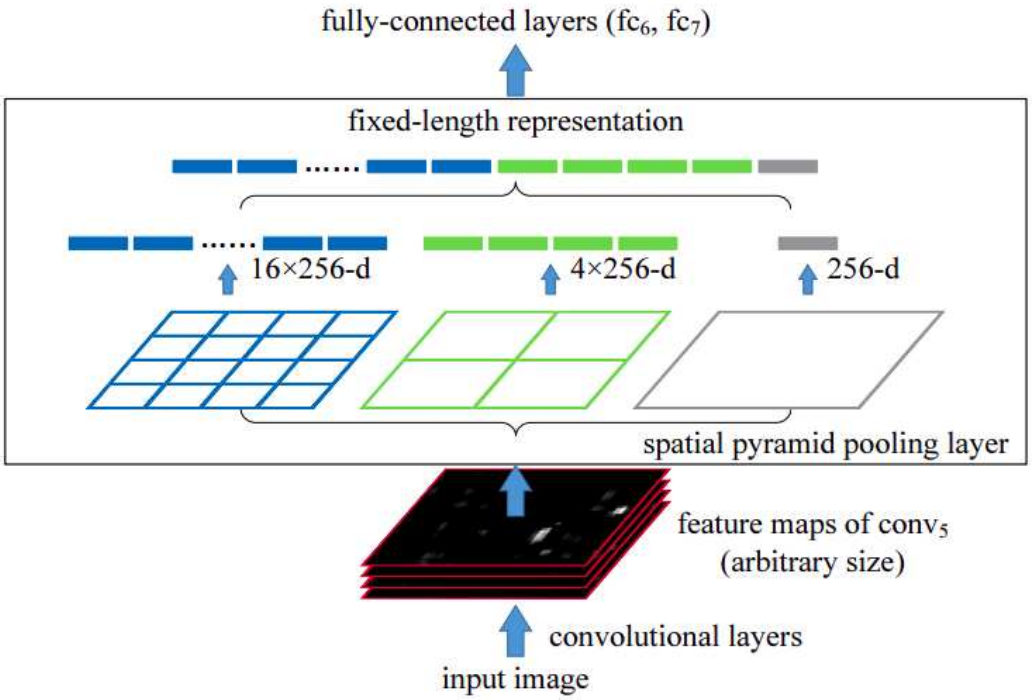

Figure 2.7: SPP-net architecture [6].

SPP-net incorporates a post-processing layer that improves localization through bounding box regression. It also follows a multistage training process, although the fine-tuning specifically targets the fully connected layers [6]. In comparison to the R-CNN model, SPP-net provides notable benefits, including faster processing speed while maintaining comparable accuracy. One significant advantage is its capability to handle images of any shape or aspect ratio, avoiding distortions caused by input warping. However, SPP-net shares some drawbacks with R-CNN, such as the need for multistage training, computational complexity, and longer training time [42].

Fast R-CNN

Fast R-CNN, introduced in [7], is an advancement over the previous R-CNN model that addresses its limitations and significantly improves both speed and accuracy. Unlike R-CNN, which processes each region proposal individually, Fast R-CNN adopts a more efficient approach by sharing convolutional feature computation across the entire image. The model takes the entire image as input and extracts convolutional feature maps using a deep CNN, such as Visual Geometry Group (VGG) or Residual Network (ResNet). These feature maps are then used to generate Region of Interest (ROI) proposals using a Region Proposal Network (RPN). The ROI pooling layer is employed to extract fixed-length feature vectors from each proposal, allowing subsequent layers to perform classification and bounding box regression tasks, as shown in Figure 2.8. Fast R-CNN replaces the need for individual forward passes through the CNN for each proposal in R-CNN, resulting in a significant speed improvement. Fast R-CNN offers several advantages. Firstly, the shared computation of convolutional features across the entire image makes it computationally more efficient than its predecessor. It eliminates the redundant computation of features for overlapping regions and enables end-to-end training of the entire network. Secondly, the ROI pooling layer ensures that the network is able to handle variable-sized proposals and produce fixed-length feature vectors, enabling seamless integration with subsequent layers. Moreover, Fast R-CNN introduces a multi-task loss that combines both classification and bounding box regression losses, facilitating joint optimization of these tasks during training [7].

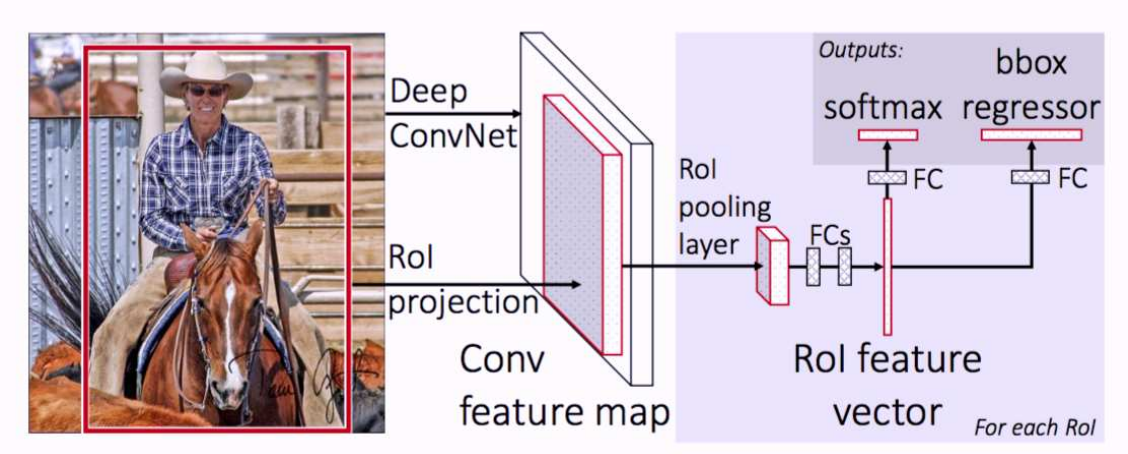

Figure 2.8: Fast R-CNN architecture [7].

Faster R-CNN

Faster R-CNN, introduced in the paper [43], is an enhanced version of the Fast R-CNN framework that improves the efficiency and accuracy of object detection. It addresses the limitation of the region proposal step in Fast R-CNN by introducing RPN that shares convolutional features with the subsequent object detection network. The key difference between Faster R-CNN and Fast R-CNN lies in the approach to generating region proposals. In Fast R-CNN, region proposals are generated using an external method, such as Selective Search, which adds computational overhead. In contrast, Faster R-CNN integrates the RPN directly into the network architecture, allowing for end-to-end training and faster computation.

The RPN in Faster R-CNN operates by sliding a set of anchor boxes over the convolutional feature maps and predicting whether each anchor contains an object of interest or not. The RPN generates region proposals based on these predictions, which are then refined by the subsequent object detection network to obtain accurate bounding box localization and object classification. Compared to Fast R-CNN, Faster R-CNN offers several advantages. Firstly, it eliminates the need for external region proposal methods, leading to a significant improvement in speed. Secondly, the shared convolutional features between the RPN and the object detection network enable better feature representation and information sharing. This results in improved accuracy and robustness in detecting objects of various sizes and aspect ratios[43].

DetectoRS

DetectoRS is an advanced object detection framework introduced in the research paper [44]. It builds upon the Faster R-CNN architecture and incorporates several novel components to enhance detection accuracy and robustness. One key difference between DetectoRS and Faster R-CNN is the introduction of a cascaded structure in DetectoRS. While Faster R-CNN has two stages (region proposal and classification), DetectoRS adds an additional refinement stage. This cascaded architecture allows DetectoRS to iteratively improve the localization and classification of objects, resulting in higher accuracy. Another notable feature of DetectoRS is the Residual Feature Aggregation (Residual Feature Aggregation (RFA)) module. This module aggregates features from different levels of the network, capturing multi-scale contextual information. By integrating features from various scales, DetectoRS gains a better understanding of object context and relationships, leading to improved detection performance.

DetectoRS also introduces the Dynamic Scale Adjustment (DSA) module, which adaptively adjusts the anchor scales based on the distribution of object sizes in the training dataset.

This enables the model to handle objects of different scales effectively and improves detection accuracy across a wide range of object sizes. Furthermore, DetectoRS incorporates an innovative training strategy called Intersection over Union (IoU)-balanced sampling. This strategy addresses the issue of imbalanced positive and negative samples during training by balancing the selection of samples based on their IoU with ground-truth objects. By giving more attention to challenging cases, DetectoRS improves its ability to localize objects accurately. In comparison to Faster R-CNN, DetectoRS achieves state-of-the-art performance on benchmark datasets and demonstrates significant improvements in accuracy and robustness. The cascaded structure, RFA module, DSA module, and IoU-balanced sampling collectively contribute to its superior performance, making DetectoRS an advanced and effective object detection framework [44].

2.7.2 One-stage detectors

One-stage detectors, as an alternative approach to object detection, operate by performing the detection process in a single step. They employ a unified neural network that simultaneously proposes regions of interest and classifies them. This is achieved by dividing the image into a grid of cells, with each cell being responsible for predicting the presence of an object and estimating the bounding box coordinates associated with it. By employing a grid-based structure, one-stage detectors offer a computationally efficient method for processing the entire image in a single forward pass. The network's predictions for each cell directly provide information about the presence and location of objects, eliminating the need for a separate region proposal stage. This streamlined approach contributes to the speed and simplicity of one-stage detectors. While one-stage detectors have demonstrated impressive performance in various object detection tasks, they do face certain challenges. The simultaneous nature of region proposal and classification can make it more difficult to accurately localize objects, particularly in scenarios where there are numerous overlapping or small objects. However, advancements in network architectures and training techniques have led to the development of more powerful one-stage detectors that offer competitive performance [39].

YOLO

YOLO is a popular one-stage object detection framework that has gained significant attention in the computer vision community. It revolutionized the object detection field by introducing a real-time detection approach that achieves high accuracy while maintaining impressive speed.

The YOLO framework takes an image as input and divides it into a grid as illustrated in Figure 2.9. Each grid cell is responsible for predicting bounding boxes and class probabilities for objects present in that cell. YOLO predicts multiple bounding boxes per grid cell, along with their corresponding confidence scores. This allows for the detection of multiple objects in a single pass through the network. One of the key advantages of YOLO is its speed. By performing detection in a single step, YOLO is able to process images in real time, making it suitable for applications that require fast and efficient object detection. This speed is achieved by leveraging a lightweight network architecture that sacrifices some localization accuracy for improved inference time [8]. However, YOLO does face challenges when it comes to detecting small objects

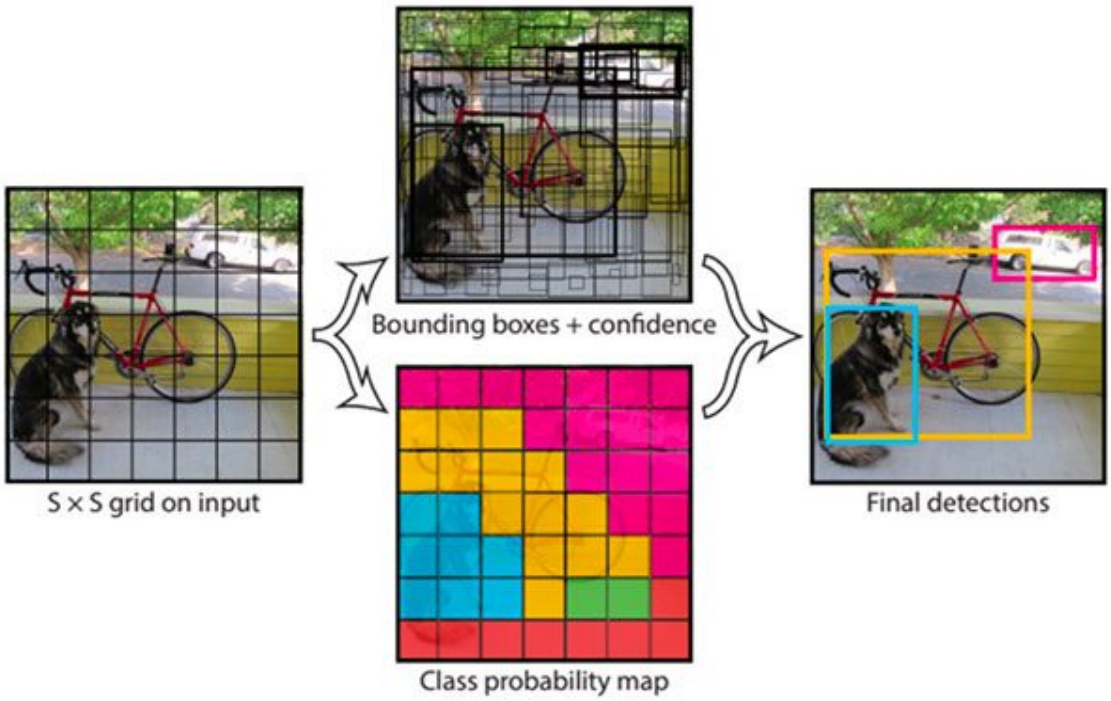

Figure 2.9: A simplified depiction of the pipeline of the YOLO object detector [8].

or objects with complex shapes. Due to its coarse grid structure, YOLO may struggle to accurately localize such objects. Additionally, the fixed grid size limits its ability to handle objects at different scales. Over the years, YOLO has evolved with several versions, including YOLOv2, YOLOv3, and the latest iteration, YOLOv4. These versions introduce various improvements such as better feature extraction, anchor box adjustments, and network architecture enhancements, resulting in improved accuracy and performance. YOLO has been widely adopted in various domains, including surveillance, autonomous driving, and video analysis, where real-

time object detection is crucial. Its combination of speed and reasonable accuracy has made it a popular choice for applications that require efficient and rapid detection of objects in images and videos [45].

YOLOv7

YOLOv7 is an advanced system for detecting objects in real time, which has become very popular in the field of computer vision. The research paper introduces a new way to design the system and make it work faster and more accurately. The researchers want to make the system better than other object detection systems that are currently available. To improve the performance of the system, the researchers came up with some new methods. These methods help the system detect objects more accurately without making them slower. They used tools and techniques that are easy to train and can adapt to different situations. The researchers also solved some problems that came up during their work, like how to replace certain parts of the system and how to assign labels to objects correctly [46].

The YOLOv7 system is designed to work quickly and accurately. It uses resources efficiently and makes good use of the computer's power. The researchers created some new methods, called "extend" and "compound scaling," that help the system work even better. YOLOv7 is faster and more accurate than other object detection systems, especially when working at speeds between 5 and 120 frames per second. It has the highest accuracy among real-time object detection systems that work at 30 frames per second or higher on the GPU V100. The research shows that object detection methods need to keep evolving. The researchers discovered some new problems that need to be solved. They also found ways to make the system better and more accurate. YOLOv7 is an example of how these improvements can lead to better object detection results in real-time applications [46].

SSD

SSD is another widely used one-stage object detection framework that has made significant contributions to the field. It is known for its efficient and accurate detection capabilities, particularly for objects at different scales. The SSD framework follows a similar concept to other one-stage detectors, where it divides the input image into a grid of cells. However, SSD goes beyond the conventional approach by employing a set of pre-defined anchor boxes with different aspect ratios and scales. These anchor boxes serve as references for predicting the location and size of objects within each grid cell. In each grid cell, SSD predicts the presence of objects,

the offsets to adjust the anchor boxes, and the class probabilities for multiple object categories. This multi-scale approach enables SSD to effectively handle objects of various sizes, improving its accuracy in detecting both small and large objects [47].

One of the notable advantages of SSD is its flexibility in capturing object features at different resolutions. It employs feature maps at multiple scales, allowing the network to detect objects at various levels of granularity. This multi-scale feature extraction helps in accurately localizing objects and reducing false positives. SSD has undergone several iterations, including SSD300 and SSD512, with variations in network architectures and anchor box configurations. These iterations aim to strike a balance between accuracy and speed, catering to different application requirements. The SSD framework has achieved remarkable performance across various benchmark datasets, demonstrating its effectiveness in object detection tasks. It provides a robust solution that combines speed, accuracy, and the ability to handle objects at different scales, making it a popular choice in computer vision applications such as robotics, surveillance, and real-time video analysis [47].

RetinaNet

RetinaNet is an advanced object detection algorithm that addresses the challenges faced by one-stage detectors operating on a dense sampling of possible object locations. The algorithm aims to achieve a balance between speed and accuracy, surpassing the performance of existing approaches. A key contribution of RetinaNet is the introduction of a novel loss function called focal loss, which effectively tackles the class imbalance issue encountered during training. By down-weighting the contribution of well-classified examples and focusing on hard examples, the focal loss enables the model to prioritize challenging instances, leading to improved detection accuracy. To evaluate the effectiveness of the focal loss, RetinaNet is designed with a Feature Pyramid Network (FPN) as its backbone. The FPN efficiently constructs a multi-scale feature pyramid from a single input image, allowing RetinaNet to detect objects at different scales effectively [48].

RetinaNet consists of a backbone network and two task-specific subnetworks. The backbone network computes a convolutional feature map, while the subnetworks perform object classification and bounding box regression. This design, tailored for one-stage, dense detection, achieves remarkable accuracy and runtime performance on the challenging COCO dataset. In conclusion, RetinaNet presents a significant advancement in the field of object detection. By addressing the class imbalance problem through the focal loss and leveraging the FPN backbone, RetinaNet demonstrates superior performance compared to previous approaches. Its

ability to strike a balance between speed and accuracy makes it a promising choice for various real-time object detection applications [48].

CenterNet

CenterNet [49] is a deep learning framework for object detection that achieves state-of-the-art performance by directly predicting object centers and bounding box sizes. It eliminates the need for anchor boxes and complex post-processing steps, simplifying the detection pipeline. CenterNet is based on a key insight that object detection can be formulated as a keypoint estimation problem, where the task is to locate the center point of each object and estimate its bounding box size. The CenterNet framework consists of a fully CNN architecture that takes an input image and generates a set of heatmaps representing the object centers. These heatmaps indicate the likelihood of each pixel being the center of an object. Additionally, CenterNet predicts the offset vectors and dimensions of the bounding boxes relative to the object centers. To train CenterNet, ground truth annotations are generated by assigning each object's center to the corresponding heatmap. The network is trained using a keypoint regression loss and a box size regression loss, optimizing the model to accurately predict object centers and bounding box sizes. During inference, the predicted object centers are used to generate the final detection results.

2.7.3 Transformers Based Detectors

Transformers have revolutionized the field of computer vision, including the task of people detection in images. Traditionally, CNNs were widely used for image recognition tasks, but transformers have emerged as a powerful alternative that has achieved remarkable success. Transformers were initially introduced for natural language processing tasks, where they demonstrated exceptional performance in tasks such as machine translation and language generation. However, researchers soon realized that the self-attention mechanism in transformers could be applied to image-understanding tasks as well. The key component of transformers is the self-attention mechanism, which allows the model to capture long-range dependencies within the input sequence. In the context of images, the input is typically divided into a grid of patches, which are then linearized and processed by the transformer model as shown in Figure 2.10. The self-attention mechanism enables the model to focus on relevant regions of the image when making predictions, leading to improved performance in tasks like people detection [9].

In the context of people detection, transformers have demonstrated impressive capabilities.

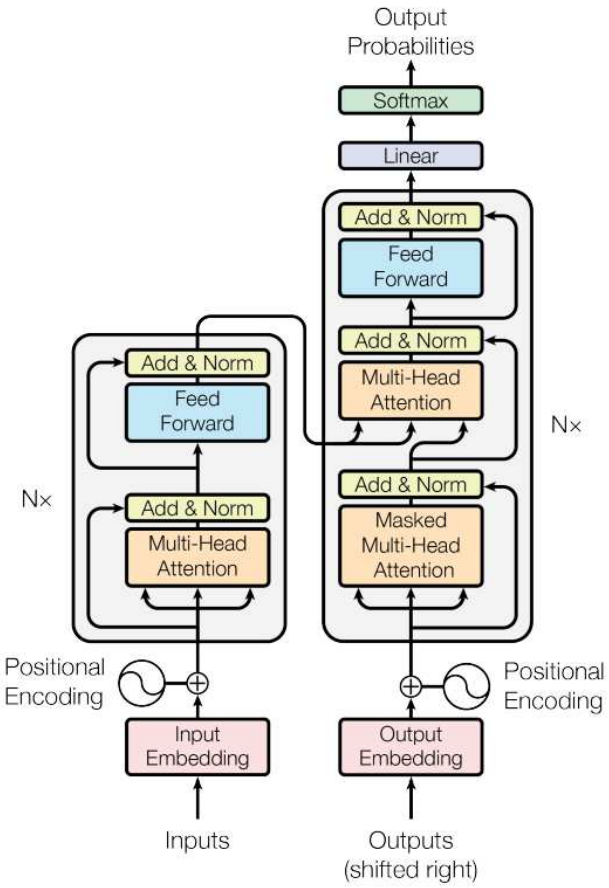

Figure 2.10: The Transformer - model architecture [9].

They can learn to detect people in images by capturing the intricate relationships between different image regions, understanding context, and effectively modeling both local and global dependencies. By considering the entire image as a sequence of patches, transformers can extract meaningful representations of various spatial scales, enabling them to detect people at different resolutions and in various poses. Transformers have also facilitated advancements in other aspects of people detection, such as instance segmentation. By augmenting the detection task with a pixel-level segmentation head, transformers can generate precise masks outlining the exact boundaries of people in images. This enables a fine-grained understanding of object shapes and provides more detailed information for downstream applications. In summary, transformers have significantly impacted the field of people detection in images. Their ability to capture global dependencies, model context, and generate precise predictions has led to substantial improvements in accuracy and efficiency. As research in transformers progresses,

we can expect further advancements and refinements in people detection and other computer vision tasks [50].

