UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Mahir Gulzar

Object Detection Using LiDAR and Camera Fusion in Off-road Conditions

Master's Thesis (30 ECTS)

Supervisor: Tambet Matiisen

Tartu 2019

Object Detection Using LiDAR and Camera Fusion in Off-road Conditions

Abstract:
Since the boom in the industry of autonomous vehicles, the need for precise environment perception and robust object detection methods has grown. While we are making progress with state-of-the-art in 2D object detection with approaches such as convolutional neural networks, the challenge remains in efficiently achieving the same level of performance in 3D. The reasons for this include limitations of fusing multi-modal data and the cost of labelling different modalities for training such networks. Whether we use a stereo camera to perceive scene's ranging information or use time of flight ranging sensors such as LiDAR, the existing pipelines for object detection in point clouds have certain bottlenecks and latency issues which tend to affect the accuracy of detection in real time speed. Moreover, these existing methods are primarily implemented and tested over urban cityscapes.
This thesis presents a fusion based approach for detecting objects in 3D by projecting the proposed 2D regions of interest (object's bounding boxes) or masks (semantically segmented images) to point clouds and applies outlier filtering techniques to filter out target object points in projected regions of interest. Additionally, we compare it with human detection using thermal image thresholding and filtering. Lastly, we performed rigorous benchmarks over the off-road environments to identify potential bottlenecks and to find a combination of pipeline parameters that can maximize the accuracy and performance of real-time object detection in 3D point clouds.

Keywords:
Object detection, sensor fusion, LiDAR, camera, frustum, off-road scenarios, ROS.

CERCS: P170 Computer science, numerical analysis, systems, control

Objektituvastus maastikul kasutades lidarit ja kaamerat

Lühikokkuvõte:
Seoses hüppelise huvi kasvuga autonoomsete sõidukite vastu viimastel aastatel on suurenenud ka vajadus täpsemate ja töökindlamate objektituvastuse meetodite järele. Kuigi tänu konvolutsioonilistele närvivõrkudele on palju edu saavutatud 2D objektituvastuses, siis võrreldavate tulemuste saavutamine 3D maailmas on seni jäänud unistuseks. Põhjuseks on mitmesugused probleemid eri modaalsusega sensorite andmevoogude ühitamisel, samuti on 3D maailmas märgendatud andmestike loomine aeganõudvam ja kallim. Sõltumata sellest, kas kasutame objektide kauguse hindamiseks stereo kaamerat või lidarit, kaasnevad andmevoogude ühitamisega ajastusprobleemid, mis raskendavad selliste lahenduste kasutamist reaalajas. Lisaks on enamus olemasolevaid lahendusi eelkõige välja töötatud ja testitud linnakeskkonnas liikumiseks.

Töös pakutakse välja meetod 3D objektituvastuseks, mis põhineb 2D objektituvastuse tulemuste (objekte ümbritsevad kastid või segmenteerimise maskid) projitseerimisel 3D punktipilve ning saadud punktipilve filtreerimisel klasterdamismeetoditega. Tulemusi võrreldakse lihtsa termokaamera piltide filtreerimisel põhineva lahendusega. Täiendavalt viiakse läbi põhjalikud eksperimendid parimate algoritmi parameetrite leidmiseks objektituvastuseks maastikul, saavutamaks suurimat võimalikku täpsust reaalajas.

Võtmesõnad:
Objektituvastus, sensorite ühitamine, lidar, kaamera, maastikukeskkond, ROS.

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

List of Abbreviations

Abbreviation	Explanation
SAE	Society of Automation Engineers
ROS	Robot Operating System
UGV	Unmanned Ground Vehicle
LiDAR	Light Detection and Ranging
RANSAC	Random sample consensus
SVM	Support Vector Machine
HOG	Histogram of Oriented Gradients
CNN	Convolutional Neural Network
R-CNN	Region Convolutional Neural Network
FOV	Field of View
DBSCAN	Density-based spatial clustering of applications with noise
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
RPN	Region Proposal Network

Table of Contents

1 Introduction

1.1 Motivation and Background

Development of a fully autonomous vehicle where an intelligent agent can fully perceive the environment and execute control decisions in real-time is among one of the most challenging tasks for AI researchers. In fact, this research domain can be subdivided into different categories of computer science and robotics, such as computer vision, machine learning, deep learning, sensor simulation etc. The development architecture of an autonomous vehicle is divided into different submodules which work in a closed loop. These modules are environment perception, vehicle localization, path planning and motion control (Figure 1.1).

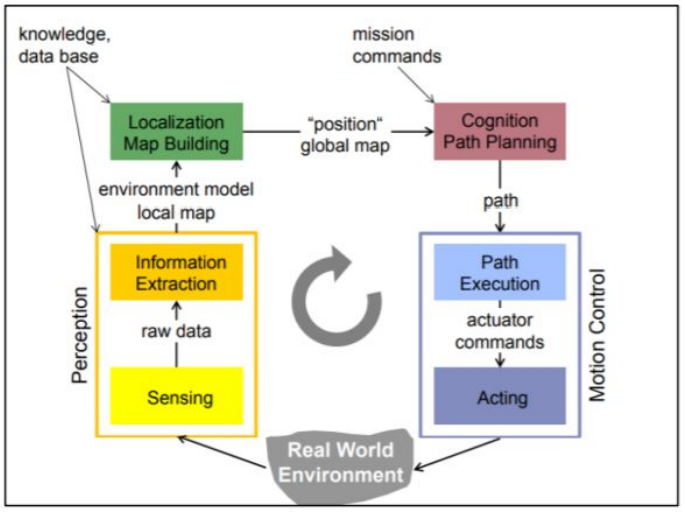


Figure 1.1. The basic framework of an autonomous vehicle [1].

The main focus of this thesis is the perception module. The perception module detects objects surrounding the vehicle e.g humans, vehicles, trees, buildings etc and might also include object tracking as well. Our research revolves specifically around detecting humans in real-time using data from multiple modalities e.g. camera and ranging sensors in off-road scenarios. The top priority of a driverless vehicles is to reduce human error to ensure safety of humans [2], so we consider this as a benchmarking parameter in this thesis, mainly because even small errors in detection can cause fatalities.

When it comes to detection, computer vision based approaches are very popular nowadays. In recent years the emergence of fusion-based methods in multi-modal systems became useful in different applications such as environment mapping [3] and autonomous driving [4]. Using a combination of cameras with different ranging sensors (such as LiDAR) is a way for an autonomous vehicle to detect objects with distance information. A camera gives us high-quality scene image and a ranging sensor provides depth or distance information of the environment. Single sensor data is often not enough for the complete perception of the surrounding environment of an autonomous vehicle mainly because of the limitations of each sensor [5]. Multi-modality (data from multiple sensors) comes into play for perceiving different attributes of the environment to make

precise control decisions. On the other hand, multi-modal data has its own limitations [6] which will be discussed in later chapters.
Instead of considering the problem of autonomy of a vehicle, in general, we look at it from the perspective of the levels of autonomy. The Society of Automotive Engineers (SAE) has defined a very useful classification system for different levels of driving automation [7]. These levels are listed in Table 1.1 below.

Table 1.1. SAE levels of driving automation summary See [7, p. 17].

SAE Level	Name	Description	Execution	Environment Monitoring	Fallback performance of dynamic task	Driving Modes
Human driver monitors driving environment						
0	**No Automation**	Full-time performance by human overall dynamic tasks	Human Driver	Human Driver	Human Driver	N/A
1	**Driver Assistance**	Driving mode specific execution by a driver assistance with an expectation that human driver will perform all remaining aspects of dynamic tasks	Human Driver and System			Some driving modes
2	**Partial Automation**	Driving mode specific execution by one or more driver assistance with an expectation that human driver will perform all remaining aspects of dynamic tasks	System			
Automated driving system monitors the driving environment						
3	**Conditional Automation**	Driving mode specific performance by an automated driver system with the expectation that a human driver will respond appropriately to a request to intervene	System	System	Human Driver	Some driving modes
4	**High Automation**	Driving mode specific performance by an automated driver system even if a human driver does not respond appropriately to a request to intervene			System	Many driving modes
5	**Full Automation**	Full-time performance of an automated driving system of all aspects of a dynamic driving task in all environmental conditions.				All driving modes

Audi claims to have reached to the level 3 of the classification table [8] i.e. able to drive autonomously in good conditions on highways and with light traffic in the city, while requiring safety driver to be alert at all times. Even though achieving level 3 is a considerable advancement in the automotive industry, we still lack in achieving the same level of autonomy in off-road scenarios i.e. these advancements are applicable only in well structured urban environments. There are several reasons for this:

- Off-road scenarios, in general, don't have any structural aspect to the environment i.e. there are no lane markers and road signs to assist the vehicle. The terrain is uneven compared to the roads in an urban environment and there

is no clear distinction between drivable and non-drivable areas i.e. it is fine to drive through bushes sometimes.
- There is no fixed point of reference for vehicle localization (trees and bushes move slightly all the time compared to buildings in urban scenarios which stay still).
- The scarcity of off-road datasets.

1.2 Problem Statement

Our work is primarily focusing on object detection in off-road scenarios. Another challenge is to identify potential bottlenecks in our pipeline and tweak the environment parameters to maximize the performance gain of the detection.

There are two core questions which we will address in this study:

1. *How well can we detect humans in real-time using mature 2D object detection methods to assist locating humans in 3D in off-road scenarios?*
2. *What are the best parameters for stable and robust benchmarking of object detection using Robotic Operating System (ROS)?*

In this thesis, a real-time model is defined as a minimum of 10 frames per second.

This study is a part of a collaborative project "Applied research on system of sensors and software algorithms for safety and driver assistance on remotely operated ground vehicles for off-road applications" with Milrem Robotics. The aim of this research is to develop a baseline for fast and robust object detection in off-road scenarios for an unmanned ground vehicle (UGV). The main objective is to maximize the performance of 3D object detection using existing state-of-the-art 2D detection methods with camera and LiDAR fusion. Inspired from frustum pointnets [9] method, which currently ranks at the 4th position for detecting pedestrians in KITTI dataset [10], we develop simple 3D object detection pipeline for detecting humans in off-road scenarios. This thesis also addresses the issues related to latency and stability of benchmarking results in the perception module faced specifically while working with ROS. For this, we perform extensive benchmarking over various parameters to give conclusive results and to maximize the performance of our detection pipeline.

1.3 Structure

We divide this work into the following chapters:

- **Chapter-2:** Overview of the sensors involved in the object detection process and their limitations. Camera-LiDAR calibration and fusion challenges.
- **Chapter-3:** Literature review of the existing approaches for object detection using cameras and LiDARs.
- **Chapter-4:** An overview of the chosen datasets. Architecture and flow of our perception module pipeline in ROS. A comparison of baselines with detection pipeline and a description of benchmarking setup with parameters involved.
- **Chapter-5:** Benchmarking results of our methodology over different datasets and detection methods used.

- **Chapter-6:** In the last chapter we conclude our work with a summary of our module design, discuss limitations of it and possible future improvements.

2 Sensor fundamentals

This chapter gives a brief overview of the sensors used in this project, their working principles as well as their limitations under which our detection pipeline is built upon.

2.1 Camera and Ranging Sensors

2.1.1 Camera

Optical sensors are very common for data acquisition among different available sensors. Cameras give a high-resolution 2D understanding of the environment depending on the type of camera (RGB, thermal etc.). Various computer vision based approaches can be applied to images taken from both thermal and RGB cameras but choosing the right type of camera depends on the use case. For example, detecting humans in low light situations is comparatively difficult from an RGB image than from a thermal image (Figure 2.1).

Figure 2.1: RGB vs Thermal in a low light scene [11].

Our study involves working with both RGB and thermal cameras. Thermal imagery can be used to detect people in a wide range of environmental conditions and can have performance gains over RGB [12]. RGB cameras work with visible reflected light over three channels (red, green and blue) which makes it dependent to day and night light conditions. On the other hand, RGB images have an advantage over thermal imagery in resolving ambiguous situations such as tracking people in occlusions [13]. This makes RGB a better choice for object tracking as RGB channels have more information about the object compared to a single channel thermal image. Table 2.1 summarizes the pros and cons of RGB and thermal cameras.

Table 2.1: RGB vs thermal pros and cons [13].

Camera Type	Pros	Cons
Thermal	Easier segmentation Independent of light	Re-identification difficult Expensive Fair resolution False information due to infrared reflectivity

RGB	Re-identification possible Cheap sensor High resolution	Sensitive to light Privacy issues Shadows

2.1.1.1 Camera as a ranging sensor:

The real-world scenes are three-dimensional i.e. the depth information of the scene is crucial for object detection. A camera works on the principle of pinhole camera model where the light rays pass through an aperture (hole) and the image of 3D environment is inversely projected on an image plane (Figure 2.2). The parameters of pinhole model are determined by (intrinsic) camera matrix that encodes focal length and sensor size of the camera. Furthermore, to fuse the output of two sensors placed at a certain distance in world space, the sensors undergo through a procedure of calculating external (extrinsic) parameters. This process of determining the extrinsic and intrinsic camera parameters is called calibration.

