Pedestrian Detection and Tracking for Mobility On Demand

by

Andrés Michael Levering Hasfura

B. S. in Electrical Engineering and Computer Science
MIT (2014)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2016

Author .....
Department of Electrical Engineering and Computer Science
January 29, 2016

Certified by .....
Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Pedestrian Detection and Tracking for Mobility On Demand

by

Andrés Michael Levering Hasfura

Submitted to the Department of Electrical Engineering and Computer Science
on January 29, 2016, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This paper presents a pedestrian detection and tracking system to be used aboard mobility on demand systems. Mobility on demand is a transportation paradigm in which a fleet of vehicles is shared among a community, with rides provided upon request. The proposed system is capable of robustly gathering pedestrian paths in space using 2D LiDAR and monocular cameras mounted onboard a moving vehicle. These gathered pedestrian paths can later be used to infer network traffic to learn to anticipate the location of ride requests throughout a day. This allows mobility on demand systems to more efficiently utilize resources, saving money and time while providing a more favorable experience for customers. The onboard LiDAR is used to cluster and track objects through space using the Dynamic Means algorithm. Pedestian detection is performed on images from the mounted cameras by extracting a combination of histogram of oriented gradients and LUV color channel features which are then classified by a set of learned decision trees. Temporal information is leveraged to acheive higher detection quality by accruing classification votes. Both a standard fusion technqiue and a novel extrinsic calibraion error-resistant fusion method are tested to fuse camera and LiDAR information for pedestrian path collection. The novel error-resistant fusion system is shown to outperform standard fusion techniques under both normal conditions and when synthetic extrinsic calibration noise is added. System robustness and quality is demonstrated by experiments carried out in real world environments, including the target environment, a university campus.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

First, I would like to thank my advisor, Professor Jonathan How. He is a fantastic mentor and thinker, and helped provide the motivation and direction necessary to accomplish this work.

I also want to thank my labmates at the ACL. I learned a great deal from interacting with you all, for that I cannot thank you enough. Shih-Yuan for providing guidance and support, patiently helping whenever needed, Justin for working with me closely when our paths overlapped and helping me think through issues when they didn't. Thank you Mike, Kris, Steven, Brett, Mark, Byron, and Jack for providing moral support.

Finally, I would like to thank the people closest to me. My parents, Roberto and Teresa, my brother, Alejandro, and my ever-supportive girlfriend, Katy. Without you all nothing would be possible.

This research is supported by grant from the Ford Motor Company through the Ford-MIT Alliance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Pedestrian detection and tracking is of paramount importance in many modern applications, including automation of vehicles, surveillance, and more. Pedestrian path data can be used anywhere from short term path prediction for collision avoidance, to long term understanding of network pedestrian traffic to utilize city resources most effectively. The application of focus in this work is mobility on demand systems for locations with heavy foot traffic, such as a university campus.

Mobility on demand (MOD) is a transportation paradigm in which a fleet of vehicles is shared among a community, with rides provided upon request. A key factor for the success of a MOD system is its ability to optimally distribute vehicles in response to customer demand. In current mobility on demand systems, a user can request a ride from a fleet of vehicles that randomly patrol some predefined network. These requests are filled but can potentially be slow, in part due to the lack of understanding of likely request locations.

Our vision to alleviate this problem is to study pedestrian flow through a given network to produce a model which can accurately predict the most common locations for a ride request at a given time. To do this, pedestrian path data needs to be recorded and analyzed. One such solution to gather pedestrian paths would be the installation of sensors throughout a network, but this does not scale and raises privacy issues. Another solution would be to have some number of sensors moving through the environment, recording flows in different locations and different times. As autonomous

vehicles approach reality, this solution becomes more tractable. A mobility on demand system can utilize the sensor suite already installed aboard an autonomous vehicle to capture pedestrian paths, and this data can be used to model traffic flow, and finally preempt ride requests to maximize resource efficiency.

Real-time collection of pedestrian traffic data in cluttered campus-like environments on a mobile platform poses a significant challenge. Simple single sensor solutions are currently not possible for tracking pedestrians in space. Camera data cannot provide depth, 2D LiDAR cannot reliably classify, and 3D LiDAR produces too much data to handle in real-time. Therefore, multisensor approaches must be used which require synchronization and coordination across different domains. Sensors are affected by weather conditions, changes in illumination, movement or terrain irregularities, surrounding buildings, among many other factors. Pedestrian walkways are dense and chaotic as few laws govern the movement of pedestrians and travel in tight, occluded groups is commonplace. Furthermore, the need for real-time processing onboard a mobile sysem makes this task evermore challenging. A fast and accurate system is required.

This work presents a robust solution to pedestrian detection and tracking using sensors likely found onboard autonomous vehicles, 2D LiDAR and cameras. The system uses LiDAR to cluster and track objects through space using the Dynamic Means algorithm [1], which models cluster evolution to provide temporal association. Images from the mounted cameras are used to classify pedestrians using the VeryFast detector [2]. The output of these two modules are fused using both a standard technique and a novel extrinsic calibration error-resistant fusion scheme. Temporal information is leveraged to acheive higher detection quality by accruing classification votes.

The three main contributions from this work are as follows. This work provides:

1. A robust but inexpensive solution to pedestrian detection and tracking in the difficult cluttered environment of interest, capturing over 90% of pedestrians with only an average of 1.5 false detections per minute using commodity sensors and one commodity computer.

2. A fast solution capable of processing 90 images per second and 50 LiDAR revolutions with one GPU/CPU.

3. A novel extrinsic calibration error resistant fusion scheme which outperforms traditional traditional fusion schemes.

Related work is discussed in section 2, an overview of the methodology is described in section 3, sections 4, 5, and 6 discuss each module in more depth, experiments and results are discussed in sections 7 and 8 respectively, and a conclusion is provided in section 9.

Chapter 2

Related Work

Pedestrian detection and tracking using camera and LiDAR combines many submodules to form one, cohesive system which outperforms any of the submodules individually. This section will discuss important work from each of the relevant submodules, the algorithms involved in the submodules, then the main contributions to pedestrian tracking with camera and LiDAR.

2.1 Related Vision Work

The largest and most explored submodule is pedestrian detection using only camera. Over time many different methodologies have been explored, including sliding window approaches, image resizing approaches, feature rescaling approaches, and more.

Successful pedestrian detection started with sliding window, image rescaling search with haar-like features and an SVM classifier [3]. This was the first instance in which semi-accurate pedestrian detection was acheived.

Shortly after, Viola-Jones developed a detector which did not require resizing of the image, and therefore could very quickly process images. It too used a sliding window technique and haar-like features to detect pedestrians [4], but suffered in quality.

In 2005, Dalal and Triggs pioneered a new feature set that performed better than any previous attempts, dubbed histogram of oriented gradient (HOG) features [5].

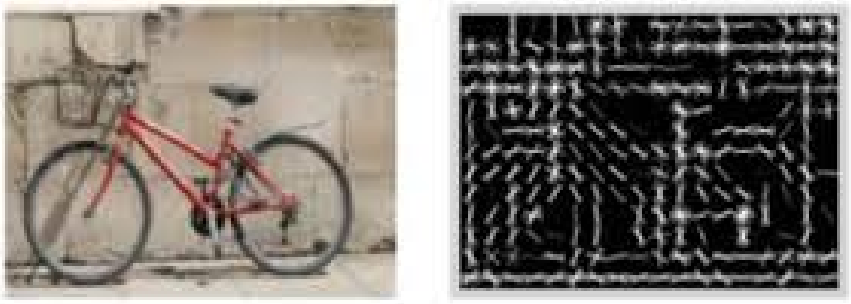


Figure 2-1: Visualization of HOG feature responses on an image of a bicycle

These features responded to strong gradients in the directions of interest, as seen in figure 2-1. SVMs were used to do the classification after the feature transformation.

Many derivatives of Dalal and Triggs' work followed, but the biggest performance boost came from training deformable part based models. Part based models train different detectors for different parts of an object, then produce an overall score based on the parts' scores and the expected distance between each part [6]. Despite the performance boost, part based models still operated quite slowly.

Boosts in speed came from discovering that it may not be necessary to resize an image to every scale, but instead rescaling fewer times and interpolating between image sizes [7], or conversely, by keeping the same image size but training multiple classifiers for different sized pedestrians and interpolating between the classifiers themselves [2].

Most recently research in pedestrian detection is focused on deep learning. This occured due to the recent increase in the number of hand labeled training images and GPU/CPU improvements. These neural nets are outperforming other classical methods in quality. There are many papers attempting to provide solutions to pedestrian detection using recurrent neural nets [8, 9] and convolutional neural nets [10]. In this work neural nets were not used for pedestrian detection because of speed concerns, although as more effort is directed toward decreasing computational complexity, they could soon be a very strong choice for similar work.

For a more in-depth literature review of the pedestrian detection using images is found here [11].

2.2 Related LiDAR Work

To make sense of the range data, a common first step is to segment the points into clusters in order to try to represent solid body objects in space. There exist many techniques to cluster range data, but only 2D point cloud clustering is discussed, as it is most relevant to this work.

Euclidean distance based methods segment objects if the difference in distance between two sequential points exceeds a threshold [12]. This method is very powerful for applications which use 2D LiDAR because range points are provided in order, so performing euclidean clustering can be as simple as taking the difference between two adjacent points in the list and checking if it is greater than some threshold. Drawbacks of euclidean clustering in this application are susceptible to noise and lack of inherent temporal association.

Region growing methods randomly assign seed points which serve as the center of clusters. Neighbors are then added to the cluster if the difference in distance does not exceed a threshold [13]. In this application region growing is not as useful as it requires more time than euclidean clustering, while not compensating with any inherent temporal association or noticeably greater clustering quality.

Some recent methods model temporal and spatial evolution of clusters. This can be very powerful for temporally consistent datasets, such as in our data. Dynamic Means is one such algorithm [1]. A potential drawback of using Dynamic Means is that it favors clusters in which points are near the centroid, as opposed to segmenting at boundaries based on distance difference thresholds. This can be a problem if one is trying to cluster long objects such as walls and cars, but turns out to be accurate when only trying to cluster pedestrians due to the natural circular shape. Due to the preference for pedestrian shaped clusters and baked-in temporal association while providing high quality, Dynamic Means was used to power the clustering and temporal association in this work. See [14, 15, 16] For a more in depth review of clustering methods.

In order to gather pedestrian paths, there is also a need to be able to track these

clusters in space. There are a number of different ways in which to do so. One of the most common methods is Kalman filtering [17], which is an estimation method that takes both a cluster's dynamics and sensor measurements to estimate the location of the cluster. There are extensions and derivatives of Kalman filtering, such as extended Kalman filtering (EKF) [18] and unscented Kalman filter [19], which more accurately model non-linear systems.

More recently work has been introduced which utilizes more sensors to provide more accurate tracking. One such example is this work which combines shape, color, and motion of an object to perform tracking using a combination of LiDAR and camera [20]. See [21] for a more thorough coverage of tracking techniques.

Although not nearly as extensive as image based pedestrian detection, There has also been some work to detect pedestrians in LiDAR space.