DETR

DEtection TRansformer (DETR) is a pioneering object detection framework that introduced the Transformer architecture as the main building block in the object detection pipeline as shown in Figure 2.11. It was proposed by researchers at Facebook AI Research (FAIR) in 2020 and marked a significant departure from traditional object detection approaches that relied heavily on CNN. Traditionally, object detection frameworks used CNNs for both feature extraction and region proposal generation. These frameworks often required additional components, such as region proposal networks (RPNs) or anchor-based methods, to identify potential object locations in an image. This led to complex and multi-stage pipelines. DETR revolutionized object detection by leveraging the Transformer architecture, which was initially popularized in natural language processing tasks. The Transformer architecture is based on the self-attention mechanism that allows the model to capture long-range dependencies between elements in a sequence. In the context of object detection, the input image is divided into a set of fixed-size spatial embeddings called "object queries." [10]

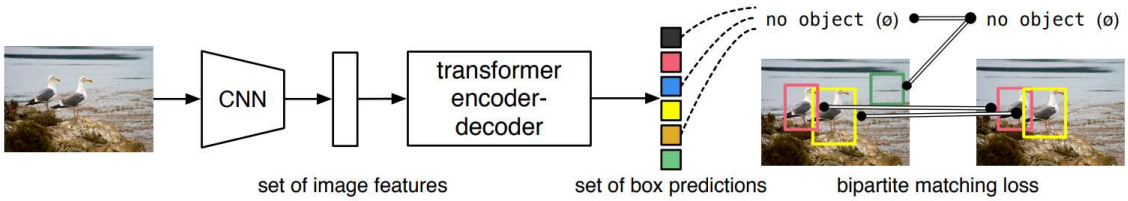

Figure 2.11: Model overview of DETR [10].

In DETR, the image is processed by a CNN backbone to obtain a set of feature maps. These feature maps are then transformed into a sequence of position embeddings, which are combined with the object queries. The resulting sequence is passed through a transformer encoder-decoder architecture. The transformer encoder processes the sequence, enabling the model to capture contextual information and relationships between objects. The transformer decoder takes the encoded sequence and generates predictions for object classes and bounding box coordinates. One of the key advantages of using Transformers in DETR is the ability to handle object detection as a set prediction problem. Unlike anchor-based methods that require

predefined anchor boxes, DETR directly predicts the number of objects and their positions in a single shot, without any handcrafted priors [10].

During training, DETR employs a bipartite matching mechanism that assigns the ground truth objects to the predicted objects based on a Hungarian matching algorithm. This allows the model to learn to associate the correct objects, overcoming the lack of a predefined anchor mechanism. DETR achieved impressive results, demonstrating competitive performance with state-of-the-art object detection frameworks while offering a simpler and end-to-end trainable pipeline. It showcased the effectiveness of using Transformers for dense prediction tasks like object detection and inspired subsequent research on using Transformers in other computer vision tasks. Since the introduction of DETR, numerous variants and extensions have been proposed, further enhancing the capabilities and performance of Transformer-based object detection models [50].

ViT

Vision Transformer (ViT) is a transformer-based model as depicted in Figure 2.12 that was originally designed for image classification tasks, but it has also been adapted for object detection. The ViT architecture replaces the traditional CNN backbone with a transformer encoder, allowing it to capture both local and global contextual information in images. In the context of object detection, ViT has been modified and extended to enable it to handle both classification and localization tasks. The ViT transformer for object detection follows a two-step process: pretraining and finetuning. During pretraining, ViT is trained on a large dataset of images using a self-supervised learning approach. This involves splitting each image into a set of fixed-size patches, which are then linearly embedded into a sequence of vectors. These patches, along with learnable position embeddings, are fed into the transformer encoder [11].

The transformer encoder processes the sequence of patch embeddings, capturing relationships between patches and extracting meaningful representations. However, ViT lacks explicit spatial information, which is crucial for object detection. To address this, positional encodings and relative positional encodings are often utilized to incorporate spatial information into the transformer. After pretraining, the pre-trained ViT model is fine-tuned for the object detection task. This involves adding additional components, such as a detection head, to the model. The detection head typically consists of a set of learnable layers that take the encoded sequence from the transformer and predict the presence, class, and location of objects within the image. To localize objects, the detection head generates bounding box coordinates (e.g., xmin, ymin, xmax, ymax) for each predicted object. These bounding boxes are then refined and adjusted

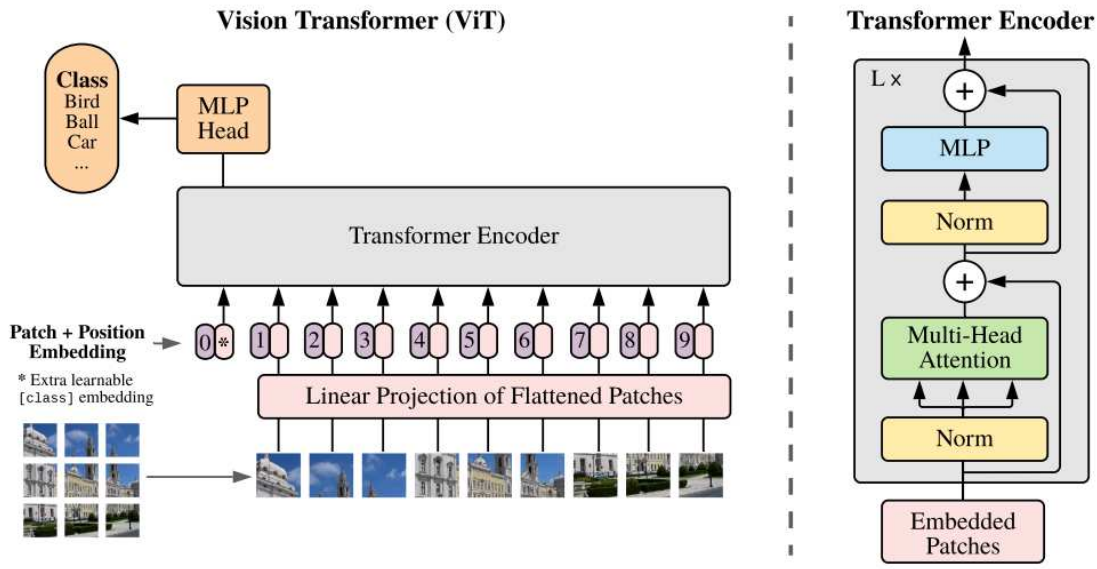

Figure 2.12: Model overview of ViT [11].

during the optimization process [11].

One common approach to incorporating spatial information in ViT for object detection is to use a combination of positional encodings and position-sensitive embeddings. These embeddings divide the image into a grid and encode position-specific information for each grid cell. This allows the model to reason about the spatial layout of objects in the image. While ViT for object detection has shown promising results, it often requires large-scale datasets and substantial computational resources for both pretraining and fine-tuning. However, it offers the advantage of end-to-end training, simplicity, and the ability to capture long-range dependencies, making it a valuable alternative to traditional CNN-based object detection models. Overall, the ViT transformer for object detection demonstrates the potential of transformers in dense prediction tasks and opens up new avenues for research in object detection using self-attention mechanisms [11].

Swin ViT

Swin Transformer (Swin) is a variant of ViT model that has been specifically designed for object detection tasks. Swin builds upon the concept of ViT but introduces hierarchical partitioning and shifting windows to address the computational and memory limitations of processing high-resolution images. The key idea behind Swin is to divide the input image into smaller

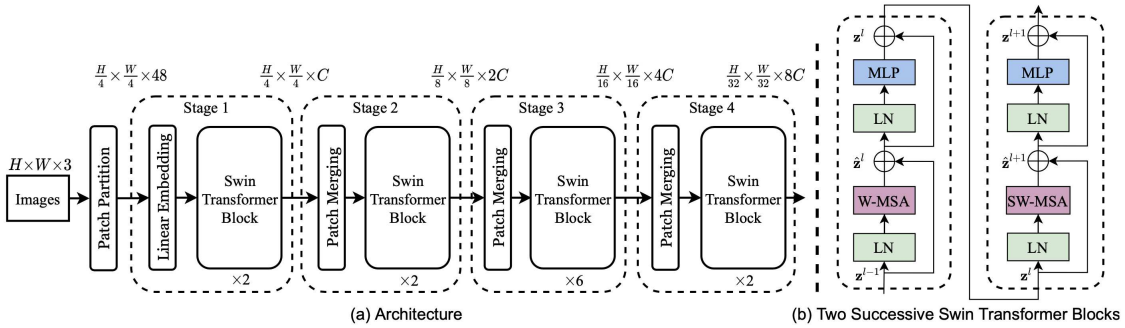

Figure 2.13: Architecture of Swin ViT [12].

non-overlapping patches, similar to ViT. However, instead of treating all patches equally, Swin introduces a hierarchical structure by grouping patches into stages. Each stage consists of a set of patch embeddings and a Swin Transformer block as depicted in Figure 2.13. The Swin Transformer block consists of multiple attention layers, each performing self-attention within a local window. These attention layers enable the model to capture both local and global dependencies within the image. By incorporating local attention windows, Swin significantly reduces the computational requirements compared to a traditional global self-attention mechanism [12].

To capture information across different stages, Swin employs a shifting operation that enables communication between adjacent stages. The shifting operation allows patches in one stage to attend to patches in the neighboring stages, facilitating the propagation of information across different scales and resolutions. During training, Swin is typically pre-trained on a large-scale dataset using a self-supervised learning approach similar to ViT. This pretraining helps the model learn meaningful representations of the input data. The pre-trained model is then fine-tuned for the object detection task by adding additional components, such as a detection head, to predict the presence, class, and location of objects within the image. The detection head takes the patch embeddings from the Swin Transformer blocks and generates bounding box predictions for each object in the image. These predictions are refined and adjusted during the optimization process to improve localization accuracy. Swin Transformer for object detection has shown competitive performance, particularly in scenarios where high-resolution images and fine-grained details are crucial. By incorporating hierarchical structures and local attention mechanisms, Swin enables efficient processing of large images while maintaining strong representation power [12].

2.8 Object Tracking

Object tracking is the process of continuously locating and following an object of interest within a video or image sequence. It is a vital component in computer vision, surveillance, robotics, and autonomous systems, offering a wide range of applications and benefits.

Video Surveillance:

In the domain of video surveillance, object tracking plays a crucial role in enhancing security measures. By tracking objects of interest, such as people or vehicles, surveillance systems can monitor their movements, identify suspicious behavior, and track individuals across different camera views. Object tracking enables automated surveillance tasks, improves situational awareness, and helps security personnel respond effectively to potential threats [51].

Autonomous Vehicles and Robotics:

Object tracking is of paramount importance in autonomous vehicles and robotics. By accurately tracking objects like pedestrians, vehicles, or obstacles in real-time, autonomous systems can perceive their surroundings and make informed decisions. Object tracking enables vehicles to navigate safely, avoid collisions, and plan optimal trajectories based on the detected objects. In robotics, object tracking allows robots to interact with their environment, manipulate objects, and collaborate with humans effectively [52].

Augmented Reality and Virtual Reality:

Object tracking is a fundamental aspect of Augmented Reality (AR) and Virtual Reality (VR) experiences. By tracking physical objects or markers, AR/ VR systems can precisely align virtual objects with the real world. This alignment facilitates realistic interactions, such as overlaying virtual information on real-world objects or enabling virtual objects to respond to physical gestures and movements. Object tracking ensures accurate registration and seamless integration of digital content, enhancing the immersive and interactive nature of AR/ VR applications [53].

Sports Analysis and Motion Capture:

In the realm of sports analysis, object tracking plays a significant role in tracking athletes' movements and actions. By tracking players or objects within a sports scene, coaches and analysts

can gather valuable data on performance, tactics, and player interactions. Object tracking enables detailed analysis, including player tracking, ball trajectory estimation, and team behavior understanding. Moreover, object tracking is extensively utilized in motion capture systems, allowing for the recording and reproduction of human movements for animation, gaming, and biomechanical analysis [54].

Human-Computer Interaction:

Object tracking enables natural and intuitive interactions between humans and computers. By tracking hand movements, gestures, or body poses, cameras can capture user input without the need for physical devices, such as keyboards or controllers. Object tracking facilitates gesture recognition, hand tracking, and body tracking, enabling users to interact with interfaces, control virtual objects, and play games through natural and immersive means. This enhances the user experience and expands the possibilities of human-computer interaction. Object tracking algorithms utilize various techniques, including feature-based methods, appearance models, motion estimation, and machine learning approaches. These algorithms encompass tasks such as object detection, initialization, tracking, and handling occlusions or object interactions. Object tracking is a fundamental task in computer vision and related fields, enabling applications that range from surveillance and autonomous systems to augmented reality, sports analysis, and human-computer interaction. Its advancements contribute to improved efficiency, safety, and user experiences in numerous domains [55].

2.8.1 Types of Object Tracking

Object tracking is a diverse field that encompasses various techniques and approaches tailored to specific tracking tasks. Let's delve into the different types of object tracking and their applications:

Image Tracking:

Image tracking involves tracking objects within a single image or a sequence of images. It focuses on determining the object's position, scale, and potentially its attributes, such as orientation or shape. Image tracking finds utility in applications like augmented reality, where virtual objects need to be aligned precisely with real-world images or markers. By accurately tracking objects within images, augmented reality systems can seamlessly integrate digital content into the user's environment.

Video Tracking:

Video tracking expands upon the concept of image tracking by encompassing the tracking of objects across a sequence of frames in a video. It entails estimating the object's position, scale, and appearance consistently as it moves through different frames. Video tracking plays a crucial role in surveillance systems, motion capture, autonomous vehicles, and numerous other applications where objects are in motion. By tracking objects in videos, systems can analyze behavior, detect anomalies, and enable autonomous navigation.

Single Object Tracking:

Single object tracking involves monitoring and following a specific object of interest throughout a video or image sequence. Typically, the object is manually identified or initialized at the beginning, and the tracking algorithm continues to track it over time, even when facing challenges like changes in appearance or occlusions. Single object tracking is relevant in applications such as object detection, visual servoing, and human-computer interaction. It enables precise tracking of a designated object, facilitating intuitive interactions and accurate analysis.

Multiple Object Tracking:

Multiple object tracking focuses on simultaneously tracking multiple objects within a video or image sequence. It involves detecting and tracking multiple objects while maintaining their identities over time, even in crowded scenes or situations involving occlusions. Multiple object tracking finds applications in diverse fields, including surveillance, traffic monitoring, crowd analysis, and robotics. This type of tracking allows systems to monitor and interact with numerous objects, leading to enhanced situational awareness and improved decision-making.

These types of object tracking can utilize various algorithms and techniques, such as feature-based tracking, optical flow, Kalman filters, particle filters, deep learning, or hybrid approaches combining multiple methods. The choice of tracking technique depends on the specific requirements of the application, the available data, and the computational resources. By employing the appropriate tracking methods, systems can achieve accurate and robust object tracking across a wide range of applications.

2.8.2 Challenges for object tracking

Developing an effective object tracking algorithm is a complex task due to several challenges that arise in real-world environments. While tracking an object on a straight road or in a simple setting may seem straightforward, numerous factors complicate object tracking in practical scenarios. Recognizing and understanding these common challenges is crucial for designing algorithms that can address these issues effectively. Let's delve into the specific challenges faced in object tracking:

1. **Occlusion:** Occlusion arises when an object being tracked is partially or completely hidden by other objects in the scene. Maintaining the object's identity and location becomes difficult when occlusion occurs. Overcoming occlusion necessitates the ability to handle object appearance changes, track multiple objects simultaneously, and predict an object's location based on its previous trajectory.

2. **Scale and viewpoint changes:** Objects often undergo changes in scale (size) and viewpoint (orientation) as they move. These variations pose a challenge in maintaining a consistent representation of the object and accurately tracking it. A robust tracking algorithm must be capable of handling scale variations and viewpoint changes to ensure reliable tracking.

3. **Illumination variations:** Changes in lighting conditions, such as variations in brightness, shadows, and reflections, significantly impact an object's appearance. Such illumination variations make it difficult for the tracker to distinguish the object from the background and other similar-looking objects. Robust object tracking algorithms should be able to adapt to these lighting variations and effectively handle different lighting conditions.

4. **Fast motion:** Objects moving rapidly can introduce motion blur, making it challenging to accurately track their position and maintain their identity. Additionally, fast motion can lead to substantial displacements between consecutive frames, posing difficulties for the tracker to estimate the object's new location. Tracking algorithms must be designed to handle fast motion and accurately predict an object's position despite motion blur or large displacements.

5. **Complex object interactions:** Real-world scenarios often involve multiple objects interacting with one another. Such interactions can include occlusions between objects, object-to-object occlusions, object mergers, or object splits. Tracking algorithms need to effectively handle these complex interactions to accurately track individual objects and maintain their identities throughout.

6. **Real-time processing:** Many applications necessitate real-time object tracking, where the tracking algorithm must process video frames at a sufficiently high frame rate. Real-time tracking imposes additional constraints on the algorithm's computational efficiency, making it challenging to develop methods that are both accurate and efficient in real-time scenarios.

7. **Initialization and re-detection:** Object tracking often require an initial bounding box or region to initiate the tracking process. However, in certain cases, initializing the tracker can be challenging, especially when the object's appearance is ambiguous or similar to other objects in the scene. Furthermore, if the tracker loses track of the object due to tracking failures, it must possess the capability to re-detect or re-acquire the object to continue tracking it accurately.

To address these challenges, researchers and practitioners focus on developing robust tracking algorithms that can handle various scenarios, adapt to changing conditions, and effectively model object appearance and motion. Innovative techniques, such as deep learning-based approaches, are being explored to tackle these challenges and enhance the performance of object tracking systems.

2.8.3 Object Tracking Algorithms and Techniques

Object tracking algorithms and techniques refer to a wide range of approaches used to locate and follow objects in videos or image sequences. These methods aim to accurately estimate the position, motion, and other relevant attributes of the tracked objects over time. Here are explanations of some commonly employed object tracking algorithms and techniques:

Feature-based

Feature-based tracking is a popular approach in object tracking that focuses on identifying and tracking distinctive features or keypoints on the object of interest. These features serve as reference points that can be reliably detected and matched between consecutive frames, allowing the tracking algorithm to estimate the object's motion and location over time. In feature-based tracking, the first step is to extract relevant features from the initial frame or a region of interest containing the object. Common feature extraction techniques include corner detection algorithms like Harris corner detector or Shi-Tomasi algorithm, which identify points where there are significant changes in intensity or gradient. Other methods, such as Scale-Invariant Feature Transform (SIFT) or Speeded Up Robust Feature (SURF), can also be used to extract more

robust features that are invariant to scale, rotation, and affine transformations. Once the features are extracted, they are matched between subsequent frames to establish correspondences. This matching process involves finding the best matches between the features in the current frame and those in the previous frame. Various matching algorithms, such as nearest neighbor matching or Random Sample Consensus (RANSAC), can be employed for this purpose.

After the feature matching, the tracking algorithm estimates the motion of the object by analyzing the displacements of the matched features. Common techniques used for motion estimation include optical flow, which computes the displacement vectors of the features, and geometric transformations like affine or projective transformations. Overall, feature-based tracking provides a robust and reliable approach to object tracking. By focusing on distinctive features, it can handle changes in appearance, occlusions, and partial object visibility. However, its performance may be affected by challenging conditions such as rapid motion, illumination changes, or complex backgrounds. Therefore, combining feature-based tracking with other techniques, such as deep learning or particle filters, can enhance its accuracy and robustness in various tracking scenarios.

Optical Flow

Optical flow [56] is a computer vision technique used to estimate the motion of pixels between consecutive frames in a video or image sequence. It provides a dense representation of the apparent motion in an image by assigning a motion vector to each pixel, indicating the direction and magnitude of its displacement. Optical flow analysis plays a crucial role in various applications such as object tracking, motion analysis, video stabilization, and autonomous navigation. The basic principle behind optical flow estimation is the assumption of brightness constancy, which states that the intensity of a pixel remains constant as it moves across frames. Optical flow algorithms aim to solve the aperture problem, where the apparent motion of a small image patch cannot be uniquely determined. To address this, several methods have been developed [56]. One popular approach is the Lucas-Kanade method, which assumes that the motion between frames is small and relatively smooth. It formulates the problem as an overdetermined system of equations and uses the least squares method to estimate the optical flow. The Lucas-Kanade method assumes a local constant velocity model, where the motion of a pixel is approximated as a linear combination of the motion of neighboring pixels.

Another commonly used optical flow method is the Horn-Schunck algorithm, which imposes smoothness constraints on the estimated flow field. It incorporates a global smoothness constraint by assuming that neighboring pixels have similar motion vectors. The Horn-

Schunck algorithm formulates optical flow estimation as a variational problem and solves it by minimizing an energy functional [56]. There are also more advanced optical flow techniques, such as the Farneback method, which employs polynomial expansion to capture complex motion patterns, and deep learning-based approaches that use CNNs to learn optical flow directly from training data. Despite their effectiveness, optical flow algorithms may face challenges in the presence of occlusions, textureless regions, rapid motion, or significant illumination changes. Robustness can be improved by combining optical flow with other techniques, such as tracking algorithms or depth information from additional sensors [56].

Kalman Filter

Kalman filters [57] are recursive estimation algorithms used for tracking and predicting the state of a dynamic system over time. They are particularly useful in scenarios where measurements are noisy or incomplete. The Kalman filter operates by combining predictions from a dynamic model with measurements to produce an optimal estimate of the system's state. The Kalman filter consists of two main steps: the prediction step and the update step. In the prediction step, the filter predicts the current state based on the previous state and the system's dynamics. The prediction is made using the state transition matrix, usually denoted as F, and the control input, denoted as B. The predicted state estimate is represented by $\hat{x}_k^-$, where k denotes the current time step [57].

After the prediction step, the filter incorporates measurements to update the state estimate in the update step. The update is performed using the measurement matrix, denoted as H, and the measurement noise covariance matrix, denoted as R. The updated state estimate is denoted as $\hat{x}_k$, and it is calculated based on the predicted state estimate, the measurement, and their respective covariances. The Kalman filter maintains two key parameters: the state estimate $\hat{x}_k$ and the error covariance matrix P_k. The error covariance matrix represents the uncertainty associated with the state estimate. Through the prediction and update steps, the Kalman filter continuously adjusts these parameters to provide an optimal estimate of the system's state [57]. The equations for the prediction and update steps of the Kalman filter can be summarized as follows:

Prediction:

$$\hat{x}_k^- = F\hat{x}_{k-1} + Bu_k$$

$$P_k^- = FP_{k-1}F^T + Q$$

Update:

$$y_k = z_k - H\hat{x}_k^-$$

$$S_k = HP_k^- H^T + R$$

$$K_k = P_k^- H^T S_k^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k y_k$$

$$P_k = (I - K_k H) P_k^-$$

In the above equations, Q represents the process noise covariance matrix, z_k is the measurement vector, y_k is the measurement residual, S_k is the innovation covariance, and K_k is the Kalman gain. The Kalman gain determines the weight given to the measurement update and depends on the uncertainties of the prediction and the measurement. These equations illustrate the fundamental operations performed by the Kalman filter, enabling it to estimate and track the state of dynamic systems accurately [57].

