Software Requirements Specification for ESF : Equivariant Sensor Fusion for Autonomous Driving Object Detection

Alaap Grandhi

April 16, 2025

Contents

Revision History

Date	Version	Notes
February 9, 2025	1.0	Initial SRS draft/document.
April 16, 2025	2.0	Updated SRS according to comments.

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

Throughout this document SI (Système International d'Unités) is employed as the unit system. In addition to the basic units, several derived units are used as described below. For each unit, the symbol is given followed by a description of the unit and the SI name.

symbol	unit	SI
m	length	metre

1.2 Table of Symbols

The table that follows summarizes the symbols used in this document along with their units. The symbols are listed in alphabetical order.

symbol	unit	description
f_x	$\mathbb{R}_{\geq 0}$ pixels	The focal length of the given camera in the x direction.
f_y	$\mathbb{R}_{\geq 0}$ pixels	The focal length of the given camera in the y direction.
c_x	$\mathbb{R}$ pixels	The principal point of the given camera in the x direction.
c_y	$\mathbb{R}$ pixels	The principal point of the given camera in the y direction.
s	$\mathbb{R}$ pixels	The skew of the given camera representing how far from perpendicular the x and y directions are.
$\mathbf{K}$	$\mathbb{R}^{3\times 3}$ pixels	The intrinsic matrix of the given camera representing how 3D points are projected onto the 2D image plane.
n_p	$\mathbb{R}_{\geq 0}$	The number of pixels in a given input image.
$\mathbf{P_i}$	$\mathbb{R}^{2\times n_p}$ pixels	The set of x,y coordinates for each of the n_p pixels in a given image.
$\mathbf{P_c}$	$\mathbb{R}^{3\times n_p}$ m	The set of back-projected x,y,z coordinates for each of the n_p pixels in a given image. This is just X_c, Y_c, and Z_c concatenated.
$\mathbf{X_c}$	$\mathbb{R}^{n_p}$ m	The set of back-projected x coordinates for each of the n_p pixels in a given image.
$\mathbf{Y_c}$	$\mathbb{R}^{n_p}$ m	The set of back-projected y coordinates for each of the n_p pixels in a given image.
$\mathbf{Z_c}$	$\mathbb{R}^{n_p}$ m	The set of back-projected z coordinates for each of the n_p pixels in a given image.
D	$\mathbb{R}^3 \to \mathbb{R}^3$	The distortion model of the given camera (accounts for lens effects).

$\mathbf{P_w}$	$\mathbb{R}^{3\times n_p}$ m	The set of back-projected, world reference frame x,y,z coordinates for each of the n_p pixels in a given image.
$\mathbf{T_{c,w}}$	$\mathbb{R}^{4\times 4}$	The extrinsics matrix that maps x,y,z points from the given camera's frame of reference to the world reference frame.
$\mathbf{R_{c,w}}$	$\mathbb{R}^{3\times 3}$	The rotation matrix between the given camera's frame of reference and the world reference frame.
$\mathbf{t_{c,w}}$	$\mathbb{R}^3$	The translation vector between the given camera's frame of reference and the world reference frame.
$\mathbb{P}$	$\mathbb{R}^{n_c}$	A predicted probability distribution over the set of n_c discrete classes.
α_{gd}	$\mathbb{R}$	A hyperparameter for gradient descent that controls how large each gradient update is.
L	$\mathbb{R}^{n_\theta} \to \mathbb{R}$	A loss function that maps the parameters of a model to a single estimated loss value.
n_θ	$\mathbb{R}_{\geq 0}$	The number of learnable parameters for a given model.
θ	$\mathbb{R}^{n_\theta}$	The learnable parameters for a given model, where the number of parameters is n_θ.
TP	$\mathbb{N}_0$	The number of true positives for a given classification task. That is, the number of positive predictions that corresponded to positive ground truth.
FP	$\mathbb{N}_0$	The number of false positives for a given classification task. That is, the number of positive predictions that corresponded to a negative ground truth.
FN	$\mathbb{N}_0$	The number of false negatives for a given classification task. That is, the number of positive ground truths that were not predicted to be positive.
$\mathbf{d_c}$	$\mathbb{R}^{n_p}$ m	The estimated depth values for each of the pixels in a given image.
g	$\mathbb{R}_{\geq 0}$	The number of scalar values/parameters used to define a bounding box (dataset-dependent). For the nuScenes dataset, this is 10 corresponding to the 3D center position, height, width, length, and a quaternion angle representation.
n_b	$\mathbb{R}_{\geq 0}$	The number of ground truth bounding boxes.
$n_{\hat{b}}$	$\mathbb{R}_{\geq 0}$	The number of predicted bounding boxes.
n_c	$\mathbb{R}_{\geq 0}$	The number of object classes (typically 3 corresponding to cars, pedestrians, and cyclists).
$\mathbf{B}$	$\mathbb{R}^{g\times n_b}$	The set of ground-truth bounding box attributes (i.e. center, height, width, etc.), where there are n_b ground truth bounding boxes.

$\hat{\mathbf{B}}$	$\mathbb{R}^{g \times n_{\hat{b}}}$	The set of predicted bounding box attributes, where there are $n_{\hat{b}}$ predicted bounding boxes.
$\hat{\mathbf{C}}_{\text{dist}}$	$\mathbb{R}^{n_c \times n_{\hat{b}}}$	The set of predicted bounding box class probability distributions.
$\hat{\mathbf{c}}$	$\mathbb{N}_0^{n_{\hat{b}}}$	The set of predicted bounding box classes.
$\mathbf{c}$	$\mathbb{N}_0^{n_b}$	The set of ground truth bounding box classes.
α_{fl}	$\mathbb{R}$	A hyperparameter for controlling the influence of the focal loss term.
γ	$\mathbb{R}$	A hyperparameter for controlling the effect of easy examples on the focal loss. In practice, this is class specific and is set to the inverse class frequency.
β_1	$\mathbb{R}$	A hyperparameter for changing the inertia of the first moment in ADAM.
β_2	$\mathbb{R}$	A hyperparameter for changing the inertia of the second moment in ADAM.
τ_{IoU}	$\mathbb{R}$	An IoU threshold that is used to determine whether a predicted and a ground truth bounding box sufficiently overlap to be considered paired.
h	$\mathbb{R}_{\geq 0}$	The height of a given input image.
w	$\mathbb{R}_{\geq 0}$	The width of a given input image.
n_l	$\mathbb{R}_{\geq 0}$	The number of points in an input lidar pointcloud.
$\mathcal{I}$	$\mathbb{R}^{h \times w}$	An input image with height h and width w.
$\mathcal{P}$	$\mathbb{R}^{3 \times n_l}$	An input lidar pointcloud with n_l points.