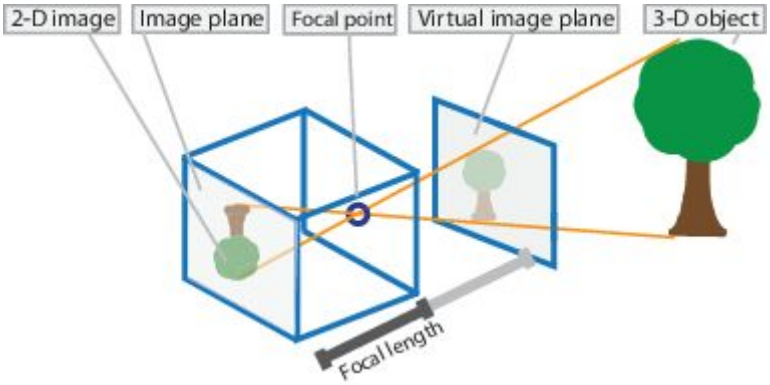


Figure 2.2. Pinhole camera model [14].

Cameras can be categorized into monocular and stereo, where stereo vision is used to perceive depth or range of the scene using two cameras similar to the way humans perceive through stereoscopic vision the distance of the objects in the world.

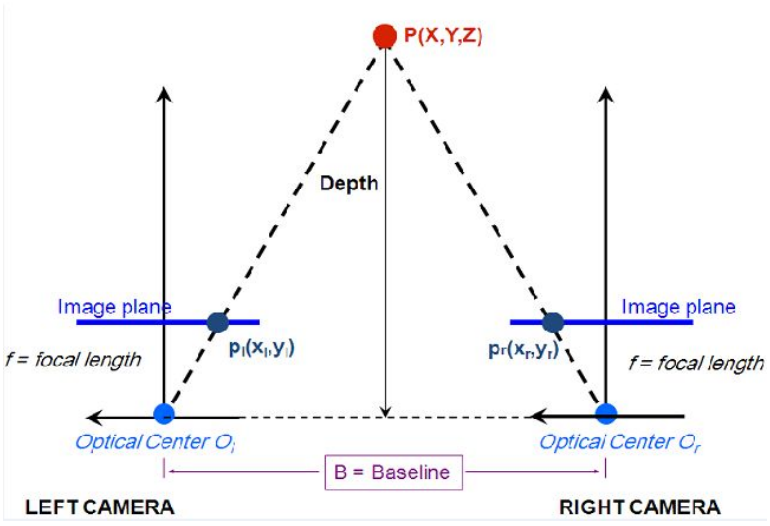


Figure 2.3. Ideal stereoscopic geometry [15].

In practice the stereo imaging for sensing range information in robotics has difficulties [16]. The main problem for stereo methods is the reliable matching of image pixels for the same object (point P in Figure 2.3) i.e. the depth information is prone to errors and less accurate. Also, the constraint of lighting conditions applies in stereo vision too. On the other hand, if mapped accurately, the stereo vision theoretically gives a high-resolution depth mapping up to infinite range - something no other range sensor gives at the moment [16]. The performance of camera sensor, in general, varies with use case scenarios and properties. Figure 2.4 shows a performance chart of camera sensor.

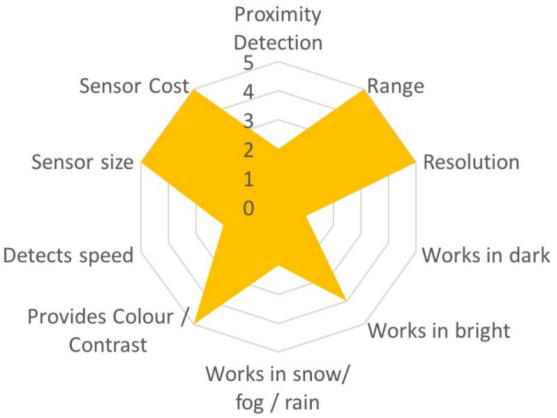


Figure 2.4. Camera vision performance chart with different use-cases on a scale of 0 to 5 (where 5 is the best performance) [17].

2.1.2 LiDAR

LiDAR (LIght Detection And Ranging) is a sensor based on time of flight principle and is very recently introduced compared to the camera. Time of flight is a method of measuring the distance of an object from the sensor. This is done in four steps:

1. emit signal (Laser) pulse from the sensor,
2. catch reflected laser back to the pulse source,
3. measure the time taken by the laser pulse to come back,
4. lastly, calculate distance using the distance formula.

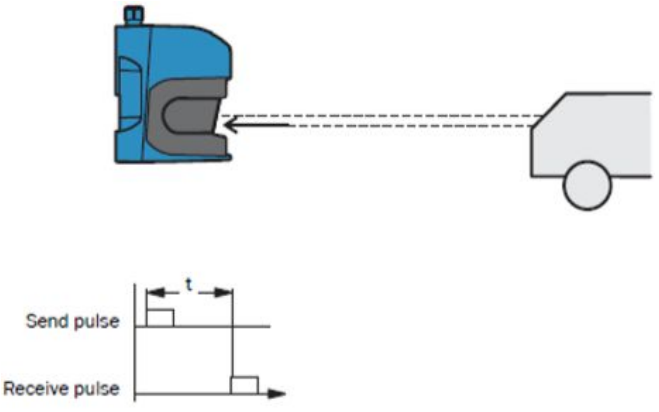


Figure 2.5. Measuring principle of LiDAR [18].

Another well known ranging sensor built on top of time of flight principle is radar (RAdio Detection And Ranging) which uses radio wave as a signal carrier whereas LiDAR uses light as a signal carrier (hence the rhyming name LiDAR). The output of LiDAR is a sequence of points consisting of four values: distance, horizontal/vertical angles and reflectivity [18]. The reflectivity of a surface depends on its texture, material and color. One of the most common ways to represent LiDAR data is point cloud due to a wide variety of available open source libraries. In point cloud format the data is represented in the form cartesian coordinate system i.e. x, y and z. Figure 2.6 shows an example frame of point cloud in urban environment

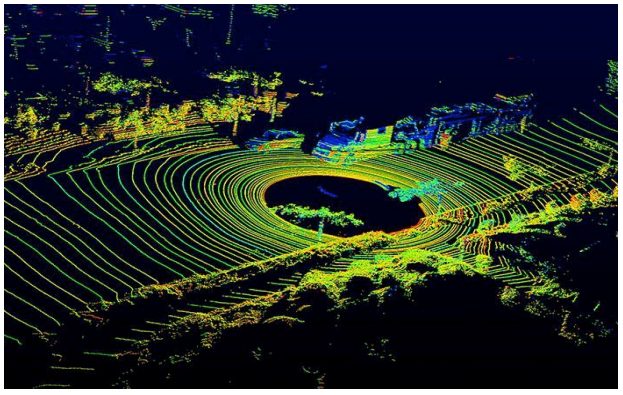


Figure 2.6. Velodyne 64 beam lidar scan [19].

In our study, we will be working on point-clouds retrieved from 32- and 16-beam Velodyne rotating LiDARs. A rotating multi-beam LiDAR provides a 360-degree field of view of the scene with a vertical resolution equal to the number of the beams, while the horizontal resolution depends on the speed of rotation. Almost all the LiDARs available today in the commercial market operate in near-infrared light spectrum thus their output power is limited for eye safety constraints.
LiDARs are typically less affected by weather conditions when compared to cameras however they lack the ability to capture the textures of the scene objects (Figure 2.7).

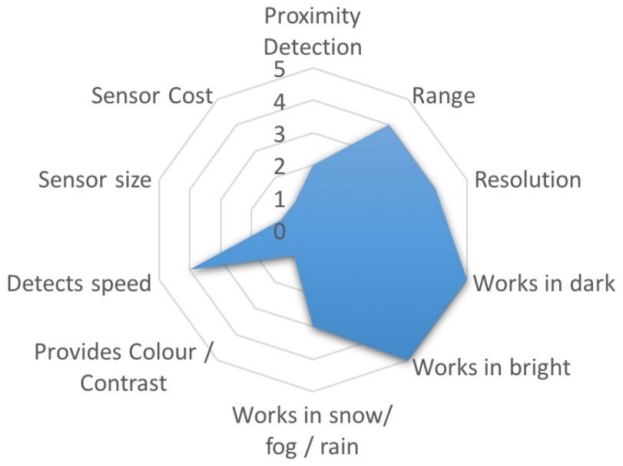


Figure 2.7. LiDAR performance chart with different use-cases on a scale of 0 to 5 (where 5 is the best performance) [17].

Additionally, a single attribute of dimension and shape cannot give accurate results in object classification. Moreover, point-clouds are sparse i.e. there is always a level of uncertainty between the scanned beams, especially for far away points. More recent advancement in LiDAR technology is Flash LiDARs which can produce a high-resolution object detail similar to that of cameras. These LiDARs were not used in the project.

2.2 Extrinsic Calibration and Sensor Fusion

At this point, we can already draw a distinct conclusion that we cannot completely rely on one vision sensor in order to do accurate and robust object detection due to the limitations of each sensor. The following table summarizes the comparisons of the sensor technologies discussed previously.

Table 2.2. A comparison of vision sensors adapted from [20].

Sensor	Principle	Computational Complexity	Characteristics
Camera	Ambient visible light intensity	Moderate	High resolution, high vertical FOV, affected by lighting conditions
Stereo camera	Multiple cameras for binocular vision	High	Similar to cameras with the addition of depth information, depth accuracy up to 50m
3D LiDAR	Measure distance & reflectance from 600-100 nm laser signal	Low	Sparse, range up to 200m, 360 surround view using a single mounting system.

Considering the characteristics of each sensor, i.e. if we get a high-resolution image of the environment from the camera and accurate depth information from LiDAR, fusing the two modalities can exploit their complementary strengths. In this context, to combine two different data modalities, we find the correspondence between the underlying data points so that we can transform from one modality to another.

In this work, the LiDAR and cameras are rigidly connected, i.e. to find correspondence between them a transformation matrix is needed to convert points from LiDAR coordinate system to camera coordinate system.. This step is often referred to as LiDAR-Camera extrinsic calibration and it requires the camera intrinsics that was explained earlier in this chapter. There are different methods for extrinsic calibration of LiDAR and camera but the one which is commonly used is by taking planar checkerboard pattern as a fiducial target. Each LiDAR point has an intensity value which helps to pinpoint the corners of the checkerboard squares in LiDAR data. The next step is to convert the LiDAR point cloud into a 2D image plane using camera projection model. After obtaining a 2D image from 3D point cloud a point to point correspondence of 2D LiDAR intensity image and the camera image is found. For this purpose, an intensity image of interpolated LiDAR points is matched with an automatically detected checkerboard pattern in camera (Figure 2.8). This correspondence is used to calculate the transformation matrix between LiDAR and camera using a RANSAC algorithm [21]. Once the correspondence is found we get a

transformation matrix for converting between LiDAR and camera coordinate system. Figure 2.9 shows an example of projected camera image on LiDAR point cloud.

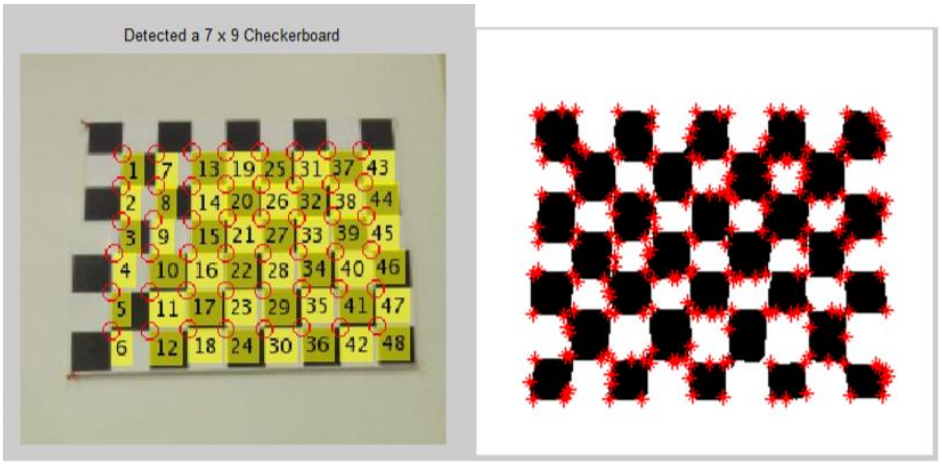


Figure 2.8. Correspondence between the camera image and intensity image (LiDAR) on a planar checkerboard pattern [3].

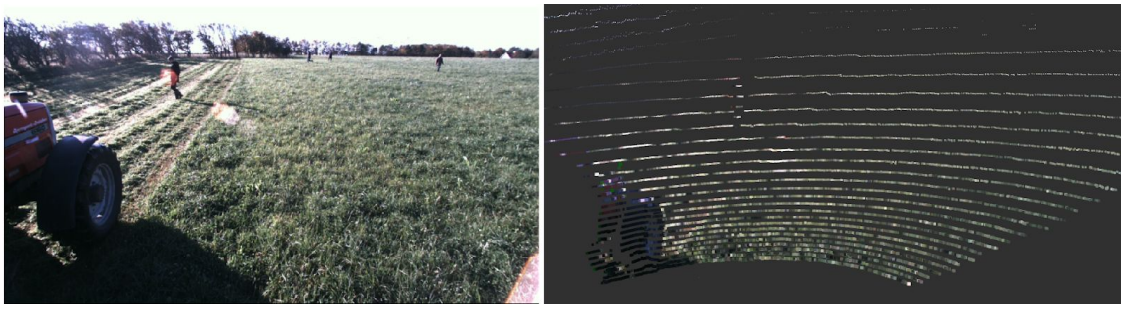


Figure 2.9. An example of our fusion node on FieldSAFE dataset [22], on left is the camera image and on right is the fused image with LiDAR points colored according to the corresponding pixels.