Unscented Kalman Filter

The Unscented Kalman Filter (UKF) [58] is an extension of the traditional Kalman filter that addresses the limitations of linearization in non-linear systems. It provides a more accurate estimation of the state and covariance by approximating the system's non-linear transformation using a set of representative points called sigma points. The UKF operates by propagating these sigma points through the non-linear functions to compute the predicted state and covariance and then updating them based on measurements.

To explain the UKF, let's denote the state as x and the measurement as z. The algorithm involves the following steps:

1. **Sigma Point Generation:** The first step is to generate a set of sigma points that capture the distribution of the current state estimate and covariance. These sigma points are calculated using the mean state estimate $\hat{x}$ and the covariance matrix P. The sigma points are typically chosen to represent the mean and the spread of the state distribution [58].

2. **Prediction Step:** Each sigma point is propagated through the non-linear process model to obtain the predicted sigma points. These predicted sigma points are then used to estimate the predicted mean state $\hat{x}_{\text{pred}}$ and the predicted covariance matrix P_{pred}. The prediction step also involves incorporating process noise [58].

3. **Update Step:** The predicted sigma points are mapped to the measurement space using the non-linear measurement model. This yields the predicted measurement sigma

points. From these points, the predicted measurement mean $\hat{z}_{\text{pred}}$ and the predicted measurement covariance matrix P_{zz} are estimated [58].

4. **Kalman Gain Calculation:** The Kalman gain K is computed by correlating the predicted measurement sigma points with the predicted state sigma points. It determines the weight given to the predicted measurement information during the update step [58].

5. **State and Covariance Update:** Finally, the updated state estimate $\hat{x}$ and the updated covariance matrix P are calculated using the Kalman gain, the predicted measurement residual y, and the predicted measurement covariance matrix P_{zz}.

The equations for the prediction and update steps of the UKF are as follows:
Prediction:

$$X_k = f(X_{k-1}, u_{k-1}) \quad \text{(propagate sigma points)}$$

$$\hat{x}_k = \sum_{i=0}^{2n} w_i^m X_{k,i} \quad \text{(predicted mean)}$$

$$P_k = \sum_{i=0}^{2n} w_i^c (X_{k,i} - \hat{x}_k)(X_{k,i} - \hat{x}_k)^T + Q_k \quad \text{(predicted covariance)}$$

Update:

$$Z_k = h(X_k) \quad \text{(measurement sigma points)}$$

$$\hat{z}_k = \sum_{i=0}^{2n} w_i^m Z_{k,i} \quad \text{(predicted measurement mean)}$$

$$P_{zz,k} = \sum_{i=0}^{2n} w_i^c (Z_{k,i} - \hat{z}_k)(Z_{k,i} - \hat{z}_k)^T + R_k \quad \text{(predicted measurement covariance)}$$

$$P_{xz,k} = \sum_{i=0}^{2n} w_i^c (X_{k,i} - \hat{x}_k)(Z_{k,i} - \hat{z}_k)^T \quad \text{(cross-covariance)}$$

$$K_k = P_{xz,k} P_{zz,k}^{-1} \quad \text{(Kalman gain)}$$

$$\hat{x}_k = \hat{x}_k + K_k(z_k - \hat{z}_k) \quad \text{(updated state)}$$

$$P_k = P_k - K_k P_{zz,k} K_k^T \quad \text{(updated covariance)}$$

In these equations, $f(\cdot)$ represents the non-linear process model, $h(\cdot)$ represents the non-linear measurement model, w_i^m and w_i^c are the weights associated with the sigma points, and Q_k and R_k represent the process noise covariance and measurement noise covariance, respectively

[58]. By incorporating the sigma points and approximating the non-linear transformations, the Unscented Kalman Filter provides a more accurate estimation of the state in non-linear systems compared to the traditional Kalman filter.

2.8.4 Particle Filter

Particle filters, also known as sequential Monte Carlo methods, are a class of recursive filtering algorithms used for estimating the state of a system in the presence of non-linearities and uncertainties. They are particularly effective when the system or measurement models are non-linear or when the probability distributions involved are non-Gaussian. Particle filters represent the probability distribution of the state using a set of particles, where each particle represents a possible state hypothesis. The particle filter algorithm operates in a sequential manner, updating the particles as new measurements become available. The key steps of the particle filter can be summarized as follows [59]:

1. **Initialization:** Initially, the particles are generated from an initial prior distribution. These particles are typically sampled randomly or based on prior knowledge about the system. Each particle is associated with a weight, indicating its importance or likelihood.

2. **Prediction:** In the prediction step, the particles are propagated forward in time using a process model. The process model accounts for the system dynamics and includes any known controls or inputs. Each particle is updated based on the system dynamics, introducing uncertainty and variability.

3. **Weight Update:** Once a new measurement is obtained, the particles' weights are updated based on the likelihood of the measurements given the particles' states. The likelihood is computed using the measurement model, which relates the system's state to the observed measurements. The weights are adjusted to reflect the consistency between the predicted measurements and the actual measurements.

4. **Resampling:** To focus computational resources on the more likely state hypotheses, the particles are resampled according to their weights. The resampling step selects particles with higher weights more frequently, while particles with lower weights are discarded. This process effectively concentrates the particle distribution around the more probable state hypotheses.

5. **Estimation:** Finally, the estimated state is computed based on the resampled particles. This can be done by taking the mean or weighted average of the particles' states, providing an approximation of the true state.

Particle filters provide a flexible and robust framework for state estimation in non-linear and non-Gaussian systems. They are widely used in various applications such as object tracking, simultaneous localization and mapping (SLAM), and robotics. Although there is no single formula that fully captures the particle filter algorithm, the following equations illustrate the main steps involved [59]:

Prediction:

$$x_k^{[i]} = f(x_{k-1}^{[i]}, u_k) + \varepsilon_k^{[i]}$$

Weight Update:

$$w_k^{[i]} = \frac{p(z_k|x_k^{[i]})}{\sum_{j=1}^{N} w_{k-1}^{[j]}}$$

Resampling:

$$\text{Draw } i \text{ with probability } \propto w_k^{[i]}$$

Estimation:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^{[i]} x_k^{[i]}$$

In these equations, $x_k^{[i]}$ represents the state of the i-th particle at time k, u_k denotes the control input, z_k represents the measurement at time k, $w_k^{[i]}$ represents the weight associated with the i-th particle at time k, $p(z_k|x_k^{[i]})$ denotes the likelihood of the measurement given the particle's state, and $\varepsilon_k^{[i]}$ represents the process noise added during the prediction step. These equations highlight the core operations involved in particle filtering, providing a foundation for implementing and understanding this powerful state estimation technique [59].

2.8.5 Deep learning algorithms for object tracking

Deep learning algorithms have revolutionized many fields, including object tracking. These algorithms leverage the power of artificial neural networks, specifically deep CNNs, to learn and predict the motion and appearance of objects in videos. Deep learning-based object tracking algorithms can handle complex scenarios, handle occlusions, and adapt to changes in object appearance. One of the key advantages of deep learning-based object tracking algorithms is their ability to learn rich and discriminative features from large-scale training datasets. These algorithms can capture high-level semantic information and generalize well to unseen objects and diverse environments. However, training deep learning models for object tracking requires a large amount of annotated training data and significant computational resources.

Deep learning algorithms for object tracking can be further improved by integrating other techniques such as online adaptation, motion models, and attention mechanisms. These additions can enhance the tracker's robustness, handling variations in object appearance, occlusions, and scale changes. Overall, deep learning-based algorithms have significantly advanced the field of object tracking, providing more accurate and reliable tracking solutions compared to traditional methods. Their ability to learn and adapt to object appearance variations makes them suitable for a wide range of applications.

Deep Sort

Deep SORT (Simple Online and Realtime Tracking) is an algorithm designed for object tracking that combines a deep appearance descriptor with the SORT (Simple Online and Realtime Tracking) algorithm. It aims to improve the tracking performance by associating detected objects over time using a deep neural network-based appearance metric. In Deep SORT, the algorithm begins by using a detection algorithm to detect objects in each frame. These detections are then associated with existing tracks using a combination of motion information and appearance similarity. The deep appearance descriptor, obtained from a pre-trained deep neural network, is used to compute the similarity between the detected objects and existing tracks. This allows for accurate matching even in cases where objects undergo significant appearance changes [60].

To maintain track identities across frames, Deep SORT utilizes a Kalman filter-based tracking framework. The Kalman filter predicts the next state of each object track based on its motion information, while the deep appearance descriptor is used to update the track's appearance representation. The algorithm also incorporates techniques such as track management, track initiation, and track termination to handle challenging scenarios such as occlusions and the temporary disappearance of objects. Deep SORT has been shown to achieve robust and accurate object tracking results, especially in complex scenarios with occlusions and appearance changes. It has been widely adopted in various applications, including video surveillance, autonomous driving, and robotics, where real-time and reliable object tracking is crucial [60].

Siamese Networks

Siamese Networks [61] are a class of deep neural networks designed for tasks that require measuring similarity or dissimilarity between pairs of inputs. They consist of twin subnetworks with shared weights, hence the name "Siamese." Each subnetwork processes one input, and

the final layers of the network compare the learned representations of the inputs to compute a similarity score. The training of Siamese Networks involves presenting pairs of inputs, where one input belongs to the same class or category, and the other belongs to a different class. The network learns to minimize the contrastive loss between similar pairs, encouraging the learned representations to be close together, while maximizing the loss for dissimilar pairs, pushing the representations further apart. This way, the Siamese Network learns to capture discriminative features and extract similarity information from the inputs. Siamese Networks have been successfully applied to various tasks, including face recognition, signature verification, image retrieval, and few-shot learning. They excel in scenarios where training data is limited or when measuring similarity is a primary objective. Siamese Networks offer a flexible framework for learning powerful representations that can measure similarities between pairs of inputs, enabling applications that require similarity-based decision-making [61].

Fair MOT

Fair MOT (Fair Multi-Object Tracking) [13] is an object tracking approach that addresses the challenge of joint optimization between object detection and re-ID tasks in a single network. It builds upon the anchor-free object detection architecture called CenterNet and introduces several modifications to achieve accurate and fair tracking results. In Fair MOT, the backbone network utilizes ResNet-34 with an enhanced version of Deep Layer Aggregation (DLA) to fuse multi-layer features. This enhanced DLA incorporates skip connections and deformable convolutions to improve feature alignment and adjust receptive fields dynamically. The resulting model, named DLA-34, strikes a balance between accuracy and speed [13].

The detection branch of Fair MOT as shown in Figure 2.14 is built on top of CenterNet, where three parallel heads estimate heatmaps, object center offsets, and bounding box sizes. The heatmap head uses a logistic regression with focal loss to estimate object locations. The box offset and size heads refine localization accuracy by mitigating quantization errors and estimating object dimensions. Fair MOT includes a re-ID branch to generate features that distinguish objects. re-ID features are extracted using a convolution layer on top of the backbone features. The branch employs a classification task, treating object instances of the same identity as the same class and computing the re-ID loss based on the identity embedding vectors [13].

Training Fair MOT involves joint optimization of the detection and re-ID branches using a combination of losses. The network balances the tasks using an uncertainty loss that adjusts the importance of each task. In addition, a single-image training method is proposed for image-level object detection datasets, where each object instance is treated as a separate class. Overall,

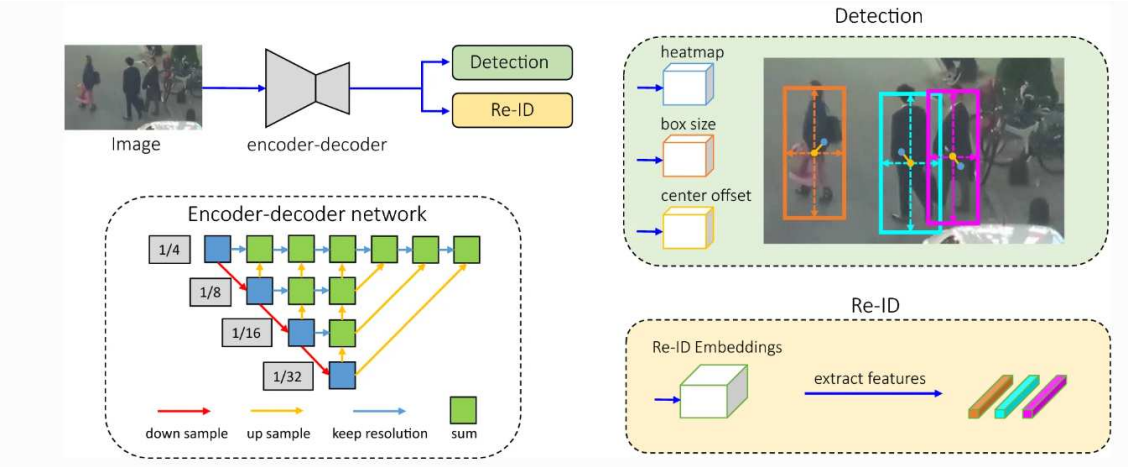

Figure 2.14: Overview of the one-shot tracker Fair MOT [13].

Fair MOT provides a comprehensive solution for multi-object tracking by integrating object detection and re-ID tasks while ensuring fairness between the two. It achieves state-of-the-art performance on various public datasets and provides source code and pre-trained models for further use and evaluation [13].

2.9 Laser Rangefinder

A laser rangefinder, also known as a laser distance meter or LiDAR sensor, is a device that uses laser technology to accurately measure distances to objects or surfaces. It operates on the principle of time-of-flight, where a laser beam is emitted towards a target, and the time it takes for the beam to travel to the target and back is measured to calculate the distance. The components of a laser rangefinder typically include a laser emitter, optics system, photodetector, timing circuitry, and a distance calculation unit. Laser rangefinders find applications in surveying, military targeting, robotics, sports, and industrial automation, providing precise distance measurements for various purposes. They offer advantages such as high accuracy, fast measurement speeds, and non-contact operation, but their performance can be influenced by atmospheric conditions and the reflectivity of the target surface. Laser rangefinders are essential devices used in a wide range of applications. They utilize laser technology to accurately measure distances by calculating the time it takes for a laser beam to travel to a target and back. The key components of a laser rangefinder include a laser emitter, optics system, photodetector, timing circuitry, and distance calculation unit. These devices are commonly used in surveying,

military targeting, robotics, sports, and industrial automation [62, 63].

In surveying and mapping, laser rangefinders aid in creating 3D models, terrain mapping, and site planning by providing precise distance measurements. In military and defense applications, laser rangefinders are integrated into weapons systems and reconnaissance equipment for target acquisition, artillery aiming, and range finding. They play a crucial role in robotics and autonomous systems by enabling real-time mapping, obstacle detection, and navigation for robots and autonomous vehicles. Laser rangefinders are also popular in sports like golf and hunting, where accurate distance measurements are essential for targeting and shot planning. In industrial and manufacturing settings, they contribute to precise positioning, quality control, and object detection in automated processes. Laser rangefinders offer numerous advantages, including high accuracy, fast measurement speeds, and the ability to operate without physical contact with the target. However, their performance can be affected by factors such as atmospheric conditions (e.g., fog, dust) and the reflectivity of the target surface. Despite these considerations, laser rangefinders are versatile devices that provide reliable and precise distance measurements, enhancing efficiency and safety across various industries and applications [62, 63].

2.10 People Detection in 2D Range Data

People detection in 2D range data refers to the process of detecting and identifying human figures or individuals in a two-dimensional representation of the surrounding environment, typically obtained from sensors such as LiDAR or depth cameras. This technique is widely used in various applications, including robotics, surveillance systems, and autonomous vehicles. The process of people detection in 2D range data involves several key steps [64]:

1. Data Acquisition: The first step is to acquire the 2D range data from a sensor, such as a LiDAR or depth camera. These sensors provide depth information for each point in the observed scene, allowing the construction of a 2D range image or point cloud.

2. Preprocessing: The acquired 2D range data often requires preprocessing to enhance the quality and remove noise. Common preprocessing steps include filtering outliers, noise removal, and downsampling the data to reduce computational complexity.

3. Segmentation: In this step, the 2D range data is segmented to separate different objects or regions of interest. Segmentation techniques aim to identify distinct clusters or groups of points that are likely to belong to the same object. Various methods such as clustering algorithms or region-growing techniques can be employed for this purpose.

4. Feature Extraction: Once the regions of interest are identified, relevant features are extracted from each segment to characterize the potential presence of people. These features may include geometric properties (e.g., height, width, aspect ratio), statistical properties (e.g., mean, variance), or texture-based features.

5. Classification: The extracted features are then utilized to classify the segmented regions as either people or non-people. This is typically achieved using machine learning algorithms such as SVM, random forests, or deep learning-based approaches. These algorithms are trained on labeled data, where human-labeled segments are used for positive samples, and non-human segments or background regions are used for negative samples.

6. Post-processing: Once the classification is performed, post-processing steps are often employed to refine the detection results. This can involve techniques such as clustering nearby detections, spatial filtering to remove false positives, or temporal filtering to maintain consistency over time in video sequences.

7. Tracking: In scenarios where the 2D range data is obtained from a moving sensor or in dynamic environments, people tracking can be applied to associate detections across consecutive frames and maintain their identities. Various tracking algorithms, such as Kalman filters or particle filters, can be used for this purpose.

It is worth noting that the effectiveness of people detection in 2D range data heavily depends on the quality and resolution of the acquired data, as well as the robustness of the employed feature extraction and classification techniques. Additionally, lighting conditions, occlusions, and complex scenes can pose challenges in accurate people detection. Overall, people detection in 2D range data provides valuable information about the presence and location of individuals, enabling various applications in areas like robotics navigation, safety systems, and crowd monitoring [64].

2.10.1 Deep Learning Algorithms in 2D range data

A deep learning algorithm for detecting people in 2D range data utilizes CNNs or similar architectures. The algorithm is trained on a labeled dataset of 2D range data samples, along with annotations indicating the presence or absence of people. The network learns to extract relevant features from the range data and map them to the correct person/non-person labels. During testing, the trained network takes preprocessed range data as input and generates predictions for person detection. The algorithm's performance is evaluated using metrics like precision, recall, and F1 score, and can be refined through adjustments to hyperparameters, network architecture, and the use of techniques such as transfer learning or data augmentation.

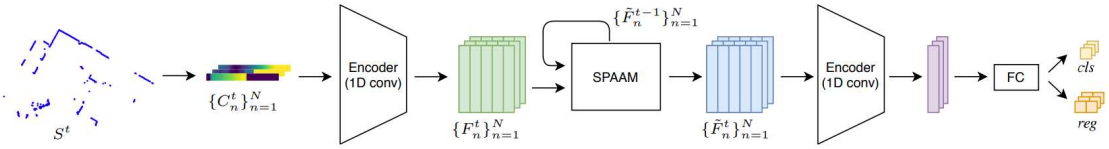

Figure 2.15: Overview of the DR-SPAAM architecture [14].

2.10.2 DR-SPAAM

Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM) [14]is a proposed method for person detection using 2D LiDAR data that takes a forward-looking approach as illustrated in Figure 2.15 to overcome the alignment problem and improve efficiency. It keeps the intermediate features from the backbone network as a template and recurrently updates this template when a new scan becomes available. The updated feature template is then used for detecting persons in the current scene. This forward-looking paradigm allows the aggregation of temporal information without the need for explicit alignment, resulting in faster processing [14]. The DR-SPAAM detector outperforms existing state-of-the-art methods on the DROW dataset while being approximately four times faster. It achieves a high frame rate of 87.2 FPS on a laptop with a dedicated GPU and 22.6 FPS on an NVIDIA Jetson AGX embedded GPU. The code for DR-SPAAM, implemented in PyTorch, and pre-trained models are released by the authors. The method utilizes a similarity-based spatial attention module to learn to associate misaligned features from a spatial neighborhood. It allows the network to effectively aggregate information from previous scans by attending to relevant features. Additionally, an auto-regressive model is used to update the representation, enabling information aggregation forward through time. Overall, DR-SPAAM presents an efficient and effective approach to person detection using 2D LiDAR data, outperforming existing methods while maintaining real-time performance [14].

2.11 Projection from World to Image Plane with Pinhole Camera

In computer vision and computer graphics, a pinhole camera model is commonly used to represent the projection of a 3D point in the world onto a 2D point in the image plane. The pinhole camera model is a simplified model that assumes a small aperture (pinhole) through which light passes, resulting in a projection onto a flat image plane. In the pinhole camera model, the cam-

era is represented as a point called the ”camera center” or ”optical center” (C). The image plane is a 2D plane perpendicular to the camera’s optical axis. The projection of a 3D point P in the world (X, Y, Z) onto the image plane results in a 2D point (x, y) in the image. The camera center, the 3D point, and its projection onto the image plane form a right triangle [15].

The basic principle is illustrated in Figure 2.16.

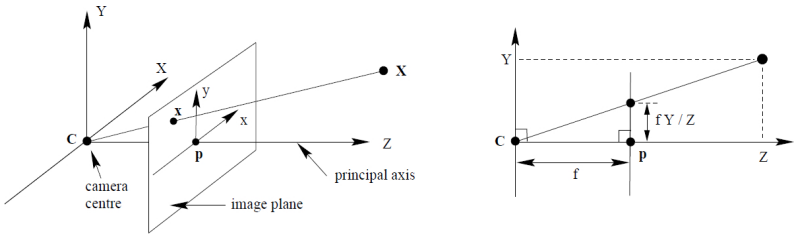

Figure 2.16: Pinhole camera model [15].