1.3 Abbreviations and Acronyms

symbol	description
A	Assumption
DD	Data Definition
GD	General Definition
GS	Goal Statement
IM	Instance Model
LC	Likely Change
PS	Physical System Description
R	Requirement
SRS	Software Requirements Specification
ESF	Equivariant Sensor Fusion for Object Detection
TM	Theoretical Model

1.4 Mathematical Notation

A list of the typographic conventions used for mathematical notation is provided as follows:

- Matrices are represented as bold uppercase letters, while vectors are represented as bold lowercase letters
- Scalars are unbolded and can be in either uppercase or lowercase
- A hat is used to denote a variable that is an estimator for another variable (i.e. $\hat{B}$ is an estimator for B).
- The matrix product is represented by placing two matrices adjacent to each other (AB), while the Hadamard product is represented by placing a circle between the two matrices ($A \circ B$)
- Matrices and vectors are indexed using square brackets and subscripts are only used for the clarification of symbols.

2 Introduction

Over the past few years, we have seen more and more autonomous vehicles being allowed onto the road. With this increase, the need for perception methods that can effectively use readings from sensor modalities like Camera and LiDAR to inform these vehicles of their surroundings has similarly increased. These sensor modalities can provide useful information on their own, but they are inherently prone to failure stemming from weather conditions or unforeseen road conditions. In contrast, methods combining the readings from both of these modalities are able to mitigate individual sensor modality failures. The aim of this project is to develop a system that can effectively combine information from Camera and LiDAR sensors to determine the dynamic objects in an autonomous vehicle's surroundings.

Now that the problem has been setup, the rest of this introduction section will outline the purpose of the document, the scope of the project, the desired characteristics of the reader, and a high-level overview of the remaining sections of the document. The template for this report is based on Smith and Lai (2005); Smith et al. (2007); Smith and Koothoor (2016).

2.1 Purpose of Document

The purpose of this document is to lay the foundation for the rest of the equivariant sensor fusion project. By outlining the system, constraints, requirements, and mathematical models necessary to frame the problem, this document will serve as the basis for future design documents and for the design of the final solution.

2.2 Scope of Requirements

This project is scoped to only consider scenarios where both images from multiple camera views (not just one) and a pointcloud from LiDAR are available at all points in time. While I have previously stated that solutions should be robust to noise and disturbances in either sensor modality, it is assumed that neither sensor will completely cut out at any point in time. That is to say that it is assumed that both forms of input will always be inserted into the model in the expected format, even if one of those inputs is highly corrupted or full of zeros. The problem is also constrained to one where model training and inference happen on different random subsets of the same dataset. The effects of changing domain (driving in Canada vs driving in Europe) are out of scope.

2.3 Characteristics of Intended Reader

The readers of this document should have an understanding of partial derivatives as would be covered in any first year undergraduate calculus course. Additionally, readers should have a first year understanding of matrix algebra. Beyond these basic requirements, an understanding of transformation matrices corresponding to a second year robotics or computer

vision course would help the reader understand the physical system. Similarly, a basic understanding of machine learning from an introductory course would make the optimization structure clearer to the reader.

2.4 Organization of Document

Now that the setting and scope of ESF have been discussed, the remainder of this document will delve into the physical setup, the intended requirements, and the math underlying the problem in more detail. Specifically, these sections will include:

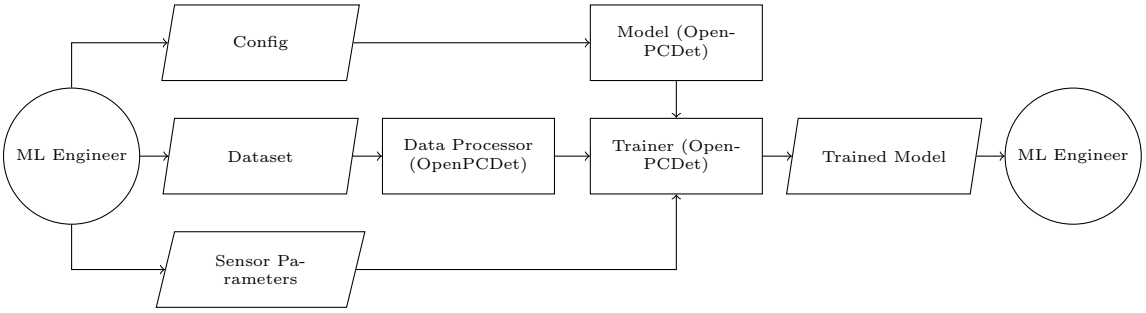
- **General System Description**: A high-level overview of the system setup, encompassing the structure of the system, the characteristics of the intended user, and the constraints on the system.
- **Specific System Description**: A more in-depth mathematical setup for the problem. First the problem's physical structure is described before progressively refining mathematical models that can be used to address the goals for the ESF project.
- **Requirements**: A description of the functional and non-functional requirements that describe qualities the software should have.
- **Likely Changes**: A list of expected changes to the modelling and design of the problem and requirements.
- **Unlikely Changes**: A list of unlikely but possible changes to the modelling and design of the problem and requirements.
- **Traceability Matrices and Graphs**: A set of tables tracking which other sections each section references.

