Person Tracking and Following with 2D Laser Scanners

Angus Leigh

Master of Science

School of Computer Science

McGill University

Montreal, Quebec

2015-06-15

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

ACKNOWLEDGEMENTS

This thesis would not have been possible without the help of a great many people who provided me with inspiration, direction and encouragement.

Foremost, I would like to thank my supervisor, Joelle Pineau. You've been the ideal mixture of supportive, easy-going and motivating during my master's. You were at once hands-off, letting me pursue the research ideas I was most interested in, and engaged, constantly providing me with valuable feedback on whatever we were working on. You listened to my ideas from day one so I would grow and learn to think for myself rather than follow orders. You were patient when research went slow and were quick to offer praise when things went well. I could not have asked for more from a supervisor.

To my closest collaborators on the SmartWheeler project, Andrew and Martin: It was a good two years! Thanks for making it fun.

Special thanks to Bennie, Ghitza and Qiwen for helping with the benchmark data collection in this thesis.

To everyone in the RLLab, I'm grateful for all the thought-provoking discussions with you. You're the reason the RLLab is such a mentally-stimulating and enjoyable research environment.

Finally, I want to thank my family for their loving support over that past two years. Thanks, Ma and Pa - you guys are "the best of the best of the best."

ABSTRACT

Having accurate knowledge of the positions of people around a robot provides rich, objective and quantitative data that can be highly useful for a wide range of tasks, including autonomous person following. The primary objective of this research is to promote the development of robust, repeatable and transferable software for robots that can automatically detect, track and follow people in their environment. The work is strongly motivated by the need for such functionality onboard an intelligent power wheelchair robot designed to assist people with mobility impairments.

In this thesis we propose a new algorithm for robust detection, tracking and following from laser scanners mounted at leg height. We develop a set of quantitative benchmarks based on the CLEAR MOT metrics which demonstrate that the approach is more effective than currently available alternatives in a variety of environments, both indoor and outdoor, and on different robot platforms (the intelligent power wheelchair and a Clearpath Husky). To the best of our knowledge, these are the first benchmarks which evaluate person tracking performance from a *mobile* robot with laser scanners at leg height. We further integrate our novel person tracking system with a person following control system to demonstrate its usefulness in this application. The method has been implemented in the Robot Operating System (ROS) framework and will be publicly released as a ROS package.

ABRÉGÉ

Une connaissance précise de la position des personnes autour d'un robot fournit des données riches, objectives et quantitatives, qui peuvent être hautement utiles pour une grande variété de tâches, incluant le suivi automatique de personnes. L'objectif premier de cette recherche est de promouvoir le développement de logiciels robustes, reproductibles et transférables, pour que des robots puissent automatiquement détecter, localiser et suivre des gens dans leur environnement. Ce travail est fortement motivé par le besoin d'une telle fonctionnalité, à bord de fauteuils roulants électriques intelligents conçus pour assister les personnes à mobilité réduite.

Dans cette thèse nous proposons un nouvel algorithme pour la détection, la localisation et le suivi à partir de capteurs lasers situés à hauteur de la jambe. Nous développons un ensemble de mesures de référence quantitatives basées sur les métriques CLEAR MOT, qui démontrent que notre approche est plus efficace que les alternatives actuelles, ce dans une variété d'environnements, à la fois intérieurs et extérieurs, et sur différentes plateformes de robots (le fauteuil roulant électrique intelligent et un Clearpath Husky). Au mieux de notre connaissance, nous sommes les premiers à évaluer la performance du suivi de personnes depuis un robot *mobile* équipé de capteurs lasers à hauteur de la jambe. De plus, pour en démontrer l'utilité, nous intégrons notre nouveau système de localisation de personnes avec un système de contrôle automatique permettant au robot de se déplacer en suivant une personne. Cette méthode a été implémentée avec le Robot Operating System (ROS), et sera publiquement disponible sous forme d'un paquet ROS.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

Having accurate knowledge of the positions of people over time provides rich, objective and quantitative data that can be highly useful for a wide range of applications. More specifically, the capability to autonomously detect, track and follow a person has been identified as an important functionality for many assistive and service robot systems [16, 34]. Over the last decade, significant progress has been made in developing person detection and tracking algorithms, often with the aim of improving human-robot interaction or robot navigation in populated environments [2, 44].

Yet most of the work to date is not easily transferable to new applications: algorithms are tested on a single robot in a single environment (if at all, sometimes only in simulation under artificial conditions), in many cases the code has not been made publicly available, datasets collected during validation sessions are not shared, and quantitative comparisons to existing algorithms are not performed. For example, to the best of our knowledge, no research groups have been able to replicate recently published person tracking results on different robot platforms. This is further reflected in the fact that, as far as we know, there does not yet exist a robust, open-source, laser-based person-following system for a mobile robot that has demonstrated capability to work indoors, outdoors and in cluttered and crowded areas.

To address this, a novel method and software architecture for autonomous person tracking and following with 2D laser scanners is proposed in this thesis. It builds on recent results in the literature [2, 44], extending them in several directions to improve accuracy and reduce the number of errors. One contribution of this work is a novel tracking method which uses tracking of both legs jointly, rather than individually, such as in [2, 44], to improve reliability, especially in cases of self-occlusion. It also integrates local occupancy grid maps to improve data association by disallowing the initiation or continuation of people tracks in occupied space. However, unlike the approach in [62] which constructs occupancy grid maps of all scan points, the approach in this thesis only uses non-human scan points and therefore does not require people to move continuously to be tracked. The proposed approach also integrates the method of object tracking present in Robot Operating System (ROS) leg_detector package in which all detected objects are tracked in every frame, included non-human objects, improving data associated in cluttered environments. Finally, a closed-loop control algorithm allowing the robot to autonomously follow tracked individuals is incorporated.

A major asset of the proposed approach is the ability to work on multiple different robots, under various operating conditions. We present extensive empirical results, validating the performance of the approach on two different robot platforms over 45 minutes of data collected across different environments, both indoor and outdoor, with moving and stationary robots, and varied crowd and obstacle conditions. These results show the benefit of the proposed tracking approach compared to the existing open-source implementation in ROS, especially in the application of person following. Another contribution of this work is a public release of all datasets and code used during our investigations as an open-source ROS package to facilitate ongoing research in this area in the future. These datasets include, to the best of the our knowledge, the first laser-based person tracking benchmarks collected on a moving robot.

While the work in this thesis focuses specifically on the problem of accurate tracking and following of people by social and assistive robots, we expect our work to have significant applications outside of this field, including for security (e.g. tracking intruders), entertainment (e.g. developing interacting exhibits), marketing (e.g. location-aware personalized advertisement), rehabilitation (e.g. assessment of patients' locomotion patterns following an injury [42]), and beyond.

1.1 Problem Description

The problem of interest can be decomposed into three sub-problems: (1) detection, (2) tracking, and (3) following. For reasons of modularity and robustness, these problems are tackled with three separate (though connected) modules. Such modularity is consistent with most of the literature on the topic, and allows development of a solution that is transferable to a large range of applications. For example, a security robot may require only detection and tracking; a social robot on the other hand may ask the user to *manually* specify what person to follow (e.g. from an image, or with a gesture command), and use autonomous behavior only for tracking and following.

For the purposes of this thesis, attention is focused on systems with planar laser sensors. Such sensors are widely available on autonomous robots, especially those deployed in public and urban environments, due to their reliability and accuracy for mapping and navigation tasks. Laser sensing is also computationally cheap to process, lighting invariant, and functional under diverse operating conditions. Further, the wide field of view of modern laser scanners allows the robot to follow in close proximity with less risk of losing people out of frame.

This work aims to build and test a system that can achieve person tracking and following under diverse conditions: different robot platforms, indoors and outdoors, single or multiple people, cluttered with stationary or moving obstacles, without an *a priori* map of the environment. In support of this, the proposed system is deployed and validated on two different robot platforms, each in a different environment.

1.2 Problem Statement

Specifically, this work proposes a novel solution to the issue of autonomously detecting, tracking and following people from any generic mobile robot platform equipped with planar laser scanners mounted at leg height.

1.3 Contributions

The contributions of this work include:

- A novel method for detecting and tracking people from mobile robots using planar laser scanners mounted at leg height.
- A set of benchmarks demonstrating its empirical improvement over existing open-source methods and to allow future methods to compare against ours.
- A software program utilizing the CLEAR MOT metrics to automatically evaluate new person-tracking method's performance on the benchmarks.
- A set of software tools for creating and annotating new benchmarks.

- Validation of our method on two separate robotic platforms, including the Smartwheeler and a Clearpath Husky.
- A method for integration of the person tracking method with a person-following navigation system, demonstrated on a Clearpath Husky.

Finally, this work is an integral component in the development of the Smartwheeler intelligent powered wheelchair [57], providing it with the important functionality of autonomous person following [34].

1.4 Publications

Many of the chapters in this thesis contain work previously published by the authors at the 2015 International Conference on Robotics and Automation, under the title *Person tracking and following with 2D laser scanners* [43]. Section 6.3.4 also contains work previously published by the authors at the 2014 IROS Workshop on Rehabilitation & Assistive Robotics, under the title *Laser-based person tracking for clinical locomotion analysis* [42].

CHAPTER 2
Background

This thesis builds on several existing concepts, frameworks and models from probability, robotics, artificial intelligence and machine learning. To provide the necessary primer and context for the novel algorithms presented in this work, this section outlines the relevant prior work. It begins with an introduction to a modern robotics software development framework, followed by occupancy grid mapping, target tracking and, finally, supervised machine learning.

2.1 Robot Operating System

The Robot Operating System (ROS) is a software framework used extensively in the implementation of this work [58]. It has also seen widespread adoption in other robotics research and industrial applications since its release in 2007 [55]. At its core, it was designed with the purpose of facilitating the interoperation and communication of discrete robotic software systems. Under the ROS framework, individual processes (referred to as *nodes* in the ROS community) can communicate either through shared memory (via *nodelets*) or through a network-based, TCP message passing system. ROS facilitates distributed computing as nodes can be programmed and compiled once, then run on any single computer as well as any network-connected computer with no modification to the source code. Further, it is language-agnostic, meaning nodes can be written in Python, C++, LISP, MATLAB and Java and communicate with other nodes via a common message-passing interface, regardless of the language any are written in.

Due it its growth and widespread adoption, ROS also contains many existing, well-tested, open-source libraries and tools that can vastly simplify the implementation of new robotic systems. These libraries include camera drivers, laser scanner drivers, a coordinate frame transform library [23], localization algorithms [24], simultaneous localization and mapping (SLAM) algorithms [38, 27] and navigation algorithms [47].

The modularity of systems developed with ROS is perhaps one reason why many such open-source libraries exist. Instead of having a robot run one monolithic program containing all the functionality of the system, ROS programs are typically composed of several modules which communicate via message passing. This modularity facilitates code charing of individual components and it is our hope that it will likewise facilitate the sharing of this work.

Furthermore, under the ROS framework, software developers can harness the benefits of multiple languages, writing code in whichever language is most appropriate for the task at hand. For example, in the implementation of this work, C++ is used when fast processing time is needed and Python is used when fast prototyping and development times are higher priorities.

2.2 Occupancy Grid Mapping

Occupancy grid maps are an integral component of modern robotic systems and are utilized in this work [68, 66]. Essentially, they are methods for constructing maps containing estimates of occupied and freespace areas nearby a robot using noisy sensor observations.

Specifically, it is assumed that a set of known robot poses $\mathbf{x}_{1:t}$ and noisy sensor measurements $\mathbf{z}_{1:t}$ are given, and we wish to estimate our belief in a map of the nearby environment $\mathbf{m}$. The canonical approach is to first decompose the nearby environment into discrete, non-overlapping cells $\mathbf{m} = \{m^1, m^2, ..., m^N\}$, where each cell $m^i \in \{0, 1\}$ is a binary variable indicating whether that location is freespace or occupied. This space can then be reasoned over in a probabilistic manner using the posterior probability

$$p(\mathbf{m}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = p(m^1, m^2, ..., m^N|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}), \tag{2.1}$$

which can be factorized by the chain rule to yield

$$p(\mathbf{m}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = p(m^1|m^2, ..., m^N, \mathbf{x}_{1:t}, \mathbf{z}_{1:t})p(m^2|m^3, ..., m^N, \mathbf{x}_{1:t}, \mathbf{z}_{1:t})...p(m^N|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}). \tag{2.2}$$

However, without further simplification, reasoning over this belief space quickly becomes intractable because it grows exponentially with the number of cells in the decomposition. Thus, it is common to adopt a (naive) assumption of independence between cells

$$p(m^i \perp\!\!\!\perp m^j)\ \forall i \in 1, 2, ..., N, j \in 1, 2, ..., N, i \neq j. \tag{2.3}$$

The posterior then simplifies to

$$p(\mathbf{m}|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = \prod_{i=1}^{N} p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}). \tag{2.4}$$

This can then be evaluated in practice using a recursive binary Bayes filter on each cell of the map and performing an update on each cell after each received measurement [66]. Specifically, the posterior probabilities in the product of Equation 2.4 can alternatively be express via the Bayes rule as

$$p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1}, m^i)p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{z}_{1:t-1})}. \tag{2.5}$$

Applying the Markov assumption, we have

$$p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t, m^i)p(m^i|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{x}_t)}. \tag{2.6}$$

Then, applying the Bayes rule to the measurement model

$$p(\mathbf{z}_t|\mathbf{x}_t, m^i) = \frac{p(m^i|\mathbf{x}_t, \mathbf{z}_t)p(\mathbf{z}_t|\mathbf{x}_t)}{p(m^i)} \tag{2.7}$$

results in

$$p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}) = \frac{p(m^i|\mathbf{x}_t, \mathbf{z}_t)p(m^i|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})}{p(m^i)}. \tag{2.8}$$

In practice, it is common to perform all calculations in log-space to prevent floating-point rounding errors when storing numbers close to 0, as is commonly the case with probabilities. As such, our measurement update becomes

$$\mathrm{logit}(p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t})) = \mathrm{logit}(p(m^i|\mathbf{x}_t, \mathbf{z}_t)) + \mathrm{logit}(p(m^i|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})) - \mathrm{logit}(p(m^i)), \tag{2.9}$$

where logit is the log-odds ratio ($\mathrm{logit}(x) = \frac{x}{1-x}$), $p(m^i|\mathbf{x}_t, \mathbf{z}_t)$ is the sensor-dependent, measurement probability, $p(m^i|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1})$ is the map cell belief from the previous timestep and $p(m^i)$ is the prior belief in the map cell, typically set to between $0.2-0.5$, depending on the assumed concentration of objects in the map area [66].

All that remains is to define $p(m^i|\mathbf{x}_t, \mathbf{z}_t)$ for the specific sensor which is used to construct the map. In the case of planar laser scanners, the main sensor employed in this work, it is calculated via ray-tracing laser scan points from the laser centre. If a ray from the laser stops near the position of cell m^i, then it is assumed m^i is an obstacle and thus $p(m^i|\mathbf{x}_t, \mathbf{z}_t)$ takes a value greater than $p(m^i)$. If the ray continues unhindered beyond the position of cell m^i, then it is assumed m^i is freespace and $p(m^i|\mathbf{x}_t, \mathbf{z}_t)$ takes a value less than $p(m^i)$. Finally, if the ray does not intersect the position of cell m^i, then typically $p(m^i|\mathbf{x}_t, \mathbf{z}_t)$ takes the value of $p(m^i)$, which results in $\text{logit}(p(m^i|\mathbf{x}_{1:t}, \mathbf{z}_{1:t}))$, from Equation 2.9, maintaining its previous state.