2.11.1 Camera Calibration

Camera calibration is the process of estimating the intrinsic and extrinsic parameters of a camera in order to accurately interpret the relationship between the 3D world and the 2D image captured by the camera. This calibration is essential for various computer vision tasks such as 3D reconstruction, object tracking, and augmented reality [65]. Intrinsic parameters represent the internal characteristics of the camera, including its focal length, principal point (the image center), and lens distortion coefficients. Extrinsic parameters, on the other hand, define the camera’s position and orientation in the 3D world. The most commonly used camera calibration model is the pinhole camera model. According to this model, the 3D world coordinates (X, Y, Z) are projected onto the 2D image plane (x, y) using the following equations:

$$x = \frac{f_x X}{Z} + c_x$$

$$y = \frac{f_y Y}{Z} + c_y$$

where: - (x, y) are the pixel coordinates in the image. - (X, Y, Z) are the corresponding 3D world coordinates. - (c_x, c_y) are the coordinates of the principal point (image center). - (f_x, f_y) are the focal lengths of the camera.

To account for lens distortion, the equations are modified as follows:

$$\acute{x} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

$$\acute{y} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

where: - $(\acute{x}, \acute{y}) is the distorted pixel coordinates. - r^2 = x^2 + y^2$. - k_1, k_2, k_3 are the distortion coefficients.

Camera calibration is typically performed using a calibration pattern with known 3D positions. By observing this pattern from different angles and positions, the corresponding 2D image coordinates can be matched with the known 3D coordinates, allowing the estimation of intrinsic and extrinsic parameters [65].

The calibration process involves the following steps: 1. Capture multiple images of the calibration pattern from different viewpoints. 2. Detect and extract the calibration pattern corners in each image. 3. Match the 2D corners with their corresponding 3D positions. 4. Use a calibration algorithm (e.g., Zhang's method or Tsai's method) to estimate the camera parameters.

Once the intrinsic and extrinsic parameters are determined, they can be used for various computer vision tasks. For example, the intrinsic parameters are necessary for correcting lens distortion, and the extrinsic parameters enable the transformation of 2D image coordinates into 3D world coordinates. Here's the formula for the pinhole camera model with lens distortion:

$$\acute{x} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$
$$\acute{y} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right)$$

Note: The above formulas assume that the camera has the same focal length in both the x and y directions (i.e., $f_x = f_y$). If the camera has different focal lengths, you can adjust the formulas accordingly [65].

2.11.2 Homography Matrix

A homography matrix, denoted as H, is a 3x3 matrix that represents the transformation between two different image planes. It defines the projective mapping between the coordinates of points in one plane to the corresponding points in the other plane. The homography matrix is used in computer vision tasks such as image registration, image stitching, and object recognition. To compute the homography matrix using the Singular Value Decomposition (SVD) method, we start with a set of corresponding points in the two image planes. Let's consider two sets of 2D points [66]:

In the source plane (Plane A):

$$\mathbf{p}_A = \begin{bmatrix} x_{A_1} & y_{A_1} \\ x_{A_2} & y_{A_2} \\ \vdots & \vdots \\ x_{A_n} & y_{A_n} \end{bmatrix}$$

In the destination plane (Plane B):

$$\mathbf{p}_B = \begin{bmatrix} x_{B_1} & y_{B_1} \\ x_{B_2} & y_{B_2} \\ \vdots & \vdots \\ x_{B_n} & y_{B_n} \end{bmatrix}$$

To find the homography matrix, we need at least 4 corresponding points. We can set up a linear system of equations based on these point correspondences:

$$\begin{bmatrix} x_{A_1} & y_{A_1} & 1 & 0 & 0 & 0 & -x_{A_1}x_{B_1} & -y_{A_1}x_{B_1} \\ 0 & 0 & 0 & x_{A_1} & y_{A_1} & 1 & -x_{A_1}y_{B_1} & -y_{A_1}y_{B_1} \\ x_{A_2} & y_{A_2} & 1 & 0 & 0 & 0 & -x_{A_2}x_{B_2} & -y_{A_2}x_{B_2} \\ 0 & 0 & 0 & x_{A_2} & y_{A_2} & 1 & -x_{A_2}y_{B_2} & -y_{A_2}y_{B_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{A_n} & y_{A_n} & 1 & 0 & 0 & 0 & -x_{A_n}x_{B_n} & -y_{A_n}x_{B_n} \\ 0 & 0 & 0 & x_{A_n} & y_{A_n} & 1 & -x_{A_n}y_{B_n} & -y_{A_n}y_{B_n} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = 0$$

where h_{ij} are the elements of the homography matrix and 0 is a zero vector.

We can rewrite the above equation as $\mathbf{Ah} = 0$, where $\mathbf{h}$ is the column vector of unknowns, and $\mathbf{A}$ is the coefficient matrix.

To solve for the homography matrix, we perform Singular Value Decomposition on matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U\Sigma V}^T$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{\Sigma}$ is a diagonal matrix with singular values on the diagonal.

The homography matrix [66] can be computed as the right-singular vector corresponding to the smallest singular value. We reshape this vector into a 3x3 matrix, resulting in the homography matrix **H**.

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

It is important to note that the homography matrix **H** is only determined up to a scale factor. In practice, the matrix is often normalized by dividing all elements by the value of h_{33}, resulting in a homography matrix with $h_{33} = 1$. Once the homography matrix is computed, it can be used to transform points between the two image planes by multiplying the homography matrix with the coordinates of a point in the source plane:

$$\begin{bmatrix} x_B \\ y_B \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix}$$

This equation provides the transformation of a point from the source plane (Plane A) to the destination plane (Plane B) using the computed homography matrix. By computing the homography matrix using SVD, we can accurately map points between different image planes, allowing for various applications in computer vision, such as image registration and stitching [66].

3
Methodology

The methodology chapter of this thesis presents a framework for auto-labeling panoramic videos for people tracking. The framework utilizes sensor fusion and calibration techniques, along with the integration of state-of-the-art deep learning-based person detection methods, specifically You Only Look Once (YOLO)v7 [46] and Distance Robust SPatial Attention and Auto-regressive Model (DR-SPAAM) [14]. Auto-labeling panoramic videos is a critical aspect of creating a dataset for people tracking in service robotics. The framework aims to automate the process of labeling panoramic frames with accurate person locations, overcoming the challenges associated with the wide field of view and distortions introduced by omnidirectional cameras. By integrating sensor fusion and calibration techniques, the framework not only enhances the accuracy and reliability of person detection and tracking but also leverages laser data to provide labels in the robot space.

The YOLOv7 algorithm, known for its real-time object detection capabilities, is utilized for detecting persons in panoramic frames. This deep learning-based approach provides efficient and robust detection performance. Additionally, the DR-SPAAM method, which is based on deep learning and sequential point association with appearance modeling, leverages 2D range sequences obtained from a laser scanner to detect persons in the environment. By integrating these advanced person detection techniques, the framework improves the precision and effectiveness of auto-labeling panoramic videos for people tracking. Throughout this chapter, we will delve into the details of the framework, explaining the implementation of sensor fusion and calibration techniques to address the distortions caused by omnidirectional cameras. We

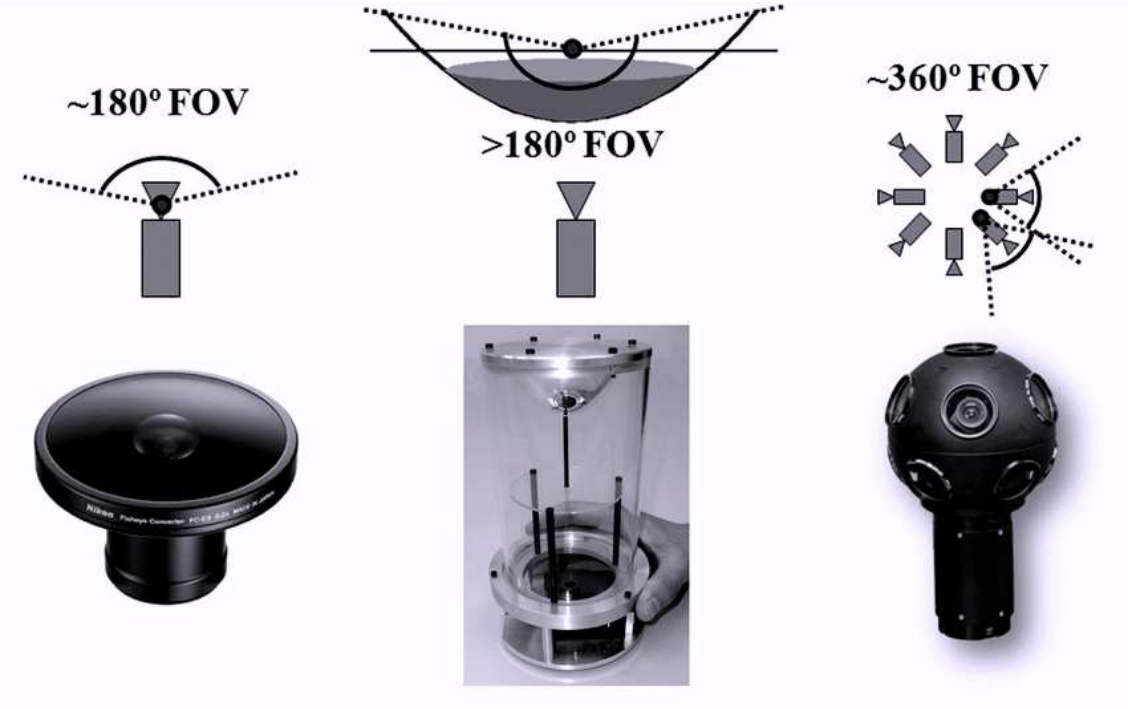

Figure 3.1: Dioptric camera, Catadioptric camera, and Polydioptric camera.

will also provide a comprehensive overview of the YOLOv7 and DR-SPAAM algorithms, describing their integration into the auto-labeling process.

3.1 Omnidirectional Camera

Traditional cameras have a restricted field of view, which means they can only capture a limited portion of the surrounding environment. However, by using fisheye lenses, it becomes possible to capture images that cover a larger field of view, reaching up to 360 degree. This expanded view allows for better situational awareness and object detection. Nevertheless, achieving a complete 360 degree visual perception often necessitates the use of multiple cameras, which introduces challenges in image processing. Combining and synchronizing the data from multiple cameras requires complex calibration techniques to ensure accurate spatial alignment and fusion of information. Furthermore, processing the large amount of data generated by multiple cameras can be computationally intensive.

Omnidirectional cameras, also known as omni cameras, possess the remarkable ability to cap-

ture light from all directions, offering coverage of an entire sphere. Several techniques have been developed to obtain 360 degree images, each with its own advantages and limitations, they showed in Figure 3.1. Dioptric cameras combine shaped lenses, such as fisheye lenses, to achieve a field of view larger than 180°, while catadioptric cameras utilize specially shaped mirrors to provide a 360 degree horizontal field of view and over 100 degree vertical field of view. Polydioptric cameras use multiple cameras with overlapping fields of view to achieve a true omnidirectional field of view, although they tend to be more costly due to the increased number of sensors required [67].

In recent years, there has been a surge in the popularity of 360 degree cameras, particularly in the consumer market, offering more affordable alternatives compared to traditional omni cameras. One prevalent type is the dual-fisheye camera, which utilizes two fisheye lenses with a field of view of approximately 220° each. By stitching the two fisheye images together, a spherical image is created, though the stitching process may introduce some artifacts. Nevertheless, modern solutions often incorporate on-camera automatic stitching, reducing the processing time required in subsequent computer vision stages. For this reason in our project, we have selected the Ricoh Theta Z1 [1] as our omni camera model. The Ricoh Theta Z1 is a dual-fisheye camera that enables real-time streaming via a USB connection. It generates an equirectangular projection of the surrounding environment, offering a 360 degree horizontal field of view and a 180° vertical field of view. The equirectangular format, which represents a sphere on a flat surface, eliminates the need for complex steps involved in obtaining a cylindrical projection from a pair of fisheye images. The versatility and convenience of the Ricoh Theta Z1 make it well-suited for our research purposes.

3.2 Robotic platform

In our thesis, we relied on the robust SUMMIT-XL STEEL [2] robot platform (Fig 3.3) as the cornerstone of our research and data collection. This platform is specifically designed for applications in logistics and indoor transport. To enhance the robot's perception abilities, we integrated a laser range finder system into the SUMMIT-XL STEEL platform. This system was comprised of two Laser HOKUYO-10LX [3] sensors strategically positioned on the robot, pro-

[1] https://theta360.com/it/about/theta/z1.html
[2] https://robotnik.eu/products/mobile-robots/summit-xl-steel-en/
[3] https://hokuyo-usa.com/products/lidar-obstacle-detection/ust-10lx

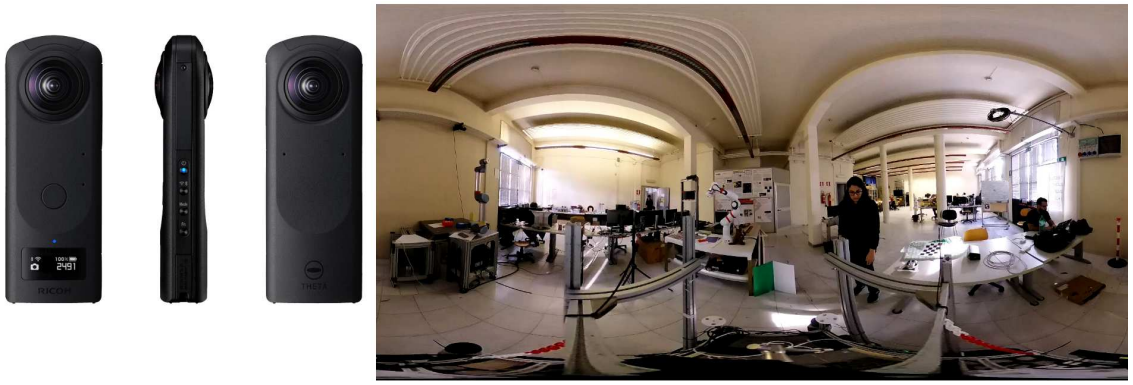

Figure 3.2: The Ricoh Theta Z1 camera and a sample picture.

Figure 3.3: The SUMMIT-XL STEEL robot.

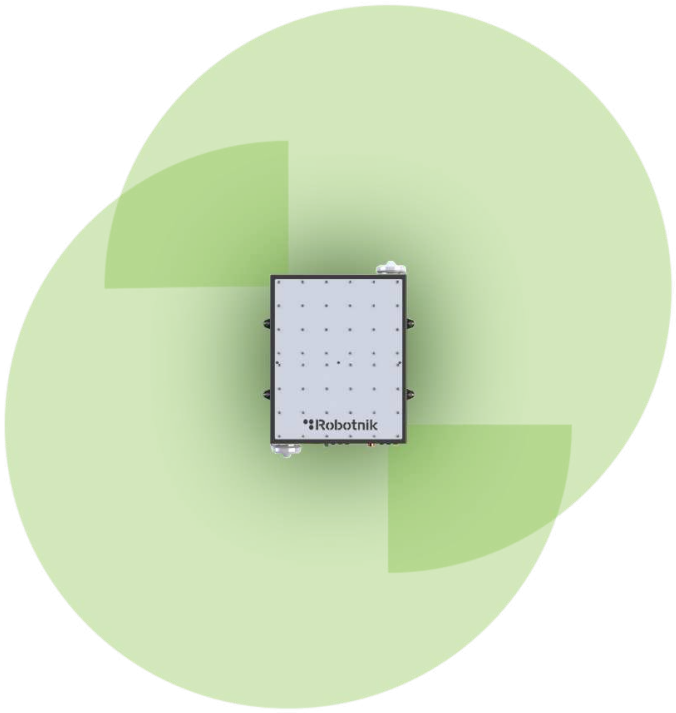

Figure 3.4: The laser's field of view.

viding accurate distance measurements and an expanded field of view. The Laser HOKUYO-10LX sensors emitted laser beams that covered a wide 270-degree field of view (as depicted in Fig 3.4) and precisely measured the time it took for the beams to reflect back. This allowed the robot to calculate precise distances to objects in its surrounding environment.

Our primary objective was to employ this laser range finder system for detecting the positions of individuals in the vicinity of the robot. By analyzing the distance measurements acquired from the sensors, we were able to determine the exact locations of people relative to the robot's position. Through the seamless integration of the laser range finders into the SUMMIT-XL STEEL robot platform, we significantly enhanced its perception capabilities. This enabled accurate detection and tracking of individuals in close proximity to the robot. The combination of the sturdy robot platform and the laser range finder system provided a dependable and adaptable solution, empowering us to effectively detect and monitor the presence of humans throughout the entirety of our study.

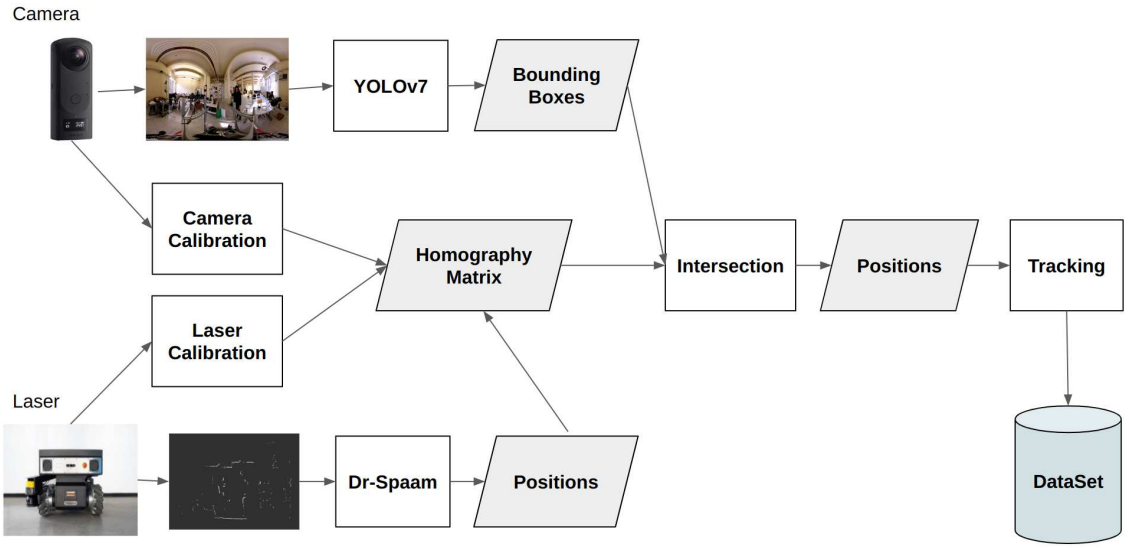

Figure 3.5: The framework for autolabeling.

3.3 Overview of the Proposed Framework

This section provides an overview of the framework designed for autolabeling panoramic videos and determining the position of each person within the world frame, as depicted in Figure 3.5. The subsequent sections will delve into each step in detail. To begin, we gather data using an omnidirectional camera and a 2D laser rangefinder, enabling us to capture a 360 degree field of view. We employ YOLOv7, a powerful object detection model, to detect people in each frame of the video. Additionally, we utilize DR-SPAAM to detect people in the 2D range dataset that corresponds to each frame.

By computing the homography matrix, we can transform the individuals detected by DR-SPAAM to the image frame and select the positions associated with the individuals identified within the bounding boxes obtained from YOLOv7. To achieve effective people tracking, we employ two techniques: the unscented Kalman filter [58] and the Global Nearest Neighbor (GNN) method. The unscented Kalman filter is utilized for predicting and handling uncertainties regarding people's motion across consecutive laser scans. On the other hand, the global nearest neighbor method is employed to resolve the data association problem between these consecutive scans. By combining these steps, our framework enables the automatic labeling of panoramic videos while providing accurate positioning information for each person within the world frame. The subsequent sections will provide a comprehensive explanation of each component in the process.

3.4 Camera and Laser Calibration

The fusion of heterogeneous sensors presents a significant opportunity to enhance the environmental perception capabilities of mobile robots. One of the main challenges in our setting is calibrating the depth scan information and plane image information obtained from these different sensors. Calibration is a fundamental step to align and synchronize the measurements from multiple sensors so that they can be effectively fused together. In our case, we want to accurately map the 3D depth scans from the laser to the corresponding 2D images captured by the camera. This calibration process is crucial because it enables the fusion of depth information with visual data, allowing for more accurate and robust perception. By aligning the sensor data correctly, the robot can associate depth information with objects and features identified in the camera images, enabling more accurate localization, mapping, and object recognition. This section focuses on addressing this challenge and proposes a technique for accurate calibration in panoramic images [68].