Detection systems in point cloud space are very similar due to the relatively small amount of information a single slice of range data contains. The largest difference in detection methods lies in the feature sets proposed. When using 2D LiDAR, features vectors are usually composed of geometric attributes of the cluster, such as total number of points, variance, and skew, and motion features, such as speed and acceleration [22, 23].

This paper [24] performs an in depth study on whether or not it is possible to do pedestrian detection in 2D LiDAR space reliably enough to not need supplementary help from another sensor. They extract different feature sets and use five different learning methods, Naive Bayes, gaussian mixture module classifier, minimization of inter-class inference for neural networks, FLDA, and SVMs to try to separate objects. In the end, this work concludes that 2D LiDAR does not provide sufficient information to accuratly classify clusters in space alone.

3D LiDAR provides more information about objects in space, but similar to 2D LiDAR, not much innovation is done outside of developing novel feature vectors composed of motion and geometry [25, 26]. Currently no one has tried to use nerual nets to learn 3D LiDAR pedestrian features, most likely because there is no large source of point cloud pedestrian samples to train a deep net.

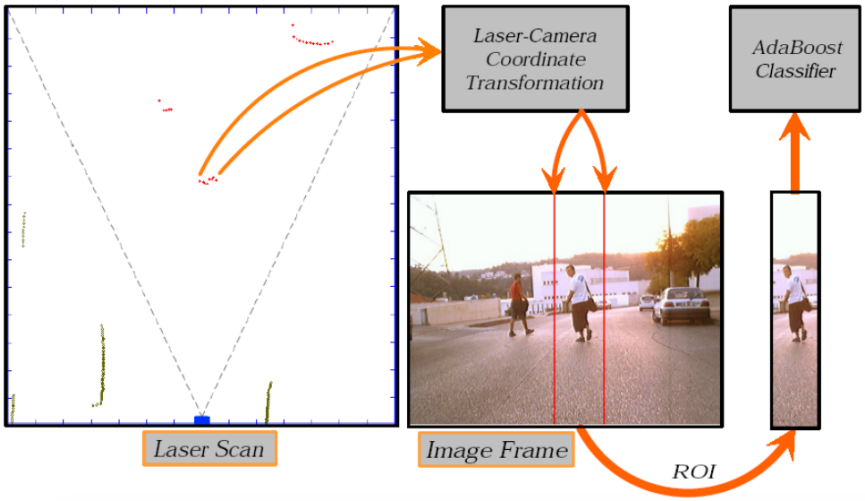


Figure 2-2: Using the assumption that a pedestrian cannot exist without a corresponding cluster in space one can cut down search space using cluster information. Region of interest within and image is found by projecting clusters from space onto the image, as shown in the figures. This figure is copied from [27]

2.3 Related Multisensor Work

When trying to get the position of an object in the world, a single monocular camera is not enough, and pedestrian detection using a single LiDAR is either too slow or not good enough. In this work, the combination of camera and LiDAR are used to compliment each other's shortcomings. There are many other previous systems that use this same sensor combination.

Some of the first work to attempt this is [27] which used 2D LiDAR and monocular cameras to track pedestrians. In this work the authors use AdaBoost for the pedestrian detection in images, a GMM classifier in pointcloud space, and use a Bayesian-sum decision rule to take a weighted sum of votes of the two detectors. Additionally, they use the clusters in space to define areas in an image in which pedestrians may exist, so as to cut down the search space in the image to speed up the overall system. They called this technique finding the image's region of interest (ROI), depicted in figure 2-2. This provided a speed boost which allowed the system to run in real-time back in 2007.

Other systems have built on the core ideas of this paper by swapping out the submodules with more modern alternatives. These two papers use HOG+SVM to do

the classification in the image space, and uses a unique hand crafted feature set to do classification in 2D LiDAR space, again using ROI techniques to cut down search space. They fuse the outputs of the two modules using a few different techniques, and make conclusions about the most accurate for the application [23, 28].

[29] combines ideas similar to the other papers, but uses radar to increase effective field of view, and uses motion models to more accurately track objects. Additionally, this work aims to detect and track more than just pedestrians, but also bikes and cars. For this they train three part based models, one for each object of interest, and also use the motion models for object detection in the LiDAR space to get a better classification rate. Altogether, The use of radar and motion models allows the system to track cars up to 200m, and pedestrians and bicycles for up to 20 meters.

Chapter 3

System Overview

This section provides a description of the vehicle used for the system, the target environment, and a methodology overview for the process by which pedestrian paths are extracted from raw sensor data.

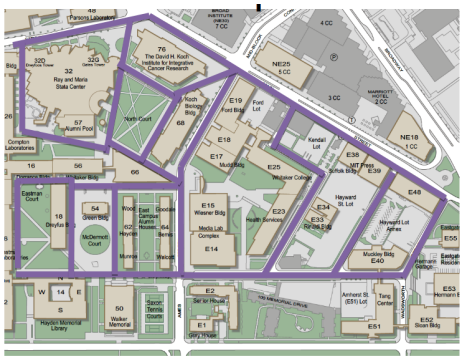
The proposed system was constructed on a Polaris GEM vehicle to allow for entrance within campus confines. The GEM is pictured in figure 3-2. Three commodity web cameras are mounted to cover 270 degrees around the front and sides of the vehicle, and two LiDARs are mounted, one on the roof and one on the hood, with the same angle of view. This is replicated on three identical GEMs to provide the mobility on demand fleet, as seen in figure 7-4. An closer image of a GEM with mounted sensors is seen in figure 3-1.

The MIT campus is chosen as a test ground to pioneer the mobility on demand system because of the periodic traffic patterns and plentiful source of pedestrians. The proposed operation environment is pictured in figure 3-3a, and the area in which experiments are conducted is pictured in figure 3-3b. These areas are heavily used by pedestrians commuting to and from classes.

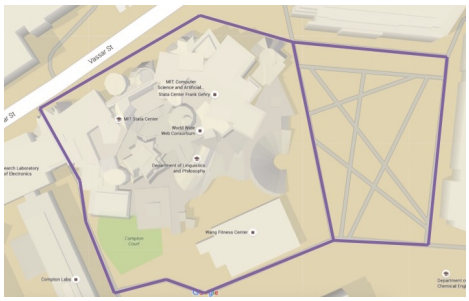
Before attempting to collect pedestrian paths, a map of the experimental area must be made in order to provide the vehicle a global frame while navigating. To generate the map, data from the roof-mounted LiDAR to avoid most moving objects (pedestrians, bicyclists, vehicles). The LiDAR points are fed into an odometry package, the output of which is in turn fed to the a gmapping package which correcting

Figure 3-1: Image of Polaris GEM with mounted sensors

Figure 3-2: Image of Polaris GEM, the type of vehicle used in this mobility on demand system.



(a) Depicts the overall area of operation for the final mobility on demand system.



(b) Depicts the area in which experiments were conducted.

Figure 3-3: Operating environment

Figure 3-4: Occupancy grid produced from map making process. This is the functional area in which all experiments are performed.

drift and produces an occupancy grid map. Upon return the system can then localize to the prebuilt map using particle filter techniques to obtain a global position. Shown in figure 3-4 is a visualization of the occupancy grid produced by the map creation process.

Given a global location, the remainder of the pipeline is focused on detecting and tracking pedestrians in the local frame. There are three main submodules involved in the tracking of pedestrains in the local frame, the LiDAR module, the vision module, and the fusion module.

The LiDAR module receives sequential distance measurments from the hood-

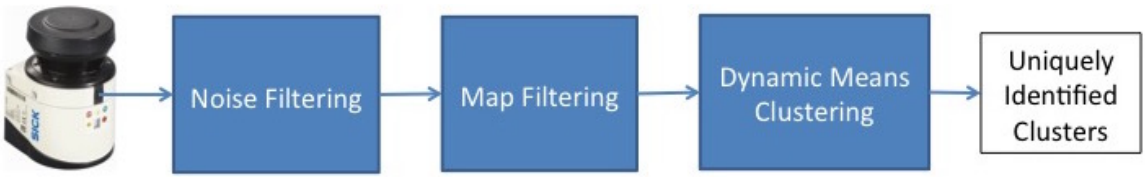


Figure 3-5: Schematic of LiDAR module

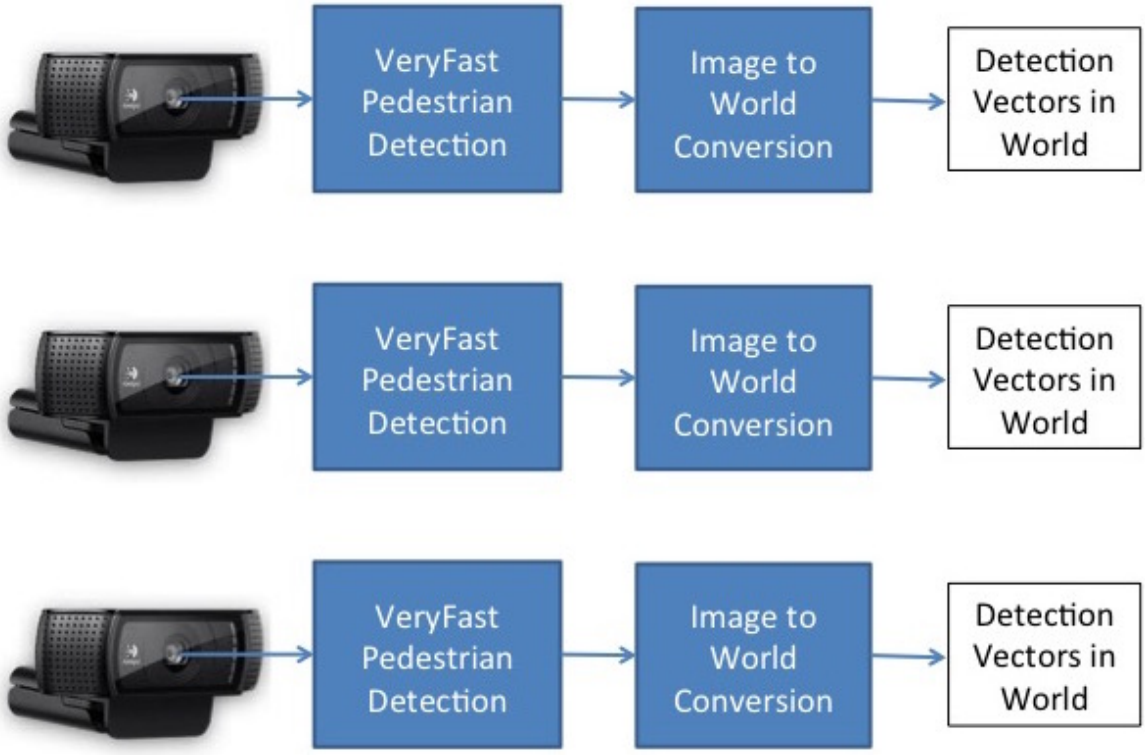


Figure 3-6: Schematic of vision module

mounted LiDAR and produces temporally-evolving uniquely-identified clusters. There are three major elements in the LiDAR pipeline. The first step removes any points deemed to be noise. The second step removes any remaining points which belong to the pre-made map based on the assumption that the map contains no pedestrians. Finally, the remaining points are clustered using Dynamic Means [1], a fast, general purpose clustering algorithm which also models temporal and spacial evolution of clusters. More in depth discussion of each of the stages and algorithms involved are presented in section 4, and a depiction of the pipeline can be seen in figure 3-5.