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

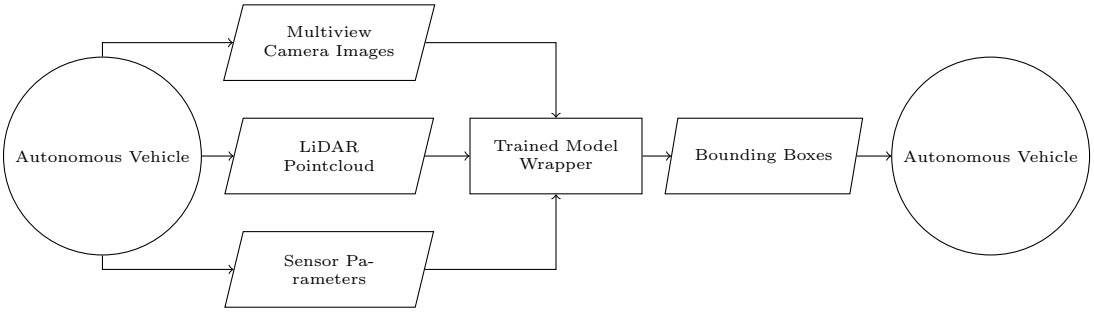
3.1 System Context

3.1.1 Training Pipeline



- ML Engineer Responsibilities:
  - Design the configuration file
  - Choose the appropriate dataset
  - Obtain the sensor parameters for the dataset
- Config Responsibilities:
  - Describe how the model should be setup
  - List the components that should be included in the model
- Dataset Responsibilities:
  - Pair each set of multiview camera images with its corresponding LiDAR point-cloud and ground truth bounding boxes
- Sensor Parameters Responsibilities:
  - Characterize settings of each sensor (camera or LiDAR)
- Data Processor Responsibilities:
  - Process the raw dataset into a form the trainer can accept
  - Detect type mismatch
  - Move data to appropriate device (i.e. GPU)
- Model Responsibilities:
  - Learnable function mapping input sensor data to output bounding boxes
- Trainer Responsibilities:
  - Use the dataset to refine the model over time
- Trained Model Responsibilities:
  - Characterize the final best version of the learnable model

3.1.2 Inference Pipeline



- Autonomous Vehicle Responsibilities:
  - Obtain the relevant sensor information and relay it to the trained model
- Multiview Camera Images Responsibilities:
  - RGB camera images showing different views from the autonomous vehicle (in front, behind, to the right, etc)
- LiDAR Pointcloud Responsibilities:
  - 3D Point Cloud representation of the autonomous vehicle's surroundings
- Sensor Parameters Responsibilities:
  - Characterize settings of each sensor (camera or LiDAR)
- Trained Model Wrapper Responsibilities:
  - Function mapping input sensor data to output bounding boxes
  - Use the trained model from the training pipeline
- Bounding Boxes Responsibilities:
  - Characterize the dynamic objects in the autonomous vehicle's surroundings

3.1.3 Intended Use Cases

This software will primarily be used for two different purposes. For the most part, the software will be used for research purposes. Researchers in the domain of autonomous driving perception will augment and use the software to run experiments and gather results for publication purposes (to further the state-of-the-art in the field). The software is also intended to be used as an integrated part of safety-critical autonomous driving systems down the line but the details of such an integration are beyond the scope of this project. Thus, if the system is integrated into an autonomous driving system, the company doing the integration is expected to address the associated safety concerns (not this document).

3.2 User Characteristics

Since the algorithm will be used as part of a bigger system (integrated into the autonomous vehicle) in inference, it does not really make sense to define the skills expected for that half of the pipeline. If used on its own in inference, no additional skills would be necessary to use the trained model.

When it comes to the training pipeline, the desired characteristics of the machine learning engineer would depend on their interest in the project. If they simply desire to tweak a config file to train a model on a pre-existing dataset, then they would only need a high-level understanding of the model hyperparameters corresponding to a third or fourth year undergraduate machine learning course.

If they intend to modify parts of an existing model or train on a new dataset for research purposes, they would need an understanding of the math underlying computer vision corresponding to a fourth year Computer Vision and Robotics course. Additionally, they would need an understanding of Deep Learning based approaches to Computer Vision corresponding to a fourth year undergraduate or first year graduate course (specialized to the topic).

3.3 System Constraints

C1: The chosen solution shall be designed as a learned model inside the OpenPCDet framework Team (2020). Rationale provided in RN1.

C2: The chosen solution shall be trained on either the NuScenes Caesar et al. (2020) or the Waymo Sun et al. (2020) Dataset. Rationale provided in RN2.

C3: The chosen learned solution shall be optimized using the ADAM Optimizer. This in turn also means that optimization will be done using gradient-descent as ADAM is a gradient-descent algorithm. This refines the first constraint above to require that the model is differentiable. Rationale provided in RN3.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. This is followed by the solution characteristics specification, which presents the assumptions, theories, definitions and finally the instance models.

4.1 Problem Description

Project is intended to provide a framework/library for determining the set of dynamic objects surrounding an autonomous vehicle given readings from its onboard sensors. In inference, it will use a set of sensor readings to predict the bounding boxes of the cars, pedestrians, and cyclists in an autonomous vehicle's surroundings. In training, it will use a given autonomous

driving dataset with paired sensor and ground truth bounding boxes to produce a trained model for this task.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- **Autonomous Vehicle**: A vehicle that can navigate and maneouver on the road without requiring human input.
- **NuScenes and Waymo**: Two commonly used open-source object detection datasets for autonomous driving Caesar et al. (2020) Sun et al. (2020).
- **Pointcloud**: A way of representing a 3D scene that uses a set of xyz points to denote locations where any solid object is present.
- **LiDAR**: A sensor that uses beams of light from a laser to obtain a 3D pointcloud representation of its surroundings.
- **Bounding Box**: A 2D or 3D box that completely encompasses some given object as compactly as possible.
- **Multiview Camera Images**: Camera images taken from multiple different viewing angles in close proximity, all concatenated together.
- **Pinhole Camera Model**: A simple model for describing how 3D points map to 2D points in the image plane.
- **Reference Frame**: An abstract coordinate system with its own origin and base x, y, z vectors.
- **Classification**: A task that involves assigning one label from a discrete set of labels to a given input.
- **Regression**: A task that involves predicting a continuous value for a given input.
- **Loss**: A term used to describe how poorly a system performs for some set of input, given some known ground truth.
- **Depth**: In the context of this report, depth is used to refer to how far away a given point is from some camera's origin.
- **OpenPCDet**: An open-source repository that provides extensive support for performing 3D object detection from camera and/or LiDAR input Team (2020).
- **BEVFusion**: A state-of-the-art method for performing 3D object detection from camera and LiDAR input Liang et al. (2022).

4.1.2 Physical System Description

The physical system of ESF includes the following elements:

PS1: The setup of sensors on top a standard autonomous vehicle (taken from NuScenes Caesar et al. (2020)).