In the end, the main purpose of fusion in this study is to propagate the detection results from one modality to another thus achieving 3D object detection from the predictions obtained from state-of-the-art 2D object detection methods.

2.3 Data Fusion Challenges

Multi-modal data streams are different from each other in terms of data format, geometrical alignment and spatial resolution.
LiDAR data has an element of sparsity while imaging output is dense i.e. of high resolution. While LiDAR and camera extrinsic calibration is a result of a rigid body transformation as discussed previously, the transformation does not solve issues regarding the uncertainty of sensor data. In order to make efficient use of data fusion by heterogeneous sensors like camera and LiDAR, the sensors must be aligned in three ways:

- **Resolution-wise:** Both modalities should have similar spatial density or resolution. The LiDAR points are always more sparse than camera images, also

the far away LiDAR points are less dense than near points. This makes the resolution gap more for far away objects than near. (Figure 2.10).

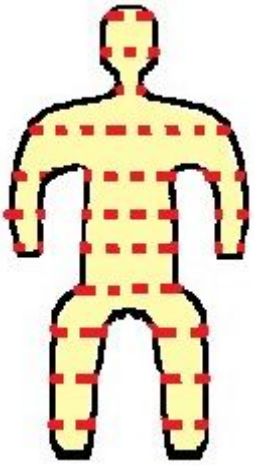


Figure 2.10: An illustration of spatial alignment problem where red dots represent LiDAR points and yellow highlighted area shows the resolution gap between LiDAR and camera which varies when human is near or far from LiDAR.

- **Spatially:** A pair of data points from both modalities correspond to a single point in world space (Figure 2.11).

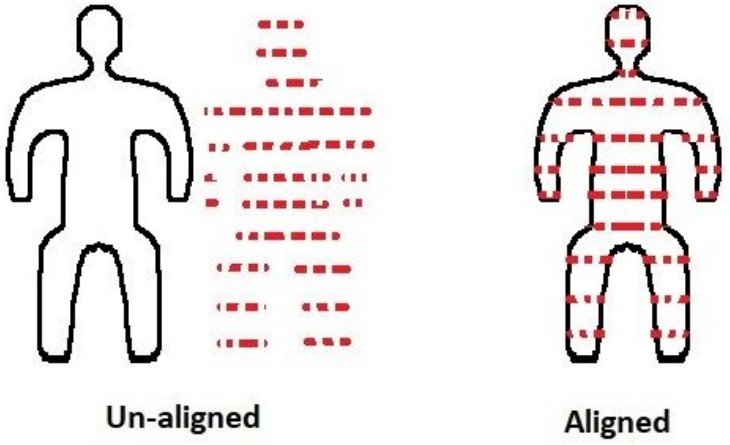


Figure 2.11: An illustration of spatial alignment problem. Red dots represent LiDAR points and human figure represents the camera output. On the left the LiDAR and camera are not horizontally aligned, on the right they are correctly aligned.

- **Temporally:** Two frames of different modalities showing a specific scene must have the same time-stamps (Figure 2.12).

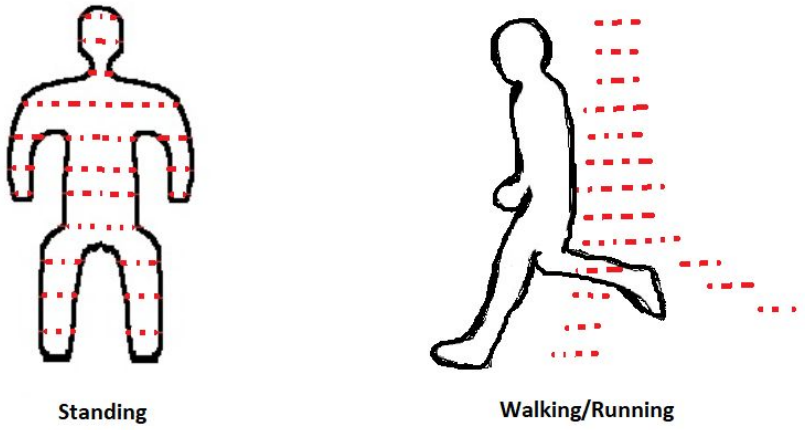


Figure 2.12: An illustration of temporal alignment problem where LiDAR points are perfectly aligned when human is standing and unaligned while human is walking/running due to delay in the acquisition.

Here, spatial alignment can be perfected with the efficient extrinsic calibration of the sensors, whereas resolution-wise alignment problem is often formulated as regression-based missing value prediction by interpolation on the measured data points [23]. Lastly, the temporal alignment problem is mainly caused in motion due to the long acquisition time and can be solved using time synchronization methods, i.e. triggering all sensors on common timing signals.

3 Related Work

Reliable object detection has been a core topic in the field of mobile robotics and computer vision. The research in object detection methods using camera and LiDAR fusion (specifically pedestrian detection) has seen considerable progress over the last decade. Most of the methodologies for 3D object detection rely on state-of-the-art 2D object detectors. For this reason, this chapter is divided into two sections: An overview of 2D object detectors followed by 3D detection approaches.

3.1 2D Object Detectors

Classical object detection methods had been popular among researchers before current deep learning based object detectors. [24] used Histogram of Oriented Gradients (HOG) to train an SVM classifier on KITTI dataset to extract feature-maps from RGB and depth images for pedestrian detection. Similarly [25] used multiple HOG detectors for detection and tracking of people using RGB-D data.

Since deep learning approaches took over as current state-of-the-art object detectors for imaging, researchers have been trying to exploit different network architectures for multi-modal perception in autonomous driving. Considering only RGB camera images, it is worth mentioning that convolutional neural networks (CNN) [26] have emerged as efficient algorithms for object detection. Object classification, detection and localization are mostly addressed by training a CNN to generate semantic segmentation or bounding boxes around predicted objects. Semantic segmentation makes predictions inferring labels for every pixel in the image thus giving a fine-grained understanding of the environment but is limited in giving instance level information of same classes. [27] introduced a flavour of CNNs named Faster-RCNN (an improved version of Fast-RCNN [28]), which is a region proposal network that predicts bounds and scores for multiple objects in an image. Faster-RCNN takes detection as a classification problem by generating object proposals before being sent to the classification head of the network. Mask-RCNN [29] is an extension to Faster-RCNN that predicts instance segmentation i.e. acquiring pixel level labels of each instance of the same classes. Mask-RCNN works in two stages i.e. Faster-RCNN for generating bounding boxes as ROI (Regions of Interest) and a second stage to generate segmentation mask inside bounding boxes of the predicted objects. Another category of 2D detectors poses detection as a regression problem. The most well-known is YOLO [30] which is a single neural network that predicts bounding boxes and class probabilities for different regions of an image. YOLO is comparatively faster than previously mentioned networks because it makes predictions with a single network evaluation, unlike Faster-RCNN and Mask-RCNN which work in multiple passes.

The speed dominance of YOLO over other existing methods makes it a perfect candidate for real-time object detection on camera images in our human detection pipeline.

3.2 3D Detection Approaches

Despite good performance, previously mentioned 2D object detectors are difficult to adapt for 3D object detection. Researchers have approached 3D object detection in various ways.

Image-based methods *[31], [32]* work on RGB images only i.e. they take shape or occlusions as priors and estimate 3D bounding boxes from 2D. A similar category of detectors includes handcrafted features or priors for proposal generation based methods that estimate the geometry shape of the objects [33].

Clustering based approaches perform ground removal in LiDAR space. [20] extracted ground from LiDAR points and estimated 3D bounding boxes from candidate clusters. [34] applied radially bounded nearest neighbour clustering of 3D LiDAR points for object detection as well as tracking in real-time. Clustering whole point cloud is very time consuming, computationally expensive and may not be efficient in real time. [35] downsampled the point cloud and made pedestrian detection more robust by adding information from projected proposed 3D human clusters into classification scores. [36] used a similar clustering based method by fusing LiDAR points with mosaiced images to extend the camera field of view to detect moving objects.

Monocular based proposal generation approaches leverage 2D detection methods by proposing regions of interest on the camera image. They became trendy from [37]. Frustum PointNets [9] is an inspiration of a similar approach where 2D region of interests are projected to 3D LiDAR space making 3D frustum proposals where a frustum represents 3D LiDAR modelled space that appear inside 2D bounding box on screen (often referred to as view frustum in computer graphics consisting a clipped 3D tilted pyramid with two parallel near and far planes). Proposed frustums, in the end, are segmented for bounding box regression using PointNets [38].

Bird's eye view-based approaches project LiDAR points to top-down view to train RPN (Region Proposal Network) over feature maps extracted from bird's eye view, LiDAR front view and camera images [39], [40]. [41] transformed feature maps extracted from camera image into top-down view image feature maps thus implicitly learning the depth of the image. Despite being ranked among one of the best approaches on KITTI dataset for detecting cars, these methods are not efficient in detecting smaller objects such as pedestrians etc.

3D based methods work only with single modality i.e. LiDAR data and are recent to compared to other approaches. [42] train a 3D object classifier using SVM on hand-crafted features of a point-cloud. [43] is an extension to the previous approach by replacing SVM with CNN. A more recent approach is voxelization of point clouds to train a generic 3D detection network thus removing the need for manual feature engineering [44], [45]. Depth based 3D bounding box prediction methods take top-down view of LiDAR data as a 2D image, project point cloud to the front view, obtaining a 2D point map then apply a fully convolutional network on the 2D point map and predict 3D bounding boxes from the convolutional feature maps [46].

This thesis combines two approaches discussed above, first 2D regions of interests are proposed the same way as they are proposed in Frustum PointNets [9] leveraging a mature 2D object detector. These regions are extruded to 3D LiDAR space but instead of filtering LiDAR points using PointNets we cluster them thus reducing the computational cost as clustering is cheaper than running full-blown neural network.

Furthermore, as we only cluster points lying inside the proposed extruded frustums, this also makes it computationally efficient instead of clustering whole point clouds which many clustering based approaches do.

4 Detection Pipeline

This chapter describes the specifications of the datasets used, followed by an overview of complete detection pipeline algorithm in ROS.

4.1 Dataset description

The detection methods discussed previously have their pros and cons depending on various factors such as labelling cost, real-time speed, dataset category etc. KITTI [10] benchmark ranking gives us a good in-depth guideline of different detection approaches but limits us with only urban scenarios. Here we take data as a specification approach and trim down the search for both 2D and 3D detection methods on the basis of off-road scenarios.

4.1.1 FieldSAFE Dataset

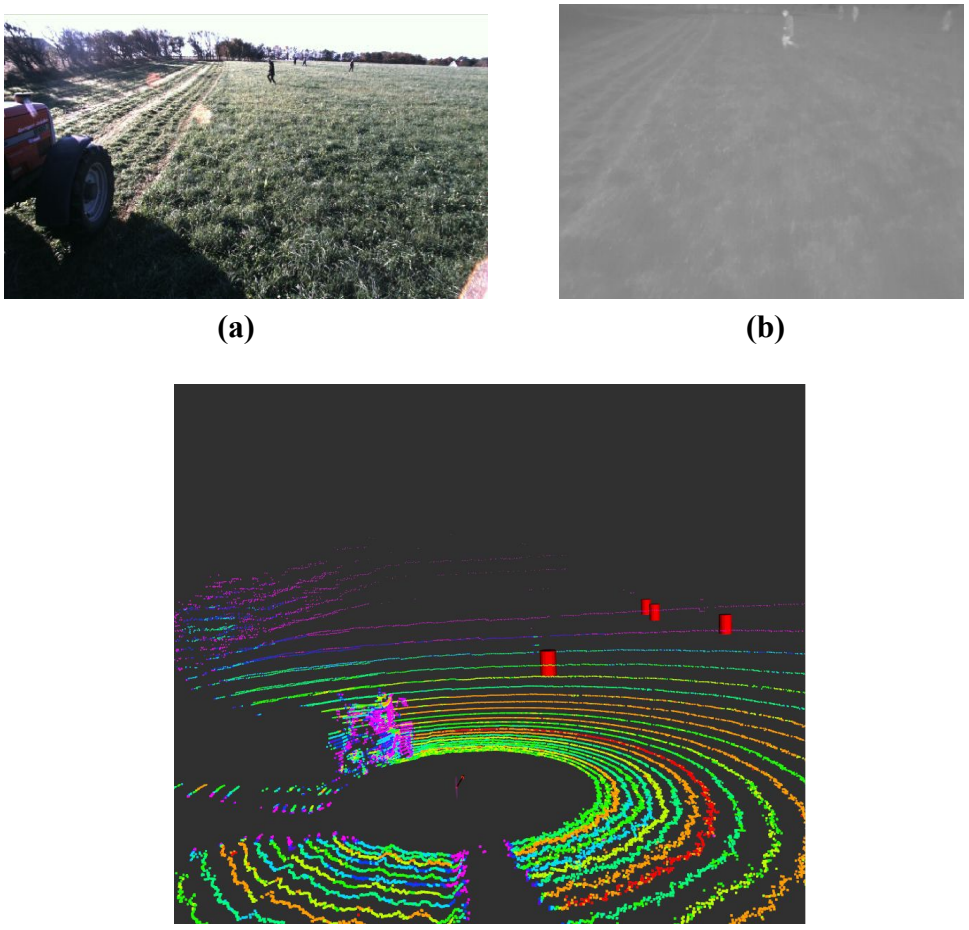
One of the key factors in environment perception is the type and specification of the dataset. There are a very limited number of available off-road datasets. Here we choose FieldSAFE [22] dataset which is mainly purposed for obstacle detection in agriculture scenarios and is, therefore, the closest match to off-road scenarios we are aiming for. The specifications of different sensor modalities in FieldSAFE dataset are summarized in Table 4.1.