This sub-section serves as a brief overview of occupancy grid mapping and the reader is encouraged to review [68, 66] for a more in-depth analysis. This technique will be revisited in Section 3.2.3 of this thesis.

2.3 Target Tracking

Target tracking is the process of estimating the pose (i.e., position and orientation) of one or many objects based on noisy observations from a sensor. In practice, the task is often non-trivial due to limited visibility of the sensor, noise in its readings, occlusions of the targets, difficulty in distinguishing multiple targets, and difficulty in differentiating targets from background objects. In this section we present the Kalman filter state estimation method, along with the Nearest Neighbour and

Global Nearest Neighbour data association approaches, which are three commonly-used frameworks for target tracking and will be utilized by the algorithms presented in this thesis.

2.3.1 The Kalman Filter

In the case of state estimation of a single tracked target with noisy observations, a Kalman filter can provide an estimate of the target's state which is more accurate than any single observation alone [35]. Furthermore, the filter provides optimal estimates in the case of a linear system with Gaussian noise on process and observation variables. Note that, although it is optimal for state estimation of such systems, it can still be employed on systems which are actually non-linear or are disturbed by other types of noise and produce useful, albeit non-optimal, state estimates.

The Kalman filter uses a hidden Markov model (HMM) to model the underlying dynamics of the tracked object [15]. In common with HMM conventions, it is assumed that $\mathbf{x}_k$ is the hidden state (such as the position and velocity of an object) and $\mathbf{z}_k$ is the noisy observation of the object. Note that in this case we are assuming discrete-time intervals between observations and state updates of the system, where $k \in \mathbb{N}$ is an index for each timestep. Further extensions for the Kalman filter for continuous-time systems exist, however they will not be described in this work.

The assumed model for how the system evolves over time is given by

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \tag{2.10}$$

where $\mathbf{F}$ is the state transition model (which we assume here is constant, although it can alternatively be modelled as time-dependent), $\mathbf{B}$ is the input model, $\mathbf{u}_k$ is the input at timestep k and $\mathbf{w}_k$ is a stochastic process noise which is assumed to have a zero-mean Gaussian distribution $\mathbf{w} \sim \mathcal{N}(0, \mathbf{Q})$ with covariance $\mathbf{Q}$ (which can also be modelled as time-dependent but will be assumed constant here). This model thus combines a linear, deterministic component with a noise component to account for uncertainties in the process.

Observations are assumed to be generated according to the model

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{2.11}$$

where $\mathbf{H}$ is the observation model (which we assume here is constant, although it can alternatively be modelled as time-dependent) and $\mathbf{v}_k$ is a stochastic observation noise which is assumed to have a zero-mean Gaussian distribution $\mathbf{v} \sim \mathcal{N}(0, \mathbf{R})$ with covariance $\mathbf{R}$ (which can also be modelled as time-dependent but will be assumed constant here).

The Kalman filter then derives an estimate of the hidden state $\hat{\mathbf{x}}_k$ along with an uncertainty covariance $\mathbf{P}_k$ using these models together with new observations. At every timestep, the state estimate is first updated using the state transition model

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k. \tag{2.12}$$

Note that $\hat{\mathbf{x}}_{k|k-1}$ refers to the estimated state at timestep k, using only observations up to timestep $k-1$ and that $\hat{\mathbf{x}}_{k|k}$ is the estimated state at timestep k, using observations up to timestep k. Using this notation, $\hat{\mathbf{x}}_{k|k}$ is written interchangeably with $\hat{\mathbf{x}}_k$

Further, the state covariance is updated using the state transition model according to

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^T + \mathbf{Q}. \tag{2.13}$$

Note that both of these initial steps are *predictive* and do not utilize new measurement information from the current timestep k.

Next, a measurement update is made using the latest sensor information to correct the state estimate and covariance. First, the measurement residual is determined according to

$$\mathbf{y}_k = \mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_{k|k-1}. \tag{2.14}$$

Intuitively, this reflects the amount of "surprise" in the latest observation. That is, how different our actual observation $\mathbf{z}_k$ is from what we expect it to be based on our state estimate $\hat{\mathbf{x}}_{k|k-1}$.

Next, the residual covariance and optimal Kalman gain are calculated by

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \tag{2.15}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\mathbf{S}_k^{-1}. \tag{2.16}$$

Finally, we can produce an updated state estimate and covariance for the current timestep

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k \tag{2.17}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k|k-1}. \tag{2.18}$$

2.3.2 Data Association

The Kalman filter is a useful tool for state estimation of a single target where observations are known unambiguously to have originated from the target. However, in the case of multiple tracked targets, where observations can originate from all targets and ambiguity exists regarding which target produces which observation, a data association method is required. This section briefly outlines the fundamentals of data association as well as two possible approaches for resolving this problem.

Formally, data association is concerned with finding a correspondence between a given set of previously tracked objects (referred to henceforth simply as *tracks*) $\mathbf{X}_{k-1}$ and a set of detections, or observations, in the current timestep $\mathbf{Z}_k$. This assignment can then be used to obtain an updated set of tracks $\mathbf{X}_k$ by performing a measurement update on each track using its assigned observation, as is shown in Kalman filter framework in Section 2.3.1.

In the simplest case, the number of tracks always equals the number of detections per timestep and there exists a one-to-one mapping between the two. More complicated cases arise, for example, when new objects can enter and leave the sensor's field of view, necessitating track initiation and deletion algorithms. Further issues emerge when spurious observations can arise from non-tracked, background objects or when the tracked objects can be temporarily occluded. In such cases, the assignment space for matching tracks to observations should be expanded to include these possibilities in order to achieve best performance.

There exist a number of approaches for handling this data association problem as well as numerous configurations for each approach depending on the problem, sensor configuration and assumptions adopted [4]. In the following two subsections, we introduce two possible methods: the Nearest Neighbour and the Global Nearest Neighbour data association approaches.

Nearest Neighbour

The Nearest Neighbour (NN) approach is perhaps the simplest assignment method for performing data association [18]. Using this approach, one first calculates a distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ between every track and every observation

$$D^{i,j} = d(\mathbf{X}^i, \mathbf{Z}^j) \;\; \forall i = 1, 2, ..., N, j = 1, 2, ..., N. \tag{2.19}$$

Then, one greedily matches detections to tracks by sequentially selecting the closest, unassigned track-detection pair from $\mathbf{D}$ until all are assigned.

Several metrics exists for the track-observation distance calculation [9], perhaps the simplest of which is the Euclidian distance

$$d(\mathbf{X}^i, \mathbf{Z}^j) = d_{Euclid}(\mathbf{X}^i, \mathbf{Z}^j) = \|\mathbf{X}^i - \mathbf{Z}^j\|_2. \tag{2.20}$$

In many cases, however, one can achieve more accurate assignments by also utilizing information from uncertainty covariances from the track state estimates [18], such as those provided by the Kalman filter in Section 2.3.1. In such cases, the Euclidian distance can be replaced with the Mahalanobis distance, which is calculated by

$$D^{i,j} = d_{Mahalanobis}(\mathbf{X}^i, \mathbf{Z}^j) = \sqrt{(\mathbf{Z}^j - \mathbf{X}^i)^T (\mathbf{S}^i)^{-1} (\mathbf{Z}^j - \mathbf{X}^i)}, \tag{2.21}$$

where $\mathbf{S}^i$ is the innovation covariance for track $\mathbf{X}^i$.

The NN data association approach benefits from a simple implementation, however, due to its greedy assignment method, it is suboptimal [4].

Global Nearest Neighbour

The Global Nearest Neighbour (GNN) approach is a data association method which minimizes the cumulative distance of all assignments in the matrix $\mathbf{D}$. This typically results in less assignment errors compared to the NN approach [39].

In the literature, finding the matchings in $\mathbf{D}$ which minimize the cumulative distance is typically referred to as the *linear assignment problem* or simply the *assignment problem* [14], and there exist a number of published methods and libraries for finding the optimal solution. For example, two commonly used methods are the Munkres assignment algorithm [40] and the Jonker-Volgenant assignment algorithm [33].

While the GNN approach is typically superior to the NN approach at producing accurate assignments, it suffers from a slower computation time. Specifically, the best solutions for the GNN approach scale at a rate of $\mathcal{O}(n^3)$ [40], where n is the number of tracked objects, whereas efficient NN approaches can scale at a more modest rate of $\mathcal{O}(n^2)$.

Other Data Association Approaches

In addition to the GNN and NN approaches, there exist many other data association techniques, such as the Probabilistic Data Association Filter [5] and the Multi-Hypothesis Tracker [60, 19]. However, these approaches will not be utilized in this thesis as the method presented in this work would not benefit from these alternative approaches in its current form. For more in-depth analyses in this area, the reader is encourged examine existing seminal work [4].

Track Initiation and Deletion

For each data association approach, there exist several methods for handling the special cases of track initiation, deletion, occlusion, or false position observations.

Tracks can be initiated in the simplest case by starting a new track for every unassigned measurement at each timestep [39, 52]. A more conservative approach, which is useful in situations where noisy background measurements can arise, is to first start a *potential* track for every unassigned measurement, then to only intiate a *confirmed* track once the potential track has been assigned measurements across multiple timesteps [29].

To remove tracks when objects leave the sensor's field of view, common approaches include deleting targets which have not been seen for a threshold number of scans scans or deleting targets which have a position covariance larger than a threshold [9].

In other works, occlusions are commonly handled as missed observations so when a target is occluded for a long period of time, its track is deleted [51]. However, more complicated methods also exist for occlusion reasoning which use prior information of the environment to potentially improve tracking and reduce the number of tracks deleted during temporary occlusions [46].

Finally, false positive measurement observations are commonly handled implicitely with the track initiation scheme: any targets which are not promoted from potential targets to confirmed targets are assumed to have been false positives [1].

2.4 Supervised Machine Learning

In the general task of supervised machine learning, a set of labelled samples is provided which are assumed to be drawn independently and identically distributed (i.i.d.) from an underlying probability distribution [8]. Each sample $\{\mathbf{x}, y\}$ consists of a set of M features $\mathbf{x} \in \mathcal{X}$ and a label $y \in \mathcal{Y}$. In general, the feature set $\mathbf{x}$ can consist of variables from a number of domains, such as categorical (e.g., $\mathbf{x} \in \mathbb{N}^M$), real (e.g., $\mathbf{x} \in \mathbb{R}^M$), integer (e.g., $\mathbf{x} \in \mathbb{Z}^M$) or some mixture across various domains. It is assumed to be visible when samples are provided from outside the training set. The label y, on the other hand, is assumed to be hidden when samples are provided from outside the training set. It is therefore typically the target variable which one desires to predict in practice. In the case of classification, it is typically categorical (e.g., $y \in \mathbb{N}$), whereas in the case of regression, it is typically real-valued (e.g., $y \in \mathbb{R}$).

The general goal of supervised learning is thus to use a provided set of labelled training data $\mathcal{D} = \{\mathcal{X}^i, \mathcal{Y}^i\}_{i=1}^N$ to learn a function $f : \mathcal{X} \to \mathcal{Y}$, which is able to predict labels $\mathcal{Y}$ as accurately as possible on unseen data using the given features $\mathcal{X}$.

In the following subsections we review two types of supervised learning algorithms: the decision tree classifier as well as the random forest classifier. First, the decision tree classifier is presented as it is used as an integral component of the random forest classifier, then the random forest classifier is presented as it is the method chosen for supervised learning in this work. Rationale for choosing the random forest classifier over other methods include its simple implementation, accuracy of prediction, robustness to hyper-parameter selection and invariance to the scaling and types of input features [22].

2.4.1 Decision Tree Classifier

Decision tree classifiers are a class of supervised machine learning models which predict output classes using sequences of binary decisions learned on the input features [11]. The decisions are learned by greedily finding the feature and decision threshold for any of the features which splits the data as much as possible into its distinct classes. This greedy partitioning of the data is then repeated and applied recursively to the nodes (i.e., subgroups) of data produced in the previous step until all nodes are fully split into distinct classes or a termination criterion is met.

Several possible metrics exist to determine the optimality of the decision tree split, such as Gini impurity [11] and information gain [59]. Here, we present the Gini impurity metric as it is used in the algorithms in this thesis and is the default metric in widely-used machine learing software implementations [56, 10]. Its calculation is also very similar to the information gain calculation and it has been shown in other works that the choice of metric has had little influence on practical performance [65].

As defined in [11], the Gini impurity for a set of categorical samples, $\mathbf{y} = \{y^1, y^2, ..., y^N\}$, is

$$I_G(\mathbf{y}) = 1 - \sum_{i=1}^{M} (\beta^i)^2, \tag{2.22}$$

where each $y^i \in \{1, 2, ..., M\}$ is a category label, and

$$\beta^i = \frac{\sum_{j=1}^{N} I(y^j = i)}{N} \tag{2.23}$$

is the fraction of samples in the set with a category label of i (i.e., the empirical frequency). Note that $I(\cdot)$ is used here as a binary indicator function, with $I(\text{true}) = 1$

and $I(\text{false}) = 0$. In essence, Gini impurity measures the expected fraction of incorrectly labelled samples if labels were chosen randomly from the empirical frequency of each category in the set.

To use this metric to determine a decision feature and threshold for splitting the data, one can exhaustively test all possible splits, each resulting in two new distinct sample sets. The feature and threshold which provides the lowest average Gini impurity across both sets is then selected.

As decision trees tend to be low-bias, high-variance classifiers, they are prone to overfitting [30]. To counter this, common strategies are to either use a set of termination criteria that encourage training to stop prematurely or to implement methods such as post-pruning. Example termination criteria include the decision tree exceeding a threshold maximal depth, the tree exceeding a threshold maximum number of total leaf nodes, nodes having less than a threshold minimum number of samples or nodes having less than a threshold minimum number of samples required to exist after a split.

2.4.2 Random Forest Classifier

A random forest classifier is an ensemble classification model which uses a census voting from a set of stochastically-trained decision trees to classify new samples [13].

It is trained by constructing a set of many decision trees, with the exact number of trees specified by a hyperparameter. Each decision tree is trained using a subset of size N samples drawn with replacement from the original training data, where N is the total number of samples in the training data. This is an example of the more general method called bagging, which is used to reduce variance in supervised learning models [12].

Individual decision trees are then trained according to the standard training method presented in Section 2.4.1 with the minor difference being that only a random subset of features is considered for each split, where the number of features to use is specified by a hyperparameter. Using random subsets of features in this manner increases bias of the individual decision trees, which is undesirable. However, this is compensated by a reduction in correlation among trees in the random forest and therefore a reduction in variance in the overall ensemble of decision trees. Further, each individual decision tree has similar termination criteria to those outlined in Section 2.4.1.

At test time, a given sample can be classified using the set of decision trees by aggregating the classification output of each individual tree. Methods for combining outputs from all trees include a majority voting scheme, or by averaging the probabilistic prediction from each tree [56].

Hyperparameters for the random forest classifier, such as the termination conditions and the number of features to use on each split can be determined using cross validation. However, the hyperparameter specifying the number of decision trees to use in the random forest classifier can be treated differently than the others. In general, increasing the number of decision trees will increase classification accuracy on test data asymptotically to an upper limit, with the only downside to using more decision trees being a slower computation time. Therefore, this parameter can be set by plotting the out-of-bag error (the classification error on samples only tested on trees which did not use them for training) vs. the number of decision trees and selecting a reasonable amount at which the out-of-bag error plateaus [32].