The vision module receives images from the three cameras mounted on the GEM and produces vectors into the world corresponding to pedestrians detected in each

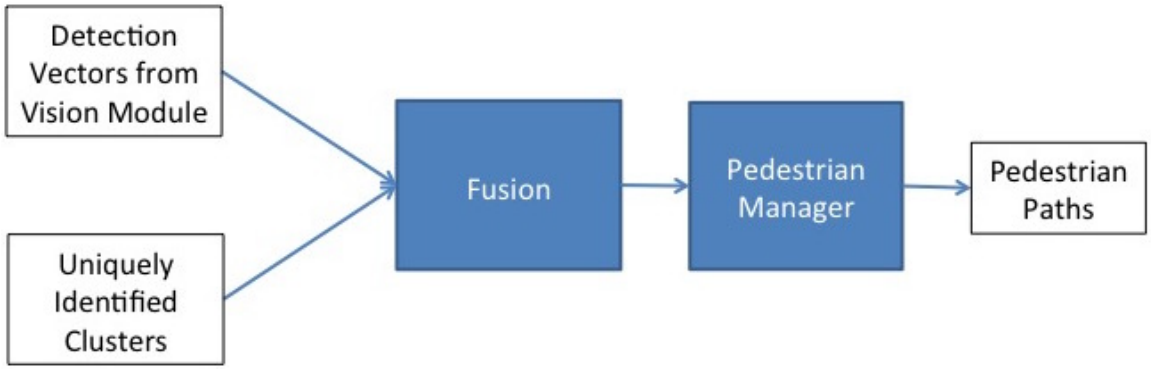


Figure 3-7: Schematic of fusion module

image. Images received from a camera are input to the VeryFast pedestrian detector [2] which produces bounding boxes indicating locations of pedestrians in the image. By modeling each camera as a pinhole camera, each bounding box in an image can be converted into three vectors pointing from a camera into the world corresponding to the left, middle, and right edge of the pedestrian's torso. More in depth discussion of each of the stages, models, and algorithms involved are presented in section 5. The vision pipeline is pictured in figure 3-6.

The fusion module receives the output of the vision and LiDAR modules (vectors corresponding to pedestrian detections and uniquely identified clusters) and produces the desired system output, pedestrian paths. There are two core steps in the fusion module: (1) to properly associate a pedestrian detection with a corresponding cluster and (2) to record cluster paths and the associated accrued hit counts. Once a cluster has been associated with pedestrian vectors enough times, the cluster is labeled a pedestrian and its path is recorded into a database for future analysis.

For the pedestrian vector to cluster association, two different methods are tested, the traditional maximum likelihood fusion and a novel probabilistic fusion. Maximum likelihood fusion provides a hit to the cluster with minimum distance to pedestrian vectors. Probabilistic fusion uses the same distance metric but instead of providing a full hit to the closest cluster, provides partial hits to clusters with magnitude inversely related to the distance from the pedestrian vectors. This can be powerful when extrinsic calibration is not perfect, as seen in section 8.

The second core submodule manages clusters paths and accrued hits. This module records the paths of all clusters, and writes a cluster's path to a database only if the cluster's hit count exceeds a hit count threshold. The output of this module can then be analyzed for applications such as mobility on demand. More in depth discussion of each of the stages and comparison between two fusion types are presented in section 6, and a depiction of the fusion pipeline is pictured in figure 3-7.

A visual summary of the overall system is shown in figure 3-9, and a visualization of the system running in the target environment is provided in figure 3-8

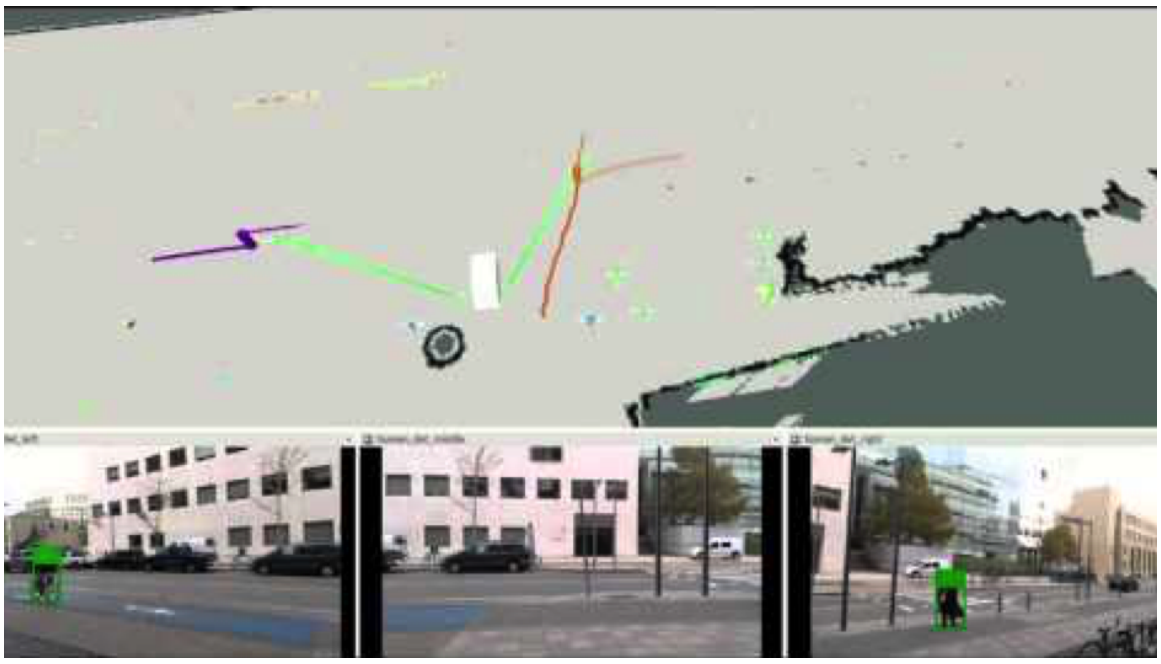


Figure 3-8: Snapshot of the system visualization. The white box represents the vehicle, green arrows represent directions of pedestrian detections, cylinders represent detected pedestrians with paths trailing, images from the three cameras on bottom of screen with pedestrians outlined in green (faces covered for privacy reasons).

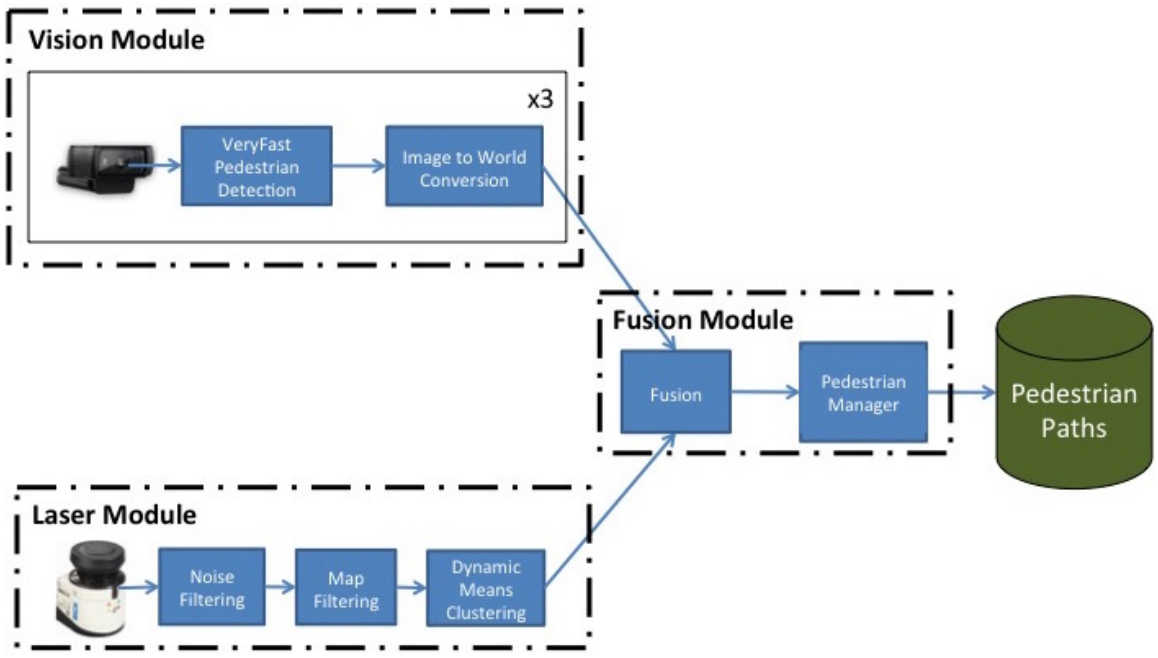


Figure 3-9: Schematic of overall system architecture

Chapter 4

LiDAR Module

The LiDAR module takes as input a list of sequential range points in a 270° sweep from the hood-mounted LiDAR and produces a set of uniquely-identified temporally-consistent clusters. This process requires multiple steps, first to remove unecessary points, then to perform the clustering and temporal association using the Dynamic Means algorithm. Figure 3-5 visualizes the LiDAR module pipeline.

4.1 Preprocessing

Initially the data received by a 2D LiDAR contains noise and other data of no use to detecting and tracking pedestrians. Raw data points go through two preprocessing steps before finally being used for clustering. First is noise removal, second is map filtering. These are discussed in the following subsections.

4.1.1 Noise Removal

LiDAR measurements will always contain noise. One of the more frequent sources of LiDAR noise are "ghost points" [30], which are erroneous measurements that occur at the boundary of two objects. Other sources of measurment error exist, such as interference from the upper LiDAR [31], and weather conditions such as rain, puddles, bright sun, white snow, and more. False measurements can complicate computation

further down the pipeline and cause error in future calculations. To remove noise from the raw data a statistical analysis is performed on each point's neighborhood, and any point that does not meet certain criteria is removed.

This is done by first finding the approximate density of the raw data by computing each point's distance to neighbors. Assuming this produces a gaussian distribution, points are removed from the raw data if the average distance to its neighbors is a certain number of standard deviations away from the mean of the distribution. The remaining points are then passed to the next stage in the LiDAR module.

4.1.2 Map Filtering

The output of the noise removal step is then input to the map filtering stage. The map filtering stage attempts to remove all points from the LiDAR range data that correspond to static objects that are represented in the pre-built map. This step is very crucial, as it removes a majority of the remaining points, allowing for the clustering step to run real-time.

Checking if a point belongs to the map is relatively easy, if a point from the input cloud is inside a voxel centered at any point on the map, the point is removed. Once noise and points that correspond to the map are removed from the original data, the remaining points are ready for clustering.