Table 4.1. FieldSAFE dataset sensing modalities [22].

Sensor	Model	Resolution	FOV in degrees	Range in meters	Data-rate / fps
Stereo Camera	**Multisense S21 CMV2000**	1024 x 544	85 x 50	1.5-50	10
Web camera	**Logitech HD Pro C920**	1920 x 1080	70 x 43	n/a	20
360 camera	**Giroptic 360cam**	2048 x 833	360 x 292	n/a	30
Thermal camera	**Flir A65, 13mm lens**	640 x 512	45 x 37	n/a	30
LiDAR	**Velodyne HDL-32E**	2172 x 32	360 x 40	1-100	10
Radar	**Delphi ESR**	16 targets/frame 16 targets/frame	90 x 4.2 20 x 4.2	0-60 0-174	20 20

The FieldSAFE dataset comprises of almost two hours of raw multi-modal sensor data in grass field where all the sensors are mounted on a ploughing machine attached to a moving tractor. There are humans as dynamic obstacles which move around the field doing various poses while the sensors are recording the data. One of the pros of working with FieldSAFE dataset is the availability of almost 10 minutes of 3D human ground

truth coordinates which saves us the time to manually label human locations in 3D. Figure 4.1 shows an example of the modalities we used from FieldSAFE dataset.



(a) **(b)**

(c)

Figure 4.1. An example of FieldSAFE dataset multiple modalities where (a) shows stereo-left camera image, (b) shows the thermal image output for the same frame and (c) represents LiDAR point-cloud with red cylinders showing 3D human ground-truth locations

4.1.2 Milrem Robotics Dataset

We intend to deploy the detection pipeline of human detection on Milrem Robotics UGV, called Themis. The purpose of initially working with FieldSAFE dataset was to refine our perception module and then perform benchmarks over Milrem datasets to compare the results. Table 4.2 shows sensor specifications in Milrem dataset.

Table 4.2. Milrem dataset sensing modalities.

Sensor	Model	No. of sensors	Resolution /points	FOV in degrees	Range in meters	Datarate / fps
Camera	SF332X-10X-					

	NVIDIA 2 Mega Camera	2	1920x1208	120x60	n/a	30
LiDAR	Velodyne LiDAR VLP-16 PUCK	2	300,000 points, 16 beams	360 x 30	100	20

Figure 4.2. An example of Milrem dataset modalities where on left is the LiDAR point-cloud and on right is the corresponding frame output of RGB camera image.

Milrem dataset comprises of almost 10 minutes of raw multi-modal sensor data in snowy roads, snowy terrain with trees (Figure 4.2) where all the sensors are mounted on a UGV. There are humans as dynamic obstacles which move around the UGV and do different poses similar to what we have in FieldSAFE but this dataset has only limited amount of human ground truth locations (50 frames).

4.2 Detection Algorithm

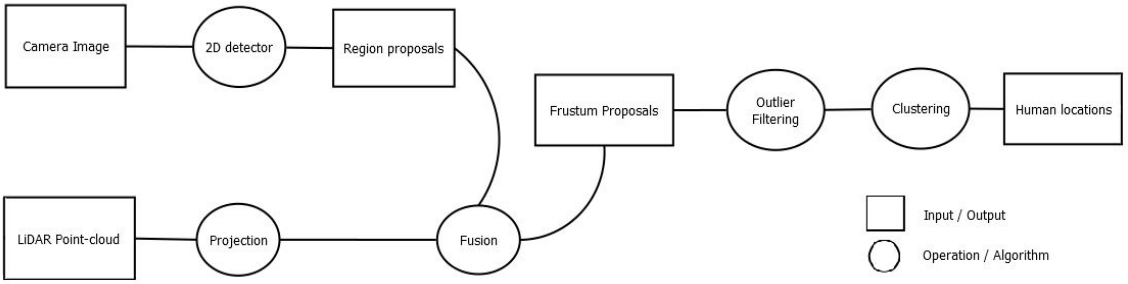


Figure 4.3. ROS detection pipeline.

The detection pipeline (Figure 4.3) can be divided into the following main steps:

4.2.1 Preprocessing

Before using raw images to generate region proposals, the images are preprocessed if required, e.g. images are rectified to remove lens distortion. Additionally, in case of

FieldSAFE dataset, thermal images were converted from 16-bit to 8-bit format and flipped in x, y-axis because of incorrect orientation.

4.2.2 2D region proposals

Once we get undistorted images the first step is to propose 2D regions on images. There are two ways of doing it:

- Bounding boxes
- Object masks

As discussed in the previous chapter that at this stage we need a mature 2D object detector. In our pipeline, we used bounding box based region proposals and tried two pre-trained neural networks i.e. YOLO v2 and v3 for generating 2D region proposals. Figure 4.4 shows an example of predicted bounding boxes from YOLO v3.

Figure 4.4. An example of 2D bounding boxes proposed by YOLO v3 on FieldSAFE stereo left RGB camera image. Here, the bounding boxes are stretched horizontally and contracted vertically because of the reasons explained in section 4.4.1.

4.2.3 Projection, Fusion & Frustum proposals

While the bounding box proposals are generated from camera images, at the time of raw image acquisition, the LiDAR point cloud data is also acquired in parallel. First, the LiDAR points are transformed from LiDAR coordinate system to camera coordinate system using the transform acquired through extrinsic calibration (discussed in chapter 2). Then, using projection matrix extracted from camera information, the point-cloud with all the points having z value (forward-backward direction) greater than 0 are projected to an image plane of the same camera to which it is intended to be fused (Figure 4.5). The reason for removing points with z less than 0 is to get rid of the unnecessary transformations of LiDAR points which lie behind the camera plane. After projection of point-cloud data on the image plane, the next thing is to perform two-step filtering i.e. first filter out all the LiDAR points which exceed image height and width giving us only those LiDAR points which have correspondence with camera image and then filter all LiDAR points which lie inside proposed regions.

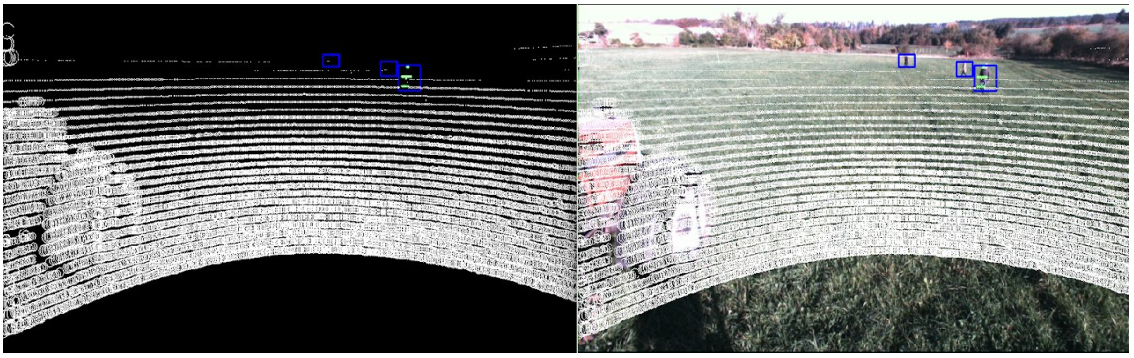


Figure 4.5. Projection of LiDAR points to camera image plane.

Once we get the filtered points which lie inside 2D region proposals, we implicitly get frustum proposals i.e. all filtered points after extrusion makes 3D frustums (Figure 4.6) of the regions where the humans are most likely to be found considering we have accurate prediction by the 2D detector.

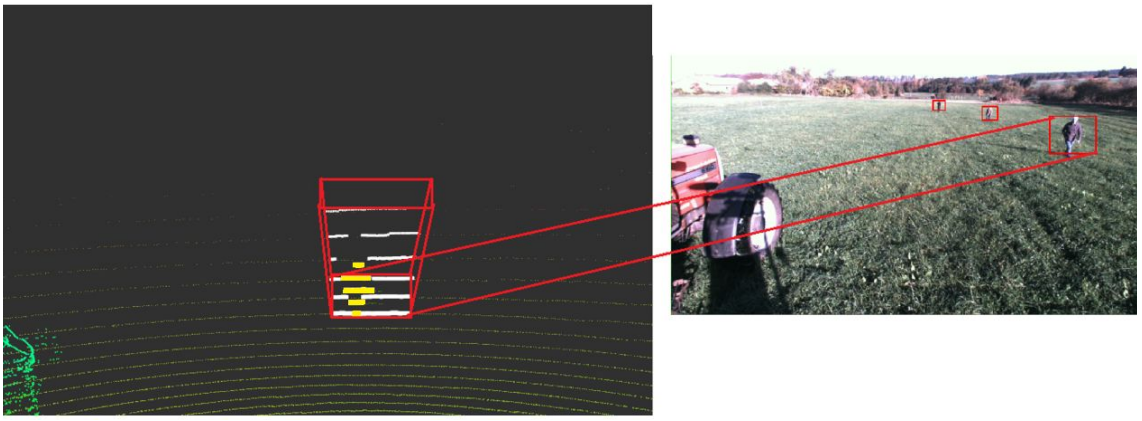


Figure 4.6. An illustration of extruded filtered LiDAR points as proposed frustum, where yellow pixels in LiDAR show human but is cluttered by background points.

4.2.4 Filtering & Clustering

The frustum proposals acquired in the previous step have LiDAR points which happened to lie inside proposed frustums but are not actually part of human figures. If we would average over coordinates of all these points to get human location, it would be inaccurate due to the scattering of points in the background. In order to localize the prediction of human in proposed frustums, first, we filter the frustums i.e. consider only those frustum points which lie inside a specified benchmark radius e.g. 30 meters around the UGV. This removes all the points which are too far away and yet marked as human due to bounding box filtering. Even after filtering the points like this, human location can still be ambiguous in scenarios where human is near the vehicle and background points are spread up to the benchmarking radius. Figure 4.6 demonstrates such a scenario where human is near to the vehicle and has cluttered background points in its frustum.

This problem can be addressed in many ways such as calculating the point density of the top-down 2D LiDAR points per frustum, as human or any proposed object will have

dense points around it in a top-down view. We addressed this problem using DBSCAN [47] algorithm. The idea here is to cluster remaining frustum points according to the distance between the points in top-down view.
Once we cluster the points inside a frustum we still have to find the candidate human cluster because there can be zero or many clusters inside one proposed frustum. At this point, we take the nearest cluster as human candidate (Figure 4.7) for safety reasons. This choice is a bit prone to foreground occluder errors where other objects can sometimes be falsely predicted as humans i.e. in scenarios where humans are partially occluded by some other object. But for safety purposes, a sudden halt of the UGV is preferable to fatalities. After human candidate cluster is filtered, the candidate cluster points are averaged to make cluster centers of the human predictions (Figure 4.8, Figure 4.9).

Figure 4.7. An illustration of clustering results using DBSCAN where green cylinders are human ground truth and white pixels are clustered human points.

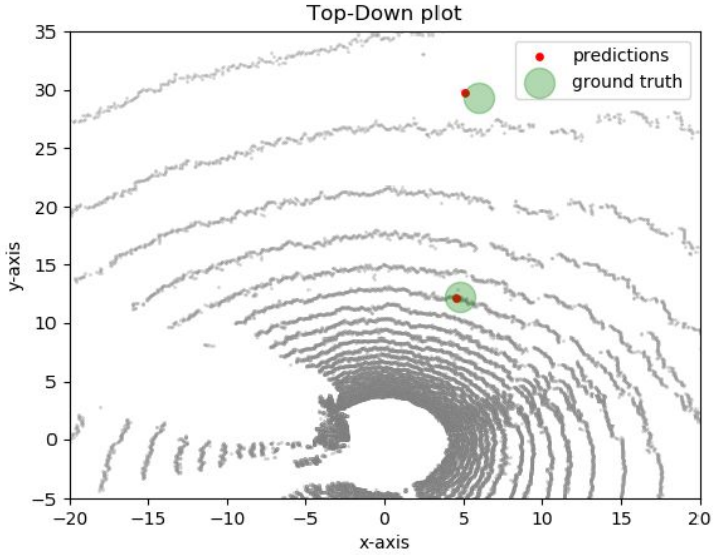


Figure 4.8 A Top-down plot of figure 4.7 with red dots representing averaged cluster centers and green circles are human ground truth scaled to the tolerance in meters.