CHAPTER 3
Joint Leg Tracker

In this section, we present our system for autonomously detecting, tracking and following people using 2D laser scanners. Our proposed system, which we call the Joint Leg Tracker, includes several components. The autonomous person detection is achieved using clustering over laser detections; confidence levels are also assessed using supervised machine learning to help prune false positives. The autonomous tracking is achieved using a combination of Kalman filter (for predictive over consecutive scans) and a GNN approach to resolve the scan-to-scan data association problem. Finally, a control algorithm is used to follow the tracked target.

Our goal in desiging the tracking system is to provide robust information on the locations and identities of people around the chair. Specifically, the system would ideally be able to detect and track anyone within range of the sensors and would neither mix up identities of people nor errorneously report non-human objects as people. This information is then needed for the autonomous following module.

The design of the autonomous following module was focused on maintaining a set distance away from the tracked person, reacting to their position changes as fast as possible, without causing overshoot or jerkiness in the robot.

3.1 Autonomous Person Detection

The laser scanner returns a vector of distance measurements taken on a plane, roughly 30cm from the ground, at a resolution of approximately 1/3°. An example image showing a set of returned scan points is shown in Figure 3–1.

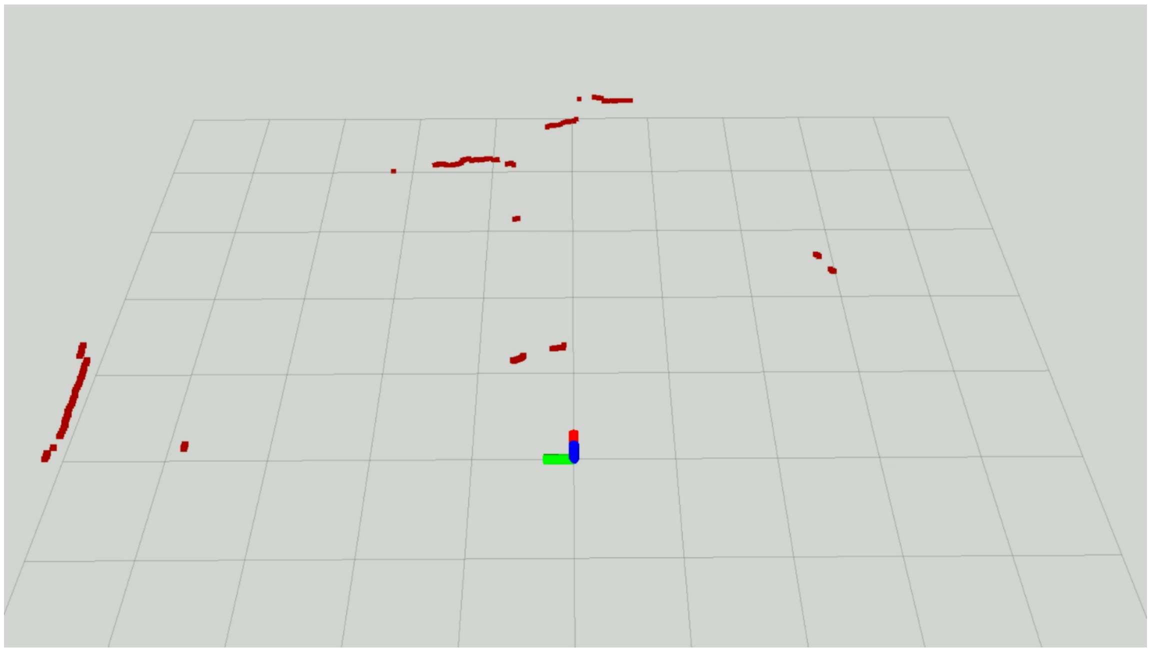

Figure 3–1: Raw scan points from an example scan. The blue-green axis shows the position and orientation of the laser scanner and scan points are shown in red.

Scan points returned are first clustered according to a fixed distance threshold, such that any points within the threshold are grouped together as a cluster. The threshold is chosen to be small enough to often separate a person's two legs into two distinct clusters, but to rarely generate more than two clusters per person. To mitigate noise, clusters containing less than three scan points are discarded in low-noise environments and clusters containing less than five scan points are discarded in high-noise environments. The positions of all clusters are then transformed into a reference frame which uses the robot's odometry to compensate for its egomotion. An example of this clustering applied to a laser scan is shown in Figure 3–2.

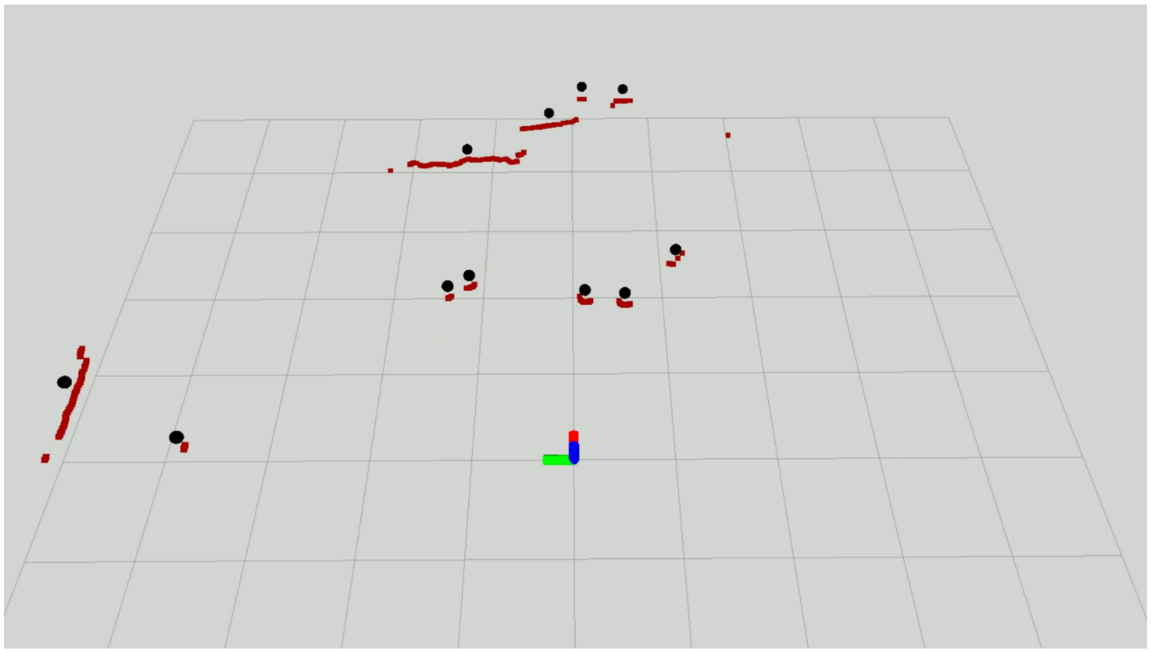

Figure 3–2: Clustering of scan points from an example scan. Cluster centroids are shown with black spheres and laser scan points are shown in red squares.

Detected cluster centroids are used as observations for the tracker. The observations at timestep k are denoted as

$$\mathbf{Z}_k = \left\{ \mathbf{z}_k^1, \mathbf{z}_k^2, ..., \mathbf{z}_k^{M_k} \right\}$$

where M_k is the total number of detected clusters at time k.

Clusters are further classified as human or non-human, based on a set of geometric features of the clusters. The features are listed in Table 3–1 and extend the set proposed in [2] and [44]. The classification is done using a random forest classifier whose training implementation is described in detail in Section 4.1.1.

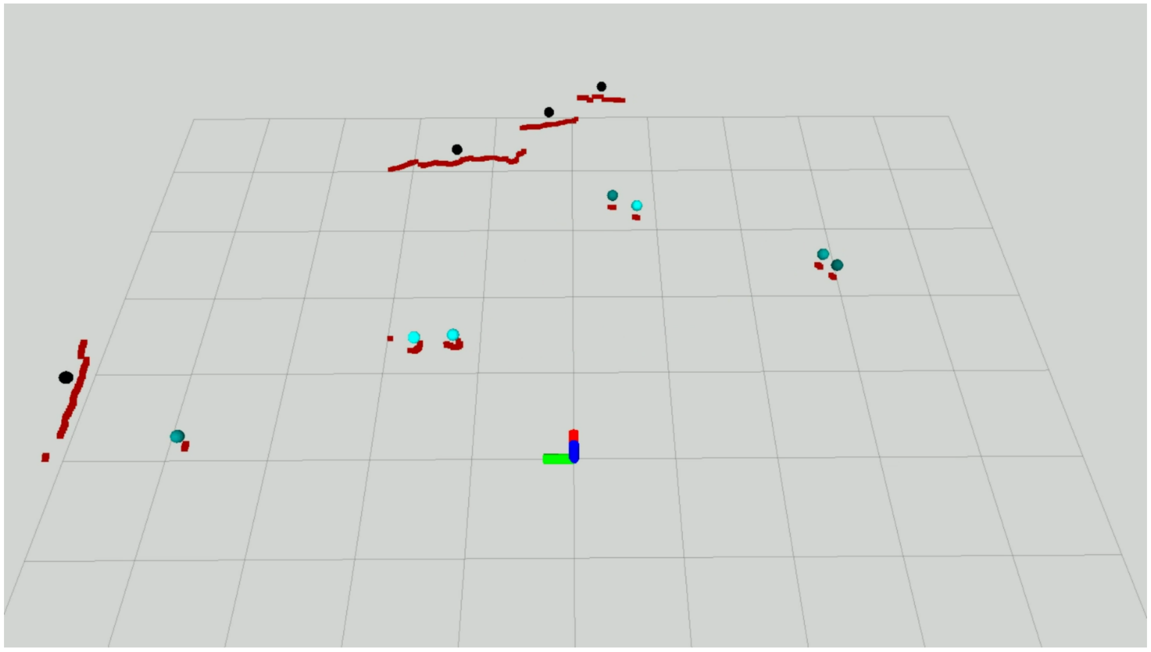

Figure 3–3: Confidence calculation output on clusters from an example scan. Lighter blue spheres indicate clusters with a higher confidence value. Note how scan clusters which have a semi-circular, curved shape in the laser scan have a greater resemblance to human legs than other clusters and therefore have a lighter blue colour.

One benefit of using an ensemble classification method is that a measure of confidence in the classification can be extracted by considering the number of individual decision trees predicting each class. A function for extracting this confidence is included in the random forest classifier implementation for OpenCV [10]. Thus, rather than using the binary output of the classifier (human/non-human), we consider the confidence level (from 0-100%) from the classifier and pass this information (along with the cluster centroid location) to the tracking module.

An example of the confidence calculation output is shown in Figure 3–3. Further, psuedocode for this module is included in Algorithm 1.

```
Data: Laser scanner reading $S_k$, robot postion estimate from odometry $O_k$
Result: Laser scan cluster centroids and confidences $Z_k = \{z_k^i, \mathsf{conf}_k^{Z,i}\}_{i=1}^{M}$

$C \leftarrow \varnothing$ // Set of clusters of raw scan points
foreach scan point $s_s$ in $S_k$ do
    $c_c \leftarrow \varnothing$ // Closest cluster
    $d_c \leftarrow D_{c,max}$ // Euclidian distance to closest cluster
    foreach cluster $c$ in $C$ do
        foreach scan point $s_c$ in $c$ do
            if $\|s_s - s_c\|_2 < d_c$ then
                $d_c \leftarrow \|s_s - s_c\|_2$
                $c_c \leftarrow c$
            end
        end
    end
    if $c_c \neq \varnothing$ then
        $c_c \leftarrow s_s \cup c_c$ // Add scan point to existing cluster
    else
        $C \leftarrow \{s_s\} \cup C$ // Create new cluster
    end
end

$Z_k \leftarrow \varnothing$ // Cluster centroids and confidences
foreach cluster $c$ in $C$ do
    $z_k \leftarrow$ Centroid($c$) // Centroid calculation
    $z_k \leftarrow$ Transform($z_k$, $O_k$) // Egomotion compensation
    $\mathsf{conf}_k^Z \leftarrow$ Confidence($c$) // Human confidence calculation
    $Z_k \leftarrow \{z_k, \mathsf{conf}_k^Z\} \cup Z_k$
end

return $Z_k$
```
Algorithm 1: LASER SCAN CLUSTERING. Note that this algorithm can result in over-segmentation in certain circumstances because it can only add scan points to existing clusters and cannot join two already existing clusters. This, however, is mitigated by the fact that scan points are iterated over sequentially in a clock-wise direction, such that cluster joining events would be a rare occurrence.

Table 3–1: Features used for confidence calculation of scan clusters.

Number of points	Width	Length
Standard deviation	Avg dist. from median	Occluded (boolean)
Linearity	Circularity	Radius of best-fit circle
Boundary length	Boundary regularity	Mean curvature
Mean angular diff.	Inscribed angular var.	Dist. from laser scanner

3.2 Autonomous Tracking of Multiple People

While our ultimate goal in designing the tracking and following system is to follow a single person, the tracking system was designed to be capable of tracking multiple people. This was done with the intention of reducing the possibility of confusing the identify of the person being followed when operating in densely populated environments.

3.2.1 Kalman Filter Tracking

The position of each detected cluster (regardless of confidence level) is individually tracked over time using a Kalman filter [35]. We refer to each Kalman filter which tracks a scan cluster over time as a *track* denoted as $\mathbf{x}_k^j$ and the set of all active tracks as

$$\mathbf{X}_k = \left\{ \mathbf{x}_k^1, \mathbf{x}_k^2, ..., \mathbf{x}_k^{N_k} \right\},$$

where N_k is the total number of tracks at time k. Further, the set of tracks $\mathbf{X}_k$ also includes a single track for each tracked person (their initiation is introduced in Sec. 3.2.4).

The Kalman filter for each track has a state estimate

$$\mathbf{x}_k^j = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

containing the position and velocity of the cluster in 2D coordinates. New tracks are initialized with a velocity of zero and existing tracks are updated with a constant velocity motion model, according to

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Process noise, $\mathbf{w}$, is assumed to be Gaussian white noise with diagonal covariance

$$\mathbf{Q} = \begin{bmatrix} q_p & 0 & 0 & 0 \\ 0 & q_p & 0 & 0 \\ 0 & 0 & q_v & 0 \\ 0 & 0 & 0 & q_v \end{bmatrix}.$$

Since only the position of each cluster is observed at each timestep, the observation matrix is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

and the observation noise has a covariance of

$$\mathbf{R} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}.$$

In tuning the Kalman filter, it was found that a small observation noise covariance ($r = 0.1^2 m$) is sufficient because position measurements from the laser scanner are highly accurate.

3.2.2 Global Nearest Neighbour Data Association

Since the system is designed to track all detected scan clusters, uncertainty arises pertaining to how detections $\mathbf{Z}_k$ should be matched to tracks from the previous time $\mathbf{X}_{k-1}$ to produce updated tracks $\mathbf{X}_k$ for the current time. This can potentially become an intractable combinatorial optimization problem. To address this, we use a GNN data association approach which is solvable in polynomial time ($\mathcal{O}(max(N_k, M_k)^3)$), via the Munkres assignment algorithm [40].