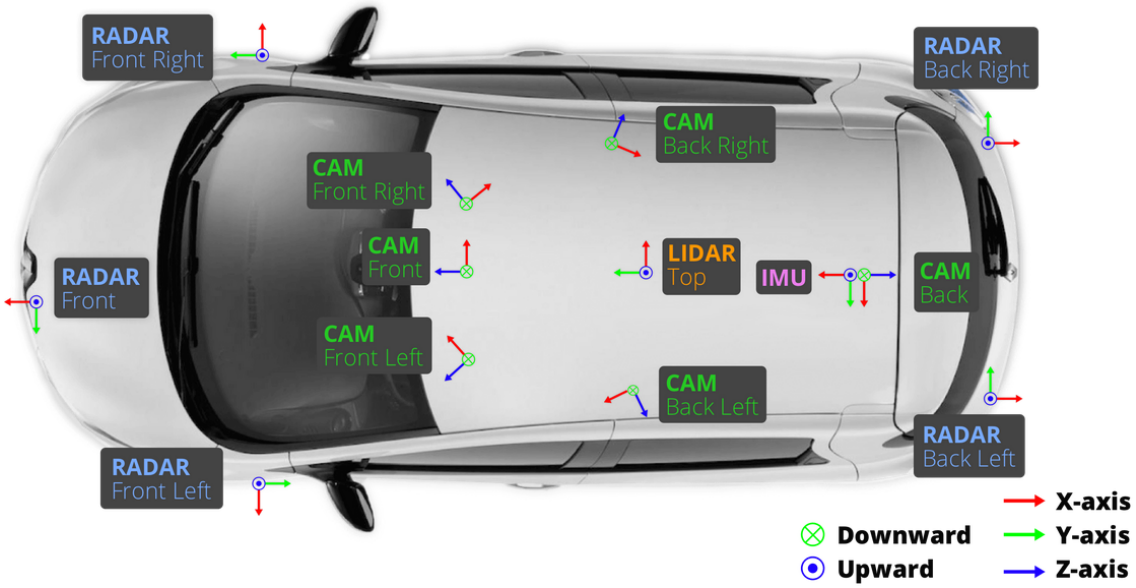


Figure 1: NuScenes autonomous vehicle sensor setup

The autonomous vehicle collects sensor readings from the sensors shown in 1. As the vehicle drives around, these sensors run at a predetermined frequency, capturing information about the vehicle's surroundings and thus enabling it to drive safely.

PS2: A camera image taken from one of the cameras on the autonomous vehicle (taken from NuScenes Caesar et al. (2020)).

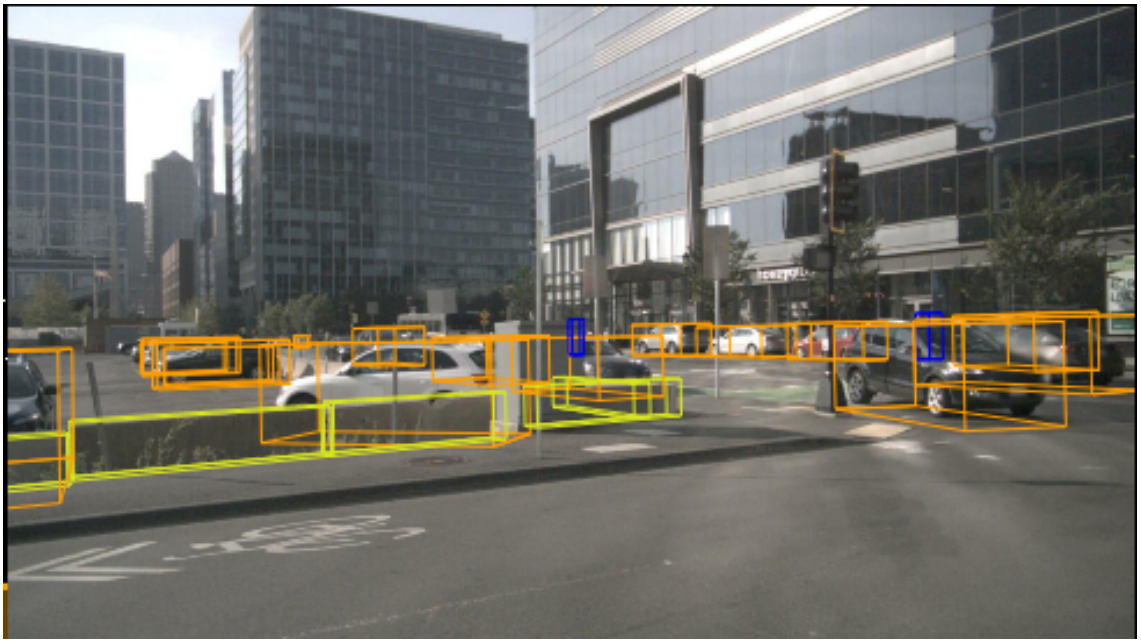


Figure 2: Camera image with bounding boxes taken from NuScenes

The cameras on the autonomous vehicle each capture images that look similar to the image shown in 2. Note that while boxes are shown in this image to highlight the other vehicles and the pedestrians in the image, the image input into the system will not have these.

PS3: A LiDAR pointcloud obtained from the spinning LiDAR sensor on top of the autonomous vehicle (taken from NuScenes Caesar et al. (2020)).