3.5 Camera Calibration

Camera calibration is a fundamental step in accurately capturing and representing the geometry of a scene. When dealing with panoramic images, camera calibration becomes even more critical to correct lens distortions and other camera-related artifacts. In this thesis, our initial approach involved calibrating the panoramic image using a standard method and applying the procedure for computing the homography matrix, as depicted in Figure 3.6. However, we encountered difficulties due to distortions and significant errors in extracting the homography matrix, as illustrated in Figure 3.7. The red circles in the image represent the transformed positions of people on the robot plane using this homography matrix, but the accuracy was found to be extremely low. As a result, we decided to adopt a different technique using a cube map instead. A cube map comprises six square images, each representing one face of a cube. These faces are arranged in a cross shape, with three horizontally stacked faces and two vertically stacked faces. Each face captures the view in a specific direction: positive and negative X, positive and negative Y, and positive and negative Z. Creating a cube map involves unwrapping and projecting the panoramic image onto the six faces of the cube using a specific mapping technique. This process transforms the spherical projection of the image captured by a camera into orthographic projection images in six directions: front, back, left, right, up, and down. By treating each face of the cube as an individual image, traditional camera calibration techniques

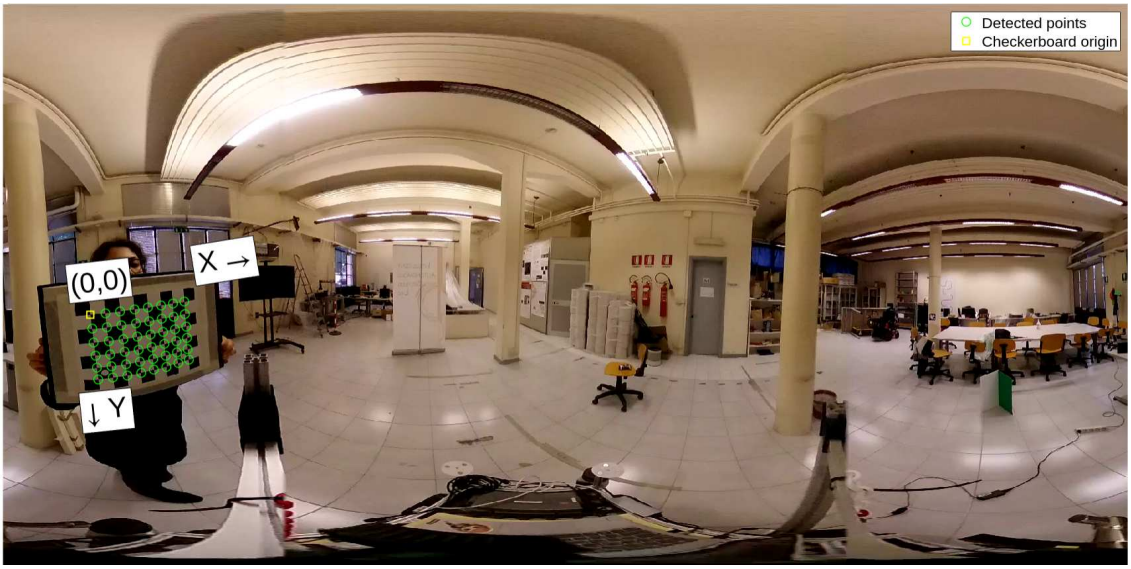

Figure 3.6: Standard calibration on the panoramic image.

can be applied to calibrate each face independently.

To begin the transformation process, the panoramic image is converted from its spherical projection to cylindrical and cubic projections, as illustrated in Figure 3.8(a). This allows for obtaining orthographic projection images of the panoramic scene in all six directions. This transformation accounts for the deviation of the camera's imaging model center from the center of each individual camera model [16].

When considering a dual-camera panoramic camera, which typically includes two ultra-wide-angle fisheye cameras positioned at the front and back, it's important to note that the optical projection of these camera lenses differs from that of regular lenses. While a regular lens follows a tangent relationship for optical projection, fisheye lens cameras primarily exhibit isometric and equiprojection characteristics. As a result, fisheye lenses offer a significantly wider field of view and introduce more distortion compared to cameras equipped with regular lenses [16].

For our research purposes, assuming that the optical centers of the integrated dual cameras align with the optical center of the camera, we can treat the two cameras as one with a larger field of view. This assumption leads to the imaging model of the camera depicted in Figure 3.8(b). Converting an equirectangular image to cubemaps involves mapping the pixels from the equirectangular projection onto the six faces of a cube:

Let's assume we have an equirectangular image with dimensions W (width) and H (height). To convert it to cubemaps, we need to compute the texture coordinates for each pixel (u, v) in

Figure 3.7: Result of transform points from the robot plane to the image plane by applying the procedure of computing H matrix on the panoramic image.

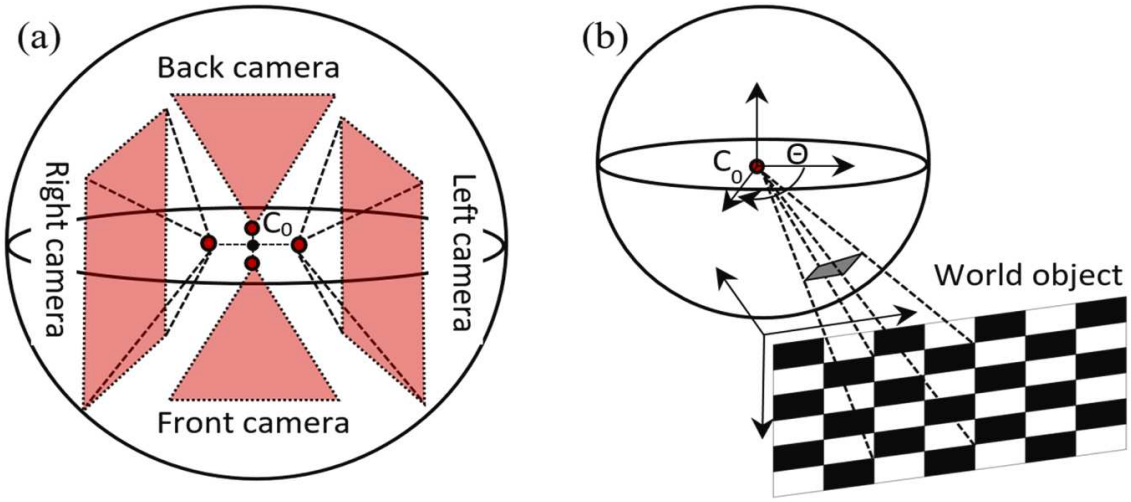

Figure 3.8: Imaging model of panoramic camera.

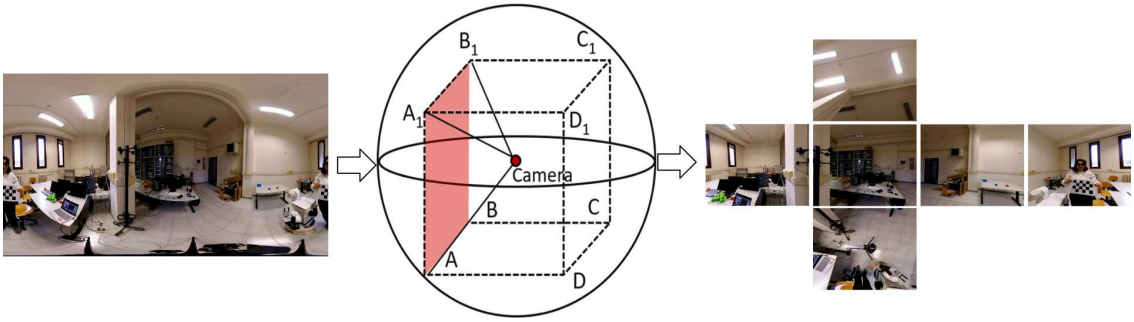

Figure 3.9: Cube projection of panoramic image.

Table 3.1: Coordinate Transformation Formulas

Faces	+X (right)	-X (left)	+Y (top)	-Y (bottom)	+Z (front)	-Z (back)
s	$\frac{-z}{x}$	$\frac{z}{x}$	$\frac{x}{y}$	$\frac{x}{y}$	$\frac{x}{z}$	$\frac{-x}{z}$
t	$\frac{-y}{x}$	$\frac{-y}{x}$	$\frac{-z}{y}$	$\frac{z}{y}$	$\frac{-y}{z}$	$\frac{-y}{z}$

the equirectangular image and map it to the corresponding face of the cube.

First, we need to calculate the 3D direction vector (x, y, z) from the cube center to the point on the unit cube's surface. This can be done using the following formulas:

$$\begin{aligned} x &= \tfrac{u}{W} \times 2 - 1 \\ y &= \tfrac{v}{H} \times 2 - 1 \\ z &= 1 \end{aligned} \tag{3.1}$$

Next, we normalize the direction vector to have a length of 1:

$$\begin{aligned} \text{length} &= \sqrt{x^2 + y^2 + z^2} \\ x &= \tfrac{x}{\text{length}} \\ y &= \tfrac{y}{\text{length}} \\ z &= \tfrac{z}{\text{length}} \end{aligned} \tag{3.2}$$

Now, we can calculate the texture coordinates (s, t) for each face of the cube based on the direction vector. The mapping formulas depend on the largest component of the direction vector:

Finally, we can calculate the pixel coordinates $(\acute{u}, \acute{v})$ in each face of the cube by multiplying the texture coordinates (s, t) by the face width $(W/4)$ and height $(H/3)$:

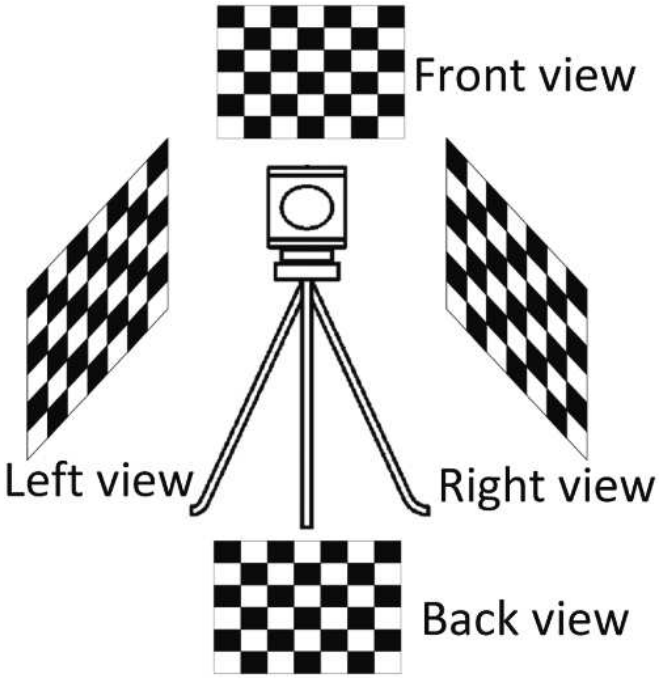

Figure 3.10: Calibration of panoramic camera [16].

$$\begin{aligned} \acute{u} &= \tfrac{s}{2} \times \left(\tfrac{W}{4}\right) + \tfrac{W}{2} \\ \acute{v} &= \tfrac{t}{2} \times \left(\tfrac{H}{3}\right) + \tfrac{H}{2} \end{aligned} \tag{3.3}$$

The given passage describes the process of mapping pixel coordinates from an equirectangular panoramic image to the corresponding faces of a cube. This mapping is done by repeating a certain procedure for each pixel in the equirectangular image. Once the equirectangular image is projected onto the six faces of a cubemap, camera calibration is performed on each face. Each face is treated as an individual pinhole camera image. However, in this calibration process, only the front, back, right, and left projection planes are calibrated. The upper and lower projection planes are typically not calibrated because they do not contain the object of interest. Figure 3.10 illustrates the calibration process. It involves capturing multiple images in the front, back, left, and right directions using a checkerboard calibration plate with a panoramic camera. This process helps calculate an intrinsic matrix for each side. The intrinsic matrix, denoted as K_{side}, represents the internal properties of the camera used to capture that specific face of the cubemap.

The intrinsic matrix K_{side} is a 3x3 matrix defined as:

$$K_{\text{side}} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

where:

- f_u and f_v are the focal lengths of the camera in the horizontal and vertical directions, respectively.
- u_0 and v_0 represent the coordinates of the optical center (principal point) of the image plane, indicating the location of the camera's center of projection.

With the intrinsic matrix K_{side} determined for each face, it can be used to compute the homography matrix. The homography matrix allows for the transformation of points from the robot plane (3D world coordinates) to the image plane (2D pixel coordinates) for labeling people in the frame. By applying the homography transformation using the appropriate intrinsic matrix K_{side} for a specific face, points in the robot plane can be accurately projected onto the corresponding image plane, enabling the labeling of individuals within the captured image. It's important to note that camera calibration is necessary to accurately estimate the intrinsic parameters, and it typically involves capturing images of a known calibration pattern and extracting the parameters from the image correspondences. The resulting intrinsic matrices provide the necessary information for subsequent transformations and analysis within the image plane.

3.6 Aligning Laser Rangefinder and Image Data: A Two-Step Mapping Process

Integrating data from a laser rangefinder and a camera requires aligning their respective coordinate systems. To address this, a two-step mapping process, as proposed by Yu et al. [68], is employed to align the laser rangefinder information with the optical image plane. The first step involves obtaining an accurate homography matrix using the camera's intrinsic matrix, transformation matrix, and rotation matrix. This matrix ensures that each data point captured by the laser rangefinder is appropriately mapped onto the coordinate system of the optical image. By establishing this alignment, the two data sources can be combined effectively. Additionally, the H matrix is calculated for each side, enabling the transformation of positions from

the laser plane to the image plane. These calibration steps ensure accurate mapping between the laser scans and the corresponding images. In the second step, since image coordinates are discrete, a grayscale interpolation method is employed for precise coordinate transformation. This interpolation method ensures that the transformed coordinates are placed accurately on the corresponding pixel points, even if the surrounding point coordinates are not integers. By accounting for the discrete nature of image coordinates, this interpolation step enhances the accuracy of mapping the laser rangefinder data onto the optical image plane. By following this two-step mapping process, we can effectively align the laser rangefinder and image data, enabling fusion and further analysis that leverages the complementary information provided by these distinct sensing modalities [68].

3.6.1 Step 1: Spatial Coordinate Transformation

In the pinhole camera model, the transformation from a point in the world frame to the image plane can be represented using a set of equations. Here we break down the equations and explain each step. The goal is to transform a 3D point (denoted as laser point (x_t, y_t, z_t) from the world frame to the image plane, obtaining its corresponding pixel coordinates (u, v). The equations are as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} f_u & 0 & \frac{u_0}{f} \\ 0 & f_v & \frac{v_0}{f} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{3.5}$$

Equaitin 3.5 relates the pixel coordinates (u, v) in the image plane to the camera coordinates (x_c, y_c, z_c) of the 3D point (x_t, y_t, z_t) introduced before expressed in the camera reference frame, for more details check Section 2.11 in Chapter 2. Here, f represents the focal length of the camera, and (u_0, v_0) denotes the principal point (the optical center of the image). The terms f_u and f_v represent the scaling factors in the x and y directions, respectively.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3.6}$$

Equation 3.5 describes the transformation from the world frame to the camera frame. The matrix on the left represents the rotation and scaling transformation, denoted as W. The vector

on the right represents the translation component, denoted as t_1, t_2, t_3.

$$\begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} . \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} + \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{3.7}$$

Equation 3.7 combines the transformations from Equation 3.5 and Equation 3.6. The left-hand side represents the scaled pixel coordinates in the camera frame, with z_c being the depth (distance from the camera to the point). The first term on the right-hand side represents the transformation from the world frame to the camera frame, and the second term represents the translation to the image plane. By using these equations, you can transform a point from the world frame to the image plane in the pinhole camera model.

3.6.2 Step 2: Grey-Scale Interpolation for Discrete Image Coordinates

After the spatial coordinate transformation, the coordinates of each pixel in the image plane may become non-integer values. To ensure accurate alignment with the exact pixel points, an interpolation method is employed. Specifically, we employed the nearest neighbor interpolation method for grayscale interpolation, as recommended by [68], to enhance the quality of our results. The process involves the following steps:

- **Obtaining the coordinates of four surrounding pixels:** The coordinates of the four pixels surrounding the laser rangefinder mapping point are determined after the spatial coordinate transformation.
- **Computing the distance to the surrounding pixels:** The distance between the laser's mapping point and each of its four surrounding pixels is calculated.
- **Substituting the laser rangefinder point coordinate:** The original coordinate of the laser rangefinder point is replaced with the coordinate of the pixel that has the minimum distance.

By employing this grey-scale interpolation method, we ensure that the laser rangefinder information is accurately mapped onto the corresponding pixel points, even when the surrounding point coordinates are not integers. This step facilitates seamless integration and precise fusion of the heterogeneous datasets [68].

3.6.3 Estimation of Intrinsic and Extrinsic Parameters

In this section, we will explain the procedure for computing the H matrix based on the transformation matrix and laser scan. First, we derive Equation 3.8 from Equation 3.7 to obtain the coordinate transform, as suggested by Yu et al. [68].

$$\begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} (f_u w_{11} + u_0 w_{31})x_t + (f_u w_{12} + u_0 w_{32})y_t + (f_u w_{13} + u_0 w_{33})z_t \\ (f_v w_{21} + v_0 w_{31})x_t + (f_v w_{22} + v_0 w_{32})y_t + (f_v w_{23} + v_0 w_{33})z_t \\ w_{31}x_t + w_{32}y_t + w_{33}z_t \end{bmatrix} + \begin{bmatrix} f_u t_1 + u_0 t_3 \\ f_v t_1 + v_0 t_3 \\ t_3 \end{bmatrix}$$

$$= \begin{bmatrix} (f_u w_{11} + u_0 w_{31})x_t + (f_u w_{12} + u_0 w_{32})y_t + (f_u w_{13} + u_0 w_{33})z_t + f_u t_1 + u_0 t_3 \\ (f_v w_{21} + v_0 w_{31})x_t + (f_v w_{22} + v_0 w_{32})y_t + (f_v w_{23} + v_0 w_{33})z_t + f_v t_1 + v_0 t_3 \\ w_{31}x_t + w_{32}y_t + w_{33}z_t + t_3 \end{bmatrix} \quad (3.8)$$

Equation 3.8 represents the transformation of coordinates from the laser rangefinder to the camera. It consists of two main parts: the transformation of the laser rangefinder coordinates to the world coordinates (represented by x_t, y_t, and z_t), and the transformation from world coordinates to camera coordinates (represented by u, v, and z_c). The transformation involves various parameters such as f_u, f_v, u_0, v_0, w_{ij}, and t_i.

Before starting the transformation process, we need to preprocess the laser rangefinder data. The laser rangefinder data is provided in polar coordinates, but for the computation of the homography matrix, it needs to be converted to Cartesian coordinates. Equation 3.9 describes the conversion of polar coordinates to Cartesian coordinates, where ρ_t represents the measured distance and θ denotes the scan angle.

$$\begin{cases} x_t = \rho_t . \cos\theta \\ y_t = \rho_t . \sin\theta \\ z_t = 0 \end{cases} \quad (3.9)$$

In their publication, Yu et al. [68] present the coordinate transformation formula (Equation 3.8) in a simplified form (Equation 3.10). This form highlights the parameters involved in the

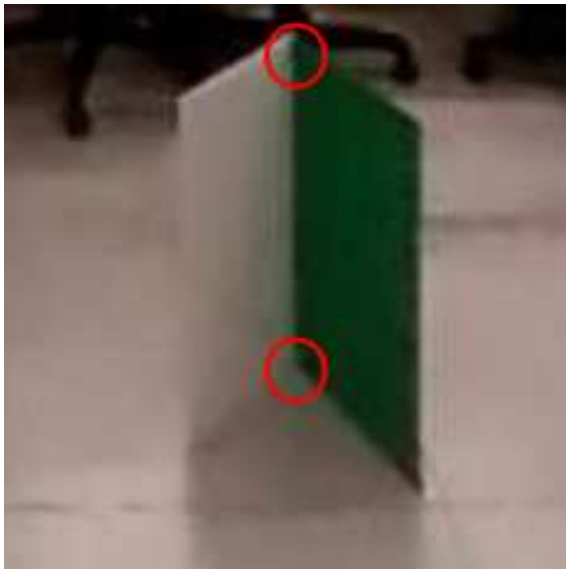

Figure 3.11: Extraction of feature line.

transformation and their relationships.

$$\begin{bmatrix} z_c u \\ z_c v \\ z_c \end{bmatrix} = \begin{bmatrix} a_0 x_t + b_0 y_t + c_0 \\ a_1 x_t + b_1 y_t + c_1 \\ a_2 x_t + b_2 y_t + c_2 \end{bmatrix} \tag{3.10}$$

In Equation 3.10, the parameters $a_0 = f_u w_{11}$, $b_0 = f_u w_{12}$, $c_0 = f_u t_1 + u_0 t_3$, $a_1 = f_v w_{21}$, $b_1 = f_v w_{22}$, $c_1 = f_v t_2 + v_0 t_3$, $a_2 = b_2 = 0$, and $c_3 = t_3$.
To determine the values of (x_t, y_t) and (u, v) in Equation 3.10, a board (depicted in Figure 3.11.a) was used. The board's intersection points with the laser rangefinder scans were extracted from RViz, representing the points in the robot plane (x_t, y_t). The image coordinates (u, v) were assumed to lie on the intersection line of the white and green boards, satisfying a linear equation (Equation 3.11).

$$Au + Bv = 1 \tag{3.11}$$

To estimate the intrinsic and extrinsic parameters of the camera, Yu et al. [68] propose a separate estimation approach. The estimation equations (Equations 3.12) is derived from Equa-

tions 3.8, 3.10, and 3.11, and they involve various parameters that need to be estimated.

$$\begin{aligned} &Af_u x_t w_{11} + Af_u y_t w_{12} + Bf_v x_t w_{21} + Bf_v y_t w_{22} + (Au_0 + Bv_0 - 1)x_t w_{31} \\ &+(Au_0 + Bv_0 - 1)y_t w_{32} + Af_u t_1 + Bf_v t_2 + (Au_0 + Bv_0 - 1)t_3 = 0 \end{aligned} \tag{3.12}$$

In Equation 3.12, we need to find the 13 parameters ($f_u, f_v, u_0, v_0, w_{11}, w_{12}, w_{13}, w_{21}, w_{22}, w_{23}$, t_1, t_2, t_3) to achieve a separate estimation of the intrinsic and extrinsic parameters. Note that we set $w_{31} = w_{32} = w_{33} = 0$ as stated in Equation 3.9 [68]. The separate estimation method considers the interrelation among the parameters, resulting in more accurate calibration. It also helps avoid convergence issues and suboptimal solutions that may arise from the mixed estimation method.

By employing the proposed separate estimation approach, we can accurately estimate the intrinsic and extrinsic parameters of the camera, leading to precise calibration and enhancing the overall quality of our study's experimental setup.

3.6.4 Identification of Characteristic Parameters (A, B, x_t, y_t)

This subsection focuses on estimating the intrinsic and extrinsic parameters of the camera used in the study. These parameters are essential for accurately determining the camera's internal characteristics and its position in the world which is essential to map laser readings into the image space. The intrinsic parameter matrix (f_u, f_v, u_0, v_0) of the camera remains constant and can be considered as known variables, which are computed through the camera calibration procedure. Therefore, in the process of estimating the transformation between the laser and image planes, the rotation matrix and translation matrix are the only parameters that need to be determined [68]. By focusing on estimating the rotation and translation matrices, we can effectively solve the transformation between the image and laser planes.