First, all tracks $\mathbf{X}_{k-1}$ are propagated to produce the state estimates $\mathbf{X}_{k|k-1}$ for time k. Next, the Munkres cost matrix is populated with the cost of assignment between every propagated track and every detection. The cost metric used in our tracking system is the Mahalanobis distance[1] between the detection $\mathbf{z}_k^i$ and the propagated track $\mathbf{x}_{k|k-1}^j$. The covariance used to calculate this distance is the innovation

[1] Note that other GNN data association approaches have recommended minimizing the square of this distance [39]. We found best results minimizing the non-squared distance in our applications.

covariance, which, in the case of the linear Kalman filter, is

$$\mathbf{S}_k^j = \mathbf{H}\mathbf{P}_{k|k-1}^j\mathbf{H}^T + \mathbf{R}. \tag{3.1}$$

where $\mathbf{P}_{k|k-1}^j$ is the prediction covariance of the propagated track $\mathbf{x}_{k|k-1}^j$. Here we use a higher noise covariance ($r = 0.5^2 m$) to allow more flexibility in associations.

A threshold is imposed such that detections outside the pth percentile confidence bounds of the expected observation are given an arbitrarily high assignment cost, $cost_{max}$. Since the Munkres algorithm requires a square matrix and N_k is often different from M_k due events such as people entering and leaving the field of view of the sensor, the cost matrix is padded with $cost_{max}$ to produce the necessary shape. When an assignment is made between a detection and a track by the Munkres algorithm and the associated cost is $cost_{max}$, the assignment is ignored. Otherwise, the track is updated with the position and the confidence-level of the detection. The position is used to perform an observation update in the track's Kalman filter to produce the updated track positions $\mathbf{X}_k$ for the current time k. Tracks without matched detections are propagated forward without observations. Detections without matched tracks spawn new tracks at their current position. The confidence of each track, denoted $\text{conf}_k^{T,j}$, is computed by the exponentially-weighted moving average

$$\text{conf}_k^{T,j} = \alpha\text{conf}_{k-1}^{T,j} + (1-\alpha)\text{conf}_k^{Z,i}, \tag{3.2}$$

where $\text{conf}_k^{Z,i}$ is the confidence of the ith detection which was assigned to the jth track and α is the parameter specifying the amount of smoothing (set to 0.95 in this work).

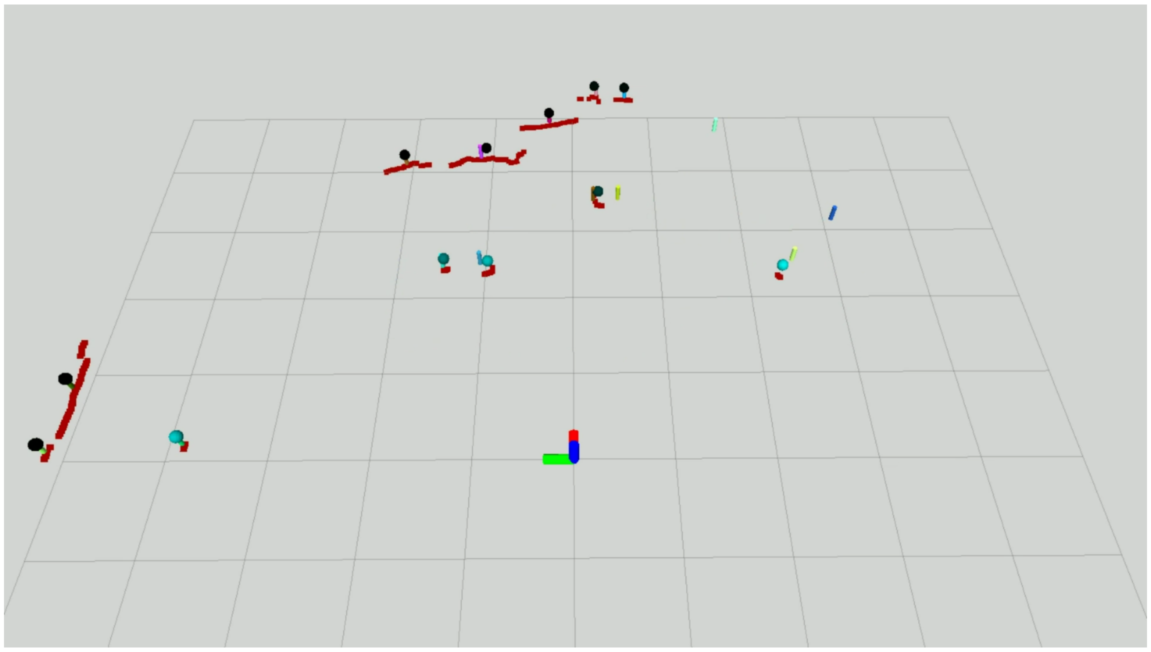

Figure 3–4: Tracked scan clusters in an example laser scan. Small colourful cylinders show the estimated positions of tracked laser scan clusters.

Further, since people are represented as a single track but people's legs can produce between 0-2 observations (depending on the number of legs visible to the laser scanner and whether or not their legs have been clustered together), an identical temporary track is created for each person before the data association. If both of the person's tracks are matched to detected scan clusters, then the tracked person's position is updated with the mean of the detections' positions. If only one track is matched to a scan cluster, then the tracked person's position is updated with the mean of the detection's position and the propagated position of the person's track. After the person's original track has been updated, the temporary track is deleted.

An example image showing the estimated positions of tracked scan clusters is shown in Figure 3–4. Further, psuedocode for the data association portion of this algorithm is shown in Algorithm 2 and psuedocode for the measurement update portion of this algorithm is shown in Algorithm 3.

3.2.3 Local Occupancy Grid Mapping

To improve robustness, an odometry-corrected local occupancy grid map is constructed and updated after every observation with any scan clusters not associated with tracked people, on the assumption that these are non-human detections. The map covers a $20m \times 20m$ square area, at a resolution of $5cm \times 5cm$/cell, and is updated periodically to remain centred at the latest position of the laser scanner. By default, all cells are assumed to be freespace until updated with a non-human observation (that is, the prior used for updates, $p(m^i)$, is less than the threshold at which we assume a cell is in freespace).

The local map is used to assist with the data association by assigning the maximum cost, $cost_{max}$, to matchings between clusters in occupied space and human tracks. This is helpful to prevent situations where a tracked obstacle cluster is split due to an occlusion and, simultaneously, a nearby person's leg is occluded, potentially resulting in a match between the person track and the obstacle cluster. The map is also used to disallow person track initiations in occupied space, which can occasionally arise from stationary non-human objects which can appear to be moving due to a changing perspective of the laser scanner.

An example map constructed from a series of laser scans is shown in Figure 3–5.

3.2.4 Person Track Initiation and Deletion

To keep the number of person tracks manageable, initiation conditions are applied, using the following criteria: (1) a pair of tracks are detected, which move a

Data: Laser scan cluster centroids and confidences $Z_k = \{z_k^i, \mathsf{conf}_k^{Z,i}\}_{i=1}^M$, estimated state, covariance, confidence and human indicator of tracked objects from previous timestep $T_{k-1|k-1} = \{x_{k-1|k-1}^i, P_{k-1|k-1}^i, \mathsf{conf}_{k-1}^{T,i}, H^i\}_{i=1}^N$, local grid occupancy map from previous timestep M_{k-1}

Result: Assignments between clusters and tracked objects A_k, track-cluster distance matrix D_k

```
$T_{k|k-1} \leftarrow \varnothing$ // Propagated tracks and uncertainties
foreach $\{x, p, c, h\}$ in $T_{k-1|k-1}$ do
    $T_{k|k-1} \leftarrow$ Propogate$(x, p, c, h) \cup T_{k|k-1}$
    if $h = True$ then
        $T_{k|k-1} \leftarrow$ Propagate$(x, p, c, h) \cup T_{k|k-1}$ // Duplicate human tracks
    end
end

$P \leftarrow |T_{k|k-1}|$
$D_k \leftarrow \{cost_{max}^{i,j}\}_{i=1,j=1}^{max(P,M),max(P,M)}$ // Track-cluster distance matrix
for $i \leftarrow 1...P$ do
    for $j \leftarrow 1...M$ do
        if $d_{Malhanobis}(x_{k|k-1}^i,\ z_k^j) < D_{Malhanobis,max}$ then
            // Disallow matchings between humans and detected
               clusters in occupied space
            if $H^i = False$ or $M_{k-1}(z_k^j) = freespace$ then
                $D_k^{i,j} \leftarrow d_{Malhanobis}(x_{k|k-1}^i,\ z_k^j)$
            end
        end
    end
end

$A_k \leftarrow$ Munkres$(D_k)$ // Minimum assignment

return $A_k, D_k$
```

Algorithm 2: Cluster to track data association.

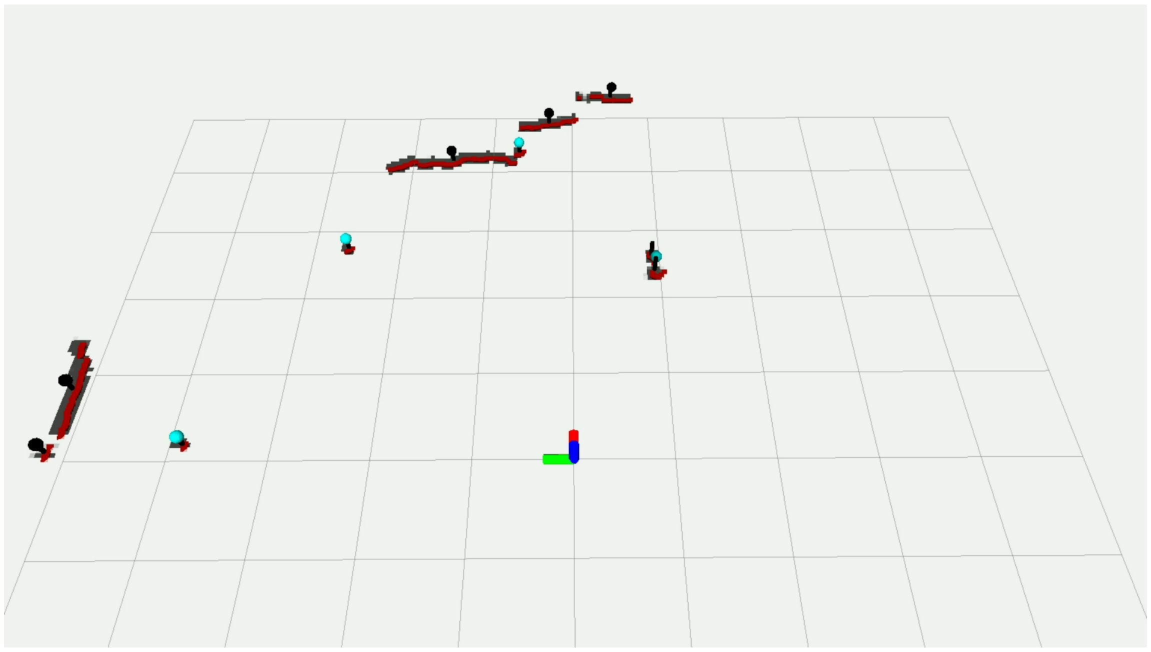

Figure 3–5: Example local occupancy grid map constructed from a series of laser scans. Black cells in the map (mostly appearing near laser scan points) show a strong occupied belief, while white cells show a strong freespace belief.

given distance ($0.5m$) without drifting apart, (2) both tracks maintain a confidence level above a threshold conf_{min}, and (3) both are in freespace (as computed from the local occupancy grid map). Upon initiating a person track, the person's position is estimated to be the mean of the positions of the pair of associated leg tracks. Then, the leg tracks are deleted and only one track is kept representing the position of the person.

Finally, human tracks are deleted when their innovation covariance $\mathbf{S}$ is greater than a threshold, or confidence over the track drops below the threshold conf_{min}.

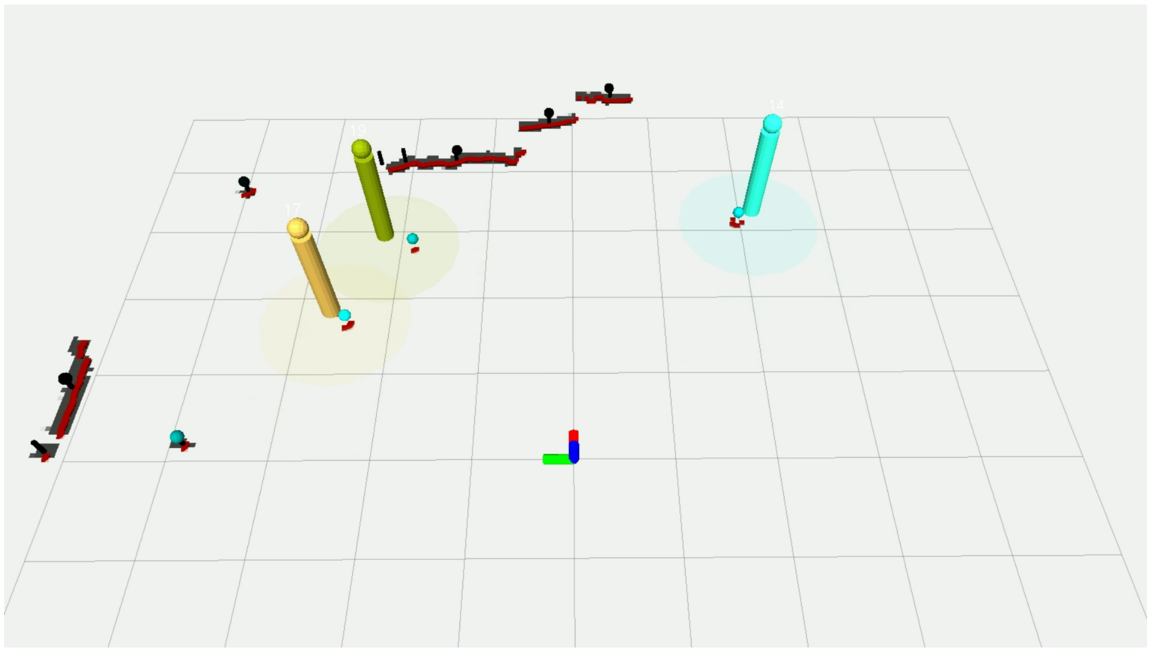

Figure 3–6: Tracked people in an example laser scan. Spheres over large cylinders show the Joint Leg Tracker's estimated positions of people in the frame.