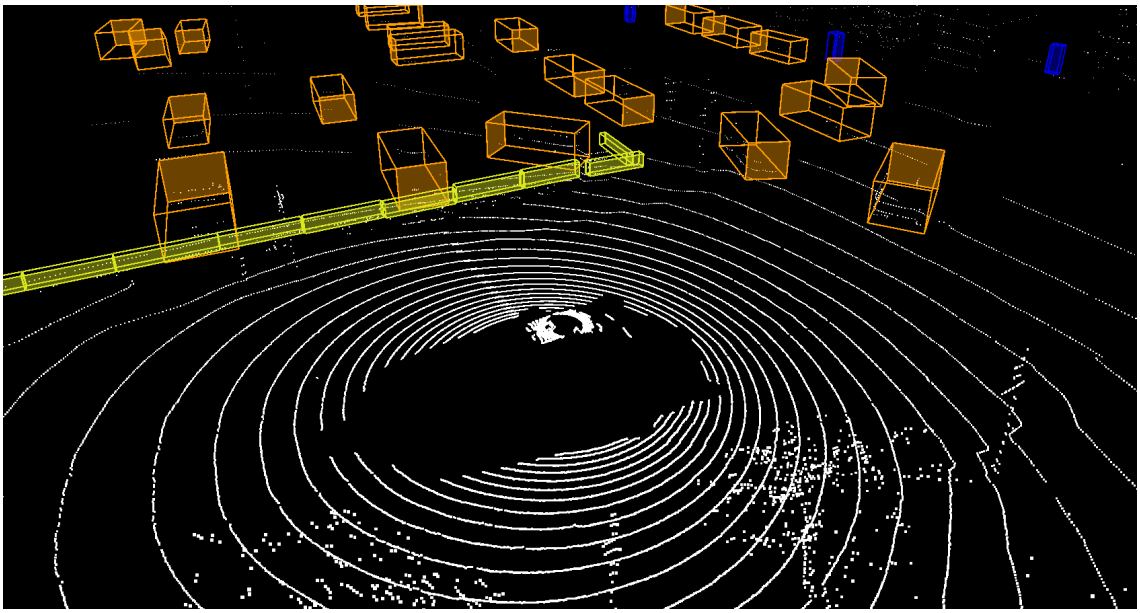


Figure 3: LiDAR pointcloud with bounding boxes taken from NuScenes

The LiDAR sensor on top of the autonomous vehicle captures information that is then converted into the pointcloud representation shown in 3. Similar to the case of the camera image shown before, the boxes in the pointcloud are for illustrative purposes and are not included in the input to the system.

PS4: The bounding boxes for the a given scene in the NuScenes dataset Caesar et al. (2020).

The LiDAR pointcloud shown in 3 and the camera image shown in 2 both have boxes surrounding the vehicles, pedestrians, and other objects of importance in the autonomous vehicle's surroundings. These boxes are called bounding boxes and they represent the output of the system.

4.1.3 Goal Statements

GS1: Given a dataset comprising of multiview camera images with paired LiDAR pointclouds and ground-truth bounding boxes, output a model that is trained to minimize the chosen error function on the input dataset.

GS2: Given multi-view camera images, a corresponding LiDAR pointcloud from an autonomous vehicle, and the trained model from GS2, output the set of bounding boxes for dynamic entities in the autonomous vehicle's surroundings.

4.2 Solution Characteristics Specification

The instance models that govern ESF are presented in Subsection 4.2.9. The information to understand the meaning of the instance models and their derivation is also presented, so

that the instance models can be verified.

4.2.1 Types

This section is not used at the moment, but it may be used in future iterations of this document.

4.2.2 Scope Decisions

This section is not used at the moment, but it may be used in future iterations of this document.

4.2.3 Modelling Decisions

This section is not used at the moment, but it may be used in future iterations of this document.

4.2.4 Assumptions

This section simplifies the original problem and helps in developing the theoretical model by filling in the missing information for the physical system. The numbers given in the square brackets refer to the theoretical model [TM], general definition [GD], data definition [DD], instance model [IM], or likely change [LC], in which the respective assumption is used. The rationale for these assumptions is provided in RN4.

A1 The camera images passed into the system are assumed to all be undistorted. That is to say that the camera images are all assumed to follow the pinhole camera model. Used in GD2.

A2 All the cameras aboard the autonomous vehicle are assumed to have near-perfectly perpendicular x and y axes. Thus, the skew parameter in the camera intrinsic matrix is assumed to be negligible. Used in GD1 and LC2.

A3 The numerical integral of the precision over a finite set of equally-spaced recall values is assumed to be close enough to the actual average precision value given that the set of recall values is sufficiently large. Used in GD2.

4.2.5 Theoretical Models

This section focuses on the general equations and laws that ESF is based on.

RefName: **TM:Intrin**

Label: 3D Position to 2D Pixel Mapping

Equation: $$\begin{bmatrix} \mathbf{P_i} \\ 1 \end{bmatrix} = D(\mathbf{K} \begin{bmatrix} \mathbf{P_c^*} \end{bmatrix}), \mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{P_c^*} \triangleq \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \\ 1 \end{bmatrix}$$

Description: The above equation describes how a 3D location $\mathbf{P_c}$ in the camera's reference frame is mapped to a 2D location in the camera's image plane $\mathbf{P_i}$. It also shows how the camera's intrinsic matrix $\mathbf{K}$ used in this equation is constructed from the camera's focal lengths f_x and f_y, the camera's skew parameter s, and the camera's principal points c_x and c_y. The outer function D represents the inherent distortion introduced due to the lens characteristics of the camera.

Notes:

Source: Barfoot (2017)

Ref. By: GD1

Preconditions for TM:Intrin:

Derivation for TM:Intrin: Not Applicable

RefName: **TM:Extrin**

Label: Camera Extrinsics

Equation: $$\begin{bmatrix} \mathbf{P_w} \\ 1 \end{bmatrix} = \mathbf{T_{c,w}} \begin{bmatrix} \mathbf{P_c} \\ 1 \end{bmatrix}, \mathbf{T_{c,w}} = \begin{bmatrix} \mathbf{R_{c,w}} & \mathbf{t_{c,w}} \\ 0 & 1 \end{bmatrix}$$

Description: The above equation gives the general relation between a point's coordinates $\mathbf{P_c}$ in some original frame and its coordinates $\mathbf{P_w}$ in some other reference frame. It also shows how the extrinsics matrix $\mathbf{T_{c,w}}$ can be decomposed into a rotation $\mathbf{R_{c,w}}$ and a translation $\mathbf{t_{c,w}}$ between the two reference frames.

Notes:

Source: Barfoot (2017)

Ref. By: IM1

Preconditions for TM:Extrin:

Derivation for TM:Extrin: Not Applicable

RefName: **TM:CE**

Label: Categorical Cross Entropy

Equation: $CE(\mathbb{P}, t) = -\log(\mathbb{P}[t])$

Description: The equation above represents a loss function that is commonly used for classification tasks in machine learning. Given a probability distribution over the set of possible classes $\mathbb{P}$ as well as the index of the correct class t ($\mathbb{R}$), it provides a measure of how poorly the probability distribution predicted the correct class. It penalizes larger errors (probabilities closer to 0) much more harshly than minor errors (probabilities closer to 1) due to the log term.

Notes:

Source: Ross and Dollár (2017)

Ref. By: IM2

Preconditions for TM:CE:

Derivation for TM:CE: Not Applicable

RefName: **TM:GD**

Label: Gradient Descent

Equation: $\theta = \theta - \alpha_{gd} \nabla L(\theta)$

Description: The above equation is the simple update rule that forms the backbone of modern machine learning. Given some differentiable loss function L that we wish to minimize, a rate-controlling parameter α, and a set of parameters θ, it updates θ in a way that would reduce the obtained loss $L(\theta)$.