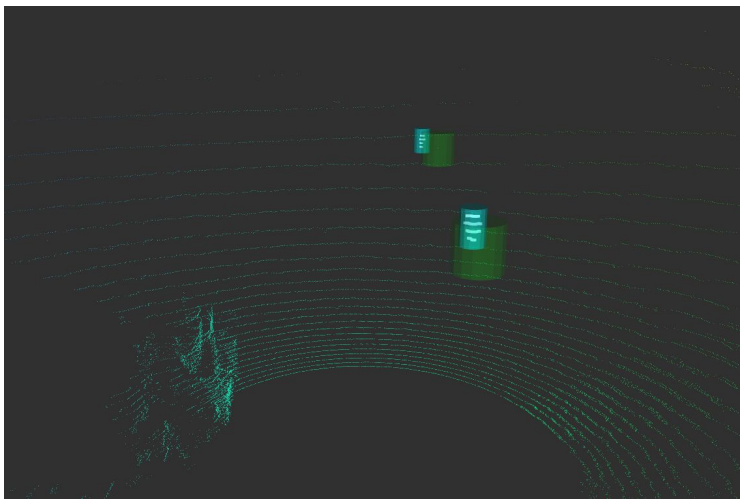


Figure 4.9. Human predictions on averaged cluster centers in 3D, where light-blue cylinders represent predictions and green cylinders represent ground-truth.

4.3 Baseline Algorithms (Mask-based)

Our detection pipeline is a follow-up of some baseline approaches which use object masks as 2D proposed regions instead of 2D bounding boxes. The initial stage involved working with thermal images and segmentation images. Additionally, YOLO predictions were also used as 2D masks. The detection flow is very similar to our current pipeline with the exclusion of frustum proposals.

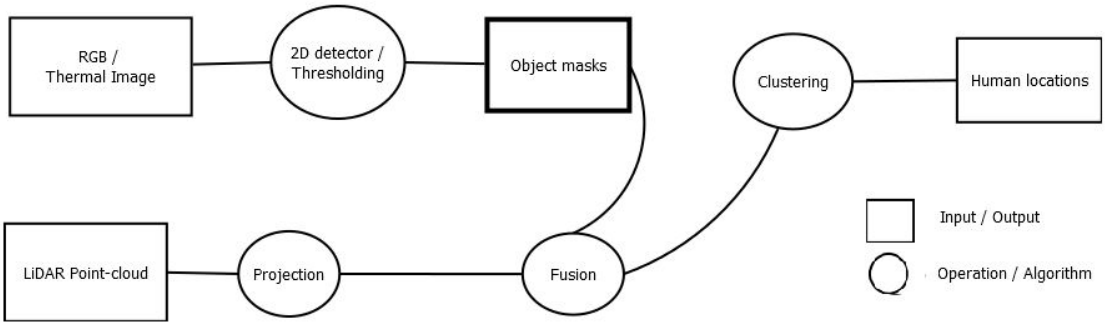


Figure 4.10. Baseline detection methods.

Figure 4.10 shows that we get the object mask either from a thermal image or from an RGB camera image segmentation result. This sub-divides the object mask retrieval into two workflows.

4.3.1 Thermal Thresholded Masks

In this workflow raw image from thermal camera is taken and preprocessed as explained earlier in this chapter, Additionally, the image is scaled into min and max range calculated over all thermal images as min-max per image can lead to different ranges of thresholding thus making it difficult to have consistent parameters of thresholding.

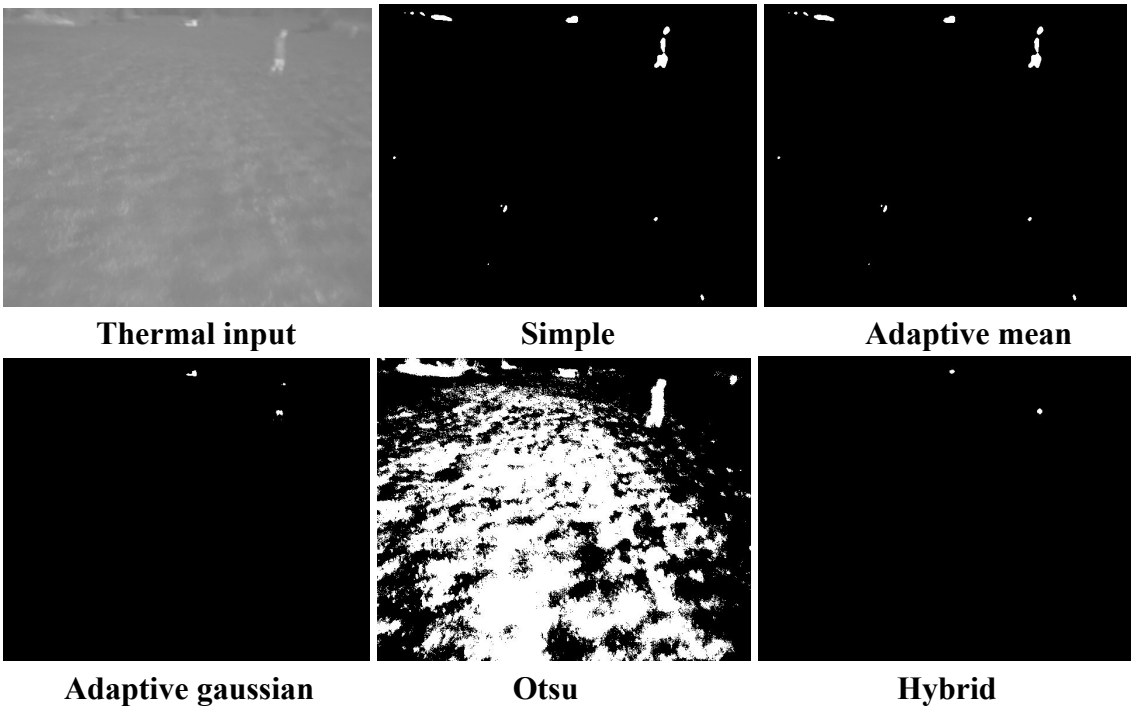


Figure 4.11. An example of thermal image thresholding methods used.

We used a simple thresholding that checks if a pixel has a value greater than a threshold and assigns it a value of 255 (white), otherwise assigns the value of 0 (black) thus generating a masked image (Figure 4.11) [48]. We also experimented with adaptive thresholding methods shown in Fig 4.11, but these did not work better than simple fixed thresholding.

4.3.2 Segmentation Masks

Thermal camera images are useful in low-light conditions but are very much affected by varying temperature of the environment. In this workflow, we tested ICNet [49] and Deeplab [50] segmentation networks to propose 2D regions as object masks. Deeplab Xception (Figure 4.12) gave the best performance among them.

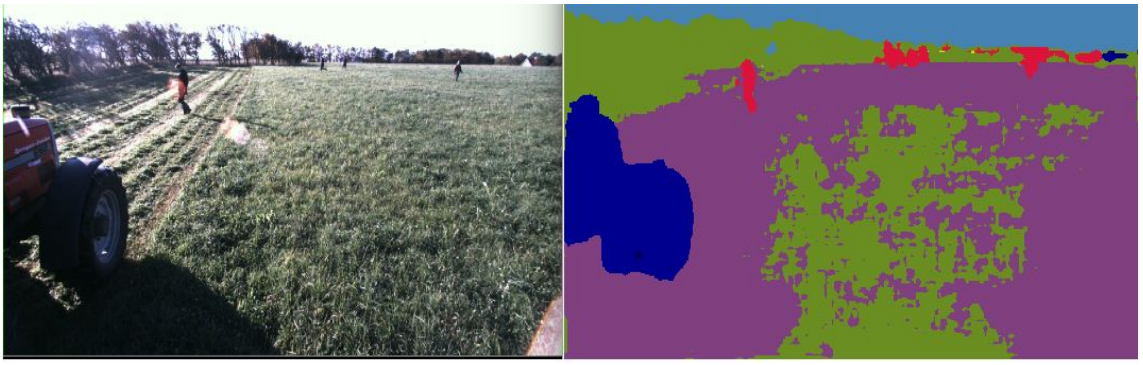


Figure 4.12: An example of human masks predicted from Deeplab Xception where red regions show human predictions.

4.3.3 Mask-based YOLO

Although a perfect segmentation can give precise human masks but tackling the alignment problem in 2D masks is tricky. In this step, we used YOLO bounding box to create 2D masks instead of segmentation masks. This method is very close to the main algorithm i.e. in this case creating object masks loses instance information while frustums in main algorithm retain instance information of each bounding box.

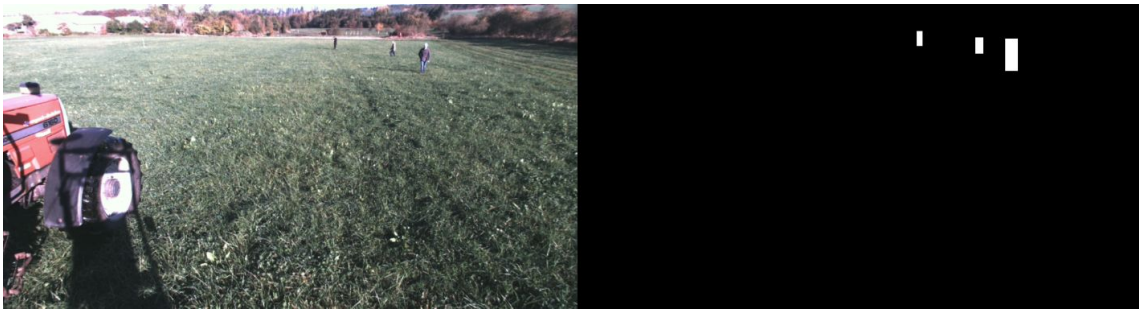


Figure 4.13: An example of human bounding box masks predicted from YOLO v3 where on left is the RGB image and on the right is the predicted human bounding masks.

4.3.4 Clustering Mask Proposals

In contrast to frustum based approach which uses nearest cluster as human, we consider every cluster in baselines as human, the reason being no information of predicted human count from the 2D mask. Clustering parameters (discussed in later sections) help to discard clusters which lie below a certain minimum size threshold. The remaining clusters are averaged to acquire a cluster center for each cluster. These cluster centers represent the predicted location of the human. Even after filtering clusters, we are still left with a considerable amount of false positives in our predictions thus reducing the performance of detection. Figure 4.14 shows an example of 2 false predictions against one human ground truth due to mask based clustering.

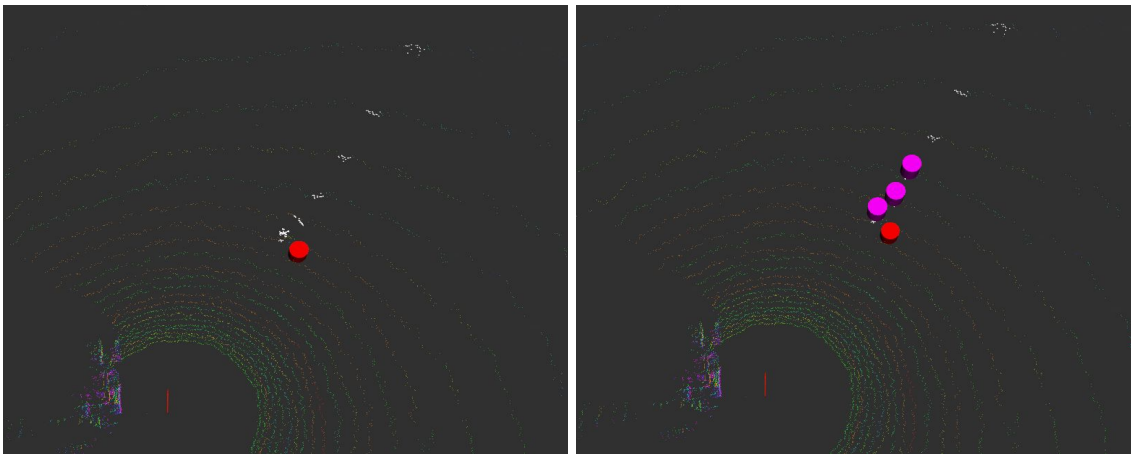


Figure 4.14: An example of mask based clustering from top-down view on FieldSAFE dataset where on left is the projected mask points and on right is the predicted human cluster centers. Here red cylinder represent human ground truth and purple cylinders represent predicted human cluster centers.

4.4 Detection Pipeline vs Baselines

4.4.1 Mask vs Bounding boxes

Segmentation networks are comparatively slower than bounding box based 2D detectors but can give pixel level labels per object. The question of choosing segmentation networks depends on the use case scenario i.e. are pixel level labels even required or we can settle with bounding boxes to have certain regions marked as humans? Additionally, we have to take in account the temporal alignment problem discussed in previous chapters i.e. what if the projected human object mask doesn't align properly with corresponding LiDAR points (Figure 2.10, Figure 2.12)?

The experiments showed that object masks did not give benefit over bounding boxes. The reason is that because of imperfect time alignment, the object masks may miss the person entirely or misplace them. Extending bounding box horizontally improved the results (Figure 4.15). On the other hand reducing the bounding box vertically decreased the number of mis-placements due to ground points, picked up by the nearest cluster heuristic.