The procedure for track initiation and deletion is described in Algorithm 4. An image showing an example of the complete tracking system's estimated position of people in a laser scan is also shown in Figure 3–6.

```
Data: Laser scan cluster centroids and confidences $Z_k = \{z_k^i, \mathsf{conf}_k^{Z,i}\}_{i=1}^{M}$,
      propogated state, covariance, confidence and human indicator of
      tracked objects from previous timestep
      $T_{k|k-1} = \{x_{k|k-1}^i, P_{k|k-1}^i, \mathsf{conf}_{k-1}^{T,i}, H^i\}_{i=1}^{P}$, assignments between clusters
      and tracked objects $A_k$, track-cluster distance matrix $D_k$
Result: Updated tracks $T_{k|k}$, matched scan clusters $Z_{matched}$
$T_{k|k} \leftarrow \varnothing$ // Updated tracks
$Z_{matched} \leftarrow \varnothing$ // Matched clusters
for $i \leftarrow 1, 2, ..., P$ do
    if IsNotDuplicate(i) then
        $j \leftarrow A_k[i]$ // Assigned cluster index by Munkres
        $z \leftarrow \varnothing$ // Cluster for measurement update
        if $D_k^{i,j} < cost_{max}$ then
            $z \leftarrow z_k^j$
            $Z_{matched} \leftarrow z_k^j \cup Z_{matched}$
        end
        if $H^i = True$ then
            $j_{duplicate} \leftarrow A_k^{Duplicate(i)}$
            if $D_k^{i,j_{duplicate}} < cost_{max}$ then
                if $z = \varnothing$ then
                    $z \leftarrow z_k^{j_{duplicate}}$
                else
                    $z \leftarrow$Avg$(z, z_k^{j_{duplicate}})$
                end
                $Z_{matched} \leftarrow z_k^{j_{duplicate}} \cup Z_{matched}$
            end
        end
        if $z \neq \varnothing$ then
            $T_{k|k} \leftarrow$ MeasurementUpdate$(x_{k|k-1}^i, z) \cup T_{k|k}$
        else
            // Update without measurement
            $T_{k|k} \leftarrow \{x_{k|k-1}^i, P_{k|k-1}^i, \mathsf{conf}_{k-1}^{T,i}, H^i\} \cup T_{k|k}$
        end
    end
end
return $T_{k|k}, Z_{matched}$
```

Algorithm 3: Track measurement update.

```
Data: Laser scan cluster centroids and confidences $Z_k = \{z_k^i, \mathsf{conf}_k^{Z,i}\}_{i=1}^{M}$,
      state, covariance, confidence and human indicator of tracked objects
      $T_{k|k} = \{x_{k|k}^i, P_{k|k}^i, \mathsf{conf}_k^{T,i}, H^i\}_{i=1}^{P}$, matched scan clusters $Z_{matched}$, local
      grid occupancy map from previous timestep $M_{k-1}$
Result: Updated tracks $T_{k|k}$ accounting for track initiation and deletion

foreach $z_k$ in $Z_k \setminus Z_{matched}$ do
    $T_{k|k} \leftarrow$ NewNonHumanTrack($z_k$) $\cup T_{k|k}$ // Unmatched clusters
end

foreach $\{x, p, c, h\}$ in $T_{k|k}$ do
    $s \leftarrow$ CalcResidualCovariance($p$)
    if $s > Cov_{max}$ or $(h = True$ and $c < \mathsf{conf}_{min})$ then
        $T_{k|k} \leftarrow T_{k|k} \setminus \{x, p, h, c\}$ // Track deletion
    end
end

for $i \leftarrow 1, 2, ..., P$ do
    for $j \leftarrow 1, 2, ..., P,\ j \neq i$ do
        if $\mathsf{conf}_k^{T,i} > \mathsf{conf}_{min}$ and $\mathsf{conf}_k^{T,j} > \mathsf{conf}_{min}$
        and $DistanceTravelledTogether(T_{k|k}^i,\ T_{k|k}^j) > D_{min}$
        and $M_{k-1}(x_{k|k}^i) = Freespace$ and $M_{k-1}(x_{k|k}^j) = Freespace$ then
            $T_{k|k} \leftarrow$ NewHumanTrack($T_{k|k}^i,\ T_{k|k}^j$) $\cup T_{k|k}$ // New person track
            $T_{k|k} \leftarrow T_{k|k} \setminus T_{k|k}^i, T_{k|k}^j$ // Remove old leg tracks
        end
    end
end

return $T_{k|k}$
```

Algorithm 4: Track initiation and deletion.

3.3 Autonomous Following: Navigation and Control

To enable the robot to autonomously follow one of the tracked people, the system has been incorporated with a person-following control system developed by N. Olmedo and H. Zhang at the University of Alberta [54], which is described in this section. Note that we do not claim the person-following control system itself is a contribution of this thesis, however, we do include the successful integration of our person tracking algorithm with it as a contribution.

To initiate person following, a single human track is selected as input to an Object Following Controller (OFC) [54]. The track can be manually selected by the robot operator, or automatically selected using a predefined decision criteria, such as picking the human closest to the robot. The OFC uses the track's latest state estimate to compute the difference (position error) from the desired position of the human with respect to the robot (position goal). The position error is then used to compute the robot's velocity setpoints for low-level actuator controllers. The motion generated decreases the position error over time and results in a trajectory that causes the robot to follow the human.

To minimize the position error of the tracked human, the OFC modulates the robot's angular and linear velocity setpoints independently. Two vectors are used for these calculations: a vector from the robot's center to the goal position, and a vector from the robot's center to the human position. The first vector is expected to be constant (as long as the goal position does not change), while the second vector changes as the human and robot move with respect to each other. The control actions aim to equalize the length of these vectors, and drive the angle between them to zero.

A Proportional-Integral-Derivative (PID) controller [3] was implemented to calculate the angular velocity setpoint using the angle between the vectors, while a second PID controller calculates the linear velocity setpoint using the difference in lengths of the vectors. Both controllers were tuned using classical Ziegler-Nichols method [71]. A dead-band zone was defined to address vibrations in the control actions when the position error is too small.

3.4 Existing Person Tracking Techniques

In additional to our novel work outlined in the previous subsections, there exist numerous previously published methods on themes of autonomous person detection, tracking and following. This section outlines many of the major prior developments in these areas. Table 3–2 is also provided to summarize the works described in this section.

Most previous work focuses on only one or two of the three identified sub-problems: detection, tracking and following. Few papers present integrated systems with state-of-the-art improvements in all three components. We focus in particular on work that uses depth sensors, such as laser or RGB-D, as those are the sensors available on our SmartWheeler robot, as well as on numerous other indoor, outdoor and assistive robots. In comparison to camera sensors, they also provide data which is lighting invariant, inexpensive to process and has a wide field of view.

3.4.1 Using Only Laser Scanners

Montemerlo et al. [49] were possibly the first to present a method for automatically locating and tracking people from laser data. They proposed using a conditional particle filter for reasoning jointly about location of the robot as well as the possible locations of people. One drawback of this work is the necessity for an *a priori* occupancy grid map of the operational environment, which is used for background subtraction to detect the person.

Schulz et al. [62] proposed to estimate the number of people in the current scan based on the number of moving local minima in the scan. Unfortunately, this requires people move continuously to be tracked, and is susceptible to poor results in cluttered environments (where the number of local minima is misleading). They also introduced a Sample Based Joint Probabilistic Data Association Filter (SJPDAF) over the observed local minima to improve tracking reliability.

Topp et al. [69] extended [62], by picking out shapes of legs and person-wide blobs in laser scans using hand-coded heuristics, to allow detection and tracking of both stationary and moving people. The approach was also combined with a person following navigation algorithm, combining both the tracked person's position as well as the location of nearby obstacles to determine suitable control. Gockley et al. [26] used a similar approach, with a few modifications, including using a Brownian motion model for the tracking component. This approach was further extended by Hemachandra [31], which improved the person-following component by proposing a navigation approach that accounts for personal space, while avoiding obstacles. Unfortunately these approaches cite tracking difficulties in cluttered conditions, since they relied primarily on detecting clusters of a heuristically-determined size in the laser scan.

More recently, Arras et al. [2] reduced this limitation by proposing a method that detects legs by first clustering scan points and then using supervised learning to learn shapes of leg clusters. Detected legs are tracked over time using constant-velocity Kalman filters and a multiple-hypothesis tracking (MHT) data association technique. This approach benefits from its ability to maintain (but not initiate) tracks of stationary people, and does not require an *a priori* occupancy grid map of the environment. Initial results for this method appear promising, but demonstrations on walking-speed robots in cluttered and crowded areas have yet to be performed. Thus, many questions remain about the robustness and generalizability of the approach.

Finally, Lu et al. [44] extended an existing ROS package originally developed at Willow Garage, that had not been formally published.[2] This method first matches scan clusters using NN data association, then determines which clusters are human legs using a supervised learning approach similar to [2]. In the data association literature, it is generally agreed that the NN filter is outperformed by more sophisticated methods, such as the GNN filter, SJPDAF or MHT [9]. Furthermore, the data association approach only considers absolute Euclidian distances while ignoring valuable uncertainty covariances that can improve tracking.

Navarro-Serment et al. [53] present the only published work, which we are aware of, that aims to track people from a mobile robot in hilly, outdoor environments using

[2] http://www.ros.org/news/2009/12/person-following-and-detection-in-an-indoor-environment.html

only 2D laser scanners. The method's applicability to cluttered or crowded environments is questionable since it uses a NN data association approach and clusters scan points with a relatively large threshold of $80cm$ and would thus be prone to clustering people together with nearby obstacles.

3.4.2 Using Other Sensor Modalities

Munaro et al. [52] proposed to track people in real-time with an RGB-D sensor on a mobile robot, demonstrating promising results with high update rates running on a CPU only. Similarly, Gritti et al. [28] demonstrated a method for detecting and tracking people's legs on a ground plane from a low-lying viewpoint with the an RGB-D sensor. However the RGB-D sensor's narrow field of view, minimum distance requirement, and inability to cope with sunlight, limit this technique's applications for person-following.

Cosgun et al. [16] presented a novel person following navigation technique which is most similar to a Dynamic Window Approach [25]. The initial detection was acquired via a user indicating the desired person they wish to follow in the an RGB-D sensor's image. Tracking was then achieved via lasers using estimated leg positions. This approach can only track one person, and uses a NN matching over a small number of laser segment features, thus is not robust in situations with multiple individuals.

Kobilarov et al. [37] developed a person-following Segway robot using an omnidirectional camera and a laser scanner. Nonetheless, variable environment lighting, backgrounds and appearances of people are factors which can be difficult to control for and can mislead vision-reliant tracking systems.

Table 3–2: Overview of related work on people tracking on mobile robots with planar laser scanners.

Paper	Sensors	Environ.	Detection	Data Assoc.	Following Control
Montemerlo et al. [49]	laser	indoor	background subtraction	Conditional PF	none
Schulz et al. [62]	laser	indoor	moving local minima	SJPDAF	none
Topp et al. [69]	laser	indoor	leg shapes (heuristic)	SJPDAF	$\sim$ potential field
Gockley et al. [26]	laser	indoor	leg shapes (heuristic)	NN	direct & path
Hemachandra et al. [31]	laser	indoor	leg shapes (heuristic)	NN	$\sim$ potential field
Arras et al. [2]	laser	indoor	leg shapes (ML)	MHT	none
Lu et al. [44]	laser	indoor	leg shapes (ML)	NN	none
Munaro et al. [52]	RGB-D	indoor	height & HOG	GNN	none
Gritti et al. [28]	RGB-D	indoor	leg shapes (ML)	NN & PDAF	none
Cosgun et al. [16]	laser & RGB-D	indoor	leg shapes (heuristic)	NN	$\sim$ dynamic window
Kobilarov et al. [37]	laser & omni-camera	outdoor	person shapes (heuristic)	PDAF	direct & path
Navarro-Serment et al. [53]	laser	outdoor	person shapes (heuristic)	NN	none

CHAPTER 4
Benchmarks and Evaluation

This chapter describes how the tracking system detailed in Chapter 3 has been implemented on multiple robots and evaluated in real-world scenarios.

4.1 Benchmarks

To objectively evaluate the performance of our tracking system against existing and future works, an objectively-evaluated benchmark is required. Some prior benchmarks for the evaluation of the person detection and tracking module are available [45], however to the best of our knowledge, they are collected from a stationary laser scanner and are at a height of $0.8m$, which is above the leg-region of many pedestrians and thus incompatible with our tracking method. The method of ground-truth annotation is also ambiguous, as it is unclear under what conditions a person is labelled as visible or occluded (an issue described in [48]). Further, the datasets are not ROS-enabled and the person-detection module used to detect people with the given laser scanner has not been released, making comparisons difficult on other platforms.

To palliate some of these gaps, and allow thorough evaluation both of our own and future methods, we collected and will be publicly sharing two new benchmarks for person detection and tracking, detailed in Table 5–1. These provide 45 minutes of data recorded onboard moving robots, are fully ROS-enabled and include annotated people tracks labelled in the laser scans using an objective, unambiguous procedure, with video data to corroborate ground-truth when necessary.

4.1.1 Robot Platforms

Two different types of robots were used for benchmark data collection: the Smartwheeler robot was used in indoor environments, while the Husky A200 was used in outdoor environments.

The SmartWheeler robot, shown in Figure 4–1, is an intelligent powered wheelchair, designed in collaboration with engineers and rehabilitation clinicians, aimed at assisting individuals with mobility impairments. The robot is built upon a commercial power wheelchair base, to which we have added onboard computing, three Hokoyu UHG-08LX laser scanners, several sonars, an RGB-D sensor, on-wheel odometry, and a touchscreen for two-way interaction. One of the important tasks for the robot is to automate navigation in challenging environments (crowded rooms, narrow spaces) to reduce physical and cognitive load on the wheelchair user [34]. The task of moving around with another individual, whether a friend or caregiver, is also one that requires substantial concentration, and wheelchair users often comment that it is difficult for them to simultaneously control their wheelchair (presumably via joystick) and hold a conversation, thus having the ability to use autonomous navigation capabilities for walking side-by-side with (or behind) another person is highly desirable [34].

A Husky A200 from Clearpath Robotics, shown in Figure 4–2, is used for the outdoor experiments in this work. It is a differential-drive robot which is used for mapping, localization, route following and autonomous navigation on rugged terrain, such as sand, gravel and grass. An on-board computer is used to interface low-level controllers and sensors, as well as to process visual and range measurements. Typically, an additional computer is mounted externally, on top of the platform for applications with high computational requirements. It is equipped with an RGB-D sensor and two Hokuyo URG-04LX-UG01 laser scanners. One laser scanner is mounted planar to the ground for detecting peoples' legs and other obstacles, while the scanner is at an angle approximately $45°$ off the horizontal for detecting peoples' upper bodies. The scanner mounted planar to the ground is used to collect the 2D scans used in this study.

The random forest classifier, introduced in Section 3.1, was trained for both robots on a set of 1700 positive and 4500 negative examples [13]. Positive examples were obtained by setting up the laser scanner in an open area with significant pedestrian traffic; all clusters which lay in the open areas and met the threshold in Sec. 3.1 were assumed to be the result of people and were used as positive training samples. Negative examples were obtained by moving the sensor around in an environment devoid of people; all clusters which met the threshold in Sec. 3.1 were used as negative training samples.

4.1.2 General Multi-person Tracking

The first benchmark, called *General multi-person tracking* is designed for evaluating performance of general multi-person tracking of pedestrians in natural environments. The benchmark is in fact composed of two datasets, one collected from a stationary robot and one collected from a moving robot. The stationary dataset includes 7 minutes of tracking data, while the moving dataset includes 5 minutes of data, and both are fully annotated, including 82 identified people tracks. The data was recorded onboard the SmartWheeler in the hallways of a university campus building during normal opening hours. Each recording includes odometry data published at 100Hz and laser data published at 7.5Hz. Video data was also captured, but was only used to as a reference for annotations of people in the laser scans, and was not included in the final curated benchmarks (though is available on demand).

All ground-truth people positions were hand-labelled in each laser scan. To address the issue of whether or not a person should be labelled as visible if they are partially occluded (an issue raised in [48]), a consistent and objective rule was used: if a minimum of at least three laser points can be clustered with a Euclidean distance of $0.13m$, and that cluster of points corresponds to a person, then they are marked as visible in the annotations.

4.1.3 Tracking for Following

The second benchmark, called *Tracking for following*, is designed to measure the performance of the detection and tracking modules when applied specifically to the task of tracking an individual person in the presence of other pedestrians. This benchmark is also composed of two separate datasets.