4.2 Clustering and Temporal Association

Once noise and points corresponding to static objects in the map have been removed, it is computationally tractable to perform clustering and temporal association of clusters. In this work, the Dynamic Means algorithm [1] is used. The Dynamic Means algorithm is based on the dependant Dirichlet process mixture model (DDPMM), which is itself a combination of a Dirichlet process mixture model (DPMM) which enables the inference of an unbounded number of mixture models and the dependant Dirichlet process (DDP) which provides a prior on the mixture model. The combination of DPMM and DDP provides the power of accurately modeling an unknown

Table 4.1: Performance of Dynamic Means algorithm compared to other bayesian non-parametric hard clustering algorithms on the automatic dependant surveillance-broadcast dataset. This table was adapted from [1].

Algorithm	% Accuracy	Time (s)
Dynamic Means	55.9	$2.7 \cdot 10^2$
DP-Means	55.6	$3.1 \cdot 10^3$
DDPGMM Gibbs Sampling	36.9	$1.4 \cdot 10^4$

number of mixtures provided by DPMM while capturing the temporal and spacial evolution and relationship provided by DDP.

Still, inference on DDPMM using Gibbs sampling is too slow for a real-time applicaiton. Dynamic Means alleviates this issue by deriving data from the low-variance asymptotic limit of the Gibbs sampling algorithm for DDPMM. This allows for the modeling power of of the DDPMM while maintaining the scalability of simpler classical hard clustering algorithms. Table 4.1 shows the performance of Dynamic Means along with other relevant bayesian non-parametric clustering algorithms on Automatic dependant surveillance-broadcast data. For a more thorough explanation of Dynamic Means see [1].

Dynamic Means outputs temporally-consistenent uniquely-identified clusters. This allows the tracking of objects in space by simply recording the locations of each cluster at every timestep.

Chapter 5

Vision Module

The vision module takes as input images from each of the three mounted cameras and produces vectors pointing from a camera into the world corresponding to the pedestrians detected in the input images. The first section discusses the pedestrian detection algorithm in the image frame, the following section discusses the camera model chosen by which vectors into the world corresponding to pedestrian detections in an image are produced.

5.1 Pedestrian Detection

The cameras are configured to produce 30 frames a second each, so the vision module needs to be able to handle 90 frames a second comfortably. This is possible due to the VeryFast pedestrian detector [2]. This section will provide a brief overview of the VeryFast pedestrian detector, but for more in depth understanding see [2].

5.1.1 Features

VeryFast uses a feature set that is exactly the same as the ChnFtrs detector [32] and FPDW detection [7].

VeryFast extracts ten features from an input image in rectangular sliding windows of constant size. These features were chosen due to the extensive work by Piotr Dol-

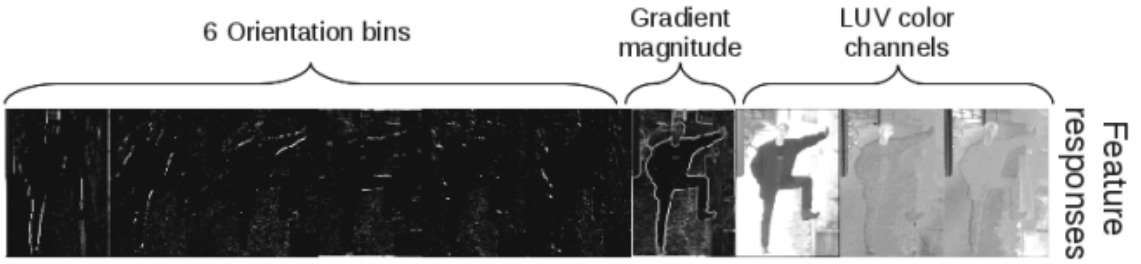


Figure 5-1: Visualization of the ten features collected by VeryFast. Left six are orientation bins, seventh is gradient magnitude, right three are LUV color channels. This figure was taken from [2]

lar [33] to find the best possible combination of features for pedestriand detection in images. The first six are the image responses at six different orientations of gradients, the seventh is image gradient magnitude with no restriction on orientation, the remaining three are the LUV color responses of the image. This feature extraction is pictured in figure 5-1.

5.1.2 Classifier

There are many different choices of possible classifiers for the resulting feature vector. In general, non-linear classifiers such as the RBF SVM perform better than linear classifiers, but at the cost of speed. In mobile robotic applications, speed is of paramount importance so VeryFast uses a linear classifier. A set of two-level decision trees are used and discrete Adaboost to learn the set of decision trees and the overall weight each tree has on the final vote. A strong classifier is produced after the decision trees have been properly weighted. For more information on discrete Adaboost, consult [34], for information on decision trees and random forests, consult [35].

5.1.3 Speed Boost

The key insight that separates the VeryFast pedestrian detector from similar detectors [7, 32] is VeryFast does not resize an image. In other work, it is common to train a classifier for a pedestrian of a certain size then resize the image itself and extract the feature response at each size. Resizing an image is very computationally expensive

due to size, so many pedestrian detectors run at speeds not suitable for real-time applications.

An alternative to resizing the image is to train many classifiers, one for each possible size of a pedestrian in an image. Although this would speed up computation during run time, this turns out to be a difficult task. First, it is difficult to find a reposity of enough training examples of pedestrians at each different size to be able to effectively train all the required classifiers. Additionally at the smaller size blur effects make properly training classifiers difficult.

Ideally, a detector could find the feature response of an image at two different sizes, then interpolate the responses at different scales. This way the image only has to be rescaled once, or only two classifiers need to be trained, and the rest can be interpolated. Unfortunately, due to discritization of pixels, scene information is lost and it is not possible to do this interpolation accurately using only two rescalings. The key insight of VeryFast is to try to train K different evenly spaced classifiers for possible sizes of pedestrian. Then there can be a reasonable approximation between image sizes, and there is no noticeable degredation in detection quality. If K remains relatively small, the loss of information due to discretization is negligable and detection remains high quality.

This provides a significant speed boost to the pedestrian detection and enables VeryFast to fit in mobile applications in which real-time speed is necessary.

5.1.4 Evaluation

VeryFast is a single part model which performs on par with parts based models, and outperforms all classical methods such as HOG+SVM and its derivatives. It slightly underperforms with respect to newer deep learning methods, but has the advantage of being much faster. A ROC curve showing performance on the INRIA dataset is shown in figure 5-2. On a single CPU+GPU, VeryFast is able to run faster than 90Hz on 640x480 images, which is what is required for this work. This speed is important for a system such as ours which leverages temporal information (accruing hit counts while a pedestrian is in the field of view).

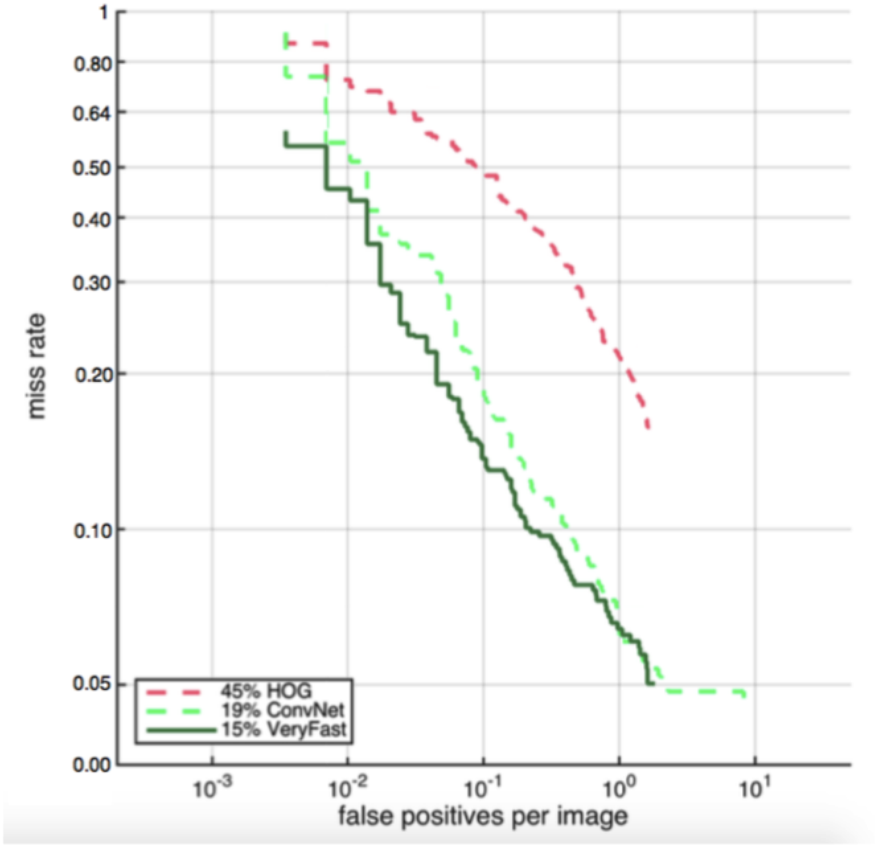


Figure 5-2: ROC curve for INRIA dataset. VeryFast outperforms HOG+SVM by a great margin and is competitive with deep learning methods.

5.2 Image to World Conversion

The output of the vision module is a set of three vectors per pedestrian detection. In order to produce these vectors, a relationship between image and world must be established.

In this work, the cameras are modeled as pinhole cameras. As seen in figure 5-3, the camera is approximated by a box allowing light in at a single point, and capturing the image on a plane at a distance equal to the camera's focal length. The camera's intrinsic values, including focal length, are found by measuring distortion and size of a known object in the world at a given distance. Using this pinhole model and intrinsic matrix, vectors can then be produced originating at the focal point and terminating at any pixel in the image. This represents the direction in the world of the object

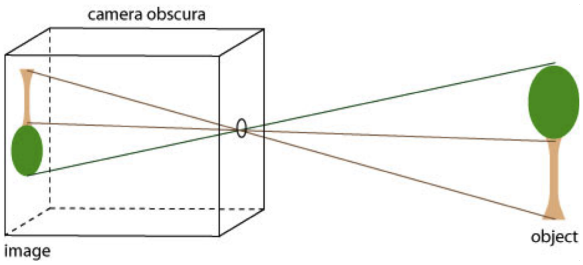


Figure 5-3: Pinhole model used to relate images to space. Using this model, a pedestrian detection can be converted into a vector pointing in space.

which produced the corresponding pixel. To produce the desired set of vectors, the system creates a vector from the focal point of the camera to the pixel at the left, right, and center of a pedestrians torso. These vectors are later used to determine which cluster in the world a detection in the camera frame is referring to.

Chapter 6

Sensor Fusion Module

The goal of the sensor fusion module is to take the world vectors produced by the vision module and associate them with a cluster produced by the LiDAR module. Once a cluster has been identified as a pedestrian enough times, the sensor fusion module records the cluster's path into a database for future analysis. This process is broken into two submodules, one which determines which cluster a pedestrian detection is referring to, and the other which keeps record of accrued hits and cluster paths.

This work explores two different methods of fusion, maximum likelihood fusion and a novel probabilistic fusion. These fusion method's differences and benefits are discussed later in this chapter.