Notes:

Source: Amari (1993)

Ref. By: IM3

Preconditions for TM:GD:

Derivation for TM:GD: Not Applicable

RefName: **TM:IoU**

Label: Intersection over Union

Equation: $IoU(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$

Description: Given two shapes in n-dimensional space A and B, the intersection over union metric ($\mathbb{R}_{\geq 0}$) provides a measure of their similarity. It tells us how much these two shapes overlap and thus it can indicate how well of an estimation one is for the other.

Notes:

Source: Team (2020)

Ref. By: IM2, IM4

Preconditions for TM:IoU:

Derivation for TM:IoU: Not Applicable

RefName: **TM:Prec**

Label: Precision

Equation: $P = \frac{TP}{TP+FP}$

Description: Precision P ($\mathbb{R}_{\geq 0}$) is a performance metric that is used in prediction tasks to determine how often a system's predictions are correct. It is calculated in the formula above using the number of true positives TP and the number of false positives FP.

Notes:

Source: Zhu (2004)

Ref. By: TM4.2.5

Preconditions for TM:Prec:

Derivation for TM:Prec: Not Applicable

RefName: **TM:Rec**

Label: Recall

Equation: $R = \frac{TP}{TP+FN}$

Description: Recall R ($\mathbb{R}_{\geq 0}$) is a performance metric that is used in prediction tasks to determine how often a system is able to detect occurrences of whatever it is predicting. It is calculated in the formula above using the number of true positives TP and the number of false negatives FN.

Notes:

Source: Zhu (2004)

Ref. By: TM4.2.5

Preconditions for TM:Rec:

Derivation for TM:Rec: Not Applicable

RefName: **TM:AP**

Label: Average Precision

Equation: $AP = \int_0^1 P(x)\, dR(x)$

Description: Average Precision AP ($\mathbb{R}_{\geq 0}$) is a performance metric that is used in prediction tasks to capture both recall and precision into a single metric. In the above formula, the symbol x is simply used to capture the vector of controllable parameters (each in R) that influence both recall and precision.

A high average precision score indicates that a system can predict most actual occurrences without many false predictions. In its purest form, it represents the area under a curve formed by plotting the precision of a system against its recall.

Notes:

Source: Zhu (2004)

Ref. By: GD2

Preconditions for TM:AP: TM4.2.5, TM4.2.5

Derivation for TM:AP: Not Applicable

4.2.6 General Definitions

This section collects the laws and equations that will be used in building the instance models.

Number	GD1
Label	**Camera Intrinsics Projection**
Units	pixels
Equation	$\begin{bmatrix} \mathbf{P_i} \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{P}^*_\mathbf{c}$, $\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, $\mathbf{P}^*_\mathbf{c} = \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \\ 1 \end{bmatrix}$
Description	This relation is a simplified version of TM4.2.5. Through assumptions A1 and A2, the distortion function and the skew parameter have been removed from the equation. The equation still works as described in TM4.2.5.
Source	N/A
Ref. By	IM1

Number	GD2
Label	**Numerical Average Precision Approximation**
Units	unitless
Equation	$AP = \sum_{k=1}^{N} (R_k - R_{k-1}) P_k$
Description	Since precision and recall cannot be expressed as direct functions of some input value, the formulation in TM4.2.5 is impossible to solve directly. As such, this relation uses a numerical approximate for the integral shown in that formulation. Through assumption A3, this numerical approximation can directly be used in place of the actual integral. Here, R ($\mathbb{R}^{N+1}_{\geq 0}$) represents a set of $N+1$ discretely sampled recall values, while P ($\mathbb{R}^{N+1}_{\geq 0}$) represents the set of $N+1$ corresponding precision values.
Source	Zhu (2004)
Ref. By	IM4

4.2.7 Data Definitions

This section is not used at the moment, but it may be used in future iterations of this document.

4.2.8 Data Types

This section is not used at the moment, but it may be used in future iterations of this document.

4.2.9 Instance Models

This section transforms the problem defined in Section 4.1 into one which is expressed in mathematical terms. It uses concrete symbols defined in Section 4.2.7 to replace the abstract symbols in the models identified in Sections 4.2.5 and 4.2.6.

Number	IM1
Label	**Camera to Reference Frame Transformation**
Input	$\mathbf{K}$, $\mathbf{T_{c,w}}$, $\mathbf{d_c}$, $\mathbf{P_i}$
Output	$\mathbf{P_w}$
Description	By combining TM4.2.5 and GD1, we can get the following equation: $$\begin{bmatrix} \mathbf{P_w} \\ 1 \end{bmatrix} = \mathbf{T_{c,w}} \begin{bmatrix} \mathbf{K} & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \begin{bmatrix} \mathbf{P_i} \\ 1 \end{bmatrix} \circ \mathbf{d_c} \\ 1 \end{bmatrix}$$ $\mathbf{P_w}$ represents a matrix of 3D points back-projected from the set of 2D pixel coordinates $\mathbf{P_i}$ and their corresponding depth estimates $\mathbf{d_c}$. Note that multiplying by the depth estimates here is the inverse operation to dividing by z when going from 3D to 2D (as seen in GD1). This allows for 2D camera features to then be projected to 3D locations so that they can be merged with LiDAR.
Sources	N/A
Ref. By	