In the Following Indoor dataset, the person to be followed was asked to walk naturally and stop periodically to interact with objects in the environment while the SmartWheeler followed either from behind or side-by-side, simulating a person-following situation (for data collection in this benchmark, the robot was manually controlled). The environment is the same university building as for the *General multi-person tracking* benchmark, and includes scenes with natural crowds and clutter, as well as five instances of pedestrians walking between the SmartWheeler and the person it was following. Odometry was collected at a frequency of 100Hz and laser scans at 15Hz. The position of the person being followed was hand-labelled in every frame. Altogether, 21 minutes of data were gathered and annotated, including 4.5 minutes of side-by-side following and 16.5 minutes of following from behind.

Figure 4–3 shows an occupancy grid map constructed using the *gmapping* ROS package [27] from scans of the university building where the data was collected.

The Following Outdoor dataset was collected with a Clearpath Husky robot at the Canadaian Space Agency in Saint-Hubert. The laser scanner was mounted level with the ground, approximately $40cm$ high. The dataset contains 12 minutes of data, also fully annotated. Laser scans were collected at a frequency of 10Hz. Odometry data was collected as well but was ultimately not used as it was found to be detrimentally inaccurate. The challenge in this dataset is dealing a high degree of sensor noise, as the laser scanner used is indoor-rated and highly susceptible to interference from the sun. In this case, the robot was controlled autonomously to follow the person using the navigation system presented in Sec. 3.3 and an earlier version of the Joint Leg Tracker presented in Sec. 3. During the experiment, the tracker failed on two separate occasions and steered the robot such that the person to be tracked was lost out of frame. Figure 4–4 shows sample images from this dataset.

4.2 Comparison Person Tracking Approaches

Quantitatively comparing to existing methods from the literature is challenging due to a scarcity of publicly available code. In our search, we were only able to find one other open-source ROS-enabled method which could be compared to ours: the ROS leg_detector package used in [44]. To adapt the ROS leg_detector to our benchmarks, all parameters were kept at their default values except the confidence threshold was lowered, as the tracker would commonly fail to initiate people tracks with its default value. Also, the clustering distance and minimum required points per cluster were set to be the same as for our method.

A variant of our algorithm, which we call the Individual Leg Tracker, is also included in the comparison. It is identical to the Joint Leg Tracker described in Sec. 3, except that all tracking is performed on an individual-leg level (not pairs of legs) and it does not use a local occupancy grid map for data association. In this case, person tracks are deleted when the two leg tracks separate beyond a distance threshold, one of the leg tracks has too low of a confidence, or one of the leg tracks is deleted due to its covariance exceeding a threshold.

Minor modifications of all methods were made to allow for repeatable results on the benchmarks. This was necessary because the tracking methods require coordinate transformations be made in real-time, which can cause benchmarks results to vary depending on the exact time these transformations are made. Each method was therefore set to use the scan header time to perform the transforms and, when they were not available, to wait for one second of real-time and, if they were still not available, the current scan was skipped (although, we found this happened only very rarely). An option was included in each method to never wait for transforms to become available but instead to always use the most recent ones. This is how the system is intended to be used in practice and was the variation used for runtime profiling.

4.3 Evaluation Metrics

4.3.1 General Multi-person Tracking

We use the CLEAR MOT metrics for quantitative evaluation of the tracking system [36]. They are commonly used metrics for multi-object tracking, and provide scorings of valid assignments, ID switches, misses, false positives (FPs) and precision of matchings cumulated from every frame. These scores can be aggregated into a combined overall multi-object tracking accuracy (MOTA) score

$$MOTA = 1 - \frac{\sum_k (ID_k + Miss_k + FP_k)}{\sum_k g_k} \tag{4.1}$$

where ID_k, $Miss_k$, FP_k and g_k are number of ID switches, misses, FPs and ground truth annotations respectively, at time k. However, the aggregation of the scorings in the combined MOTA score assumes that three types of errors (ID switches, misses and FPs) are equally unfavorable, which is arguably not true in most applications, including ours. We therefore report the raw count of each error type alongside the MOTA score.

Another relevant CLEAR MOT metric is the mult-object tracking precision (MOTP), which is defined as

$$MOTP = \frac{\sum_{i,k} d_k^i}{\sum_k c_k} \tag{4.2}$$

where c_k is the number of matchings made between estimated people positions and ground truth positions at time k and d_k^i is the distance between the ith match. Intuitively, it provides a measure of how precise a target is tracked when it is being tracked properly. In the context of person-following, a lower MOTP score would be beneficial because a more exact location estimate of the person being followed would presumably allow the following controller to track the person more precisely.

A threshold distance of $0.75m$ was used for the CLEAR MOT matching to determine whether a tracked person should be matched to an annotated ground truth person.

4.3.2 Tracking for Following

For evaluation in the *Tracking for following* scenarios, the same CLEAR MOT metrics are used. However, since we are only concerned with the tracker's ability to track a particular person (among several), only the ground-truth positions of the target person were labelled. Other pedestrians were ignored, and only person-tracking events corresponding to the target person are reported (i.e., FPs were ignored because they may have been due to unlabelled pedestrians).

Under such an evaluation framework, the relevant metrics are ID switches and misses. ID switches represent cases where the track of the target person was switched with someone or something else, and it would no longer be possible to autonomously follow them. Misses represent cases where the target person was visible in the laser scanner's field of view but was not tracked. In such cases, the robot user would not be able to lock into the target person to initiate following.

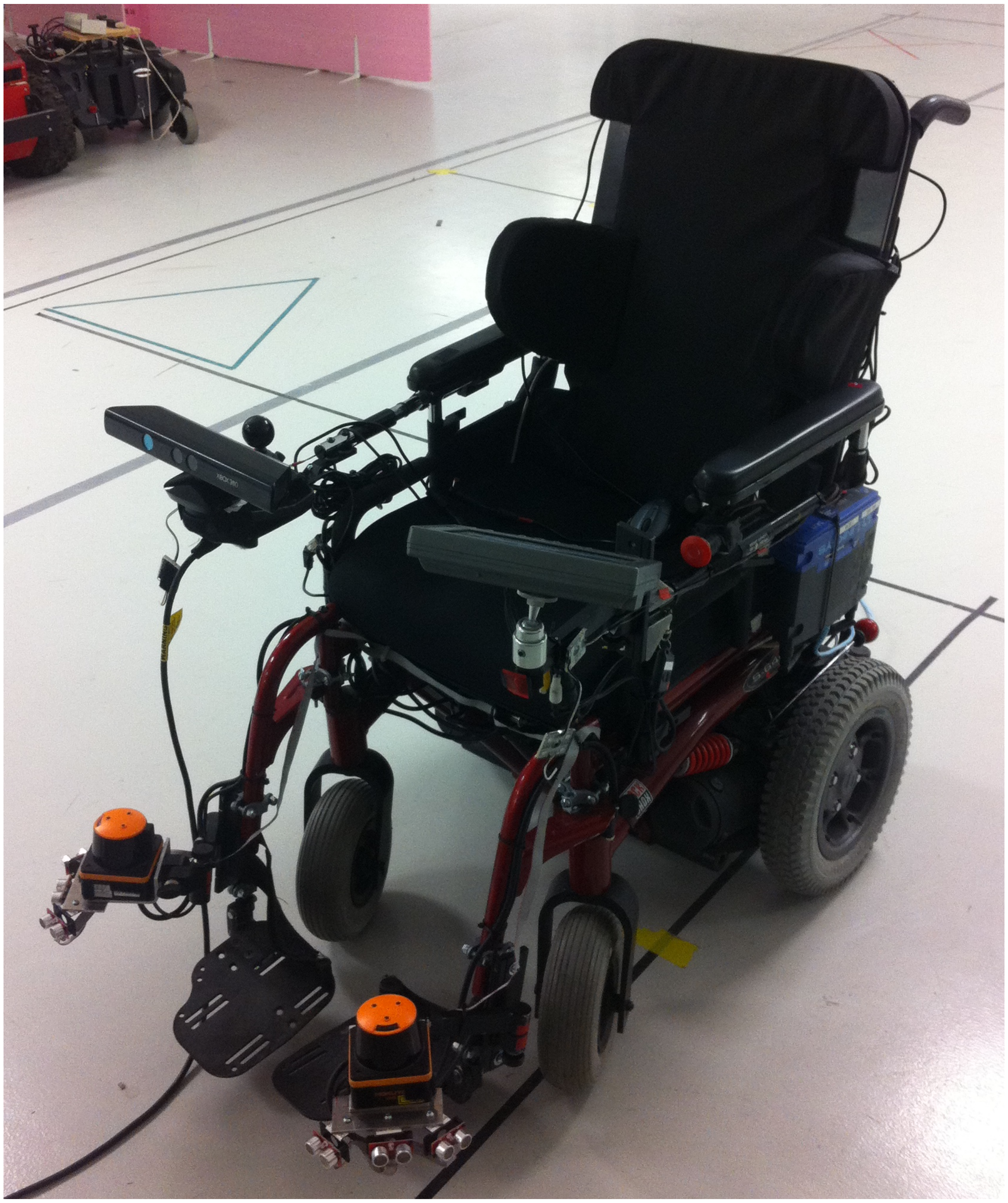

Figure 4–1: The SmartWheeler robot. Visible are two front Hokuyos laser scanners (the rear Hokuyo is hidden), the Kinect RGB-D sensor, six sonars and the touch screen.

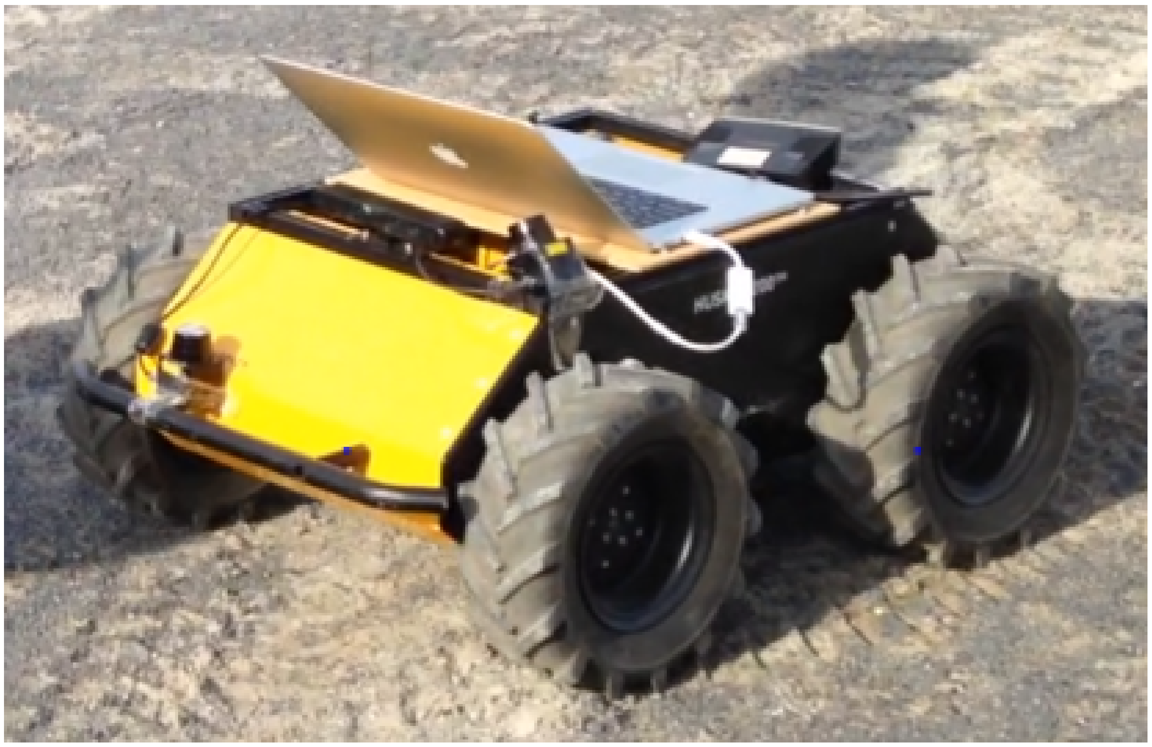

Figure 4–2: The Husky A200 from Clearpath Robotics. Visible are two Hokuyo laser scanners an RGB-D sensor. Only the laser scanner mounted on the front bumper was used in this work.

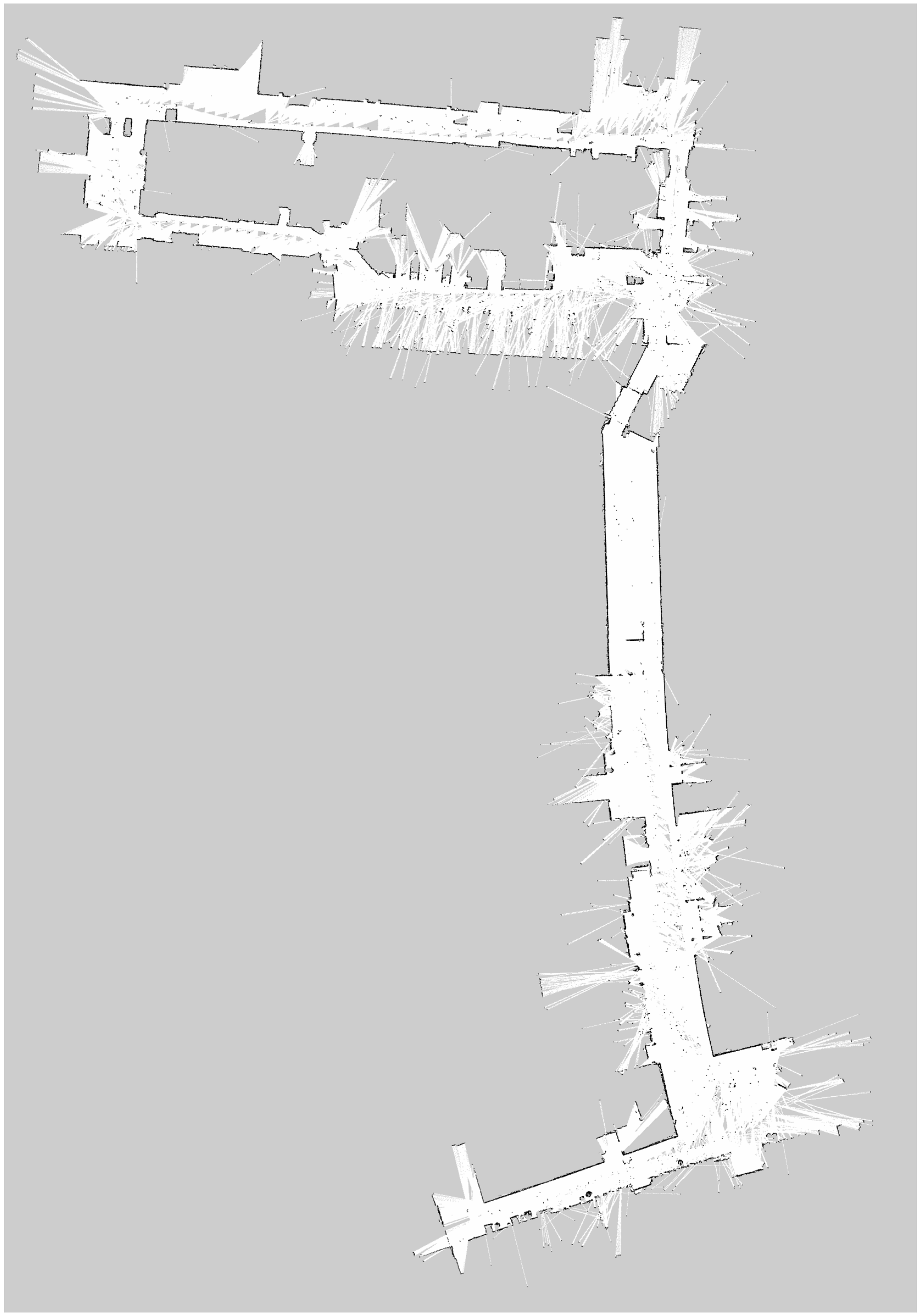

Figure 4–3: Occupancy grid map showing some of the data collection area for the *Tracking for following* benchmark in the university campus building. The map was constructed using the *gmapping* ROS package from [27]. Note that such maps are not provided *a priori* on the tracking benchmarks.