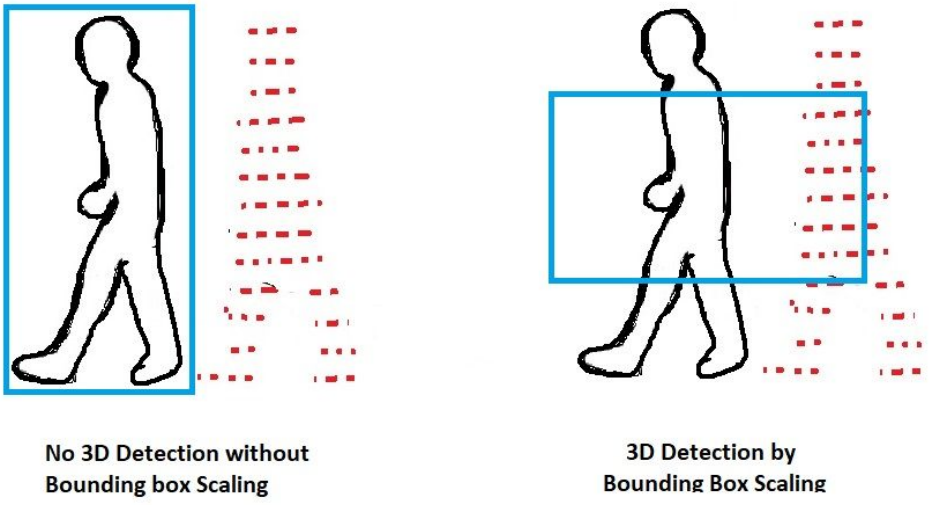


Figure 4.15. Scaling of 2D bounding box helped with misaligned human LiDAR points due to temporal lag while walking.

4.4.2 Instance vs Without Instance Information

Having no instance information with masks affects the clustering computation i.e. all clusters are considered as humans and there is no instance information to separate them. Without instance information, we cannot get a definite count of how many humans are detected and must rely on minimum number of points in cluster to filter out human clusters. We could have used instance segmentation masks using Mask-RCNN but as discussed in previous chapter, Mask-RCNN doesn't fit into our pipeline as a real-time

model due to a low speed of 5 frames per second. Even if we could tune the speed of Mask-RCNN, there would still be the misalignment problem between 2D instance mask and LiDAR human points as described in the previous section.
Here, bounding box based method with instance information gives the best combination of all. The scaled extruded bounding boxes, firstly, help to resolve misalignments as discussed above, secondly, they already tell how many humans should be expected in 3D predictions. This reduces clustering cost i.e. instead of clustering all points at once, we cluster each frustum separately to find one candidate human cluster from each frustum. Moreover, these methods are faster than instance segmentation based approaches.

4.5 Stable Benchmarking and ROS

ROS is publish-subscribe architecture in which multiple nodes behave as subscribers and publishers to pass processed information to each other as messages. It is one of the mainstream robotics middlewares used to work with sensors and low-level device control. The detection pipeline explained in the previous section is built on top of it.

ROS provides a feature to record real-time sensor data and store it into a BAG file. Every message in BAG file has a time-stamp that tells ROS when to publish a message during playback. This helps to achieve a consistent playback of the recorded streams. ROS libraries use computer's clock as source time but for playback of logged data, it is desirable to use simulated clock which helps to slow or accelerate time over system's clock [51]. Although, each message in the BAG file has defined time-stamps, the benchmarking of the pipeline parameters for both FieldSAFE and Milrem datasets showed inconsistent results. Following can be the potential causes of such inconsistency:

- Random delays in data acquisition from BAG file during playback.
- Random delays in processing (algorithm computation) even at lower BAG speed.
- Inherent asynchronous nature of ROS architecture.

ROS has its own Time Synchronization message filter that helps to alleviate such inconsistency problems by making sure that multiple messages arrive in a time-synchronized way to a particular node. In our case, we used time synchronizer while extracting data for benchmarking the 3D predictions i.e. the LiDAR point clouds, camera images, ground truth and human prediction locations are retrieved simultaneously.

4.5.1 External Setup

While time synchronization somewhat helps in solving the inconsistency problem during playback of the BAG file, yet the results still differ in each benchmark run on the same BAG. A better solution here is to perform benchmarking externally i.e. independent of ROS environment. But even then, we are still dependent on using ROS environment for extraction of sensor modalities and ground-truth of each frame into some files that can later be read by external benchmarking scripts.

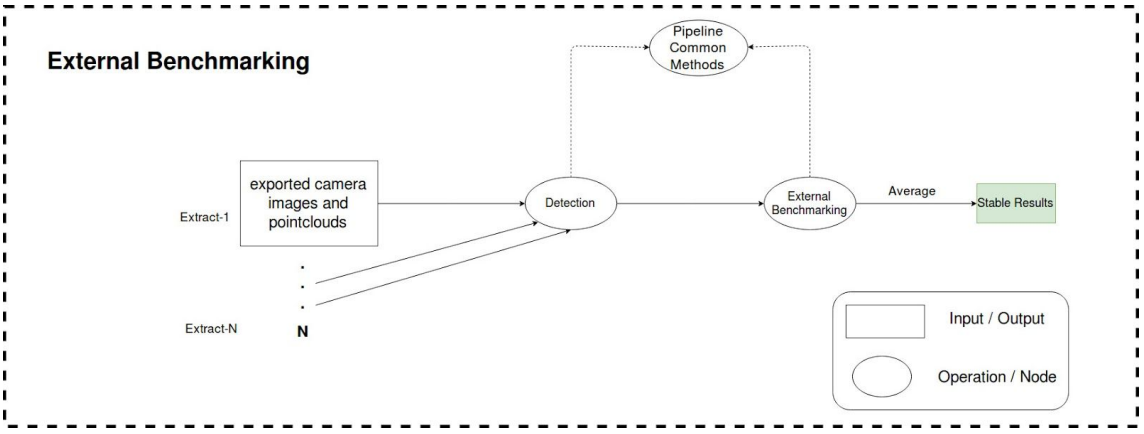


Figure 4.16. External Benchmarking Setup

For this, we created an external setup for benchmarking as shown in Figure 4.16. Here instead of acquiring predictions within ROS, we extract the intended modalities along with ground truth on a very slow playback rate to get less variance in extracted frames. The data is extracted into 3 formats per frame i.e. PCD files for LiDAR point-cloud, PNG for camera images (stereo or thermal) and NPZ for packing human ground truth locations and transformation matrices to transform between two modalities. Once we extract a dataset from ROS, all the benchmarking is performed on this dataset. This gives consistency in human predictions thus making a reliable benchmarking pipeline outside of ROS.

4.6 Parameter Description

The main purpose of making a stable benchmarking setup is to fine tune the pipeline parameters which affect the performance of 3D detection. These parameters are categorized into three main categories which are as follows:

4.6.1 ROS parameters

ROS parameters come into play during the extraction of the dataset. Tuning these parameters stabilizes the extraction of multiple datasets of the same BAG. There are two ROS parameters which affect the stability.

4.6.1.1 Play rate

The play rate controls the playback speed of the BAG file during dataset extraction. For example play rate 2 means multiplying the publishing frequency of each message inside the bag with 2 i.e. doubling the speed of playback.

4.6.1.2 Time Synchronization method

Time synchronization as discussed earlier helps to synchronize the sensor messages coming from different modalities on the basis of their time-stamps. Table 4.3 summarizes the synchronization methods used.

Table 4.3. Time synchronization methods.

Method	Type	Description
Approximate	ROS Time Synchronizer	Synchronizes message callbacks by parameter called **slop**. The slop value tells how much difference between the two message timestamps is tolerable. Additionally, it also has **queue_size** parameter that tells how many messages can be queued from each source for timestamp comparison.
Naive Cache	Custom	Proceed when all messages are available to process i.e. there is no boundness on time stamp. Here we cache and use the last frame of previously received messages until a fresh frame is received.
Naive Least Frequent	Custom	Proceed when all messages are available to process i.e. there is no boundness on time. Here we discard a frame once it is processed and wait until all messages are available to process. All frame checks are done inside the callback of the sensor modality with the **lowest** frequency.
Naive Most Frequent	Custom	Proceed when all messages are available to process i.e. there is no boundness on time. Here we discard a frame once it is processed and wait until all messages are available to process. All frame checks are done inside the callback of the sensor modality with the **highest** frequency.

4.6.2 Benchmarking parameters

Benchmarking parameters define the boundaries of radii during the evaluation of results. There are two benchmarking parameters in our pipeline:

1. **Benchmarking Radius:** Refers to the radius in meters with origin at LiDAR, inside which we evaluate our benchmarking results. This is motivated by the fact that sensor accuracy degrades over a distance, but we are actually not interested in detecting infinitely far objects. It is important to detect the objects that are within braking distance of the UGV.
2. **Tolerance:** Refers to the radius in meters with origin at ground truth human location in LiDAR frame. This tells how much distance is tolerable between 3D human ground truth and nearby human prediction. This is partly motivated by the fact that our benchmarking datasets (especially FieldSAFE) do not have perfect ground truth and certain amount of prediction errors are expected.

4.6.3 Algorithm parameters

Algorithm parameters define values which affect the performance of localizing human predictions in 3D. Table 4.4 summarizes the algorithm parameters with their usage in detection pipeline.

Table 4.4. Summary of algorithm parameters.

Parameter	Algorithm	Usage in Modality	Description
Method	2D Detection	RGB and Thermal Image	2D detection methods include, YOLOv2, YOLOv3, Deeplab Xception, Deeplab_Mobilnet, and thresholding. Where thermal images are only used for thresholding.
Epsilon	DBSCAN	LiDAR	The maximum distance in meters between two points

		point-cloud	so that they can be considered in the same cluster.
Min Samples	DBSCAN	LiDAR point-cloud	Min number of points required to make a cluster.
Scale_X	YOLO	RGB Image	Scaling factor along the width of the predicted YOLO bounding box.
Scale_Y	YOLO	RGB Image	Scaling factor along the height of predicted YOLO bounding box.
Thermal Threshold	Filtering	Thermal Image	Pixel threshold value between 0-255 for filtering 8-bit thermal image.

4.7 Evaluation Metrics

The benchmarking we performed uses *Precision*, *Recall*, and *F1-Score* as evaluation metrics. Before explaining these metrics one should understand the underlying concepts of the following:

- **True positives (TP)**: A total count of predictions which lie inside the tolerance radius of the intended human ground truth points.
- **False positives (FP):** A total count of predictions which lie outside the tolerance radius of human ground truth points
- **False negatives (FN):** A total count of the human ground truth points which had no predictions around them.

The definitions of precision, recall and f1-score are as follows:

- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

Where precision is the ratio of correctly predicted humans to the total number of predicted humans or in other words it tells how much can we rely on the predictions made (to be actually humans). Recall is the ratio of correctly predicted humans over actual number of humans or in other words this helps to identity how often actual humans are correctly predicted. Lastly, F1-score is the weighted average of Precision and Recall. F1-Score gives balanced weightage to both precision and recall. We could also incorporate F2 score to emphasize more on recall. This is due to the fact that the safety of humans is more important than mispredictions which were not actually humans. In other words we don't want to miss any humans from the environment i.e. the FN count should be least compared to FP. At the same time the precision shouldn't be too bad either because we don't want the UGV to halt too much because of many wrong human predictions. For now, we keep F1-score only to evaluate or detection pipeline.

5 Benchmarking and Performance Analysis

This chapter summarizes the benchmarking results on different pipeline parameters and analyses them for performance optimization.

5.1 Benchmarking Results

Similar to the previous section, the results of benchmarking human detection over pipeline parameters are divided into different categories depending on the type of parameters. In this section, we first summarize the main results showing the best parameters over the main algorithm vs baselines and then summarize the benchmarking results of each parameter with their performance impact analysis.

5.1.1 Dataset Benchmarks

This section shows the final benchmark results performed with the best parameters on both FieldSAFE and Milrem dataset. FieldSAFE dataset (5041 frames) was split into 50% test and 50% validation, the validation set was used for parameter tuning and test set for final results. To add diversity in validation and test split, we divided the complete dataset into 4 splits and chose split 1 and 3 as validation set whereas split 2 and 4 as the test set. Milrem dataset only had ground truth available for 50 frames. We did not split it, because parameter tuning was done on FieldSAFE anyway.

5.1.1.1 FieldSAFE

Table 5.1: Benchmark on FieldSAFE Validation Set (2520 frames)**.**

Method	Camera	Min Samples	Epsilon	Precision	Recall	F1-score
YOLO v2 Frustum	Stereo left	2	0.2	78.1%	48.3%	59.7%
YOLO v3 Frustum	**Stereo left**	**2**	**0.2**	**87.9%**	**59.7%**	**71.1%**

Table 5.2. Benchmark on FieldSAFE Test Set (2521 frames)**.**

Method	Camera	Min Samples	Epsilon	Precision	Recall	F1-score
YOLO v2 Frustum	Stereo left	2	0.2	82.4%	35.9%	50.%
YOLO v3 Frustum	**Stereo left**	**2**	**0.2**	**91.6%**	**47.8%**	**62.9%**

Table 5.1 and 5.2 shows final benchmark results on FieldSAFE validation and test sets over the best parameters listed in the next section. The results show that YOLO v3 outperformed its predecessor v2 which is expected as v3 is considerably better in performance [52]. The precision in both validation and test set is greater than recall because YOLO is mostly missing humans in 2D images rather than giving more false

predictions. Essentially, 3D prediction results imitate 2D predictions. Here, validation set has better evaluation results than test set. One of the main reason for this could be that the parameters were tuned too much on validation set thus leading to some overfitting, but on the other hand there is some slight shift in dataset too i.e. validation and test set doesn't have similar number of human ground truth distribution and poses. Additionally, YOLO is not good in recognizing humans lying on the ground or doing uncommon arbitrary poses. The test set had comparatively more such cases than validation set.