Number	IM2
Label	**Bounding Box Loss Function** L
Input	$\hat{\mathbf{B}}$, $\mathbf{B}$, $\hat{\mathbf{C}}_{\text{dist}}$, $\mathbf{c}$, α_{fl}, γ
Output	RL, FL, L
Description	Given the set of predicted bounding boxes $\hat{\mathbf{B}}$ and the set of ground-truth bounding boxes $\mathbf{B}$, the set of paired ground-truth bounding boxes $\mathbf{B}^*$ and paired ground-truth bounding box classes $\mathbf{c}^*$ will be defined as follows using the IoU function defined in TM4.2.5: $\mathbf{B}^*[,i] = \mathbf{B}[,\arg\max_j \text{IoU}(\hat{\mathbf{B}}[,i], \mathbf{B}[,j])]$, $\mathbf{c}^*[i] = \mathbf{c}[\arg\max_j \text{IoU}(\hat{\mathbf{B}}[,i], \mathbf{B}[,j])]$ Then, by using the predicted-ground truth bounding box pairings, the regression loss can be defined as follows: $RL(\hat{\mathbf{B}}, \mathbf{B}^*) = \sum_{k=0}^{n_{\hat{b}}-1} \frac{\lvert\hat{\mathbf{B}}[,k]-\mathbf{B}^*[,k]\rvert}{n}$ While this penalizes predicted boxes being far from ground truth boxes, an additional loss term is needed to ensure that boxes are classified correctly. As such, a focal classification loss using the paired ground truth bounding box classes can be defined as follows: $FL(\hat{\mathbf{C}}_{\text{dist}}, \mathbf{c}^*) = -\sum_{k=0}^{n_{\hat{b}}-1} \alpha_{fl}(1-\hat{\mathbf{C}}_{\text{dist}}[\mathbf{c}^*[k],k])^{\gamma} \log(\hat{\mathbf{C}}_{\text{dist}}[\mathbf{c}^*[k],k])$ This function is derived from the cross-entropy equation TM4.2.5, with an additional term to downweight easy examples. By combining the regression loss with the classification loss, the total bounding box loss function (to be used for optimization) can be defined as follows: $L(\hat{\mathbf{B}}, \mathbf{B}^*, \hat{\mathbf{C}}_{\text{dist}}, \mathbf{c}^*) = RL(\hat{\mathbf{B}}, \mathbf{B}^*) + FL(\hat{\mathbf{C}}_{\text{dist}}, \mathbf{c}^*)$
Sources	Ross and Dollár (2017), Team (2020)
Ref. By	IM3

Number	IM3
Label	**Model Training via ADAM Optimization** T_W
Input	θ_t, β_1, β_2, α_{gd}
Output	θ_{t+1}
Description	With the loss function established in IM2, a gradient-based method for updating a chosen system's parameters $\theta_\mathbf{t}$ can be setup as follows: $\mathbf{g_t} = \nabla_\theta L(\theta_\mathbf{t})$ Where gradient descent TM4.2.5 would simply subtract off this gradient to update the parameters $\theta_\mathbf{t}$, a modified form inspired by statistical moments can be used instead (called ADAM): $\mathbf{m_t} = \beta_1 \mathbf{m_{t-1}} + (1-\beta_1)\mathbf{g_t}$ $\mathbf{v_t} = \beta_2 \mathbf{v_{t-1}} + (1-\beta_2)\mathbf{g_t}^2$ $\hat{\mathbf{m}}_\mathbf{t} = \frac{\mathbf{m_t}}{1-\beta_1^t}$ $\hat{\mathbf{v}}_\mathbf{t} = \frac{\mathbf{v_t}}{1-\beta_2^t}$ $\theta_{\mathbf{t+1}} = \theta_\mathbf{t} - \alpha_{gd}\frac{\hat{\mathbf{m}}_\mathbf{t}}{\sqrt{\hat{\mathbf{v}}_\mathbf{t}}+10^{-8}}$ This optimization process in conjuction with the aforementioned loss function forms the backbone of the training pipeling.
Sources	Kingma (2014)
Ref. By	

Number	IM4
Label	**Model Evaluation via Mean Average Precision**
Input	$\hat{\mathbf{B}}$, $\mathbf{B}$, $\hat{\mathbf{c}}$, $\mathbf{c}$, τ_{IoU}
Output	mAP
Description	Using the classes for the predicted bounding boxes $\hat{\mathbf{c}}$ and the classes for the ground-truth bounding boxes $\mathbf{c}$, their corresponding bounding box sets can be partitioned into class-specific bounding boxes as follows: $\hat{\mathbf{B}}^{\mathbf{c}} = \{\hat{\mathbf{B}}[,i] \vert \hat{\mathbf{c}}[i] = c\}$ and $\mathbf{B}^{\mathbf{c}} = \{\mathbf{B}[,i] \vert \mathbf{c}[i] = c\}$, for all possible classes c Then, the true positives TP, false positives FP, and false negatives FN for a class can be defined as follows by using the IoU metric described in TM4.2.5: $TP(\hat{\mathbf{B}}^{\mathbf{c}}, \mathbf{B}^{\mathbf{c}}) = \sum_{k=0}^{n_{\hat{b}}-1} \max_i 1((\text{IoU}(\hat{\mathbf{B}}^{\mathbf{c}}[,k], \mathbf{B}^{\mathbf{c}}[,i])) \geq \tau_{IoU})$ $FP(\hat{\mathbf{B}}^{\mathbf{c}}, \mathbf{B}^{\mathbf{c}}) = \sum_{k=0}^{n_{\hat{b}}-1} (1 - \max_i 1((\text{IoU}(\hat{\mathbf{B}}^{\mathbf{c}}[,k], \mathbf{B}^{\mathbf{c}}[,i])) \geq \tau_{IoU})) = n_{\hat{b}} - TP$ $FN(\hat{\mathbf{B}}^{\mathbf{c}}, \mathbf{B}^{\mathbf{c}}) = \sum_{k=0}^{n_b-1} (1 - \max_i 1((\text{IoU}(\mathbf{B}^{\mathbf{c}}[,k], \hat{\mathbf{B}}^{\mathbf{c}}[,i])) \geq \tau_{IoU}))$ Using these values, the average precision values for each class can be calculate as described in GD2. The final mAP metric is then calculated by averaging these as follows: $\text{mAP}(\hat{\mathbf{B}}, \mathbf{B}, \hat{\mathbf{c}}, \mathbf{c}) = \sum_c \frac{AP(\hat{\mathbf{B}}^{\mathbf{c}}, \mathbf{B}^{\mathbf{c}})}{\vert c \vert}$
Sources	Team (2020)
Ref. By	

4.2.10 Input Data Constraints

Table 1 shows the data constraints on the input output variables. The column for physical constraints gives the physical limitations on the range of values that can be taken by the variable. The column for software constraints restricts the range of inputs to reasonable values. The software constraints will be helpful in the design stage for picking suitable algorithms. The constraints are conservative, to give the user of the model the flexibility to experiment with unusual situations. The column of typical values is intended to provide a feel for a common scenario. The uncertainty column provides an estimate of the confidence with which the physical quantities can be measured. This information would be part of the input if one were performing an uncertainty quantification exercise.

The specification parameters in Table 1 are listed in Table 2.