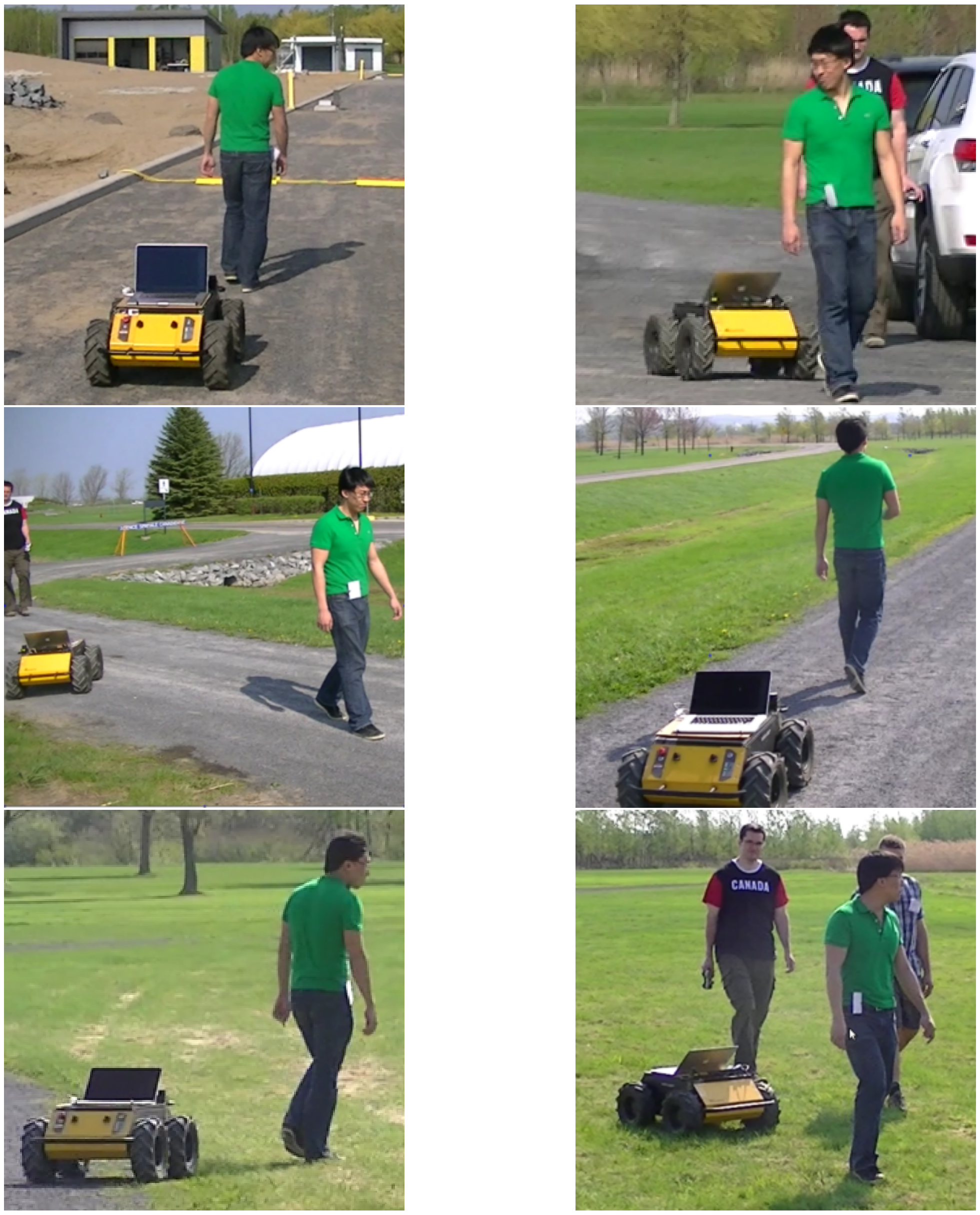

Figure 4–4: The person tracking and following system implemented on a Clearpath Husky during the collection of the Following Outdoor dataset. The robot autonomously followed the participant in an approximately $500m$ loop on gravel and grass at the Canadian Space Agency in Saint-Hubert. Two tracking failures occurred, one of which is shown in the centre-left image, which were caused by extensive sensor noise, as the laser scanner used is indoor-rated and highly susceptible to interference from the sun.

CHAPTER 5
Results and Discussion

This chapter describes, discusses and compares the results of all person-tracking methods on the benchmarks presented in Chapter 4.

5.1 Benchmark Results

Results from all datasets described in Chapter 4 are shown in Table 5–2. The Joint Leg Tracker achieves the best results in the *Tracking for following* tasks by a large margin. It suffers from only 7 ID switches in the 21 minutes of the Following Indoor dataset and makes virtually no ID switch errors in the Following Outdoor dataset (assuming the 2 ID switches which were caused by the person being tracked moving out of frame for a significant amount of time to not be preventable). It also achieves the lowest MOTP on all datasets, meaning that when a person is being tracked properly, it provides the most precise estimate of their location. This would presumably improve the performance of the following controller, which uses the location estimate to perform closed-loop control.

Its performance when applied to the *General multi-person tracking* benchmark is also favourable but by a lesser margin. It causes significantly less of every type of error on the Stationary Robot dataset compared to the others and achieves similar results to the Individual Leg Tracker on the Moving Robot dataset.

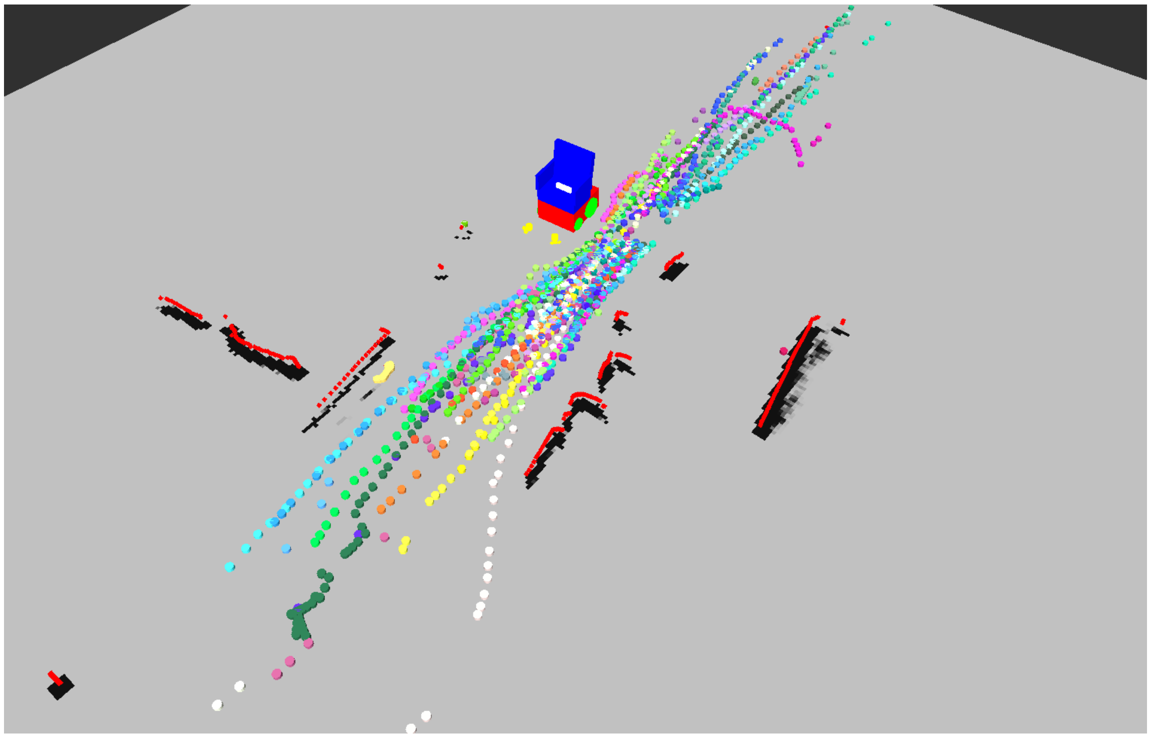

Figure 5–1: Joint Leg Tracker's cumulative estimated positions of people on the *General multi-person tracking*, stationary dataset. The estimated positions are shown in multi-coloured markers on the ground-plane. Further, the simulated model of the SmartWheeler used for data collection is shown in the top-middle of the figure. The figure also shows the lastest laser scan in red, as well as the local grid occupancy map in grey and black.

The leg_detector is outperformed in almost all respects by the other tracking methods and is generally unable to track anyone persistently in the challenging benchmark environments, as is shown by its high number of ID switches in all cases.

Fig. 5–1 shows an example output of the Joint Leg Tracker on the *General multi-person tracking*, stationary dataset. It is evident in this figure that very few FPs appear on or near stationary objects in the environment suggesting the local grid occupancy map is effective at preventing these types of errors.

5.1.1 Runtime

The average-case runtime of the Joint Leg Tracker is faster than the maximum scanner frequency in all cases when run on an Intel Core i7 CPU. In the Following Indoor dataset, which has frequent open areas containing many distinct objects to be tracked, the slowest single update is slower than the scan frequency by a factor of approximately four, though the average runtime over the entire dataset is still better than real-time. The bottleneck in these cases tends to be the tracking and data association step, which is implemented in Python. In the future we plan to test a C++ implementation and, if this does not achieve worst-case, real-time performance, limit the number or distance of clusters which are tracked.

5.2 Discussion

The overall CLEAR MOT scores presented in the benchmarks are significantly lower than those presented in [45]. However, the results are not directly comparable because of the different sensor configurations and environments. For example, the laser sensor used in the *General multi-person tracking* benchmark has a lower scanning frequency (7.5Hz vs 37.5Hz) and a shorter range ($8m$ vs $80m$) than the laser sensor from the benchmarks in [45]. The shorter range naturally increased the number of misses since pedestrians would enter and leave the field of view quicker and with fewer detections, and would therefore spend proportionally more time being tracked as non-persons before person tracks were initiated.

Table 5–1: Overview of benchmarks.

Benchmark	Dataset	Duration	Avg. estimated robot speed when moving (m/s)	Annotated people tracks	Avg. dist. to person followed (m)
General multi-person tracking	Stationary Robot	7m11s	n/a	45	n/a
	Moving Robot	5m01s	0.9	37	n/a
Tracking for following	Following Indoor	21m24s	0.9	1	1.33 ± 0.57
	Following Outdoor	12m00s	0.6	1	1.68 ± 0.30

Table 5–2: Benchmark results. Bolded values indicate best results for the given metric.

Dataset	Tracker	Valid	ID Switch	Miss	FP	MOTA	MOTP (m)	Runtime Worst/Avg (Hz)
Stationary Robot	leg_detector	427	51	1605	28	19.2%	0.28	19/25
	Individual Leg	569	10	1504	61	24.4%	0.23	14/25
	Joint Leg	**703**	**8**	**1372**	**11**	**33.2%**	**0.16**	15/25
Moving Robot	leg_detector	149	15	481	294	-22.5%	0.27	19/25
	Individual Leg	160	**2**	483	**88**	**11.2%**	0.17	12/24
	Joint Leg	**163**	**2**	**480**	97	10.2%	**0.15**	11/24
Following Indoor	leg_detector	15425	196	3425	n/a	n/a	0.15	18/25
	Individual Leg	17295	73	1678	n/a	n/a	0.13	4/19
	Joint Leg	**18554**	**7**	**485**	n/a	n/a	**0.09**	4/19
Following Outdoor	leg_detector	4835	172[1]	1172	n/a	n/a	**0.09**	22/25
	Individual Leg	5348	12[1]	819	n/a	n/a	0.11	15/25
	Joint Leg	**6073**	**2**[1]	**104**	n/a	n/a	**0.09**	18/25

[1] Due to the two cases where the person followed was lost out of frame for a significant amount of time, it is reasonable to expect a minimum of two ID switches in this dataset.

CHAPTER 6
Conclusion

6.1 Summary

This work presents a novel method for detecting, tracking and following people using laser scanners mounted at leg-height. The system integrates a novel Joint Leg Tracker algorithm with local occupancy grid maps and a method of tracking all scan clusters, including non-human clusters, to improve tracking in cluttered areas. It is fundamentally different from previous approaches, such as [2, 44], which require both legs remain visible to the laser scanner for tracking. Empirically, it has shown to be effective in the target application of autonomous person following and presents advantages in general multi-person tracking as well. The tracking method has also been verified in practice on two separate robot platforms: the SmartWheeler and a Clearpath Husky. It has additionally been integrated with an autonomous following navigation system for the Clearpath Husky deployment.

6.1.1 Contributions

The contributions of this work include the novel Joint Leg Tracker algorithm, which has shown to achieve state-of-the-art performance for person tracking using planar laser scanners on a set of objectively-evaluated benchmarks. This work also provides a public release of the benchmarks to facilitate quantitative evaluation of methods developed in the future. Further included is a validation of the presented approached on two separate robots: the Smartwheeler and a Clearpath Husky. Finally, this thesis also describes how the tracking system can be advantageously integrated with an autonomous person-following control system.

6.2 Limitations

Although we believe, and have shown in objective benchmarks, that this work presents a significant improvement over existing systems, it does suffer from several limitations which are discussed in this section.

6.2.1 Hardware Requirements

The algorithm necessitates the robot being equipped with a planar laser scanner. They tend to be fairly common on many of today's mobile robotic systems as they are often used for mapping and navigation [38, 27]. Thus, many of today's robots should be capable of running the algorithm with no physical modification, however, some still will require alteration and outfitting with the sensor.

Further, the provided software implementation of this work contains a confidence calculation module which was trained on a Hokoyu UHG-08LX laser scanner with an angular resolution of 1/3°. It is currently unknown how laser scanners of other resolutions would affect the confidence calculation, but we suspect they would require retraining of the random forest classifier used by it to achieve optimal performance.

The limitations of the laser scanner can be compared to those of a camera, which are a common alternative sensor for performing person detection and tracking in other works [61]. Cameras benefit from lower cost and possibly being less intrusive to install, however laser scanners have higher accuracy in depth measurements, wide field of views and the aforementioned existing presence on many of today's mobile robots. Thus, from a hardware procurement perspective, the preferred sensor depends on the context and use cases.

6.2.2 Laser Scanner Height

The algorithm requires the laser scanner be mounted at approximately leg-height. This is due to its necessity of two distinct tracked clusters for initiating a person track. If, for example, the laser scanner is mounted at waist-height or above, then there may only exist one tracked cluster and the system would therefore be unable to initiate a person track. This issue could potentially be remedied in future work by re-training the confidence calculation using samples from different laser scanner heights, and modifying the tracking algorithm to only require one scan cluster to initiation a person track.

6.2.3 Ground-Plane Assumption

Another limitation of the presented algorithm is that it assumes the robot is operating on an approximately planar area, with the laser scanner mounted roughly level to the ground plane. If the laser scanner were mounted at a downwards angle or the operating area were heavily sloped, ground returns in the laser scan would likely degrade performance of the tracker.