5.1.1.2 Milrem

Table 5.3. Benchmark on Milrem extract with 50 frames**.**

Method	Camera	Min Samples	Epsilon	Precision	Recall	F1-score
YOLO v2 Frustum	Mono RGB	2	0.2	89.7%	71.7%	79.7%
YOLO v3 Frustum	**Mono RGB**	**2**	**0.2**	**95.9%**	**81.4%**	**88.1%**

Table 5.3 shows benchmark results on Milrem dataset using frustum based detection with the best parameters. The results are comparatively better than FieldSAFE mainly because the humans were very close to the UGV and therefore easily detected. Also the results might be bit optimistic because of small number of benchmarked frames.

5.1.2 Frustum Based Detection vs Baselines

In this section, we summarize the results on the best parameters over both main algorithm and baselines on FieldSAFE example extracted dataset. The best-fixed parameters used during benchmarking are as follows:

- **Time Sync Method:** Naive most frequent
- **Play rate:** 0.2
- **Scale_X (Frustum):** 1.5
- **Scale_Y (Frustum):** 0.5
- **Benchmarking Radius:** 30
- **Tolerance:** 2
- **Thermal Threshold:** 160

Some parameters are specific to the type of detection methods e.g. Scale_X and Scale_Y are only used in frustum based detection methods, whereas thermal threshold is only applicable while filtering thermal images. The extracted datasets contain **387** stereo left and **597** thermal camera images. Due to time constraints, the comparison with baselines is made on a smaller version of FieldSAFE dataset.

Table 5.4. Main algorithm vs baselines with their best parameters on FieldSAFE dataset.

Method	Camera	Min Samples	Epsilon	Precision	Recall	F1-score
YOLO v3 Frustum	**Stereo left**	**2**	**0.2**	**96.7%**	**73.7%**	**83.7%**

YOLO v2 Frustum	Stereo left	2	0.2	77.1%	59.3%	67.1%
Yolo v3	Stereo left	11	0.3	97.3%	39.3%	56.0%
Yolo v2	Stereo left	11	0.3	60.6%	49.2%	54.3%
Deeplab Mobilenet	Stereo left	11	0.3	22.8%	45.8%	30.4%
Deeplab Xception	Stereo left	10	0.3	55.1%	49.4%	52.1%
Thermal Thresholding* (different extract)	Thermal	2	0.4	47.2%	69.4%	56.2%

Table 5.4 summarizes the detection results with best parameters over FieldSAFE example dataset. The highlighted methods in the above table are part of our frustum based main detection pipeline where YOLO v3 and v2 are used as 2D region proposal networks to generate 3D frustum proposals. YOLO v3 frustum based detection outperformed all other baseline methods in F1-score which is our main evaluation metric (Figure 5.1). Overall, instance based methods (frustums) work better than other mask based methods. Also, bounding boxes from YOLO v2/v3 gave better performance than pure object masks from Deeplab Xception and Mobilenet.

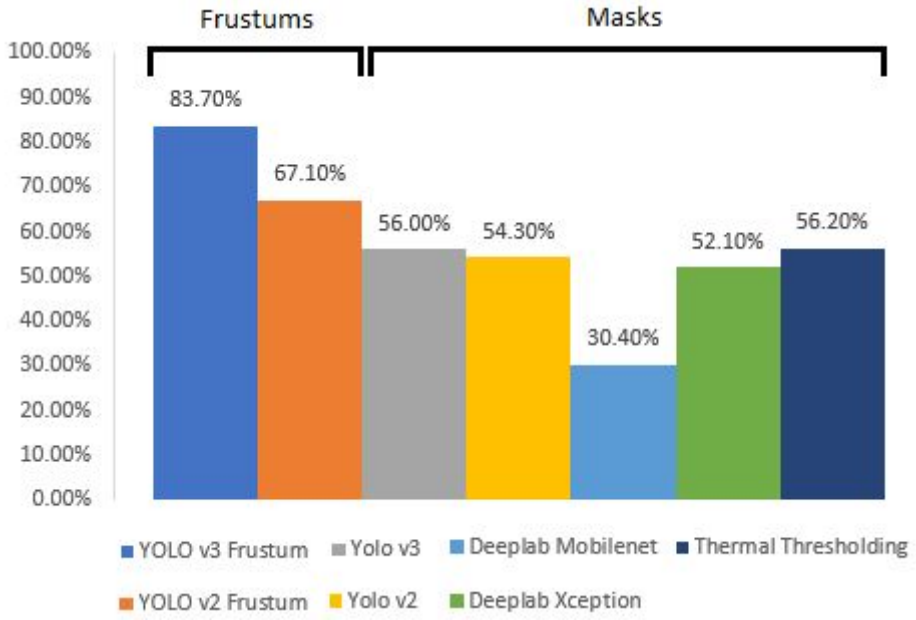


Figure 5.1. Human detection F1-score performance comparison chart of frustum based detection vs baseline approaches on example FieldSAFE extracted datasets.

5.1.3 Algorithm Parameters (FieldSAFE)

In this section, we summarize the benchmarking results to fine-tune algorithm parameters and discuss how tuning algorithm parameters improves detection performance.

5.1.3.1 Epsilon and Min Samples

Table 5.4 shows performance impact while tuning min samples and epsilon clustering parameters over simple YOLO v3 method on smaller version of FieldSAFE dataset.

Table 5.4. Epsilon and Min Sample detection performance impact.

Epsilon	Min Samples	Precision	Recall	F1-score
0.4	4	98.1%	69.6%	81.4%
0.4	3	97.7%	72.3%	83.1%
0.4	**2**	**96.9%**	**73.8%**	**83.8%**
0.3	4	98.1%	69.6%	81.4%
0.3	3	97.7%	72.3%	83.1%
0.3	**2**	**96.9%**	**73.8%**	**83.8%**
0.2	4	98.1%	69.6%	81.4%
0.2	3	97.7%	72.3%	83.1%
0.2	**2**	**96.9%**	**73.8%**	**83.8%**
0.1	4	97.9%	49.5%	65.7%
0.1	**3**	98.0%	68.4%	80.6%
0.1	2	**96.8%**	**72.2%**	**82.7%**

- Increasing min samples helps if the detected object is near the LiDAR as we get dense LiDAR points near the point-cloud origin. The farther we go the less nearby points we get due to sparsity of LiDAR points. A higher min samples value will miss far away humans while a very low value will increase the number of false positives near LiDAR origin. In other words, increasing min samples increases precision at the expense of recall while decreasing min samples increases recall, at the expense of precision.
- Epsilon value shows quite the opposite behaviour to that of min samples. A higher epsilon value increases the number of false positives in nearby detection but epsilon shows consistent results when its value is greater than or equal to 0.2 with min samples 2. On the other hand, a very low epsilon value may miss far away human points to be counted in the same cluster.
- In general, min Samples and epsilon are affected by the distance of the object as point cloud gets denser in near LiDAR beams.

5.1.3.2 Bounding Box Scale

Bounding box scale x and y are interestingly one of the main parameters that boost our detection performance. As discussed in the previous chapters, the major problem with 2D driven 3D detection is the alignment of the modalities. Most of the temporal alignment problems between LiDAR and camera are caused during the motion of an object. It is practically illogical for humans to move along the vertical axis of the camera image i.e. they move along horizontally (left-right or right-left) in front of a vehicle. This means that we mostly suffer from temporal misalignment in the horizontal axis of the LiDAR. To detect misaligned human points in 3D we scale up the width of the proposed frustum by increasing the x scale of the predicted 2D bounding box to some factor. Additionally, we reduce y scale of the 2D bounding box to reduce the height of 3D frustum which avoids misclassification of ground points as humans.

Table 5.5. Bounding box scaling performance impact.

Scale_X	Scale_Y	Precision	Recall	F1-score
1	1	94.8%	72.9%	82.4%
1.2	0.8	96.1%	73.6%	83.3%
1.5	**0.5**	**96.9%**	**73.8%**	**83.8%**

Table 5.5 shows benchmarking results on bounding box scaling parameters on an example FieldSAFE extracted dataset with 387 stereo-left images.

5.1.4 Benchmarking Parameters

This section summarizes the performance impact over LiDAR Benchmarking radius and GPS error tolerance.

Table 5.6. Benchmarking parameters performance impact onFieldSAFE dataset using YOLOv3 (387 images).

Benchmark radius	Tolerance	Precision	Recall	F1-score
10	4	96.2%	100.0%	98.0%
10	3	95.2%	100.0%	97.6%
10	2	93.5%	100.0%	96.7%
10	1	91.3%	100.0%	95.5%
20	4	98.8%	84.1%	90.9%
20	3	98.6%	83.0%	90.1%
20	2	98.6%	81.9%	89.5%
20	1	85.8%	74.1%	79.6%
30	4	99.2%	76.2%	86.2%

30	3	99.2%	75.7%	85.9%
30	**2**	**96.9%**	**73.8%**	**83.8%**
30	1	66.2%	51.6%	58.0%
40	4	98.9%	66.2%	79.3%
40	3	98.9%	66.1%	79.3%
40	2	90.3%	60.6%	72.5%
40	1	55.7%	37.4%	44.8%
50	4	98.3%	56.1%	71.5%
50	3	97.3%	55.6%	70.7%
50	2	87.3%	49.8%	63.4%
50	1	52.6%	30.1%	38.3%

The benchmarking parameters are not something to fine tune for improving detection performance, in fact, they define boundaries to our benchmarking evaluations. In our detection pipeline, we kept benchmarking radius to be 30 meters which should be enough for the UGV to come to full stop. This makes it a safety parameter instead of a performance booster. The tolerance value here compensates for FieldSAFE GPS ground truth errors i.e. we estimated the maximum discrepancy between LiDAR and GPS human location to be 2 meters. In case of Milrem dataset, we had perfect ground truth inside LiDAR point cloud so the tolerance value is comparatively kept lower on it i.e. 0.8 meters. Table 5.6 shows the potential impact of changing these parameters i.e. as benchmark radius gets smaller the LiDAR beams get denser and so average precision and average recall is boosted on the contrary reducing tolerance value decreases performance.

5.1.5 ROS Parameters

This section shows the results of benchmarking tests performed on FieldSAFE dataset over ROS parameters.

5.1.5.1 Play rate

Table 5.7. Play rate performance impact benchmarks over 5 runs (using Approx Time Sync)

Play rate	Best F1-score	Mean F1-score	Standard deviation of F1-score
0.5	76.6%	45.28%	0.2861
0.4	75.7%	70.56%	0.0792
0.3	77.9%	77.26%	0.0088
0.2	**77.8%**	**77.42%**	**0.0024**

0.1	66.8%	64.42%	0.0203

Following conclusions can be drawn from the results in Table 5.7:

- Reducing play rate gives more stable results in benchmarking
- 0.1, in this case, is an outlier i.e. the standard deviation is higher than previous play rates.

5.1.5.2 Time Synchronization

Table 5.8. Slop (Approx. Time Sync) performance impact benchmarks.

Slop	Precision	Recall	F1-score	No. of Frames
0.07	74.9%	53.6%	62.5%	388/390
0.06	89.9%	60.8%	72.5%	388/390
0.05	**93.7%**	**65.9%**	**77.4%**	**386/390**
0.04	92.0%	61.6%	73.8%	296/390
0.03	95.6%	53.6%	62.5%	219/390

Following conclusions can be drawn from the results in Table 5.8:

- Reducing slop gives better-benchmarking results but extracted frame count also decreases due to less tolerance for time synchronization error.
- Increasing slop value gives more frames but due to an increase in error tolerance, results get worse with it.
- For slop, a value which is lower and gives max frame count should be considered which in this case is 0.05 but these results may vary with datasets.

Table 5.9. Time Sync Methods performance impact benchmarks over 5 runs.

Method	Best F1-score	Mean F1-score	Standard deviation of F1-score	Average No. of Frames
Approx. Time Sync	77.8%	77.42%	0.0024	388/390
Naive Cache	79.8%	73.9%	0.0760	388/390
Naive Least Frequent	54.5%	52.65%	0.0021	360/390
Naive Most Frequent	**83.0%**	**81.26%**	**0.0202**	**387/390**

Following conclusions can be drawn from the results in Table 5.9:

- Naive approaches give comparatively better results than Approx. Time Synchronizer. Approximate time synchronization synchronizes messages w.r.t. to timestamps and should have performed better than naive approaches. But the benchmarking results show that naive approaches tend to be more stable. Any

time synchronization method in ROS would work better if the messages are synchronized on hardware level. In FieldSAFE dataset, the trigger signals for thermal and stereo camera were generated from a pulse-per-second signal from an internal GNSS in the LiDAR, which allowed exact timestamps for all three sensors [22]. Surprisingly this did not seem to help time synchronization.

- Naive least frequent method is unreliable i.e. depends on how quickly all the messages are processed.
- Naive least frequent skips a lot of frames.
- Naive most frequent gives the best performance among all available methods.

5.2 Error Analysis

In this section, we summarize frame by frame analysis on Milrem dataset to find the cause of possible errors or misdetection in each frame.

Table 5.9. Error analysis on 50 frames of Milrem dataset.

	Occlusion 2D	2D Detection	Misalignment	Occlusion 3D	Nearest Cluster
Error Count	11	13	0	6	2
Percentage of Errors	34.3%	40.6%	0%	18.7%	6.2%

Where the error causes in Table 5.9 are defined as follows:

- **2D occlusion:** Object not detected because behind another object.
- **2D detection:** Object not detected by YOLO.
- **Misalignment:** Object not detected due to temporal or spatial misalignment.
- **3D occlusion:** Two objects were detected in 2D, but because one was partly behind the other, the object positioned nearer was picked and the other was missed.
- **Nearest cluster:** Our clustering approach picks the closest cluster, sometimes there are enough points on the ground to form a cluster which makes human predictions closer than the actual human position.