The characteristic parameters A, B, x_t, y_t, which are known parameters in Equation 3.12. These parameters are calculated using the characteristic line $Au + Bv = 1$ and points on the separating intersection described in Equation 3.11. By identifying characteristic points on the characteristic line of $Au + Bv = 1$, the values of A and B can be determined. The laser rangefinder data (x_t, y_t) represents the maximum point of the intersection points of the scanning plane and the calibration plate as shown in Figure 3.12, ensuring they fall on the line Au + Bv = 1. These points are extracted from the intersection with the calibration plate. To solve and estimate the other parameters, the paper [68] states that a significant amount of experimental data is required. This data is obtained by varying the relative pose between the points and the

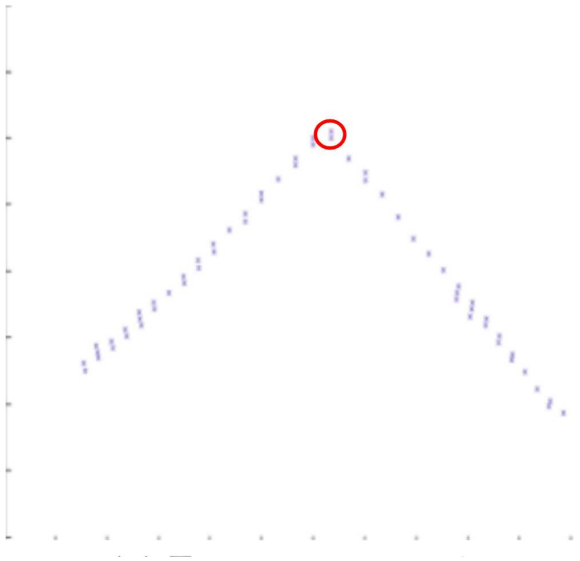

Figure 3.12: Showing (x_t, y_t) on the scan plane.

camera in different experimental setups, such as changing the object's inclination or adjusting the distance between the object and the camera. It is crucial to collect laser rangefinder and optical image data synchronously. The paper suggests that at least 15 groups of effective data are needed to establish the equations for estimating and solving the other parameters.

3.6.5 Optimizing the calibration parameters for data fusion

The process of optimizing calibration parameters for data fusion involves selecting appropriate initial values and then optimizing the parameters using certain constraints. Gaussian elimination, an algorithm for solving linear equations, is used to determine suitable initial values for the optimization process. This helps improve the performance of the parameter optimization algorithm by allowing it to search for solutions within a specific range of initial values. However, the presence of measurement errors and noise prevents the coefficient matrix from meeting the requirement of non-zero solutions. To address this, random noise is added to the equations to ensure non-zero solutions. The equations, represented as Equation 3.13, are then solved using Gaussian elimination to obtain the initial value [68].

$$y = Mx + e \tag{3.13}$$

By collecting values for (x_t, y_t) and (u, v) from different scenes for the border, we can obtain the necessary information to solve the H matrix for Equation 3.12. This can be achieved through the use of Singular Value Decomposition (SVD), which provides an initial value for the H matrix. Once the initial value is determined, the parameters' optimization process begins. The optimization is performed using two constraints. The first constraint aims to minimize the sum of squared errors, as shown in Equation 3.14, where ε = Au + Bv - 1 [68].

$$\min \left\{ \sum_{i=1}^{n} |a_i u + b_i v - 1|^2 \right\} \tag{3.14}$$

The second constraint involves minimizing the sum of distances from points to lines, as shown in Equation: 3.15.

$$\min \left\{ \sum_{i=1}^{n} |a_i u + b_i v - 1| / \sqrt{a_i^2 + b_i^2} \right\} \tag{3.15}$$

The optimization of intrinsic and extrinsic parameters is achieved using nonlinear least squares and nonlinear Gauss-Newton methods, considering the aforementioned constraints. The parameters u and v in Equations 3.14 and 3.15 are calculated based on the given formulas [68].

The optimized initial values obtained through singular value decomposition (SVD) are used as a starting point. Initially, the parameters are optimized using the nonlinear least squares method under the first constraint (Equation 3.14). Then, the optimized parameters are further optimized using the nonlinear Gauss-Newton method under the second constraint (Equation 3.15). The parameters' matrix, denoted as H, is obtained through the nonlinear least squares method, and it serves as an initial solution for the nonlinear Gauss-Newton method. This re-optimization process takes into account the constraint presented in Equation 3.15. The final result, denoted as H, is obtained by optimizing the parameters using both the nonlinear least squares and nonlinear Gauss-Newton methods. This optimized parameter matrix can then be used in a real-time data fusion system for subsequent computations [68].

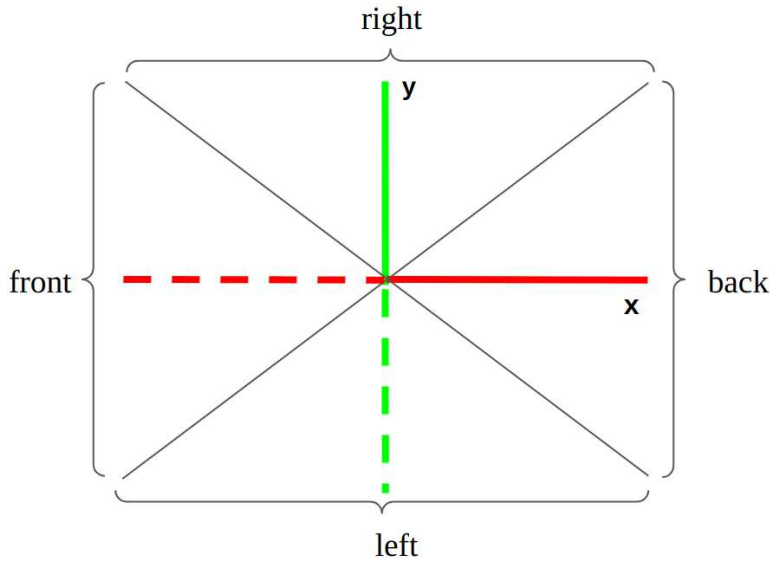

Figure 3.13: Define sides for positions in laser plane.

3.7 People Detection in Image and Laser Planes

In this section, we describe the process of detecting people in both the image and laser planes. The procedure includes camera calibration, and transforming positions between the laser and image planes. Firstly, camera calibration is performed to obtain the necessary parameters (f_u, f_v, u_0, v_0) for each side of the camera (front, left, right, and back). Next, we utilize the DR-SPAAM algorithm to detect people within the laser scan ranges corresponding to each person detected by YOLOv7 in the image frame, since noises of detection by DR-SPAAM was high we utilize YOLOv7 to filter noises. The detected people, along with their associated scan and image data, are saved in a CSV file. Each person is assigned a unique identifier for tracking purposes.

To label the people in each frame with their positions, the following steps are followed:

$$\begin{aligned} z_c &= w_{31}x_{ti} + w_{32}y_{ti} + t_3 \\ u &= [(f_u w_{11} + u_0 w_{31})x_{ti} \quad +(f_u w_{12} + u_0 w_{32})y_{ti} \quad +f_u t_1 \quad +u_0 t_3]/z_c \\ v &= [(f_v w_{21} + v_0 w_{31})x_{ti} \quad +(f_v w_{22} + v_0 w_{32})y_{ti} \quad +f_v t_2 \quad +v_0 t_3]/z_c \end{aligned} \tag{3.16}$$

1. Retrieve the image, the related laser scan, and the corresponding DR-SPAAM data for

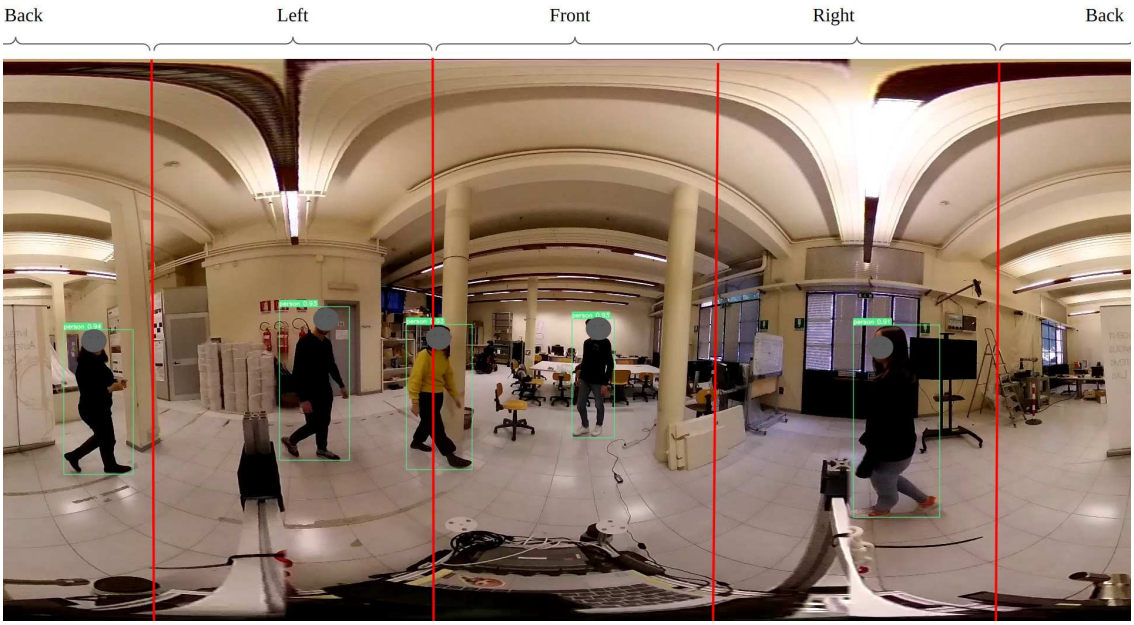

Figure 3.14: Sample of the first scenario in the detection part of the framework.

Figure 3.15: Sample of the second scenario in the detection part of the framework.

the frame under consideration.

2. Project the image onto a cube reprojection, which converts the panoramic image into a regular image for each of the four sides (front, left, right, and back).

3. Detect all persons in the panoramic image to check a person not being present on multiple sides, as depicted in Figure 3.14. This allowed us to determine the number of people within the frame. We then checked the bounding boxes for each person to ensure that they fell within the boundaries of each side. If a person with the yellow shirt in Figure 3.14, was present on two sides, we assessed the percentage of the person that belonged to each side. Based on this analysis, we removed the person from the side with the lower percentage. In the event that the percentages were equal, we randomly removed the person from one side. This approach was adopted to prevent our framework from assigning two labels to a single person.

4. Divide the positions into different sides based on Figure 3.13, which illustrates the sides in the laser plane. This division is performed using the algorithm described in Pseudocode 3.1 to determine which positions are associated with each side of the image.

5. Iterate over the sides of the image and apply the YOLOv7 object detection algorithm on each side image to detect people within the respective image regions.

6. To obtain pixel coordinates (u, v) and transform positions to the image plane, we utilize Equation 3.16. This equation incorporates the specific camera parameters (f_u, f_v, u_0, v_0) for each side (front, left, right, and back). Moreover, we leverage the H matrix associated with each side to accurately transform the DR-SPAAM data corresponding to that side. Additionally, we apply Gray-Scale interpolation to ensure precise mapping of positions to discrete image coordinates.

7. Check the intersection of the (u, v) coordinates of a position with the bounding boxes of people detected on the corresponding side. If an intersection is found, select the position with the associated DR-SPAAM data for that person. Remove the person and their DR-SPAAM position from the respective lists.

8. In cases where multiple points fall within the bounding box for a person, prioritize the point that is closer to the origin axes of the robot. This selection is based on computing the Euclidean distance, as defined in Equation 3.17. This step is necessary because, in certain scenarios, a distant point detected by the laser may erroneously fall inside a bounding box, it may a noise point in the bounding box to be closer to the origin axis of the robot which in the scenario discussed in the Chapter 4. For instance, as illustrated in Figure 3.16, there are two points within the bounding box of the person ID 3 in the image. The top point is related to an object located behind the person, while the other point, near the person's feet, accurately corresponds to the person's position.

9. If no intersection between DR-SPAAM and the bounding boxes is found in step 6, to have a position for the person to label it, convert all laser scans related to that side using Equation 3.16, and select the point closest to the origin axes of the robot based on the Euclidean distance.

10. In the scenario addressed the situation where the bottom part of a person was covered by the robot body, as illustrated in Figure 3.15. Although the image could not capture the person's position, the scan data could provide the person's location. To handle this case, we considered the position of a point within the range of u_0 (the u pixel of the left point of the bounding box) and u_1 (the u pixel of the right point of the bounding box) as an indicator of the person's position in the robot plane.

11. Furthermore, through the development phase, we identified that certain sides required an offset. We added this offset to the u and v coordinates for each side based on the experiences gained during training. Additionally, we observed that if a person was in close proximity to the robot, with |x| < 1.3 and |y| < 1.3, they would appear on the bottom side. As we did not have a homography matrix for this case, we decided to add an offset to the x and y coordinates before transforming them to u and v. However, for labeling purposes, we considered the original values for x and y. These offsets were determined based on manual observations of selected points and images for each side.

12. Save the positions, labeled by ID, for each frame in a YAML file for further analysis or processing.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.17}$$

By following this step-by-step process, we can accurately detect people in both the image and laser planes, associate their positions with real-world coordinates, and store the relevant data for subsequent analysis or use.

3.8 People Tracking and Dataset Creation

After the initial detection step, we obtain a YAML file that contains the positions of people in the real world for each frame. To track the movement of people over consecutive laser scans, we employ two methods: the Unscented Kalman filter and the GNN method. The Unscented Kalman filter is utilized to predict and handle the uncertainty associated with people's motion

Algorithm 3.1 Categorizing Points Based on Positions

```
0: procedure SIDES_POINTS(dr_spaam)
0:     back_xy ← empty list
0:     left_xy ← empty list
0:     front_xy ← empty list
0:     right_xy ← empty list
0:     for each d in dr_spaam
0:         x ← d[0]
0:         y ← d[1]
0:         if y ≥ 0
0:             if x > 0 and x ≥ y
0:                 append (x, y) to back_xy
0:             else if 0 < x < y
0:                 append (x, y) to right_xy
0:             else if x < 0 and |x| < y
0:                 append (x, y) to right_xy
0:             else if x < 0 and |x| ≥ y
0:                 append (x, y) to front_xy
0:             end if
0:         else
0:             if x > 0 and x ≥ |y|
0:                 append (x, y) to back_xy
0:             else if 0 < x < |y|
0:                 append (x, y) to left_xy
0:             else if x < 0 and |x| < |y|
0:                 append (x, y) to left_xy
0:             else if x < 0 and |x| ≥ |y|
0:                 append (x, y) to front_xy
0:             end if
0:         end if
0:     end for
0:     return back_xy, left_xy, right_xy, front_xy
0: end procedure

0: dr_spaam ← [(1, 2), (−3, 4), (0, −2), (−2, −3)] {Example input}
0: back_xy, left_xy, right_xy, front_xy ← SIDES_POINTS(dr_spaam) {Call the function}

0: Output:
0: Back: back_xy
0: Left: left_xy
0: Right: right_xy
0: Front: front_xy =0
```

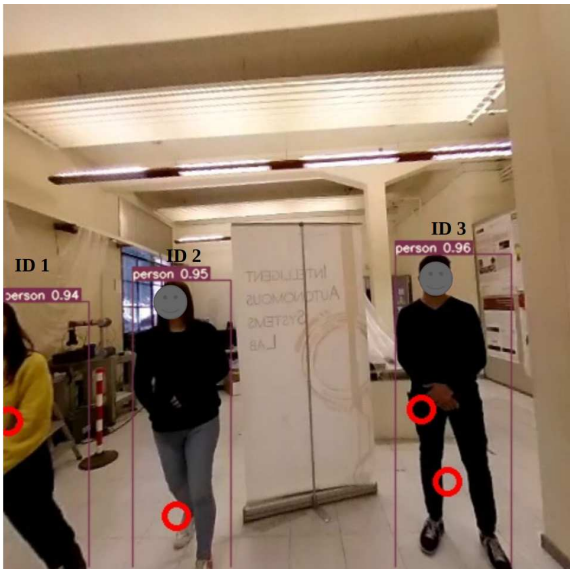

Figure 3.16: Specific scenario for people detection.

between consecutive scans. It is capable of generating estimated positions for each person in every scan, even in challenging scenarios such as occlusion or when position sensing is temporarily unavailable. This filter allows us to maintain accurate tracking of people over time.

To resolve the data association problem between consecutive scans, we employ the GNN method. This method helps in associating the detections from one scan with the corresponding individuals in the subsequent scan. By using the GNN method, we can ensure that the correct tracker is assigned to each detected person, enabling consistent and accurate tracking across frames. In our proposed algorithm, the Unscented Kalman filter serves as the basis for the people tracking process. It effectively handles various situations, including occlusions and temporary lack of position sense, to generate estimated person positions for each scan. The algorithm takes advantage of both the predictive capabilities of the Kalman filter and the data association abilities of the GNN method to ensure robust and reliable tracking results.

1. Load the input data: Read the object measurements for each frame from the YAML file from the detection part.
2. Initialize the necessary variables:
    - Set the parameters for the algorithm, such as the number of states and measurements, process and measurement noise variances, time step, and loss association threshold.

- Create empty lists to store the positions and IDs of removed objects.

3. Define the state transition function:

    - The state transition function takes the current state vector and time step as inputs.
    - Implement the function based on a motion model suitable for the application, such as a constant velocity model.
    - The function should return the predicted state vector for the next time step.

4. Define the measurement function:

    - The measurement function takes the current state vector as input.
    - Implement the function based on the available measurements for each object, such as position.
    - The function should return the measurement vector corresponding to the state vector.

5. Define the function to handle the loss of object IDs:

    - This function is responsible for removing filters associated with lost IDs and performing any additional handling or cleanup required.
    - Remove the filter from the list of filters and handle any necessary post-processing steps, such as saving or logging information about the lost track.

6. Initialize the filters:

    - Create an empty list to store the filters for each object.
    - Iterate over each frame in the input data.
    - Extract the measurements for the current frame.
    - If no filters exist:
        - Iterate over each measurement and create a new Unscented Kalman Filter (UKF) for each object.
        - Set the initial state and covariance matrix for the filter.
        - Set the process and measurement noise covariance matrices.

- Assign a unique object ID to each filter.
- Initialize the loss association counter and other relevant properties for the filter.
- Add the filter to the list of filters.

- If filters already exist:
  - Iterate over each filter and predict the next state based on the state transition function and time step.
  - Find the nearest measurement for each filter using GNN.
  - If the distance is below a threshold, update the filter with the nearest measurement.
  - Increment the appearance frame counter for each filter.
  - If a filter does not have an associated measurement, increase the loss association counter.
  - Handle the case of lost measurement association or new ID assignments by creating new filters or reassigning existing filters.
  - If a filter exceeds the loss association threshold, remove it from the list of filters using the handling loss of id function.

7. To create the dataset, save the tracker ID and its corresponding position for each frame in a YAML file.

8. Repeat the process for all frames in the input data.

In conclusion, the process of people detection in both the image and laser planes involves camera calibration, transforming positions between the laser and image planes, and utilizing the DR-SPAAM algorithm. By following a step-by-step approach, we can accurately detect people, associate their positions with real-world coordinates, and store the relevant data for further analysis. Once the initial detection is complete, the next step is to track the movement of people over consecutive laser scans. To achieve this, we employ two methods: the Unscented Kalman filter and the GNN method. The Unscented Kalman filter predicts and handles the uncertainty associated with people's motion, while the GNN method resolves the data association problem between consecutive scans.

By combining the predictive capabilities of the Kalman filter and the data association abilities of the GNN method, our algorithm ensures robust and reliable tracking results. The

Unscented Kalman filter generates estimated person positions for each scan, even in challenging scenarios such as occlusion or temporary lack of position sensing. Through the use of these methods, we can track people's movements and create a dataset that contains the positions of people in the real world for each frame. This dataset can be valuable for various applications, such as behavior analysis, crowd management, or human-robot interaction.

4

Evaluation and Experimental Results

This chapter focuses on the evaluation of the auto-labeling framework proposed in Chapter 3 for creating a dataset for tracking people with an omnidirectional camera labeled with their real position in the world. The evaluation aims to assess the effectiveness and performance of the framework in various aspects of dataset creation. The first part of the chapter provides an overview of the data collection process and the characteristics of the recorded data. It discusses the methodology used for data collection, including the setup of the data collection environment and the deployment of the necessary sensors, particularly the omnidirectional camera. The chapter also provides considerations about lighting conditions, camera positioning, and trajectories of walking people, which are crucial factors in capturing high-quality data. Following the data collection, the recorded data undergoes preprocessing to enhance its quality and usability. This section of the chapter outlines the experimental implementation of steps of the framework, and evaluation for each step. It also addresses any challenges or limitations encountered during the recording stage and presents solutions that were implemented to overcome them.