6.1 Pedestrian Detection to Cluster Association

This submodule takes the vectors produced by the vision module, which are in the camera frame, and tries to correctly associate the detection with the corresponding cluster in space. When a hit between a detection and a cluster is found, this submodule will send a message to the pedestrian manager submodule to increase the hit count of the cluster.

6.1.1 Extrinsic Calibration

To be able to associate pedestrian detection vectors originating at a camera with a cluster in space, a spacial relationship between camera and LiDAR must be developed. This is called the extrinsic relationship.

Extrininsic relationships are acquired by careful measuring pose differences between sensors. This is error prone, and error in these measurements can produce harmful results downstream, especially when operating in cluttered environments such as a university campus. Therefore, in this work a fusion method is developed and proposed that is resistant to extrinsic imperfections, noise, and drift in sensor calibrations.

6.1.2 Fusion Technique

Presented here are two fusion methods, the traditional maximum likelihood fusion and a novel probabilistic fusion. In both techniques the notion of distance between a given pedestrian detection and a given cluster is defined by the sum of differences of angles between the pedestrian detection vectors and a cluster's edges and center. This can be seen in figure 6-1, and the equation for distance is shown below

$$d = |\theta| + |\phi| + |\rho| \tag{6.1}$$

where d is the distance between cluster and detection, θ is the angle between left detection vector and cluster left edge, ϕ is the angle between center detection vector and cluster center, and ρ is the angle between right detection vector and right cluster edge.

Maximum Likelihood Fusion

Maximum likelihood (ML) fusion provides a hit only to the cluster most near the vector produced by the pedestrian detection, where distance is measured by equation 6.1. ML fusion sends a message to the pedestrian manager module to increment the hit count of the closest cluster ID. This is a very effective fusion method if extrinsic

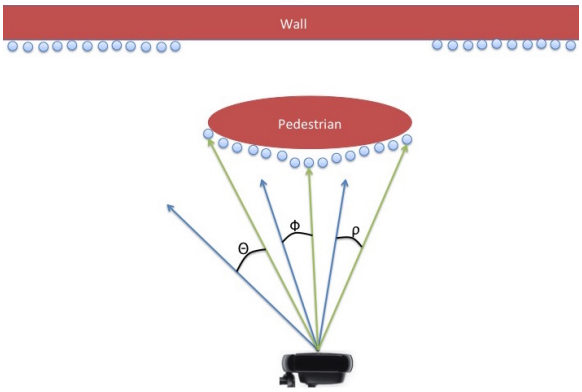


Figure 6-1: Vector association between a pedestrian detection (blue vectors) and a cluster's edges and center (green arrows). Distance is calculated as the sum of $|\phi|$, $|\rho|$, and $|\theta|$.

calibration error is negligable, but produces problems if this assumption does not hold. Consider a consistent misalignment between camera and LiDAR. This can occur for many reasons, such as poor initial extrinsic calibration, or a sensor physically drifting over time as the vehicle drives through uneven terrain. In tight, cluttered environments such as the target environment of the MOD application, as a pedestrian moves through the space, objects surrounding the pedestrian's cluster will be awarded with hits instead of the correct cluster. This issue is addressed with probabilistic fusion.

Probabilistic Fusion

In probabilistic fusion, multiple clusters receive partial hits varying inversely with distance to the detection. a cluster is awarded a hit according to

$$h = e^{-\frac{d^2}{2\sigma}} \tag{6.2}$$

where h is the hit value to be accumulated, d is the distance between a detection and cluster as defined in equation 6.1, and σ is the variance, or a measure of the rate of dropoff of the hit awarded as distance increases. Instead of one cluster receiving a full hit, clusters in the vicinity of a detection receive partial hits with exponential dropoff with respect to d, the distance. In this case if there is a constant error in calibration,

clusters surrounding a pedestrian will still receive the largest increase in hit count, but the pedestrian will always receive some portion of it. So as a pedestrian walks through the space, it will continue to accrue hits, eventually properly being labeled a pedestrian.

Additionally, in probabilistic fusion the *absence* of a significant hit on a cluster is taken into account. In this method, if a cluster is in view of the system but is not being associated with a pedestrian detection vector, the cluster's hit count is reduced. This is powerful in many scenarios. For example, if a non-pedestrian object is in view for an extended period of time and is not associated with a pedestrian detection vector, but at some point a pedestrian steps in front of the tree, this version of fusion will not detect the object as a pedestrian due to the overwhelming negative hits accumulated originally.

6.2 Pedestrian Manager

The pedestrian manager module records two things, the location of every cluster at every time step and the hits associated with cluster IDs received from the fusion submodule. Logging does not start when a cluster to be identified as a pedestrian, but instead all clusters trajectories are logged. Despite the waste of resources required to log every single cluster path, this enables the storing of a pedestrian path before the system had identified the cluster as a person. This extra distance is important for mobility on demand, because it can expose transitions and other important behavior as opposed to just a link count in the network. Periodic pruning occurs to remove inactive clusters to allow the system to run indefinitely. Once a cluster's hit count exceeds a threshold, the pedestrian manager module records the entire trajectory into a database for future analysis.

Chapter 7

Experiments

Due to a lack of established dataset to benchmark pedestrian path collection, we collected our own data, built ground truth from it, which was used to produce performance metrics.

For a mobility on demand system, the most important metric is whether or not the path of a pedestrian which entered the field of view of the proposed system is recorded. The following sections describe the platform used for data collection, data acquisition details, the production of ground truth, and finally the method by which a run is evaulated.

7.1 Platform

The platform used to collect data is a four seat Polaris GEM chosen to be the same model as campus maintanance to allow for entry into campus paths.

The GEM vehicle is equipped with two SICK LMS151 LiDARs, 270 degree sweep with 50 meter range, one mounted on the hood of the vehicle 3 feet off the ground, the other mounted 7.5 feet from the ground on the roof of the vehicle. Additionally, the vehicle is equipped with three logitech C920 HD Pro cameras, one facing front, and one facing on either side of the vehicle. One camera mount is show in figure 7-1 and the hood-mounted LiDAR is show in figure 7-2.

All processing occurs on a single Gigabyte Brix Pro, which contains an intel i7

Figure 7-1: Logitech webcam mounted on GEM vehicle

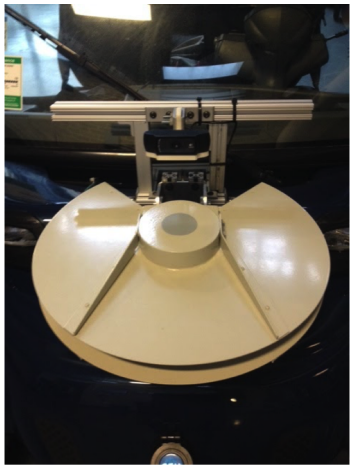


Figure 7-2: LiDAR mounted on GEM vehicle

Figure 7-3: Gigabyte Brix Pro mounted on GEM vehicle which handles all processing

Figure 7-4: Fleet of GEM vehicles to provide mobility on demand service

CPU and the Nvidia GeForce GTX 760 GPU with 16 GB of RAM, as seen in figure 7-3.

In order to properly implement a mobility on demand system, it is necessary to have multiple vehicles. Three vehicles were constructed identically, to provide a fleet. Figure 7-4 shows the fleet equipped with identical sensors.

7.2 Data Acquisition

Data was recorded by driving for approximately thirty minute segments through the target environment, the MIT campus. The GEM vehicles are localized within the pre-built map created by the roof-mounted LiDAR. The vehicle is then driven through

the area freely, encountering pedestrian walking and standing naturally. Lighting conditions change throughout the dataset, as clouds cover and uncover the sun.

No images are recorded for privacy reasons. Instead, only the *locations* of bounding boxes of each detection from the VeryFast pedestrian detector are recorded with an associated score, and thresholding can be performed further down the pipeline.

The primary dataset used for benchmarking was recorded at 3:00PM during a weekday on October 8th 2015 while MIT classes were in session.

7.3 Construction of Ground Truth

Construction of ground truth starts with a subset of the data recorded from the acquisition phase.

We opted to not hand label every frame due to the entensive manual labor required for the task. Instead, data was passed through the proposed system to produce clusters and paths associated with the clusters. In this case, every path was recorded, detected pedestrian or not. A human then watched the 3D LiDAR data associated with the data and carefully labeled which clusters and the associated paths belonged to pedestrains, and those paths were extracted as ground truth. The human watched 3D LiDAR because LiDAR provides enough information for a human to classify pedestrians, but not enough to identify the person, avoiding privacy issues.

Because the clustering and tracking algorithm is not perfect, post process path rectification was also performed. This method raises three major issues listed here:

1. Pedestrians outside of range of the LiDAR (50m) would be excluded from the ground truth

2. The clustering algorithm sometimes combines multiple pedestrians into one cluster, so only one of the pedestrians paths would be recorded

3. The clustering algorithm could assign multiple clusters to a single person over time if a pedestrian moved too fast, producing erroneous paths for the pedestrian

Each of these issues are addressed in the creation of the ground truth. The first issue is ignored as any pedestrian from ground truth which is out of the range of the system is excluded. The second issue is addressed by duplicating the combined cluster with an offset proportional to the size of the cluster in the direction perpendicular to the direction of travel. The third and final issue is addressed by noting which clusters correspond to the same person, then stitching the paths together and filling in the gap with evenly spaced points.

Once the paths of pedestrians are filtered and repaired from the original set, the ground truth is ready to be used to evaluate future runs in which pedestrian paths are automatically extracted by the system from the same input sensor data.

The resulting ground truth is 22 minutes long, with 237 unique pedestrians.

7.4 Evaluation

Evaluation compares the paths extracted by the system to the ground truth paths. In a mobility on demand system, the most important objective is to accurately extract the path of every pedestrian that enters the field of view. Other metrics which could be of interest, such as average time elapsed before detection, are not as important in this application. Because of this, the benchmarking system focuses on computing false detections (non pedestrians whose paths were recorded) and hit rate (percentage of pedestrians that entered the field of view whose paths were recorded). This is found by comparing each path produced by the system to the ground truth set and finding if a match exists.

It is not clear cut exactly how to evaluate whether two paths are the same. In this work, a path is represented as a list of timestamped x, y, z coordinates. Below are examples of issues when trying to compare paths represented by timestamped locations.

- Two clustering algorithms may have different run speeds. This would give a different number of points in the same path for the same pedestrian

- A clustering algorithm can be non deterministic, so two runs on the same dataset can give slightly different locations for cluster locations, depending on where the algorithm converges
- Should a path which is a subset of a ground truth path be counted as a match?
- Should two paths match if they have a large intersection but a larger symmetric difference?
- How many points need to match between two paths? Should it be a percentage, or a threshold?

There are many potential ways in which to determine whether or not two paths are a match. For this work, a threshold of 100 points is used with a distance tolerance of 10 centimeters and a time tolerance of 10 milliseconds. In other words, if a path had at least 100 points which were within 0.1m and 0.01s of a corresponding point in a ground truth path, then the paths are a match. Any path that was left unmatched is a false detection.