5.2.1 2D Misdetection causes

Table 5.9 shows that most of the misdetections in 3D are caused by 2D misdetections. Therefore we additionally analyzed the main causes of 2D misdetections and found following reasons:

- Blending with the background (forest) (Figure 5.2)

Figure 5.2: An example of 2D mis-detection possibly due to cloth color blending with the background, where blue bounding boxes represent ground truth and green boxes represent 2D predictions.

- Unrecognized poses (Figure 5.3)

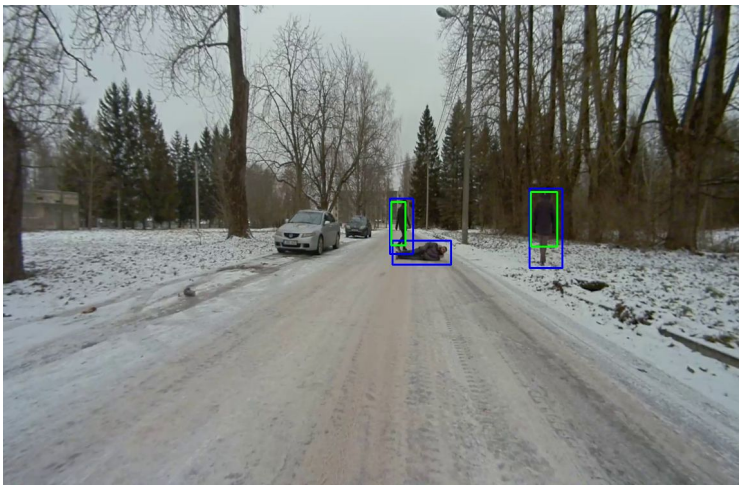


Figure 5.3. An example of 2D mis-detection due to human pose, where blue bounding boxes represent ground truth and green boxes represent 2D predictions.

- Random miss detection (Figure 5.4)

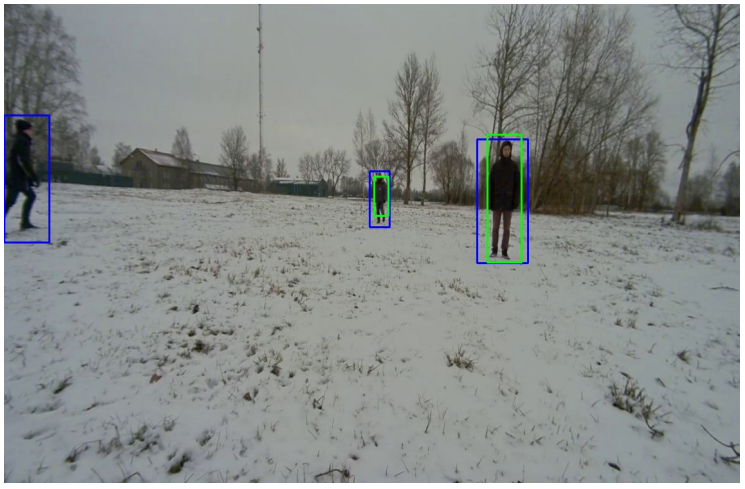


Figure 5.4: An example of random 2D mis-detection, where blue bounding boxes represent ground truth and green boxes represent 2D predictions.

For 2D driven 3D detection, we need to improve 2D detector to get rid of most the detection errors in 3D. At this point, we can safely exclude 2D and 3D occlusions from errors as we only intend to detect the humans which are not occluded by something. Lastly, the nearest cluster approach contributes very less to the total number of errors, which is one of the main things we wanted to analyze. We could apply some ground removal techniques before clustering the proposed 3D frustums to solve this type of errors.

6 Conclusions and Future Work

In this study, we experimented with state-of-the-art 2D object detection methods to achieve 2D driven 3D object detection. We summarized a frustum based approach to detect humans in off-road scenarios and discussed potential impact of various parameters to increase 3D detection efficiency. The results we gathered answers two core questions of our problem statement i.e.

- Leveraging mature 2D detection approaches can give decent 3D detection results in off-road scenarios provided we account for the alignment problems.
- Benchmarking inside ROS is unstable. This study demonstrated a potential solution i.e. an external setup that works in parallel with detection pipeline but is independent of ROS environment. Additionally, tuning pipeline parameters w.r.t. the use-case scenarios increases the performance of 3D detection.

Off-road scenarios are complex because detecting a perfect structure of small objects, e.g. humans in 3D, is often quite difficult. In this study, we reduced the complexity of this problem by detecting even a few 3D points in the proposed frustums. While we tuned the pipeline parameters in accordance with human detection, we cannot completely rely on these parameters generally. For example, if we want to predict bigger objects such as trees, cars etc we cannot just use the predicted centers of the clusters. We need to predict complete 3D bounding of the object. In case of moving obstacles such as cars, we might need to predict the orientation of the 3D bounding box too. For such cases, replacing frustum clustering with a neural network, similar to what Frustum PointNets [9], would be more appropriate.

One of the main advantages of using frustum based detection pipeline is that we can replace the 2D detection method with a much better version in the future. We could deploy a detection network that proposes 2D bounding boxes over thermal images thus giving 3D frustums from thermal image only or possibly apply some scoring technique that fuses both thermal and RGB images and gives fused proposed frustums for more better detection. We can also modify the clustering of proposed frustums to do ground removal inside frustums which would help to resolve nearest cluster misdetections discussed earlier. Additionally, while stabilizing the benchmarking setup, we deduced that speeding up the underlying algorithms can help to increase detection performance on naive time synchronization methods. A modular and simple structure of our detection pipeline makes it easy to adapt for future improvements.

7 References

[1] H. Cheng, *Autonomous Intelligent Vehicles: Theory, Algorithms, and Implementation*. Springer Science & Business Media, 2011.

[2] T. S. Combs, L. S. Sandt, M. P. Clamann, and N. C. McDonald, "Automated Vehicles and Pedestrian Safety: Exploring the Promise and Limits of Pedestrian Detection," *Am. J. Prev. Med.*, vol. 56, no. 1, pp. 1–7, Jan. 2019.

[3] J. Li, "Fusion of LiDAR 3D points cloud with 2D digital camera image," *Oakland University: Rochester, MI, USA*, 2015.

[4] F. Zhang, D. Clarke, and A. Knoll, "Vehicle detection based on LiDAR and camera fusion," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 1620–1625.

[5] N. Aranjuelo, L. Unzueta, I. Arganda-Carreras, and O. Otaegui, "Multimodal Deep Learning for Advanced Driving Systems," in *Articulated Motion and Deformable Objects*, 2018, pp. 95–105.

[6] D. Lahat, T. Adali, and C. Jutten, "Multimodal Data Fusion: An Overview of Methods, Challenges, and Prospects," *Proc. IEEE*, vol. 103, no. 9, pp. 1449–1477, Sep. 2015.

[7] O.-R. A. D. (orad) Committee and On-Road Automated Driving (ORAD) committee, "Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems." .

[8] "TechDay piloted driving – The traffic jam pilot in the new Audi A8," *Audi MediaCenter*. [Online]. Available: https://www.audi-mediacenter.com/en/techday-piloted-driving-the-traffic-jam-pilot-in-the-new-audi-a8-9276. [Accessed: 12-May-2019].

[9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11. pp. 1231–1237, 2013.

[11] Kamarul A, *Thermal Infrared vs RGB*. Youtube, 2015.

[12] J. W. Davis and V. Sharma, "Robust detection of people in thermal imagery," *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* 2004.

[13] S. Z. Nielsen, R. Gade, T. B. Moeslund, and H. Skov-Petersen, "Taking the Temperature of Pedestrian Movement in Public Spaces," *Transportation Research Procedia*, vol. 2. pp. 660–668, 2014.

[14] "What Is Camera Calibration? - MATLAB & Simulink - MathWorks China." [Online]. Available: https://ww2.mathworks.cn/help/vision/ug/camera-calibration.html?lang=en&s_tid=gn_loc_drop. [Accessed: 03-May-2019].

[15] "Stereo Vision Based Depth Estimation Algorithm In Uncalibrated Rectification." [Online]. Available: https://pdfs.semanticscholar.org/32e9/d2bfe7d347662591cfb92c8309e62ea7fe51.pdf?_ga=2.124916426.197756512.1556227394-672891054.1556227394. [Accessed: 26-Apr-2019].

[16] R. B. Fisher and K. Konolige, “Range Sensors,” *Springer Handbook of Robotics*. pp. 521–542, 2008.
[17] M. Barnard, “Tesla & Google Disagree About LIDAR -- Which Is Right? | CleanTechnica,” *CleanTechnica*, 29-Jul-2016. [Online]. Available: https://cleantechnica.com/2016/07/29/tesla-google-disagree-lidar-right/. [Accessed: 10-May-2019].
[18] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Global Edition*. Pearson, 2017.
[19] W. Cunningham, “How lasers map the world for self-driving cars - Roadshow,” *Roadshow*, 19-Dec-2016. [Online]. Available: https://www.cnet.com/roadshow/news/how-lasers-map-the-world-for-self-driving-cars/. [Accessed: 03-May-2019].
[20] A. Arya Senna Abdul Rachman, “3D-LIDAR Multi Object Tracking for Autonomous Driving: Multi-target Detection and Tracking under Urban Road Uncertainties,” 2017.
[21] R. Hartley and A. Zisserman, “The Background: Projective Geometry, Transformations and Estimation,” *Multiple View Geometry in Computer Vision*. pp. 23–24.
[22] M. F. Kragh *et al.*, “FieldSAFE: Dataset for Obstacle Detection in Agriculture,” *Sensors* , vol. 17, no. 11, Nov. 2017.
[23] V. De Silva, J. Roche, and A. Kondoz, “Fusion of LiDAR and camera sensor data for environment sensing in driverless vehicles,” 2018.
[24] C. Premebida, J. Carreira, J. Batista, and U. Nunes, “Pedestrian detection combining RGB and dense LIDAR data,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4112–4117.
[25] W. Choi, C. Pantofaru, and S. Savarese, “Detecting and tracking people using an RGB-D camera via multiple detector fusion,” in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 1076–1083.
[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6. pp. 84–90, 2017.
[27] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *arXiv [cs.CV]*, 04-Jun-2015.
[28] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
[29] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” *IEEE Trans. Pattern Anal. Mach. Intell.*, Jun. 2018.
[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *arXiv [cs.CV]*, 08-Jun-2015.
[31] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, “3d bounding box estimation using deep learning and geometry,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.
[32] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, “Data-driven 3d voxel patterns for object category recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1903–1911.
[33] T. He and S. Soatto, “Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors,” *arXiv [cs.CV]*, 11-Jan-2019.

[34] Z. Luo, S. Habibi, and M. V. Mohrenschildt, “LiDAR based real time multiple vehicle detection and tracking,” *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 10, no. 6, 2016.
[35] D. Matti, H. K. Ekenel, and J.-P. Thiran, “Combining LiDAR Space Clustering and Convolutional Neural Networks for Pedestrian Detection,” *arXiv [cs.CV]*, 17-Oct-2017.
[36] I. Malysheva and Others, “Large-scale multimodal sensor fusion and object detection,” 2017.
[37] J. Lahoud and B. Ghanem, “2d-driven 3d object detection in rgb-d images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4622–4630.
[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” 02-Dec-2016.
[39] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-View 3D Object Detection Network for Autonomous Driving,” *arXiv [cs.CV]*, 23-Nov-2016.
[40] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, “Joint 3D Proposal Generation and Object Detection from View Aggregation,” *arXiv [cs.CV]*, 06-Dec-2017.
[41] E. Schröder, M. Mählisch, J. Vitay, and F. Hamker, “Fusion of Camera and Lidar Data for Object Detection using Neural Networks.”
[42] D. Z. Wang and I. Posner, “Voting for Voting in Online Point Cloud Object Detection,” in *Robotics: Science and Systems*, 2015, vol. 1, pp. 10–15607.
[43] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, “O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis,” *ACM Trans. Graph.*, vol. 36, no. 4, pp. 72:1–72:11, Jul. 2017.
[44] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.
[45] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1355–1361.
[46] B. Li, T. Zhang, and T. Xia, “Vehicle Detection from 3D Lidar Using Fully Convolutional Network,” *arXiv [cs.CV]*, 29-Aug-2016.
[47] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, and Others, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Kdd*, 1996, vol. 96, pp. 226–231.
[48] “OpenCV: Image Thresholding.” [Online]. Available: https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html. [Accessed: 27-Apr-2019].
[49] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, “ICNet for Real-Time Semantic Segmentation on High-Resolution Images,” *arXiv [cs.CV]*, 27-Apr-2017.
[50] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-decoder with atrous separable convolution for semantic image segmentation,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 801–818.
[51] “Clock - ROS Wiki.” [Online]. Available: http://wiki.ros.org/Clock. [Accessed: 29-Apr-2019].

[52] “YOLOv3.” [Online]. Available: https://pjreddie.com/media/files/papers/YOLOv3.pdf. [Accessed: 10-May-2019].

Appendix

I. License

Non-exclusive licence to reproduce thesis and make thesis public

I, Mahir Gulzar

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

Object Detection Using LiDAR and Camera Fusion in Off-road Conditions,

supervised by **Tambet Matiisen**

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mahir Gulzar
16/05/2019