Table 1: Input Variables

Var	Physical Constraints	Software Constraints	Typical Value	Uncertainty
$p_{i,j}$	-	$p_{\min} \leq p_{i,j} \leq p_{\max}$	Any valid value	Unknown
w	$w \geq 0$	$w_{\min} \leq w \leq w_{\max}$	1920 pixels	Unknown
h	$h \geq 0$	$h_{\min} \leq h \leq h_{\max}$	1280 pixels	Unknown

Table 2: Specification Parameter Values

Var	Value
$p_{\min}$	0
$p_{\min}$	255
$w_{\min}$	24
$w_{\min}$	4096
$h_{\min}$	24
$h_{\min}$	4096

4.2.11 Properties of a Correct Solution

Rather than describe the properties of a correct solution in a table, I think It is more illustrative to describe them in words. The main properties of a correct solution in inference are that output bounding boxes must be non-overlapping and must have positive width, length, and height.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

R1: During inference, input images should be accepted in the following formats:

- PNG (Portable Net Graphics)
- JPEG (Joint Photographic Experts Group)
- TFRecord (TensorFlow Binary Record)
- PT (Pytorch Tensor File)

R2: During inference, input LiDAR pointclouds should be accepted in the following formats:

- PCD (Point Cloud Data)
- PLY (Polygon File Format)
- TFRecord (TensorFlow Binary Record)
- PT (Pytorch Tensor File)

R3: Given an input dataset, the system shall be designed to output a trained model that minimizes the combined bounding box loss IM2 on the training subset using the ADAM optimization method IM3.

R4: In inference, the system shall predict the set of dynamic object bounding boxes and visualize them on the LiDAR pointcloud that was inputted in.

5.2 Nonfunctional Requirements

NFR1: **Accuracy** The accuracy of the software shall be measured using the mAP metric IM4. The level of accuracy achieved by ESF shall be comparable to that of the BEVFusion method Liang et al. (2022) presented in the OpenPCDet repository Team (2020) (on the NuScenes Caesar et al. (2020) or Waymo Sun et al. (2020) dataset).

NFR2: **Understandability** As ESF is meant to be modified and used for research purposes, the classes and functions designed in ESF shall be fully documented in a way that minimizes the effort needed for a domain expert to understand them.

- Maintainability and Portability are inherited from the greater OpenPCDet repository Team (2020) and thus are not explicitly considered.

5.3 Rationale

This section provides rationales for the system constraints and assumptions as follows:

RN1: **Rationale for C1**: For this project, I decided to focus on learned methods specifically because I have seen learned methods for this task often presented at scientific conferences. Additionally, my main area of focus is in deep learning for computer vision, so I also chose this constraint to match my interests. With this in mind, OpenPCDet Team (2020) simplifies the process of setting up datasets and building models as it has much of the foundational code fully setup so its use was a logical constraint to make the problem more tractable in the 3 months of the term. It also provides implementations and benchmark values for reference methods like BEVFusion Liang et al. (2022), so it simplifies evaluation and verification of NFR1 as well.

RN2: **Rationale for C2**: Reference methods for this task are often evaluated on these benchmark datasets so enforcing their use simplifies the evaluation of NFR1.

RN3: **Rationale for C3**: Reference methods for this task are almost always trained with the ADAM optimizer, so allowing the use of other optimizers would simply add more complexity and ambiguity to the results obtained.

RN4: **Rationale for A1, A2, A3**: These assumptions are commonly made in open-source libraries like OpenPCDet Team (2020), so it makes sense to have them for this project as well.

6 Likely Changes

LC1: While the loss functions presented are commonly used and thus are unlikely to be removed, the greater loss function may have additional terms incorporated in IM2.

LC2: Depending on the dataset used for training and the data used for inference, the skew parameter in the intrinsic matrix may need to be considered A2.

7 Unlikely Changes

ULC1: The choice of ADAM as the optimization algorithm is unlikely to change as it allows for an easier comparison to state-of-the-art methods.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed,

the items in the column of that component that are marked with an "X" may have to be modified as well. Table 3 shows the dependencies of theoretical models, general definitions, data definitions, and instance models with each other. Table 4 shows the dependencies of instance models, requirements, and data constraints on each other. Table 5 shows the dependencies of theoretical models, general definitions, data definitions, instance models, and likely changes on the assumptions.

	TM4.2.5	TM4.2.5	TM4.2.5	TM4.2.5	TM4.2.5	TM4.2.5	TM4.2.5	TM4.2.5	GD1	GD2	IM1	IM2	IM3	IM4
TM4.2.5														
TM4.2.5														
TM4.2.5														
TM4.2.5														
TM4.2.5														
TM4.2.5														
TM4.2.5														
TM4.2.5						X	X							
GD1	X													
GD2						X	X	X						
IM1	X	X							X					
IM2			X		X									
IM3				X								X		
IM4					X	X	X	X		X				

Table 3: Traceability Matrix Showing the Connections Between Items of Different Sections

	IM1	IM2	IM3	IM4	4.2.10	R1	R2	R3	R4	NFR1	NFR2
IM1					X						
IM2											
IM3		X									
IM4											
4.2.10											
R1					X						
R2											
R3		X	X								
R4											
NFR1				X							
NFR2											

Table 4: Traceability Matrix Showing the Connections Between Requirements and Instance Models

	A3	A1	A2
GD1		X	X
GD2	X		
IM1		X	X
IM2			
IM3			
IM4	X		

Table 5: Traceability Matrix Showing the Connections Between Assumptions and Other Items

9 Development Plan

This section is not used at the moment, but it may be used in future iterations of this document.

10 Values of Auxiliary Constants

This section is not used at the moment, but it may be used in future iterations of this document.

References

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993.

Timothy D. Barfoot. *State Estimation for Robotics*. Cambridge University Press, 2017.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tingting Liang, Hongwei Xie, Kaicheng Yu, Zhongyu Xia, Zhiwei Lin, Yongtao Wang, Tao Tang, Bing Wang, and Zhi Tang. Bevfusion: A simple and robust lidar-camera fusion framework. *Advances in Neural Information Processing Systems*, 35:10421–10434, 2022.

T-YLPG Ross and GKHP Dollár. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017.

W. Spencer Smith and Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: http://dx.doi.org/10.1016/j.net.2015.11.008. URL http://www.sciencedirect.com/science/article/pii/S1738573315002582.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Àgerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

Mu Zhu. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2(30):6, 2004.