In addition to paths of pedestrians, the ground truth contains a special category filled with the paths of objects that this work does not care about, known as "don't cares". don't cares consist of paths of objects that are human, but not important to the work. The two major examples are pedestrians on bicycles and seated pedestrians. In this work, the system is neither penalized nor rewarded if it extracts the path of a bicyclist or seated person.

The dataset is also divided and evaluated with respect to the difficulty. There are four difficulties, listed here.

All – All pedestrians that enter the system's field of view.

Hard – Only pedestrians that enter the system's field of view for under two seconds.

Medium – Only pedestrians that enter the system's field of view for over two seconds.

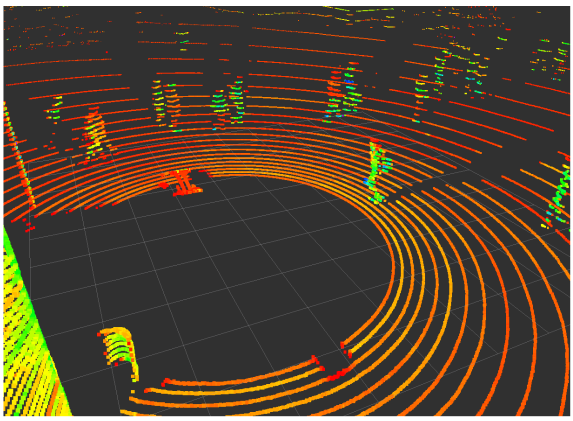


Figure 7-5: Example of an occurance in which the system has difficulty identifying all pedestrians due to frequent occlusions. Image is of a visualization of 3D LiDAR points, as images are not recorded due to privacy reasons.

Easy – Only pedestrians that enter the system's field of view for over two seconds and who's path is not 80 percent similar to another path in the ground truth set (pedestrians with similar routes are problematic due to occlusions and group clustering).

Separation into difficulty categories illuminates performance in different scenarios and conditions, and allows us to understand which scenarios are causing the greatest problems. An example of difficult situation of a bursty group of pedestrians which is isolated by the dataset separation can be seen in figure 7-5.

Chapter 8

Results

8.1 System Performance using Maximum Likelihood Fusion

The output of the proposed system using maximum likelihood fusion is tested against the ground truth while sweeping the cluster hit threshold. Results are shown in figure 8-1. Overall the system performs relatively well, capturing 90% of the pedestrians that enter the field of view while allowing under 5 false detections per minute on average.

To understand cases which cause tracking errors, the dataset is divided into categories as descibed in section 7, and the evaluation is run on each set. Figure 8-2 shows system performance for only pedestrians that fell into the hard category. These pedestrians only entered the system's field of view for fewer than two seconds.

Clearly the system struggles with this set of pedestrians. The issue lies in the time required to accrue sufficient hits. Because these pedestrians are only in range briefly, the proposed system has a short period of time in which to associate the clusters as pedestrians. In this work, these misses are accepted as part of the tradeoff of the system. In order to try to capture these pedestrian paths, it would require that few hits constitute a pedestrian, which would cause many more false detections.

Figure 8-3 shows performance on the dataset excluding hard pedestrians. By re-

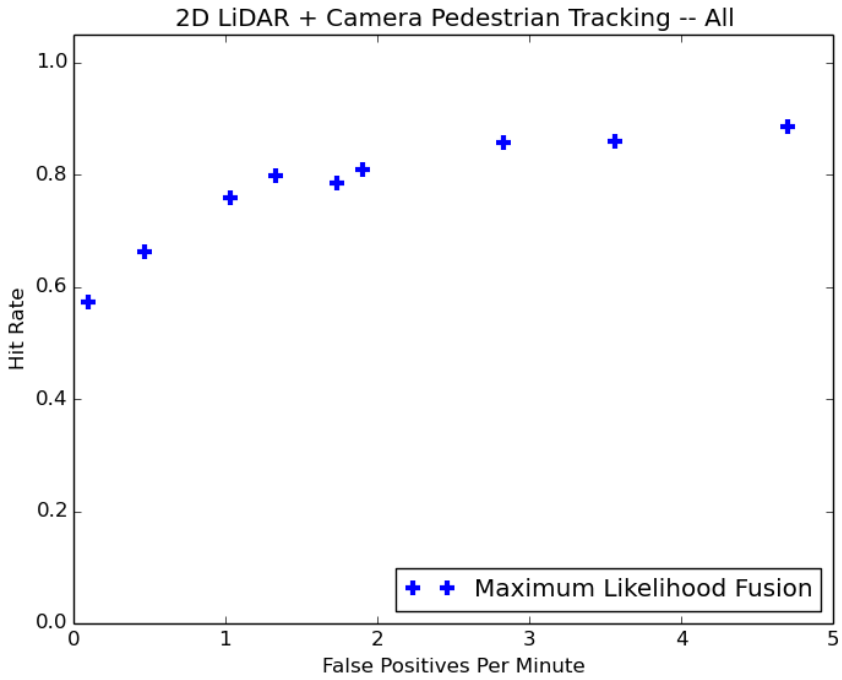


Figure 8-1: Performance of system using maximum likelihood fusion on all pedestrians from dataset

moving the hard subset of pedestrians, the maximum likelihood fusion system acheives over 80% accuracy with 1.5 false detections per minute on average.

Further removing categories of pedestrians, system performance on the subset of pedestrians in the dataset who enter the field of view for over two seconds and who's path is not over 80% similar to another path in the ground truth set is analyzed. Similar pedestrian paths were removed for the final set because pedestrians that walk in groups tend to cause problems due to occlusions and group clustering. Figure 8-4 displays performance for this subset, dubbed the easy set.

As seen in the plot, the proposed system is able to collect nearly every pedestrian in the dataset for this category with 3 false detections per minute. Although a very good result in terms of hit rate, three false detections per minute may be high for some applications. In the following section it is shown that probabilistic fusion acheives similar hit rates with many fewer false detections per minute.

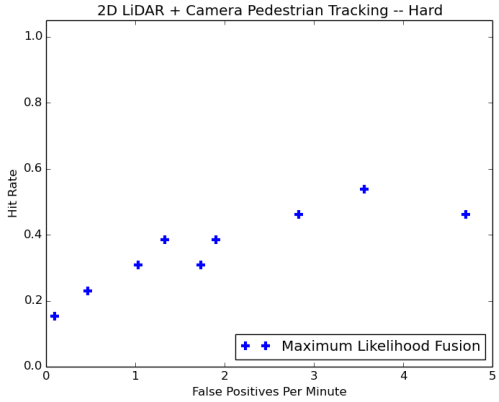


Figure 8-2: Performance of system using maximum likelihood fusion on pedestrians from dataset which entered the field of view for fewer than two seconds

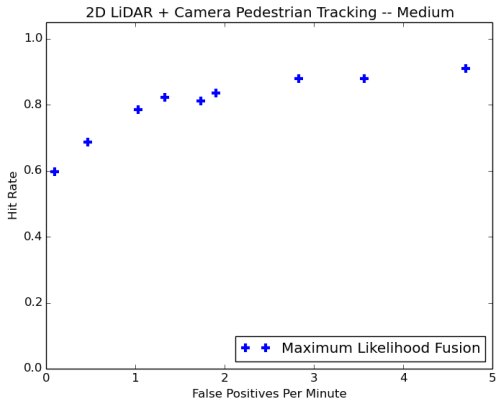


Figure 8-3: Performance of system using maximum likelihood fusion on pedestrians from dataset which entered the field of view for greater than two seconds

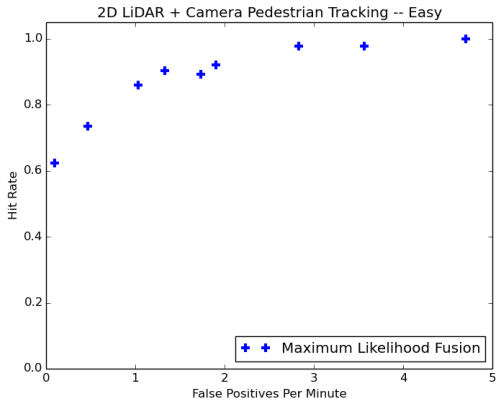


Figure 8-4: Performance of system using maximum likelihood fusion on pedestrians from dataset which entered the field of view for greater than two seconds and also did not have a path over 80% similar to another pedestrian's path

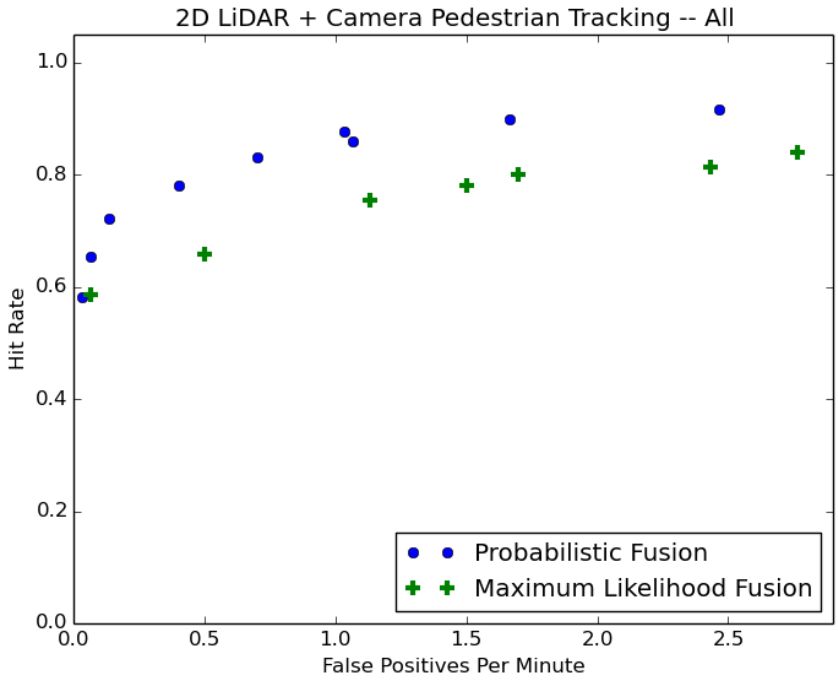


Figure 8-5: Comparison of two fusion schemes on entire pedestrian dataset between probabilistic fusion and maximum likelihood fusion.

8.2 System Performance using Probabilistic Fusion

In this section we introduce the performance of the novel probabilistic fusion and compare to the maximum likelihood performance. Figure 8-5 shows performance of maximum likelihood fusion and probabilistic fusion for the full dataset.

Probabilistic fusion outperforms maximum likelihood fusion for all values of the swept hit count threshold. This comes from a few sources. First, utilizing the information provided by the emphabsence of a hit on a cluster helped reduce false detections. Second, it is likely that, despite our best efforts, the extrinsic calibration between the sensors is actually slightly incorrect for the vehicles used to record the ground truth set. This is more problematic for maximum likelihood fusion.

Additionally, the performance of the two fusion techniques are compared in each of the dataset subsets in figures 8-6, 8-7, and 8-8.