The experimental calibration of the camera and laser components is essential for ensuring the accuracy and reliability of the auto-labeling framework. The chapter dedicates a section to describe the experimental calibration process, including the calibration setup, calibration targets, and calibration patterns used. The evaluation of the calibration results is discussed, emphasizing the achieved accuracy and identifying any potential sources of error. The evaluation and results of the framework's detection and tracking capabilities form a significant part

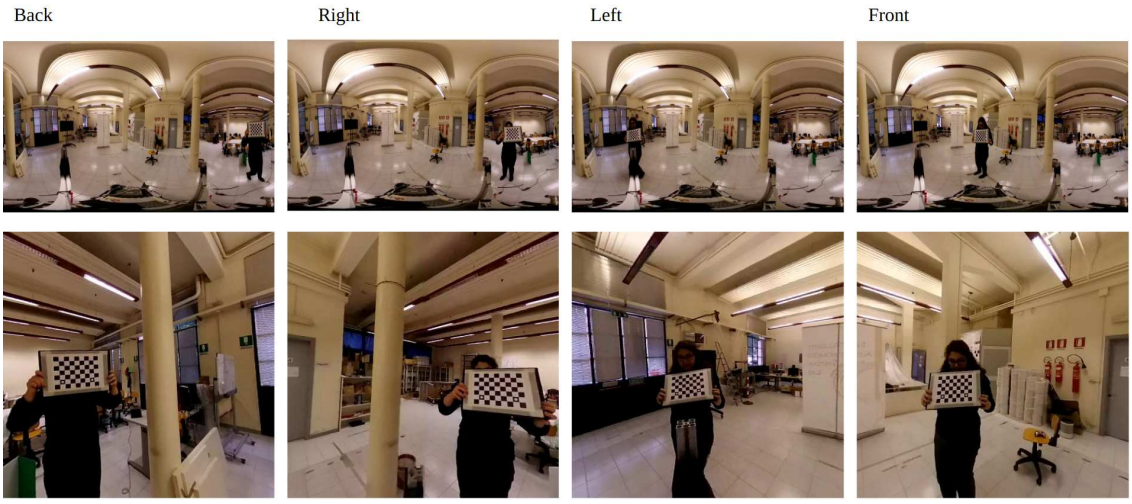

Figure 4.1: Recording data for camera calibration.

of the chapter. The performance of the framework is evaluated in terms of detection accuracy, tracking robustness, and computational efficiency. Quantitative metrics such as precision, recall, and F1 score are utilized to assess the effectiveness of the framework. This evaluation section also addresses any limitations or challenges encountered during the evaluation process. Lastly, the chapter presents the results of the dataset creation using the auto-labeling framework. It discusses the quality and completeness of the generated dataset, emphasizing its utility for training and evaluating tracking algorithms. A comparison is made between the auto-labeled dataset and manually labeled ground truth data to assess the accuracy and reliability of the auto-labeling process.

4.1 Data Collection and Characteristics

For the data collection process, we utilized the rosbag recording tool to capture data from three topics: `/odom`, `/scan`, and `/theta-camera`. The recorded data encompassed three different scenarios involving the tracking of five individuals within the lab environment.

The first recorded video had a duration of 49.8 seconds, where the robot remained stationary while the people moved around it. This scenario involved various changes in the individuals' directions, occlusions, brief stops, and instances of individuals leaving the environment. The second video, spanning 96 seconds, featured both the movement of all people and the robot itself. The robot was controlled by a person and navigated around the room, exhibiting dif-

ferent trajectories and scenarios. This scenario included challenges such as occlusions, and individuals leaving and re-entering the frame. In the third video, which lasted 165 seconds, the same scenario as the second one was replicated. However, in this case, the robot left the initial room and transitioned to an adjacent room, resulting in a change in the environment. Afterward, the robot returned to the first room, completing the cycle. This scenario encompassed the challenges present in the second video, along with the added complexity of transitioning between different environments.

Additionally, a rosbag was recorded to facilitate camera calibration. For this purpose, the camera remained in a fixed position, and a checkerboard pattern was moved around the robot, capturing data from different perspectives for each side of cubemapping panoramic image as shown in Figure 4.1. The checkerboard was placed at varying distances, and it was flipped horizontally and vertically to collect sufficient data for each side of the cube projection, as discussed in Chapter 3.

Furthermore, data was recorded using a green and white board, as depicted in Figure 3.11. The board was moved close and far from the camera, and different angles were explored for each side. This data collection process allowed for the acquisition of diverse perspectives and variations in the board's appearance. Overall, the data collection process encompassed various scenarios, movements, occlusions, and changes in the environment to capture a comprehensive range of situations for the evaluation and experimentation of the auto-labeling framework.

4.2 Experimental Camera Calibration

In our camera calibration experiment, we conducted a comprehensive procedure aimed at achieving highly accurate calibration results. To record the necessary data, we utilized a rosbag while systematically moving a checkerboard around the camera. The initial step involved converting the recorded data into individual frames, resulting in a series of panoramic images. To streamline the calibration process, we implemented cube projection, which allowed us to save four sides of the cube for each frame. To ensure efficient data management, we organized the images into separate folders for each side of the cube, making it easier to retrieve and analyze them during the calibration process. We employed the powerful camera calibration Matlab Toolbox [1], which provided us with extensive capabilities for processing the captured images.

[1] https://it.mathworks.com/help/vision/ug/using-the-single-camera-calibrator-app.html

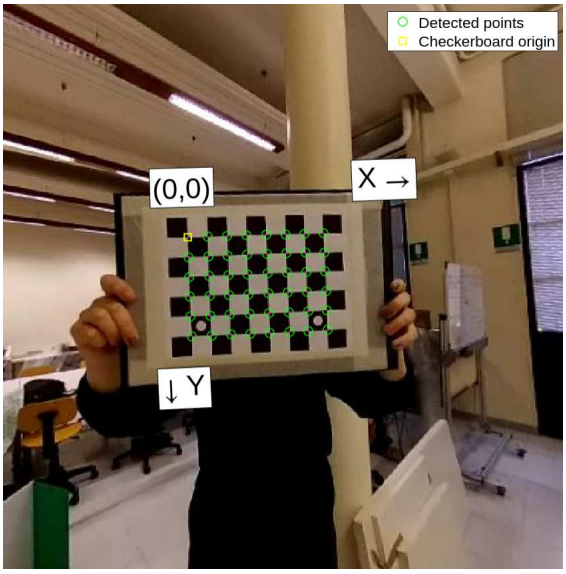

Figure 4.2: Sample of corner detection of Matlab Camera Calibration toolbox.

4.2.1 Implementation and Evaluation of Calibration

During the implementation stage, we utilized the Matlab Toolbox to analyze the images and identify the subset that exhibited accurate corner detections as depicted in Figure 4.2. By carefully selecting these images, we ensured the use of a high-quality dataset for the subsequent calibration process. Through meticulous examination, we determined that 24 images were suitable for calibrating the back side, 26 images for the left side, 21 images for the front side, and 20 images for the right side.

To quantitatively assess the accuracy of our calibration results, we calculated the intrinsic matrix for each side and evaluated the performance using the Mean Reprojection Error (MRE), which serves as a crucial metric. The MRE is a metric used to evaluate the accuracy of a 3D reconstruction or camera pose estimation algorithm. It measures the average distance between the projected 3D points and their corresponding 2D points in the image plane. The mathematical formula for calculating the Mean Reprojection Error is as follows:

$$MRE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{n_i}\sum_{j=1}^{n_i}\left(u_{ij}-u'_{ij}\right)^2+\left(v_{ij}-v'_{ij}\right)^2}$$

Table 4.1: Estimate accuracy for camera calibration for each side.

Sides	Back	Front	Left	Right	Panoramic image
Mean Reprojection Error	0.1905	0.2035	0.2061	0.2276	0.6586

Where:

- **MRE** represents the Mean Reprojection Error.
- **N** is the total number of 3D points or correspondences.
- n_i is the number of 2D points or correspondences for the ith 3D point.
- n_i is the number of 2D points or correspondences for the ith 3D point.
- u_{ij} **and** v_{ij} are the x and y coordinates, respectively, of the jth 2D point corresponding to the ith 3D point in the original image.
- $\acute{u}_{ij}$ **and** $\acute{v}_{ij}$ are the x and y coordinates, respectively, of the jth 2D point corresponding to the ith 3D point in the projected or reconstructed image.

The formula calculates the Euclidean distance between the projected 2D points ($\acute{u}_{ij}$, $\acute{v}_{ij}$) and their original 2D points (u_{ij}, v_{ij}). This calculation is performed for each 3D point and then averaged over all 3D points to obtain the Mean Reprojection Error.

The MRE provides a quantitative measure of the accuracy of the reconstruction or camera pose estimation algorithm, with lower values indicating better accuracy.

Table 4.2 provides a summary of the obtained results, showcasing the intrinsic matrix values for each side along with their respective Mean Reprojection Errors. From the table, we observe that the Mean Reprojection Errors for the different sides are as depicted in Table 4.1:

These results indicate that our camera calibration process yielded accurate estimations of the intrinsic matrix for each side. The Mean Reprojection Error for all sides falls within the range considered as good calibration based on industrial practical standards (i.e., MRE less than 1 or 2 pixels) [69] and MRE for all sides are significantly lower than the MRE for the standard calibration on the panoramic images. This level of calibration precision allows for the precise mapping of three-dimensional objects onto two-dimensional images, enhancing the overall accuracy of our camera system.

Table 4.2: Camera Calibration Results for Different Sides

Sides	f_u	f_v	u_0	v_0
Back	250.00142	253.95530	239.73133	246.91707
Front	239.72036	242.38976	237.57136	245.03967
Left	248.56713	249.78301	242.942149	233.23526
Right	253.39937	247.43437	246.43457	239.28797

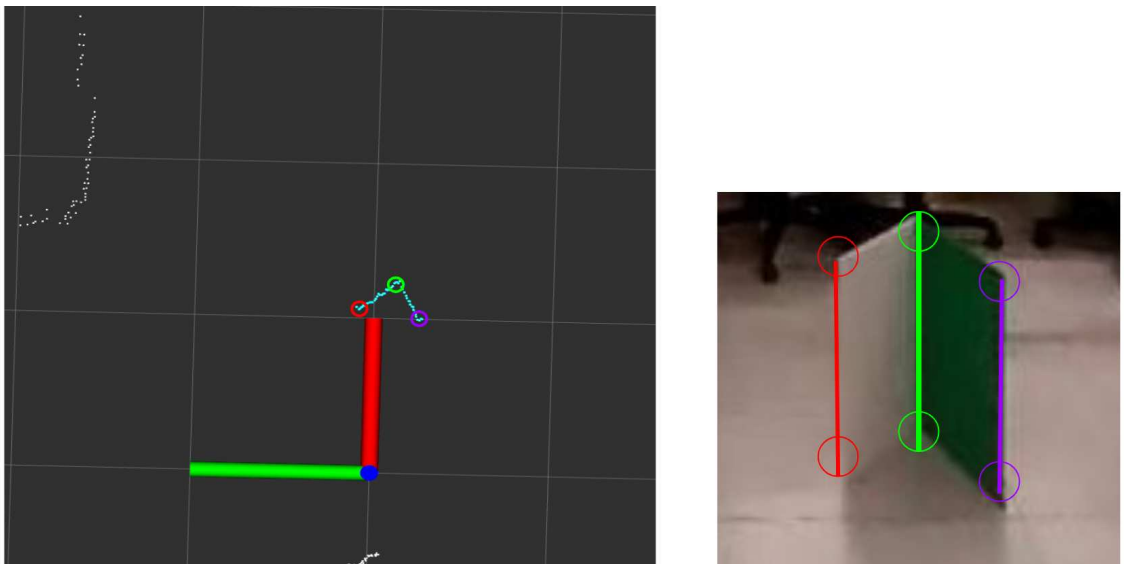

Figure 4.3: Sample of extracting points from the RViz.

4.3 Experimental Camera Intrinsic and Extrinsic Calibration

In order to calibrate the laser and obtain the homography matrix, we followed the following steps:

1. Saving rosbag as frames: We first saved the rosbag data as frames, capturing images of the green and white board to create a panoramic image.
2. Intrinsic matrix for camera calibration: To compute the homography matrix, we required the intrinsic matrix for camera calibration. This process involved mapping the panoramic images to six sides using cubemapping.
3. Calibration of laser using board images: Similar to the camera calibration, we applied the same procedure to the board images for laser calibration. Each panoramic image,

divided into six sides with cubemapping, was reprojected. We then manually extracted pixels from these images for the lines indicated in Figure 4.3 of the board, three green, red, and purple lines. This allowed us to compute A and B using Equation 3.11.

4. Extraction of scan points: From RViz, we manually extracted the positions of points where the laser scan interacted with the board, as shown in Figure 4.3. These points were used as x_t and y_t in Equation 3.16 to compute (u,v).

5. Computing H Matrix: To compute the H matrix, we started by considering Equation 3.11. For each side, we first calculated an initial value for H, denoted as H_0, using the Singular Value Decomposition (SVD) Solver in Matlab [2]. We then minimized the constraints in Equation 3.14 using the function $H_1 = fminsearch(H_0)$ in Matlab. Finally, we obtained the optimized form of H by considering the constraints of Equation 3.15 and using the function $H = lsqnonlin(H_1)$in Matlab.

6. Computing u and v: The values of u and v in Equations 3.14 and 3.15 were computed using Formula 3.16. The points extracted from RViz, representing the scan incidents with the board, were used as x_t and y_t. The values of a_i and b_i in these equations corresponded to A and B computed for each frame.

[2] https://it.mathworks.com/help/dsp/ref/svdsolver.html

Table 4.3: Number of points extract for each side for laser calibration.

Sides	Back	Front	Left	Right
Number of points	70	67	73	137

The number of points extracted from the board and RViz for each side are as shown in Table 4.3.

The obtained results for H_0, H_1, and H corresponding to each side are as follows:

Back Side H Matrix:

$$H_0 = \begin{bmatrix} 0.052972 & -0.01701 & 0.0063758 \\ 0.17646 & 0.0093073 & 0.062791 \\ 0.039367 & 0.39968 & -0.042932 \end{bmatrix} H_1 = \begin{bmatrix} 0.15887 & -0.036618 & -0.021383 \\ 0.025895 & 0.030872 & 0.16751 \\ -0.035063 & -0.16757 & 0.0027809 \end{bmatrix}$$

$$H = \begin{bmatrix} 0.15888 & -0.036621 & -0.021383 \\ 0.025895 & 0.030874 & 0.16751 \\ -0.035062 & -0.16757 & 0.002782 \end{bmatrix}$$

Left Side H Matrix:

$$H_0 = \begin{bmatrix} 0.052972 & -0.01701 & 0.0063758 \\ 0.17646 & 0.0093073 & 0.062791 \\ 0.039367 & 0.39968 & -0.042932 \end{bmatrix} H_1 = \begin{bmatrix} 0.15887 & -0.036618 & -0.021383 \\ 0.025895 & 0.030872 & 0.16751 \\ -0.035063 & -0.16757 & 0.0027809 \end{bmatrix}$$

$$H = \begin{bmatrix} 0.15888 & -0.036621 & -0.021383 \\ 0.025895 & 0.030874 & 0.16751 \\ -0.035062 & -0.16757 & 0.002782 \end{bmatrix}$$

Right Side H Matrix:

$$H_0 = \begin{bmatrix} 0.74478 & -0.26856 & 1.4652 \\ 2.985 & 0.10986 & 0.69111 \\ 0.58605 & -5.5248 & -0.30102 \end{bmatrix} H_1 = \begin{bmatrix} 1.3639 & -0.33246 & -0.18641 \\ 0.21498 & 0.26688 & 1.3911 \\ -0.24866 & 1.0998 & -0.045703 \end{bmatrix}$$

$$H = \begin{bmatrix} 1.3646 & -0.33852 & -0.18656 \\ 0.21548 & 0.26631 & 1.3902 \\ -0.23934 & 1.1006 & -0.037212 \end{bmatrix}$$

Front Side H Matrix:

$$H_0 = \begin{bmatrix} -0.19935 & -0.9924 & 0.31822 \\ 0.67668 & 0.96891 & -0.17065 \\ 0.072209 & -0.016932 & -0.23113 \end{bmatrix} H_1 = \begin{bmatrix} -0.27244 & -1.1756 & 0.64667 \\ -0.04813 & 1.1734 & -0.24977 \\ -0.040278 & -0.023284 & -0.27421 \end{bmatrix}$$

$$H = \begin{bmatrix} -0.27263 & -1.1756 & 0.64677 \\ -0.048135 & 1.1741 & -0.24661 \\ -0.039707 & -0.023353 & -0.27371 \end{bmatrix}$$

4.3.1 Evaluation of Laser Calibration

To evaluate the H matrices obtained for each side and frame, we followed a simple procedure. Using Formula 3.13, we converted the (x, y) coordinates from the robot plane, which we extracted from RViz for each side, to (u, v) coordinates on the image plane. Then, we plotted the corresponding pixel on the image. If the plotted point correctly intersected with the line on the board, it was considered a correct match, as depicted in Figure 4.4.

To evaluate the accuracy of the laser calibration, we performed a transformation of the extracted points from the robot plane to the image plane using Equation 3.16. Subsequently, we determined the number of correct points that intersected the associated line of the board on

Figure 4.4: Sample of transforming point from the robot plane to image plane to evaluate H matrix.

Table 4.4: Number of points extract for each side for laser calibration.

Sides	Back	Front	Left	Right
Accuracy	100%	94%	100%	91%

the image. This count was then divided by the total number of points for that side. By doing so, we were able to compute the accuracy of the laser calibration. The accuracy values obtained for each side are presented in Table 4.4.

4.4 Evaluation of Detection Part of the Framework

In this phase, we conducted a thorough evaluation of the detection and tracking framework we developed. The evaluation was performed using three recorded rosbag files, where the first two videos were utilized during the framework development and parameters tuning, while the third video served as the evaluation dataset. To ensure accurate synchronization between the scan data and image frames, we implemented a Robot Operating System (ROS)code that subscribed to relevant topics for both scan and image data. By comparing the header timestamps of the two topics, we were able to save the corresponding image frame and related scan data with the same ID, ensuring precise alignment.

Once synchronization was achieved, we utilized the DR-SPAAM algorithm with a detection threshold of 0.3 to identify individuals using laser scans as input. The selection of this threshold was based on stochastic analysis, as depicted in Figure 4.5. By reducing the threshold, the number of True Positive (TP) increased, but so did the number of False Positives (FPs) and False Negatives (FNs). However, below 0.3, the increase in TPswas not significant, while FP increased significantly. This would result in excessively long runtimes if we were to consider thresholds like 0.1 or 0.2. Therefore, we chose 0.3 as the threshold to ensure that the majority

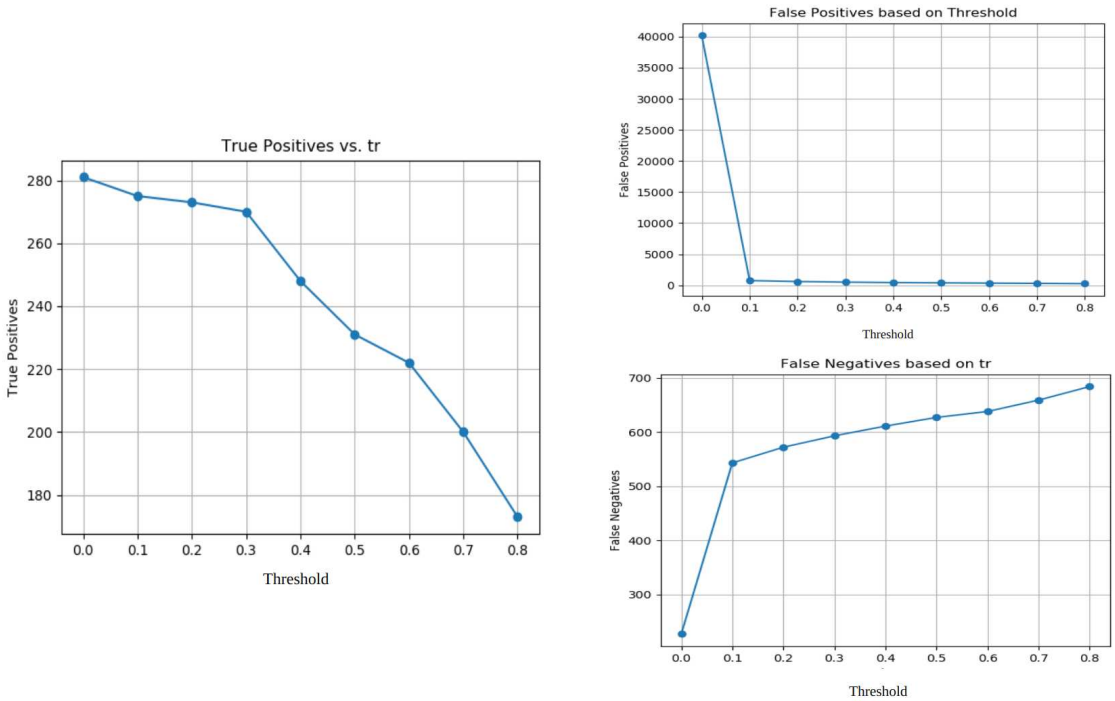

Figure 4.5: Plot of FP, FP, and False Negative for different thresholds of DR-SPAAM detection.

of people within the robot's field of view were detected by DR-SPAAM. After the detection process, we proceeded to apply our framework to the recorded data for further processing and analysis.

To assess the performance of our framework, we obtained ground truth data by manually labeling 200 frames from the third video using RViz. These labels meticulously specified the positions of people on the robot plane. Subsequently, we used this ground truth data to calculate precision, recall, and F1 score, which are commonly used metrics for evaluating detection algorithms.

Precision measures the proportion of FP detections out of the total detections made by the algorithm. It is calculated using the following formula:

$$\text{Precision} = \frac{\text{TPs}}{\text{TPs} + \text{FPs}}$$

Recall, also known as sensitivity, measures the ability of the algorithm to correctly identify positive instances out of the actual positive instances present in the dataset. It is calculated using the following formula:

$$\text{Recall} = \frac{\text{TPs}}{\text{TPs} + \text{FNs}}$$

F1 score is the harmonic mean of precision and recall, providing a balanced measure of the algorithm's overall performance. It is calculated using the following formula:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

In the detection phase, the positions of each person in each frame were saved without considering their individual IDs which are assigned later by the tracking system. The positions were stored in a YAML file, following the format:

```
frame 0:
- id0:
    x: 2.6052408912980587
    y: 0.7479766180888706
- id1:
    x: 0.5246997404650402
    y: -1.4093692363531574
- id2:
    x: -0.7034259082088902
    y: -0.05016346038490271
- id3:
    x: 0.7124919102053732
    y: 2.143921563269925
- id4:
    x: 1.8447878729708156
    y: 1.9597255230080062
```

In the ground truth YAML file, each person was assigned a unique ID, and their positions in each frame were specified based on their respective IDs. To evaluate the accuracy of the detection data, we employed the Global Nearest Neighbor (GNN) distance calculation between the positions in the detection data and the ground truth data.