6.2.4 Data Association

Another potential limitation of the presented system is the GNN data association method. While it is computationally faster than more sophisticated methods, such as the MHT and the JPDAF, quantitative experiments exist demonstrating better assignment performance of these more sophisticated data association methods in other application areas. For example, meta-results reported by [9] suggest that the JPDAF and the MHT generally track targets better in high-clutter environments with many false positive detections when used in the task of radar tracking. However, quantitative comparisons specifically in the area of people tracking in 2D laser scans would be beneficial because it is a fundamentally different domain. For example, false positive detections are produced systematically by objects in the scan, rather than randomly over the scanning area, as is often assumed in radar tracking, and can therefore be accounted for explicitly (e.g., as in our tracker or in [46]).

6.2.5 Evaluation Metrics:

While the CLEAR MOT evaluation metrics were the primary method of evaluation used in this work, other methods exist which have been used by other recently-published works in the target tracking literature. These include trajectory-based measures [70], configuration-based measures [64], and a global missmatch method [7].

As outlined in [48], some of the drawbacks of CLEAR MOT metrics include inconsistent penalizing of tracking errors in some situations with groups of people, or its lack of penalizing ID switches in quick interactions from multiple people [50]. However, we believe that despite these drawbacks, it is nonetheless a reasonable choice for quantitative evaluation because all other available alternative methods suffer from drawbacks of their own. For example, a previously used trajectory-based metric [70] lacks granularity, only discriminating tracks with three metrics: "mostly tracked", "fragmented" and "mostly lost". Another example is that the configuration-based approach of [64] provides non-intuitive measurements of FPs and misses and has nine metrics for describing results, which can make comparisons between tracking methods difficult [36].

6.3 Impact and Future Work

The system development and experiments completed thus far indicate that we have achieved a portable and reliable system. Moving forward, we intend to verify the usefulness of the system on the SmartWheeler with the target user population. One goal will be to determine if the system is capable of allowing a smart wheelchair user to comfortably carry on a conversation with a companion while autonomously following them side-by-side in busy environments. Additionally, we plan to test the following from behind functionality, comparing it with side-by-side following, to the determine conditions in which each approach is favourable.

In the following subsections, we further discuss some potential areas for future work related to this thesis.

6.3.1 Fusion with Complimentary Sensors

One failure mode of the current system is that it can lose track of people due to occlusions and cannot reidentify them again when they re-enter the field of view. To address this, we plan to examine ways the current system can be advantageously fused with information from camera or RGB-D sensors. One possibile approach would be to use the camera to learn colour features for recognizing tracked people and then use these features to perform person re-identification when new tracks are initiated.

Another failure mode is that false positive and false negative detections can occur when objects in the surroundings are confused with people. Again, fusion with a camera or RGB-D sensor could reduce some of the these cases. Both of the sensors capture colour images which can be applied to person detection modules (e.g., [63]) to detect and track people outside the range of the laser scanner, as well as to prune false positive detections within range. Further, RGB-D sensors provide a high-resolution depthmap, which has shown to be highly capable of distinguishing between people and other objects [52].

6.3.2 Incorporating *a priori* Occupancy Grid Maps

False negative and false positive detections could potentially be further reduced if this method were to utilize pre-built maps of static obstacles in the environment. This has proven to be beneficial in previous work concerned with laser-based person tracking [49].

To provide an example, a naive way to integrate *a priori* maps could be to use them in place of the local occupancy grid maps in the Joint Leg Tracker. This would require first localizing the robot accurately on the map, which could be done using an existing particle filter localization implementation in ROS [24]. Then, the *a priori* maps could potentially be more accurate than the locally constructed maps and our system might require fewer frames to perform track initation and track deletion.

Further in this line of work, the algorithm could incorporate information from large-scale maps built as the robot is in operation. For example, one could use large-scale SLAM systems, such as [38, 27], to build occupancy grid maps in real time, then use this information to supplement the tracking algorithm when revisiting prior locations. This would allow the system to not be dependant upon *a priori* maps but still benefit from large-scale maps constructed during operation.

6.3.3 Linear Assignment Problem

This work uses the Munkres algorithm to solve the linear assignment problem in the GNN data association method [40]. However, another method by R. Jonker and T. Volgenant [33] has shown to improve assignment runtime efficiency in practical applications in other works [21]. Thus, it would likely be beneficial to integrate this algorithm in a future iteration of this work. It is worth noting, however, is that the algorithm from R. Jonker and Volgenant scales with the same complexity ($\mathcal{O}(n^3)$) but its improvements over Munkres in empirical applications have nonetheless been lauded [21] due to a lower constant-multiple of the complexity.

6.3.4 Application to Clinical Locomotion Analysis

The work presented in this thesis is also under investigation for its potential to help automatically assess patients' locomotion patterns following an injury in the context of a rehabilitation therapy [42]. Previous approaches to this assessment utilized fixed motion-capture systems, such as in [6] and [41]. Since the presented approach instead uses a laser range scanner, it benefits from greater portability, less cost and allowing its users to be tracked naturally as they appear without the need for added markers.

At the clinical level, the system is being prepared for characterizing the locomotion behavior of stroke patients both in a rehabilitation hospital and in natural living spaces. The latter deployment in natural living spaces is particularly promising, as this is the type of study which was not feasible with a fixed motion capture system. Thus, it is hoped that the present tracking approach is able to capture behaviors not observed in clinical settings.

6.3.5 Application to Security, Entertainment and Marketing

We envision this work also has the potential for more wide-ranging beneficial applications in areas such as security, entertainment and marketing.

A prototype physical security robot has been designed and presented in previous works [20]. It is a mobile robot which monitors indoor building environments and has the capability of interacting with the people and objects it detects in its surroundings. The prototype relies upon 2D laser scanners as well as vision for perception and uses a laser-based person detection and tracking method. This, and other mobile security robots, may benefit from the improved tracking performance of our method.

Minerva [67] is an example of an interactive tour guide robot which could enhance the experience of museum visitors. It used laser and vision sensors to estimate the position of itself and people in a building as well as for navigation when leading people in guided tours. The robot could potentially have operated more effectively in these populated environments if it had had more accurate information regarding the locations and identities of people in its surroundings. Thus, it may have benefited from an implementation of our method.

Finally, our system could be used to improve retail sales and marketing by tracking the motion patterns of shoppers as they walk through stores [17]. This information could then be used for decision support and optimizing processes such as inventory selection and store layout.

References

[1] Kai O Arras, Slawomir Grzonka, Matthias Luber, and Wolfram Burgard. Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *International Conference on Robotics and Automation (ICRA)*, 2008.

[2] Kai O Arras, Boris Lau, Slawomir Grzonka, Matthias Luber, Oscar Martinez Mozos, Daniel Meyer-Delius, and Wolfram Burgard. Range-based people detection and tracking for socially enabled service robots. In *Towards Service Robots for Everyday Environments*, pages 235–280. 2012.

[3] Karl J Astrom. *PID controllers: theory, design and tuning.* Research Triangle Park, 1995.

[4] Yaakov Bar-Shalom and Xiao-Rong Li. *Multitarget-multisensor tracking: principles and techniques*, volume 19. Yaakov Bar-Shalom, 1995.

[5] Yaakov Bar-Shalom and Edison Tse. Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460, 1975.

[6] Roain Bayat, Hugues Barbeau, and Anouk Lamontagne. Speed and temporal-distance adaptations during treadmill and overground walking following stroke. *Neurorehabilitation and Neural Repair*, 19(2):115–124, 2005.

[7] Horesh Ben Shitrit, Jerome Berclaz, Francois Fleuret, and Pascal Fua. Tracking multiple people under global appearance constraints. In *International Conference on Computer Vision (ICCV)*, 2011.

[8] Christopher M Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[9] Samuel Blackrnan and Robert Popoli. *Design and analysis of modern tracking systems.* Artech House, 1999.

[10] Gary Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[11] Leo Breiman. *Classification and regression trees.* Chapman & Hall/CRC, 1984.

[12] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Rainer E Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems, revised reprint.* SIAM, 2009.

[15] Zhe Chen. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[16] Akansel Cosgun, Dinei A Florencio, and Henrik I Christensen. Autonomous person following for telepresence robots. In *International Conference on Robotics and Automation (ICRA)*, 2013.

[17] Doug Cox, Darren Fairall, Neil MacMillan, Dimitri Marinakis, David Meger, Saamaan Pourtavakoli, and Kyle Weston. Trajectory inference using a motion sensing network. In *Canadian Conference on Computer and Robot Vision (CRV)*, 2014.

[18] Ingemar J Cox. A review of statistical data association techniques for motion correspondence. *International Journal of Computer Vision*, 10(1):53–66, 1993.

[19] Lngemar J Cox and Sunita L Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[20] Donato Di Paola, Annalisa Milella, Grazia Cicirelli, and Arcangelo Distante. An autonomous mobile robotic system for surveillance of indoor environments. *International Journal of Advanced Robotic Systems*, 7(1):19–26, 2010.

[21] Stefan Fankhauser, Kaspar Riesen, and Horst Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *Graph-based Representations in Pattern Recognition*, pages 102–111. Springer, 2011.

[22] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[23] Tully Foote. tf: The transform library. In *International Conference on Technologies for Practical Robot Applications (TePRA)*. IEEE, 2013.

[24] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.

[25] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[26] Rachel Gockley, Jodi Forlizzi, and Reid Simmons. Natural person-following behavior for social robots. In *International Conference on Human-Robot Interaction*, 2007.

[27] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[28] Armando P Gritti, Oscar Tarabini, Jerome Guzzi, Gianni Di Caro, Vincenzo Caglioti, Luca M Gambardella, and Alessandro Giusti. Kinect-based people detection and tracking from small-footprint ground robots. In *International Conference on Intelligent Robots and Systems (IROS)*. 2014.

[29] David Hall and James Llinas. *Multisensor data fusion*. CRC press, 2001.

[30] David J Hand, Heikki Mannila, and Padhraic Smyth. *Principles of data mining*. MIT press, 2001.

[31] Sachithra Hemachandra, Thomas Kollar, Nicholas Roy, and Seth Teller. Following and interpreting narrated guided tours. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[32] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*. Springer, 2013.

[33] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[34] Dahlia Kairy, Paula W Rushton, Philippe Archambault, Evelina Pituch, Caryne Torkia, Anas El Fathi, Paula Stone, François Routhier, Robert Forget, Louise Demers, Joelle Pineau, and Richard Gourdeau. Exploring powered wheelchair users and their caregivers' perspectives on potential intelligent power wheelchair

use: A qualitative study. *International Journal of Environmental Research and Public Health*, pages 1–7, 2014.

[35] Rudolph E Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 8:35–45, 1960.

[36] Bernardin Keni and Stiefelhagen Rainer. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, pages 1–10, 2008.

[37] Marin Kobilarov, Gaurav Sukhatme, Jeff Hyams, and Parag Batavia. People tracking and following with mobile robot using an omnidirectional camera and a laser. In *International Conference on Robotics and Automation (ICRA)*, 2006.

[38] Stefan Kohlbrecher, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. A flexible and scalable slam system with full 3d motion estimation. In *International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2011.

[39] Pavlina Konstantinova, Alexander Udvarev, and Tzvetan Semerdjiev. A study of a target tracking algorithm using global nearest neighbor approach. In *International Conference on Computer Systems and Technologies*, 2003.

[40] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[41] Anouk Lamontagne, Joyce Fung, Bradford McFadyen, Jocelyn Faubert, and Caroline Paquette. Stroke affects locomotor steering responses to changing optic flow directions. *Neurorehabilitation and Neural Repair*, 24(5):457–468, 2010.

[42] Angus Leigh and Joelle Pineau. Laser-based person tracking for clinical locomotion analysis. *IROS Workshop on Rehabilitation and Assistive Robotics*, 2014.

[43] Angus Leigh, Joelle Pineau, Nicolas Olmedo, and Hong Zhang. Person tracking and following with 2d laser scanners. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[44] David V Lu and William D Smart. Towards more efficient navigation for robots and humans. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[45] Matthias Luber and Kai O Arras. Multi-hypothesis social grouping and tracking for mobile robots. In *Robotics: Science and Systems*, 2013.

[46] Matthias Luber, Gian Diego Tipaldi, and Kai O Arras. Place-dependent people tracking. *International Journal of Robotics Research (IJRR)*, 30(3):280–293, 2011.

[47] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[48] Anton Milan, Konrad Schindler, and Stefan Roth. Challenges of ground truth evaluation of multi-target tracking. In *International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013.

[49] Michael Montemerlo, Sebastian Thrun, and William Whittaker. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *International Conference on Robotics and Automation (ICRA)*, 2002.

[50] Peter Morton, Bertrand Douillard, and James Underwood. Multi-sensor identity tracking with event graphs. In *International Conference on Robotics and Automation (ICRA)*, 2013.

[51] Matteo Munaro. *Robust perception of humans for mobile robots rgb-depth algorithms for people tracking, re-identification and action recognition.* PhD thesis, University of Padova, 2014.

[52] Matteo Munaro and Emanuele Menegatti. Fast rgb-d people tracking for service robots. *Autonomous Robots*, 37(3):227–242, 2014.

[53] Luis E Navarro-Serment, Christoph Mertz, Nicolas Vandapel, and Martial Hebert. Ladar-based pedestrian detection and tracking. In *IEEE Workshop on Human Detection from Mobile Platforms*, 2008.

[54] Nicolas A Olmedo, Hong Zhang, and Michael Lipsett. Mobile robot system architecture for people tracking and following applications. In *International Conference on Robotics and Biomimetics (ROBIO)*, 2014.

[55] Caroline Pantofaru, Sachin Chitta, Brian Gerkey, Radu Rusu, William D Smart, and Richard Vaughan. Special issue on open source software-supported robotics research. *Autonomous Robots*, 34(3):129–131, 2013.

[56] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[57] Joelle Pineau and Amin Atrash. Smartwheeler: a robotic wheelchair test-bed for investigating new models of human-robot interaction. In *AAAI Spring Symposium: Multidisciplinary Collaboration for Socially Assistive Robotics*, 2007.

[58] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[59] J Ross Quinlan. *C4.5: programs for machine learning*. Elsevier, 2014.

[60] Donald B Reid. An algorithm for tracking multiple targets. *Transactions on Automatic Control*, 24(6):843–854, 1979.

[61] Mikel Rodriguez, Ivan Laptev, Josef Sivic, and Jean-Yves Audibert. Density-aware person detection and tracking in crowds. In *International Conference on Computer Vision (ICCV)*, 2011.

[62] Dirk Schulz, Wolfram Burgard, Dieter Fox, and Armin B Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *International Journal of Robotics Research*, 2003.

[63] Pierre Sermanet, Koray Kavukcuoglu, Sandhya Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[64] Kevin Smith, Daniel Gatica-Perez, Jean-Marc Odobez, and Sileye Ba. Evaluating multi-object tracking. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2005.

[65] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Addison-Wesley, 2006.

[66] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, 2003.

[67] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B Cremers, Frank Dellaert, Dieter Fox, Dirk Hähnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. Minerva: A second-generation museum tour-guide robot. In *International Conference on Robotics and Automation (ICRA)*, 1999.

[68] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[69] Elin Anna Topp and Henrik I Christensen. Tracking for following and passing persons. In *International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[70] Bo Wu and Ram Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[71] John G Ziegler and Nathaniel B Nichols. Optimum settings for automatic controllers. *Transactions of the American Society of Mechanical Engineers (ASME)*, 64(11):759–768, 1942.