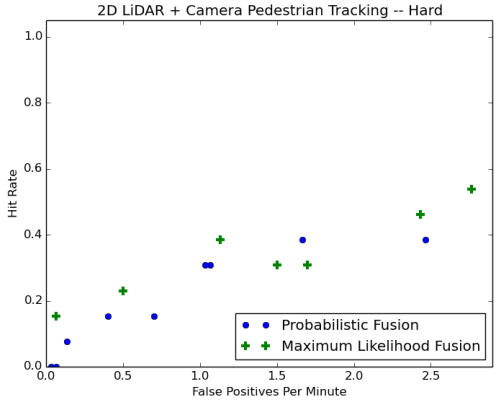


Figure 8-6: Performance comparison of the two fusion schemes on pedestrians from dataset which entered the field of view for fewer than two seconds

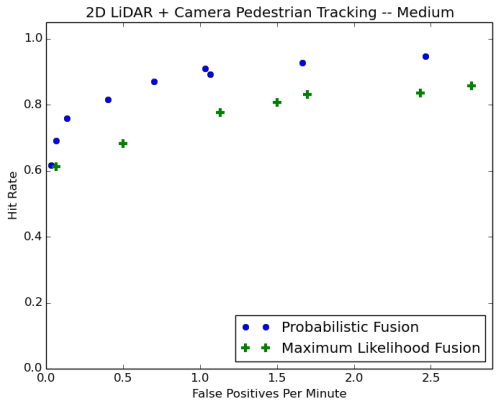


Figure 8-7: Performance comparison of the two fusion schemes on pedestrians from dataset which entered the field of view for greater than two seconds

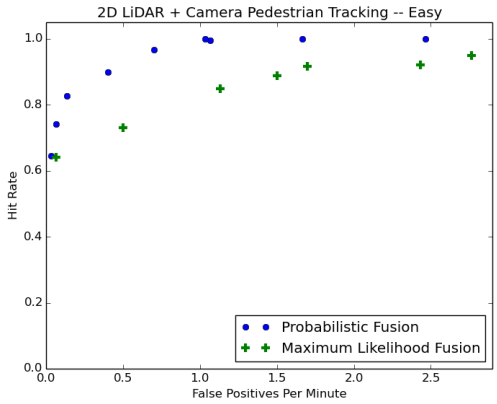


Figure 8-8: Performance comparison of the two fusion schemes on pedestrians from dataset which entered the field of view for greater than two seconds and also did not have a path over 80% similar to another pedestrian's path

The novel probabilistic fusion system clearly outperforms the maximum likelihood fusion in all subsets of the dataset except for the hard dataset, in which the results of the two fusion schemes are similar. This comes from the lack of time to accrue sufficient hits to properly differentiate the schemas. These misses are accepted as part of the tradeoff of the system, as attempting to collect them accurately would cause many more false detections.

In summary, the novel probabilistic fusion outperforms maximum likelihood fusion and acheives accurate results in the target environment. When excluding difficult pedestrians which hardly enter the field of view, the proposed system was able to accurately collect the path of every pedestrian for 22 minutes in an active university campus with under two false detections per minute. When including all pedestrians, the system was able to collect 214 of 237 (90.3% hit rate) pedestrians correctly, and produced a passable 33 false detections (1.5 false detections per minute).

8.3 Performance with Synthetic Extrinsic Calibration Error

As the vehicles are used, sensor mounts may suffer wear due to weather, road conditions, or human interaction. To explore how performance will suffer under those conditions, benchmarking was run with synthetic extrinsic calibration error. Table 8.1 shows system performance between maximum likelihood fusion and probabilistic fusion as the amount of synthetic extrinsic calibration error is swept from 0 to 20 degrees of rotational error in 2.85 degree increments.

To directly analyze system performance with respect to internal parameters, a objective function is defined

$$J = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}$$

which measures the percentage of recorded paths which belonged to pedestrians, a valuable measurment for a mobility on demand system. The higher the objective

Table 8.1: Table showing how the two fusion methods perform as synthetic extrinsic calibration error is added into the system

Extrinsic Calibration Error (Degrees)		0.0	2.85	5.7	8.55	11.4	14.25	17.1	19.95
Prob. Fusion	Hit Rate	0.932	0.940	0.937	0.932	0.886	0.768	0.696	0.656
	False Detections Per Min	3.73	3.99	3.53	3.33	3.33	3.07	3.08	2.84
ML Fusion	Hit Rate	0.873	0.831	0.747	0.667	0.637	0.574	0.540	0.515
	False Detections Per Min	3.40	4.59	5.31	5.74	5.41	5.40	5.78	5.58

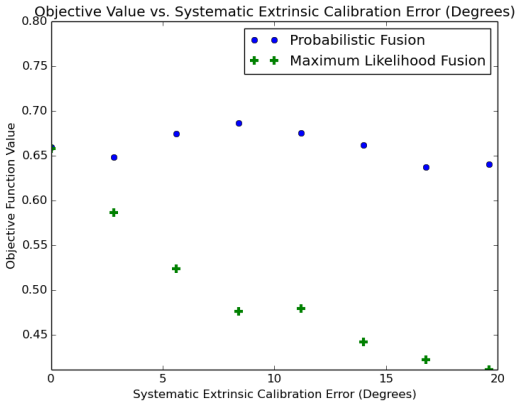


Figure 8-9: Objective value of the system using two different fusion types as systematic error is increased

value the better the system is performing. Figure 8-9 shows how the objective function varies with systematic error.

Probabilistic fusion is able to outperform maximum likelihood fusion when synthetic error is increased, as seen in figure 8-9, and even maintain acceptable system quality under many degrees of miscalibration. For a direct comparison between the two fusion types, figure 8-10 shows the ratio of objective value between the two fusion types as systematic calibration error increases.

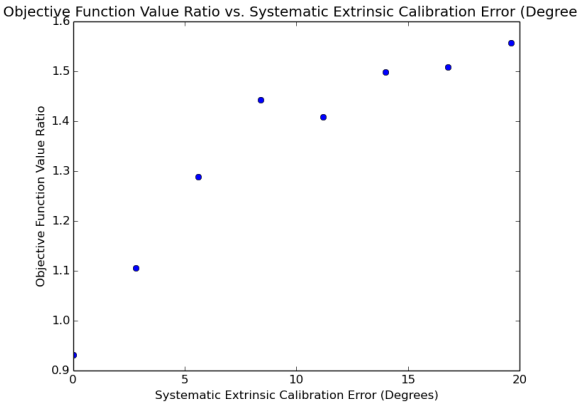


Figure 8-10: Ratio between probabilistic fusion objective value and maximum likelihood objective value as systematic error is increased

8.4 Further Probabilistic Fusion Analysis

The use of probabilistic fusion adds another free parameter to the system, σ. The role of σ can be seen in equation 6.2, and can be summarized as the rate of fall off of hit value with respect to distance from a detection.

σ represents a tradeoff. A small value of σ provides fast hit decay, and therefore produces a more exact detection. If extrinsic calibration error is negligible, this is beneficial, as the only cluster that will receive a significant hit will be the intended cluster. The issue arises if instead calibration error is non-negligible. In this case, the true pedestrian cluster will receive no hits and surrounding non-pedestrian clusters will recieve much larger hits.

Conversely, If σ is allowed to be large, hit values are spread farther from a detection. If calibration error is small the pedestrian cluster will accrue the largest hits, but surrounding clusters will also receive hits which may lead to erroneously identifying them as pedestrians. The benefit of a small σ can be seen if calibration error is non-negligable. With both a large σ and non-neglibable extrinsic error, pedestrian clusters will still be identified, despite other non-pedestrian clusters potentially being more directly targeted. This tradeoff is summarized in table 8.2

The effect of choice σ can be further seen in figure 8-11 and the corresponding table 8.3. In this figure, hit count threshold is kept constant while σ is swept from

	Negligible Calibration Error	Non-Negligible Calibration Error
Low σ	high hit rate, low false detections	low hit rate, low false detections
High σ	high hit rate, high false detections	high hit rate, high false detections

Table 8.2: Summary of the effects of the parameter σ

Sigma	0.03	0.06	0.09	0.12	0.15	0.18	0.21
Hit Rate	0.861	0.907	0.924	0.928	0.924	0.945	0.941
False Detections Per Minute	1.09	1.91	2.57	3.14	3.66	3.96	4.22

Table 8.3: Values of hit rate and false detections per minute as σ is swept under no exterinsic calibration error.

low to high under no syntheic noise. As σ is increased, the false detections increase as expected (hit rate also increases some, due to some small accidental calibration error in the original set up used to record ground truth).

Figure 8-12 and the corresponding table 8.4 show the effect of σ under synthetic extrinsic calibration error. In this figure, hit count threshold is kept constant while σ is swept from low to high under 8 degrees of synthetic extrinsic calibration noise. As σ is increased, the false detections and hit rate increase as expected, as significant hits are distributed farther from the detection.

In the end, the application determines just how to set this parameter, depending on the number of false detections permitted. The more susceptible to calibration

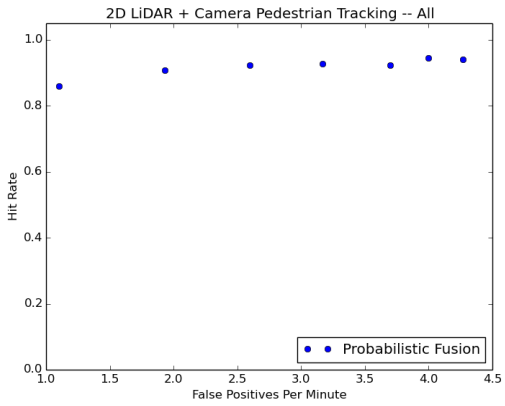


Figure 8-11: Figure displaying effects of σ under negligible calibration error

Sigma	0.03	0.06	0.09	0.12	0.15	0.18	0.21
Hit Rate	0.219	0.409	0.671	0.848	0.916	0.932	0.941
False Detections Per Minute	1.19	2.01	2.38	2.71	2.77	3.60	3.83

Table 8.4: Values of hit rate and false detections per minute as σ is swept under non-negligable extrinsic calibration error.

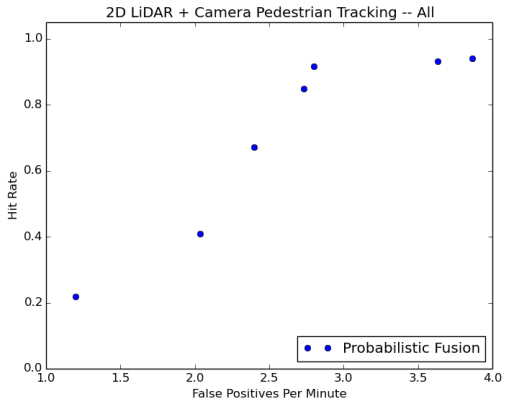


Figure 8-12: Figure displaying effects of σ under non-negligible calibration error

error or less concerned with false detections a application is, the higher σ should be set. The more harmful false detections are or less susceptible to calibration error a system is, the lower σ should be set.