The GNN distance was computed using the following formula:

$$\text{GNN distance} = \min_{i} \left\| \text{detection position} - \text{ground truth position}_{i} \right\|$$

In our study, the position detected by DR-SPAAM for a person is represented by a point, which corresponds to the red circle in Figure 4.6. If DR-SPAAM fails to detect the position of a person, our algorithm assigns the closest position to the origin axis of the robot within the bounding box to that person. This assigned position is indicated by the blue circle in Figure 4.6. The ground truth position for the person is denoted by the green circle in Figure 4.6. To determine the ground truth position, we select a point that lies in the middle with respect to the person. For the purpose of computing TPs, FPs, and FNs, we establish a threshold of 0.2. This threshold was determined by randomly selecting several points and measuring the distance between the points detected by DR-SPAAM or our algorithm and the corresponding ground truth points. The maximum distance observed was 0.2. We compare each point in the

Figure 4.6: Switch ID during track

Table 4.5: Result of detection part of the framework.

Algorithm	TPs	FPs	FNs	Precision	recall	f1-score
DR-SPAAM	270	513	593	0.344	0.313	0.327
Our	763	58	85	**0.9296**	**0.9021**	**0.9156**

detection data with the closest point in the ground truth data. If the distance between these points is less than and equal to 0.2, we consider it a FP. In our study, a FP signifies that the person has been accurately detected in the correct position.

If we encountered a detected point without a corresponding point in the ground truth data, it was classified as a FP. Conversely, if there were points in the ground truth data for which we couldn't find any close points in the detection data, they were categorized as FNs. The evaluation results are presented in Table 4.5:

Precision is a measure of the accuracy of positive predictions. In this evaluation, the precision is 0.9296, indicating that the majority of the detected points are indeed TPs. Recall, also known as sensitivity or FP rate, measures the ability to identify all positive instances. The recall in this evaluation is 0.9021, indicating that a significant portion of the ground truth points were successfully detected. The F1-score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance. The F1-score obtained in this evaluation is 0.9156, indicating good overall performance in terms of precision and recall. As shown in Table 4.5, the performance of our framework is approximately three times better than using

solo DR-SPAAM for detecting people. These evaluation results suggest that the detection part of the framework performs well, with a high number of TPs, low FPs, and FNs.

4.5 Implementation of Tracking

Following the detection phase, we utilized the output file as input for the tracking part of our framework. To track the detected objects over time, we employed the unscented Kalman filter from the Python library filterpy.kalman [3]. The tracking process involved reading each frame of data from a YAML file. For initialization, we used frame 0 as the starting point for our trackers. For each person detected in this frame, we created a filter with a unique tracker ID, initial position, a loss counter (initialized to zero to handle cases where a person is not detected in certain frames), and a list to keep track of the number of frames that the person is considered "missed."

Using a loop to iterate through subsequent frames, we implemented the tracking algorithm. This involved handling loss cases, where a person may not be detected in certain frames, and assigning the detected objects to exist trackers based on their positions. More details on these functions can be found in Chapter 3. Throughout the tracking process, we saved the tracked positions for each tracker ID in each frame. The resulting data was stored in a YAML file with the same format as the ground truth, allowing for easy comparison and analysis of the tracked positions over time.

When evaluating multi-person tracking using 2D range data, there are several evaluation metrics that can be used to assess performance. While Precision, Recall, and F1-Score are commonly explained for the detection part, there are other metrics that are frequently utilized as well. Here are a few additional commonly used metrics:

1. Multiple Object Tracking Accuracy (MOTA): MOTA quantifies the overall tracking accuracy by considering FPs, FNs, and identity switches. It is defined as the percentage of tracking errors per frame.

$$\text{MOTA} = 1 - \frac{\text{FP} + \text{FN} + \text{ID}}{\text{GT}} \tag{4.1}$$

where:

[3] https://filterpy.readthedocs.io/en/latest/kalman/KalmanFilter.html

Figure 4.7: Switch ID during tracking. a) before the person misses in the frame. b) after remerging the missed person.

- **FP**: Total number of FPs. These are cases where the tracker incorrectly detects a person that does not exist in the ground truth.
- **FN**: Total number of FNs. These are cases where the tracker fails to detect a person that exists in the ground truth.
- **ID**: Total number of identity switches. An identity switch occurs when the tracker assigns a different identity to a person that should maintain the same identity across frames.
- **GT**: Total number of ground truth trajectories. This represents the actual number of unique persons present in the scene as annotated in the ground truth data. Therefore, based on the given ground truth data, the value of GT is 5, indicating that there are 5 unique ground truth trajectories in the tracking scenario.

The MOTA metric calculates the tracking accuracy by subtracting the sum of FPs, FNs, and identity switches from the total number of ground truth trajectories, and then normalizing it to a scale from 0 to 1. A higher MOTA score indicates better tracking performance, as it means fewer tracking errors relative to the ground truth.

2. Mostly Tracked Targets (MT): MT measures the percentage of ground truth trajectories that are mostly tracked by the tracker, meaning they have an overlap greater than a specified threshold.

$$MT = \frac{N_{\mathrm{MT}}}{N_{\mathrm{GT}}}$$

Where:

- MT represents the Mean Mostly Tracked metric.

- N_{MT} is the number of mostly tracked targets.
- N_{GT} is the total number of ground truth trajectories.

By dividing the number of mostly tracked targets by the total number of ground truth trajectories, this equation provides the ratio of correctly mostly tracked targets, representing the tracking performance of the algorithm.

3. Mostly Lost Targets (ML): ML measures the percentage of ground truth trajectories that are mostly lost by the tracker, meaning they have an overlap of less than a specified threshold.

$$ML = \frac{N_{ml}}{N_G T} \tag{4.2}$$

Where:

- ML represents the Mean Mostly Lost metric.
- $N_{\mathbf{ML}}$ is the number of mostly lost targets.
- $N_{\mathbf{GT}}$ is the total number of ground truth trajectories.

Table 4.6: Performance Metrics Comparison for different multi people tracking algorithms.

Methods	MOTA (%)	MT (%)	ML (%)
AMIR3D [70]	25.0%	3.0%	27.6%
MCFPHD [71]	39.9%	25.7%	**16.8%**
GPDBN [72]	49.8%	25.7%	17.2%
MOANA [73]	52.7%	28.4%	22.0%
MPLT [74]	**54.2%**	30.6%	20.9%
Ours	30.7%	**78%**	27%

Based on the evaluation results in Table 4.5, it is evident that the tracking component of our framework has certain limitations, particularly in terms of ID switches and tracking accuracy, as indicated by the relatively low MOTA value compared to other multi-person tracking algorithms in the provided table. When compared to higher-performing algorithms such as MOANA and MPLT, our algorithm exhibits a lower overall tracking accuracy due to a higher number of identity switches. This issue arises when a person is temporarily occluded or missed in a frame, causing their appearance in a different location relative to the robot due to the robot's movement. This change in perspective poses a challenge for the tracking algorithm in

consistently assigning the same ID to the person. Consequently, the person's ID may switch in subsequent frames, resulting in an increased number of identity switches. This scenario might have occurred with other individuals in the video, further contributing to the high number of identity switches. To address this challenge, we discuss potential solutions in the future work section Chapter 5 of our thesis. By incorporating additional spatio-temporal tracking techniques, we aim to improve tracking accuracy by reducing the occurrence of ID switches. In Figure X, we illustrate an example where a person initially appears in front of the robot but reemerges on the left side, leading to an ID switch. Resolving such specific cases of ID switches can significantly enhance the performance of our algorithm.

It is important to note that our algorithm performs well in terms of MT, as indicated by the high value compared to other listed algorithms. This suggests that our algorithm effectively tracks people with a substantial overlap with the ground truth annotations. This could be attributed to the fact that the number of people we are tracking is fewer compared to other algorithms. Therefore, if we can address the specific cases of ID switches, our algorithm has the potential to yield significantly better results. Regarding the ML metric, our algorithm exhibits a relatively higher value, indicating a larger proportion of lost people during tracking compared to some other algorithms. This signifies the need for improvement in reducing the loss of people during the tracking process. In summary, while our algorithm demonstrates strong performance in terms of MT (accurately tracking people with high overlap), it falls behind in terms of MOTA (overall tracking accuracy) when compared to algorithms like MOANA and MPLT. The higher ML value suggests that our algorithm experiences more instances of losing people during tracking compared to certain other algorithms. Therefore, there is room for improvement in terms of tracking accuracy and reducing the loss of people in our algorithm.

5 Conclusion

In conclusion, this thesis provides a comprehensive overview of the research findings and contributions in the domain of autolabeling people detection using omnidirectional cameras and laser rangefinders in indoor environments. The study delved into the motivation behind this research, identified the challenges involved, and established the objectives pertaining to people detection and tracking in panoramic videos. Furthermore, a novel framework was introduced for generating a specialized dataset of panoramic images tailored for the purpose of people tracking. This dataset serves as a valuable resource for advancing research in this area. In this final chapter, we aim to summarize the key findings, discuss the implications of the research, and propose potential avenues for future studies in this field.

5.1 Conclusion

In conclusion, this thesis has aimed to contribute to the field of people detection and tracking in indoor environments using omnidirectional cameras. Panoramic cameras have gained popularity in service robotics due to their ability to capture comprehensive 360 degree views, which can be beneficial for tracking people and enabling people-guiding functionalities in autonomous mobile robots. However, a significant challenge in this field is the lack of datasets specifically designed for people tracking in panoramic videos, especially with accurate localization of individuals in the real world. To address this challenge, the developed framework in this thesis aims to facilitate the creation of a dataset for panoramic images that can automatically

label people with their precise world locations, eliminating the need for human intervention.

The framework presented in this thesis proposes an innovative approach to autolabeling panoramic videos and accurately determining the position of each person in the world frame, as opposed to the image frame as done in previous works [17, 75, 76, 77]. By utilizing an omni-directional camera and a 2D laser rangefinder, the system captures a comprehensive 360 degree field of view, enabling effective monitoring of the environment. The integration of YOLOv7, a robust object detection model, with DR-SPAAM for person detection in the 2D range dataset further enhances the accuracy of the system. The use of the homography matrix facilitates the transformation of individuals detected by DR-SPAAM into the image frame, allowing for the selection of positions associated with individuals identified within the bounding boxes obtained from YOLOv7. To ensure reliable people tracking and handle uncertainties in people's motion across consecutive scans, the framework incorporates the unscented Kalman filter and the global nearest neighbor method. These techniques aim to improve the system's tracking capabilities and robustness when dealing with varying movements of individuals.

The experimental results have provided compelling evidence supporting the effectiveness of the proposed framework. One notable advantage of the framework is the implementation of an automatic labeling technique, which leverages mobile robots and laser rangefinders. This technique significantly streamlines the labeling process, reducing the need for manual effort and intervention. Regarding the detection aspect, the results are promising and comparable to previous works. The framework demonstrates a high level of accuracy in detecting objects of interest. However, when it comes to the tracking part, there are some areas that require further improvement. Although the values of MT (mostly tracked objects) and ML (mostly lost objects) are reasonably good in comparison to previous works, as discussed in Chapter 4, the value of MOTA(Multiple Object Tracking Accuracy) is not as high as desired. One of the main reasons for this limitation is the occurrence of frequent switches in ID assignments between the robot and missed persons during the tracking process. To address this issue and enhance the overall tracking performance, future work could focus on investigating strategies to minimize switch ID occurrences.

5.2 Future Works

While our framework has shown promising results, there are several areas that can be further improved and explored in future work. One aspect to consider is the detection of objects that may partially cover a person, leading to inaccurate position labeling. Currently, our framework

does not account for such cases. To address this, a potential future direction is to incorporate YOLO object detection algorithm to detect and classify all objects in the environment. By checking the intersection between objects and people, we can ensure that each person is accurately labeled with their respective positions.

Another area for future work is the laser calibration process. In our current implementation, we manually extract points from images of the calibration board. Although we attempted line detection algorithms, we encountered challenges due to factors such as the low image quality when transformed to the sides with cube map, variations in lighting conditions, and the color of the board. To make our framework more automated, future efforts can focus on resolving these issues and developing robust algorithms for automated point extraction from images. Additionally, we can explore the possibility of using pattern-matching techniques to determine the positions of the calibration board on laser scans, further enhancing the automation of the calibration process.

In terms of tracking, we observed a low MOTA when considering the switch ID for moving the robot and people, as well as the 360 degree fields of view of the laser and camera. To address this issue, a potential improvement for future work is to incorporate feature extraction techniques on the image data. By extracting pertinent features, we can strengthen the process of assigning IDs to individuals in each frame, thereby enhancing the accuracy and reliability of the tracking results.

To expand the applicability of our framework and ensure a reliable dataset for panoramic people tracking with real positions in the robot frame, future work should involve recording more data in various scenarios. This can help to validate the framework's performance under different conditions and evaluate its effectiveness in outdoor environments. By addressing these areas and exploring these avenues for future work, we can further enhance the capabilities and accuracy of our framework, making it more robust and applicable in a wider range of real-world scenarios.

References

[1] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "Motchallenge: A benchmark for single-camera multiple target tracking," *International Journal of Computer Vision*, vol. 129, pp. 845–881, 2021.

[2] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun, "Crowdhuman: A benchmark for detecting human in a crowd," *arXiv preprint arXiv:1805.00123*, 2018.

[3] G. Mazzola, L. Lo Presti, E. Ardizzone, and M. La Cascia, "A dataset of annotated omnidirectional videos for distancing applications," *Journal of Imaging*, vol. 7, no. 8, p. 158, 2021.

[4] E. Vendrow, D. T. Le, J. Cai, and H. Rezatofighi, "Jrdb-pose: A large-scale dataset for multi-person pose estimation and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4811–4820.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[6] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1458–1465.

[7] R. Girshick, "Fast r-cnn object detection with caffe," *Microsoft Research*, 2014.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 213–229.

[11] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[13] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, "Fairmot: On the fairness of detection and re-identification in multiple object tracking," *International Journal of Computer Vision*, vol. 129, pp. 3069–3087, 2021.

[14] D. Jia, A. Hermans, and B. Leibe, "Dr-spaam: A spatial-attention and auto-regressive model for person detection in 2d range data," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 270–10 277.

[15] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[16] S. Jiang, Y. Wang, J. Zhang, and J. Zheng, "Full-field deformation measurement of structural nodes based on panoramic camera and deep learning-based tracking method," *Computers in Industry*, vol. 146, p. 103840, 2023.

[17] A. Bacchin, F. Berno, E. Menegatti, and A. Pretto, "People tracking in panoramic video for guiding robots," in *Intelligent Autonomous Systems 17: Proceedings of the 17th International Conference IAS-17*. Springer, 2023, pp. 407–424.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[19] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[20] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "Mot16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[21] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv preprint arXiv:2003.09003*, 2020.

[22] S. Zhang, R. Benenson, and B. Schiele, "Citypersons: A diverse dataset for pedestrian detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3213–3221.

[23] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2011.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[25] M. Gou, S. Karanam, W. Liu, O. Camps, and R. J. Radke, "Dukemtmc4reid: A large-scale multi-camera person re-identification dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 10–19.

[26] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[27] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[28] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 954–960.

[29] S. Bianco, G. Ciocca, P. Napoletano, and R. Schettini, "An interactive tool for manual, semi-automatic and automatic video annotation," *Computer Vision and Image Understanding*, vol. 131, pp. 88–99, 2015.

[30] Yan, J. Yang, and Hauptmann, "Automatically labeling video data using multi-class active learning," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 516–523 vol.1.

[31] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *Proceedings of the IEEE*, 2023.

[32] B. Schiele, M. Andriluka, N. Majer, S. Roth, and C. Wojek, "Visual people detection: Different models, comparison and discussion," in *Proceedings of the IEEE ICRA Workshop on People Detection and Tracking*, 2009.

[33] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1. Ieee, 2001, pp. I–I.

[34] D. P. Solomatine and D. L. Shrestha, "Adaboost. rt: a boosting algorithm for regression problems," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2. IEEE, 2004, pp. 1163–1168.

[35] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. Ieee, 2005, pp. 886–893.

[36] F. Jurie and M. Dhome, "A simple and efficient template matching algorithm," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2. IEEE, 2001, pp. 544–549.

[37] C. Zhaoyang, G. Haolin, and W. Kun, "A motion based object detection method," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, 2020, pp. 280–283.

[38] M. Szarvas, A. Yoshizawa, M. Yamamoto, and J. Ogata, "Pedestrian detection with convolutional neural networks," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, 2005, pp. 224–229.

[39] M. Haris and A. Glowacz, "Road object detection: A comparative study of deep learning-based algorithms," *Electronics*, vol. 10, no. 16, p. 1932, 2021.

[40] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[42] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, p. 103514, 2022.

[43] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[44] S. Qiao, L.-C. Chen, and A. Yuille, "Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 10 213–10 224.

[45] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[46] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[47] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[48] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[49] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[50] E. Arkin, N. Yadikar, Y. Muhtar, and K. Ubul, "A survey of object detection based on cnn and transformer," in *2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML)*. IEEE, 2021, pp. 99–108.

[51] S. Vishwakarma and A. Agrawal, "A survey on activity recognition and behavior understanding in video surveillance," *The Visual Computer*, vol. 29, pp. 983–1009, 2013.

[52] E. Khatab, A. Onsy, M. Varley, and A. Abouelfarag, "Vulnerable objects detection for autonomous driving: A review," *Integration*, vol. 78, pp. 36–48, 2021.

[53] H. Uchiyama and E. Marchand, "Object detection and pose tracking for augmented reality: Recent approaches," in *18th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 2012.

[54] E. Van der Kruk and M. M. Reijne, "Accuracy of human motion capture systems for sport applications; state-of-the-art review," *European journal of sport science*, vol. 18, no. 6, pp. 806–819, 2018.

[55] J. Yi, J. Liu, C. Zhang, and X. Lu, "Magnetic motion tracking for natural human computer interaction: A review," *IEEE Sensors Journal*, vol. 22, no. 23, pp. 22 356–22 367, 2022.

[56] D. Fleet and Y. Weiss, "Optical flow estimation," in *Handbook of mathematical models in computer vision*. Springer, 2006, pp. 237–257.

[57] G. Bishop, G. Welch *et al.*, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-23175, p. 41, 2001.

[58] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter," *Kalman filtering and neural networks*, pp. 221–280, 2001.

[59] P. M. Djuric, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo, and J. Miguez, "Particle filtering," *IEEE signal processing magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[60] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.

[61] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2, no. 1. Lille, 2015.

[62] R. W. Byren, "Laser rangefinders," *The Infrared and Electro-Optical Systems Handbook*, vol. 6, pp. 77–114, 1993.

[63] H. Lamela and E. Garcia, "A low power laser rangefinder for autonomous robot applications," in *Proceedings of the 1996 IEEE IECON. 22nd International Conference on Industrial Electronics, Control, and Instrumentation*, vol. 1. IEEE, 1996, pp. 161–167.

[64] O. M. Mozos, R. Kurazume, and T. Hasegawa, "Multi-part people detection using 2d range data," *International journal of social robotics*, vol. 2, pp. 31–40, 2010.

[65] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[66] ——, "Determining the epipolar geometry and its uncertainty: A review," *International journal of computer vision*, vol. 27, pp. 161–195, 1998.

[67] D. Scaramuzza, "Omnidirectional camera," *Computer Vision: A Reference Guide, Editors: Katsushi Ikeuchi, ISBN: 978-0-387-30771-8 (Print) 978-0-387-31439-6 (Online), Springer, April, 2014*, 01 2014.

[68] L. Yu, M. Peng, Z. You, Z. Guo, P. Tan, and K. Zhou, "Separated calibration of a camera and a laser rangefinder for robotic heterogeneous sensors," *International Journal of Advanced Robotic Systems*, vol. 10, no. 10, p. 367, 2013.

[69] Y. Ren and F. Hu, "Camera calibration with pose guidance," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 2180–2184.

[70] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 300–311.

[71] N. Wojke and D. Paulus, "Global data association for the probability hypothesis density filter using network flows," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 567–572.

[72] T. Klinger, F. Rottensteiner, and C. Heipke, "Probabilistic multi-person localisation and tracking in image sequences," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 127, pp. 73–88, 2017.

[73] Z. Tang and J.-N. Hwang, "Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d," *IEEE Access*, vol. 7, pp. 31 934–31 945, 2019.

[74] F. Yang, F. Li, Y. Wu, S. Sakti, and S. Nakamura, "Using panoramic videos for multi-person localization and tracking in a 3d panoramic coordinate," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1863–1867.

[75] R. Seidel, A. Apitzsch, and G. Hirtz, "Improved person detection on omnidirectional images with non-maxima suppression," *arXiv preprint arXiv:1805.08503*, 2018.

[76] S. Li, M. O. Tezcan, P. Ishwar, and J. Konrad, "Supervised people counting using an overhead fisheye camera," in *2019 16th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE, 2019, pp. 1–8.

[77] B. E. Demiröz, A. A. Salah, Y. Bastanlar, and L. Akarun, "Affordable person detection in omnidirectional cameras using radial integral channel features," *Machine Vision and Applications*, vol. 30, pp. 645–655, 2019.

Acknowledgments

I would like to express my heartfelt gratitude to my advisor and coadvisor, Professor Emanuele Menegatti and Professor Alberto Pretto, for providing me with the opportunity to work on this thesis in the YAS-LAB. Their guidance, expertise, and unwavering support have been invaluable throughout the entire research process. I am also deeply thankful to Dr. Alberto Bacchin for his invaluable contributions to this research. His expertise, patience, and encouragement have played a crucial role in shaping the outcome of this work. I would like to extend my sincere appreciation to the staff and administrators of the University of Padova for their assistance and cooperation in providing the necessary resources and facilities for this research. Their professionalism and dedication have greatly facilitated the progress of my thesis. I would like to acknowledge the participants of my study, whose active involvement and willingness to share their experiences have been pivotal in the success of this research. Their insights and perspectives have significantly enriched the findings and conclusions of this thesis. My heartfelt thanks go to my family and friends for their unwavering support, encouragement, and understanding throughout my academic journey. Their love, patience, and belief in me have been an endless source of inspiration and motivation. Lastly, I would like to express my sincere appreciation to all those who have directly or indirectly contributed to the completion of this thesis. Your support, encouragement, and valuable contributions have played an instrumental role in achieving this significant milestone. Thank you all for being an integral part of this incredible journey.

Sepideh Shamsizadeh