Chapter 9

Conclusions

This work has presented a pedestrian detection and tracking system using camera and LiDAR capable of performing robustly in its target environment, a university campus. The system is (1) cheap, using only off the shelf web cameras, 2D LiDARs, and a single Gigabyte Brix machine, (2) fast, the system is able to process 90 images per second and 50 LiDAR revolutions per second, and (3) high quality, capturing over 90% of pedestrians paths while capturing only 1.5 false detections per minute on average in the target environment. Additionally, a novel fusion system is presented which is capable of withstanding systematic miscalibration error between camera and LiDAR, which is a threat for heavily used systems.

This system is already deployed, providing the mobility on demand service to students at MIT, starting as a fleet of three vehicles. As the vehicles log more hours and observe pedestrian flow through the MIT network, the resulting path data will be utilized to more accurately position vehicles so as to optimize transit time and improve quality of service.

In the not-so-distant future, autonomous vehicles will be commonplace on campuses and city streets. When this day comes, the sensor suite used in the proposed system will come standard in vehicles. This work could be used to record pedestrian traffic on larger scales, expanding city-wide for use in urban planning to help make transportation and mobility truly seamless.

9.1 Future Work

In part because of the many subcomponents which make up this work, there are many interesting areas of future work for improving pedestrian detection and tracking onboard a mobility on demand vehicle.

9.1.1 Pedestrian Detection

In the current state, VeryFast pedestrian detector is used for all image processing despite no longer being the state of the art. There is a great deal of work occuring on deep neural nets for pedestrian detection, which has surpassed feature based detection. At the time of this writing, these deep learning methods were too slow for this work's purpose because of how computationally expensive deep learning is.

As more research is done to require less computation power, deep learning will likely become the obvious solution in these types of applications. [9] already has an implementation capable of running at 30Hz on a Titan X GPU. Once speed up is greater, deep learning would be a very powerful detector for mobility on demand systems not only due to the quality, but because it is also a multiclass detector, so vehicle and bicycle traffic can also be extracted.

9.1.2 Tighter Submodule Integration

Currently, the proposed system is very linear. Clustering occurs independently of pedestrian detection, pedestrian detection works independently of any information used by the clustering, then the output of the two are fused together to collect paths.

Group clustering could be aided by submodule integration. Group clusters occur frequently in college campus environments when friends walk together through the network, which causes the loss of a pedestrian path. If the pedestrian detector is consistently classifying the cluster as two pedestrians, this information can be used to split a shared cluster, thereby providing more accurate clustering.

A cluster's recorded velocity could be used to locate a lost cluster. Currently, if a cluster is temporally lost, Dynamic Means uses something similar to a nearest

neighbor search from the last known location to update cluster location, but the recorded velocity would be a better indicator of where to search for the cluster. This could cause fewer cluster fractures and produce overall better temporal association.

9.1.3 3D LiDAR Use

3D LiDAR integration would provide better clustering and temporal association, and could be used for pedestrian detection.

There is already work attempting 3D LiDAR classification [26]. Having a vote from the LiDAR module would remove total classification responsibility from the vision module. Currently, if the vision model does not find a pedestrian, no path will ever be recorded. Introducing detection in LiDAR space would remove this single point of failure.

In this system 3D LiDAR was not used as we were unable to perform real-time clustering in a 3D point cloud using Dynamic Means. Some work could be done to try to more efficiently downsample and remove map points to try to enable the real-time use of 3D LiDAR data.

9.1.4 2D LiDAR Classification

There is already work in the area of 2D LiDAR Classification. At the moment, nothing has seemed promising [24], but some classification attempts have been made by developing feature vectors from the cluster geometry and motion patterns [29, 27, 22].

Similar to the advances deep learning has provided in other fields, it could potentially perform better pedestrian detection in point cloud space. This way, no feature engineering would need to be done, the neural net would learn the best features for classification in the space. [24] attempted using a two layer fully connected neural net for 2D LiDAR detection, but with recent advances in nerual net architecture research, it is possible significant improvements to this architecture could be made, making a better classifier.

The production of the training data would not be difficult. This work presented a method to filter pedestrian clusters to produce ground truth from data recorded by utlizing the proposed system. The same process could be used to record a dataset of positive and negative examples of pedestrian clusters. Assuming two pedestrians are in view at any given time and the LiDAR spins at 40Hz, producing 100,000 pedestrians clusters would require sifting through approximately a 20 minute dataset, which is very feasible. For comparison, producing ground truth required manual filtering of a 22 minute dataset.

In the end it may be the case that there is just not be enough data in a 2D cluster to accurately distinguish clusters. In fact, it is very difficult to do so accurately even for humans. Regardless, this could prove to be an interesting area of research to improve system quality.

9.1.5 360 Degree Coverage

The final potential area of interest is the addition of rear sensor coverage. This would enable the capture of pedestrians which only walk behind the system and allow for twice the maximum tracking distance. This could be very powerful for understanding pedestrian flows in a network, as more transitions and human behavior would be captured, as opposed to shorter segments which can only serve as link counts.

Additionally, having more cameras at different angles can provide more votes on whether or not a pedestrian exists on a frame, making the overall system more reliable. Because the system is very scalable computationally, the doubling of the sensor suite would only require the addition of one more Gigabyte Brix, which is relatively inexpensive and simple.

Bibliography

[1] T. Campbell, M. Liu, B. Kulis, J. P. How, and L. Carin, "Dynamic clustering via asymptotics of the Dependent dirichlet process mixture," in *Advances in Neural Information Processing Systems*, 2013, pp. 449–457. [Online]. Available: http://papers.nips.cc/paper/5094-dynamic-clustering-via-asymptotics-of-the-dependent-dirichlet-process-mixture

[2] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *CVPR*, 2012.

[3] C. Papageorgiou and T. Poggio, "Trainable pedestrian detection," in *1999 International Conference on Image Processing, 1999. ICIP 99. Proceedings*, vol. 4, 1999, pp. 35–39 vol.4.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 1, Jun. 2005, pp. 886–893 vol. 1.

[6] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[7] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.

[8] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik, "Recurrent Network Models for Human Dynamics," 2015, pp. 4346–4354. [Online]. Available: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Fragkiadaki_Recurrent_Network_Models_ICCV_2015_paper.html

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv:1506.01497 [cs]*, Jun. 2015, arXiv: 1506.01497. [Online]. Available: http://arxiv.org/abs/1506.01497

[10] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a Deeper Look at Pedestrians," *arXiv:1501.05790 [cs]*, Jan. 2015, arXiv: 1501.05790. [Online]. Available: http://arxiv.org/abs/1501.05790

[11] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 4, pp. 743–761, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2011.155

[12] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications," *Robotica*, vol. 2005, pp. 17–25, 2005. [Online]. Available: http://www.isr.uc.pt/~cpremebida/files_cp/tech2.pdf

[13] J. C. Tilton, "Image segmentation by region growing and spectral clustering with natural convergence criterion," in *International geoscience and remote sensing symposium*, vol. 4. INSTITUTE OF ELECTRICAL & ELECTRONICSENGINEERS, INC (IEE), 1998, pp. 1766–1768.

[14] R. Xu and I. Wunsch, D., "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005.

[15] A. Nagpal, A. Jatain, and D. Gaur, "Review based on data clustering algorithms," in *2013 IEEE Conference on Information Communication Technologies (ICT)*, Apr. 2013, pp. 298–303.

[16] P. Willett, "Recent trends in hierarchic document clustering: A critical review," *Information Processing & Management*, vol. 24, no. 5, pp. 577–597, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0306457388900271

[17] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960. [Online]. Available: http://fluidsengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1430402

[18] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *AeroSense'97*. International Society for Optics and Photonics, 1997, pp. 182–193. [Online]. Available: http://proceedings.spiedigitallibrary.org/proceeding.aspx?articleid=925842

[19] E. Wan, R. Van Der Merwe, and others, "The unscented Kalman filter for nonlinear estimation," in *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*. IEEE, 2000, pp. 153–158. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=882463

[20] D. Held, J. Levinson, S. Thrun, and S. Savarese, "Robust real-time tracking combining 3d shape, color, and motion," *The International Journal of Robotics Research*, Aug. 2015. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364915593399

[21] P. Morton, B. Douillard, and J. Underwood, "An evaluation of dynamic object tracking with 3d LIDAR," in *Proc. of the Australasian Conference on Robotics & Automation (ACRA)*, 2011. [Online]. Available: http://www.araa.asn.au/acra/acra2011/papers/pap137.pdf

[22] M. Lee, S. Hur, and Y. Park, "An Obstacle Classification Method Using Multi-feature Comparison Based on 2d LIDAR Database," in *2015 12th International Conference on Information Technology - New Generations (ITNG)*, Apr. 2015, pp. 674–679.

[23] E. P. Fotiadis, M. Garzn, and A. Barrientos, "Human Detection from a Mobile Robot Using Fusion of Laser and Vision Information," *Sensors*, vol. 13, no. 9, pp. 11 603–11 635, Sep. 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/9/11603

[24] C. Premebida, O. Ludwig, and U. Nunes, "Exploiting LIDAR-based features on pedestrian detection in urban scenarios," in *12th International IEEE Conference on Intelligent Transportation Systems, 2009. ITSC '09*, Oct. 2009, pp. 1–6.

[25] D. R. Kisku, P. Gupta, and J. K. Sing, "Multibiometrics Feature level fusion by Graph clustering," *International Journal of Security and Its Applications*, vol. 5, no. 2, pp. 61–74, 2011. [Online]. Available: http://www.sersc.org/journals/IJSIA/vol5_no2_2011/5.pdf

[26] K. Cho, C. Kim, S.-H. Baeg, and S. Park, "Learned geometric features of 3d range data for human and tree recognition," *Electronics Letters*, vol. 50, no. 3, pp. 173–175, Jan. 2014.

[27] C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto, "A Lidar and Vision-based Approach for Pedestrian and Vehicle Detection and Tracking," in *IEEE Intelligent Transportation Systems Conference, 2007. ITSC 2007*, Sep. 2007, pp. 1044–1049.

[28] C. Premebida, O. Ludwig, and U. Nunes, "LIDAR and vision-based pedestrian detection system," *Journal of Field Robotics*, vol. 26, no. 9, pp. 696–711, Sep. 2009. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/rob.20312/abstract

[29] H. Cho, Y.-W. Seo, B. Vijaya Kumar, and R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 1836–1843.

[30] R. Cifuentes, D. Van der Zande, C. Salas, J. Farifteh, and P. Coppin, "Correction of Erroneous LiDAR Measurements in Artificial Forest Canopy Experimental Setups," *Forests*, vol. 5, no. 7, pp. 1565–1583, Jul. 2014. [Online]. Available: http://www.mdpi.com/1999-4907/5/7/1565/

[31] G. Kim, J. Eom, and Y. Park, "Investigation on the occurrence of mutual interference between pulsed terrestrial LIDAR scanners," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2015, pp. 437–442.

[32] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *BMVC*, 2009.

[33] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *PAMI*, vol. 34, 2012.

[34] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999. [Online]. Available: http://www.yorku.ca/gisweb/eats4400/boost.pdf

[35] L. Breiman, "Random forests-random features," *UC Berkeley, Statistics Department, Technical Report*, no. 567, 1999. [Online]. Available: http://nma.berkeley.edu/ark:/28722/bk0000n2728