Eindhoven University of Technology

MASTER

Domain Knowledge-based Drivable Space Estimation

Muñoz Sánchez, Manuel

Award date:
2020

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Data Mining Group

Department of Integrated Vehicle Safety

Domain Knowledge-based Drivable Space Estimation

Master Thesis

Manuel Muñoz Sánchez

Academic Supervisor:
dr. Decebal Constantin Mocanu

TNO Industry Supervisors:
dr. Emilia Silvas
dr. Denis Pogosov

Graduation Committee:
dr. Decebal Constantin Mocanu
dr. Emilia Silvas
dr. Marwan Hassani
dr. Denis Pogosov

Eindhoven, September 2020

CONTENTS

***Abstract*—Automated vehicles are a desirable technology since they have the potential to significantly increase road safety and additionally bring several environmental benefits. To achieve fully automated vehicles, it is imperative that an accurate world model is available uninterruptedly. To that end, current research focuses on efficient extraction of landmarks collected by several on-board sensors, which can be used to localize the ego vehicle accurately given a previously constructed map of the environment. However, these approaches are susceptible to unreliable (e.g. due to sensor failure or adverse weather conditions) or outdated (e.g. due to temporary road work) sources of information, and if a sufficient number of landmarks is not detected, they fail to meet the stringent requirements of a typical automated driving application. In this work, a novel method for robust drivable space estimation from a limited number of inputs is proposed, which ensures the availability of a minimal road model that is consistent with the current driving situation and that can be used for in-lane localization. To model the drivable space, semantic information of high-level sensed objects and assumptions based on domain knowledge are used to estimate the drivability of the space surrounding each object. These estimations are modeled as a probabilistic graph to account for the uncertainty of information from different sources, and an optimal spatial configuration of its elements is achieved via graph-based Simultaneous Localization and Mapping (SLAM) optimization. The robustness of the proposed method towards missing or unreliable inputs of different qualities has been tested extensively in a simulation environment. The results achieved in our simulations show improved robustness towards these challenging conditions, and the recovered drivable space allows for accurate in-lane localization of the ego vehicle, even in extreme cases where no prior knowledge of the road network is available.**

I. Introduction

Automated vehicles (AVs) have several benefits that have made them a research focus in recent years. Not only do AVs have the potential to increase road safety [1], but they also have additional benefits such as higher traffic throughput [2], fewer CO_2 emissions and better fuel efficiency [3].

A typical AV architecture consists of three main modules: world modeling, planning and control. To plan and carry out the most suitable course of action while sensing the environment, it is crucial to construct a coherent world model that accurately captures the physical environment surrounding the vehicle. Such a world model consists of several components, including a road model and the ego location within it. To achieve higher levels of automation (i.e. levels 3 and higher of the SAE scale [4]) it is of utmost importance to have an accurate representation of the world model, and therefore an accurate road model and ego location are required at all times. A common approach in automated driving (AD) to achieve the required accuracy in localization and road model is to first build detailed maps offline, mainly considering the geometry of road elements and surrounding static objects. Many techniques are available to extract road geometry and construct these maps, including (i) analysis of aerial images [5], (ii) recorded GPS trajectories [6]–[9], or (iii) deployment of probe vehicles equipped with various perception sensors such as cameras and LiDARs [10]. After the detailed map containing all the static features of the environment has been constructed, detected landmarks when driving are matched against the previously built map to locate the vehicle within it. These approaches allow accurate localization, but have several limitations. First, if the environment changes (e.g. construction work, temporarily blocked lanes, or newly constructed roads), the previously known map becomes outdated and another mapping process is required. Furthermore, depending on the level of detail in the constructed map this might not be sufficient, especially for urban environments in which centimeter level localization accuracy is required. In turn, if a very detailed map is used, this typically incurs a heavy computational and storage burden [11] or if the map is retrieved *on-the-fly*, long-lasting connectivity issues become a challenge.

Alternative map formats have been proposed to address these limitations [12]–[15]. These map formats group all the elements of the environment into different layers, which allows dynamically adjusting the level of detail required by the application and enriching static map information with highly dynamic objects such as other road users. These map formats rely on the assumption that the perception module is able to provide sufficient inputs to generate the world model. However, upon sensor failure (e.g. connectivity issues impeding the timely retrieval of an updated map) or other complications (e.g. adverse weather conditions complicating the detection of nearby landmarks for localization), the AV is unable to generate an accurate world model and cannot continue to operate.

To overcome these limitations, this work introduces the foundations of a novel method that allows estimation of the drivable space from a scarce number of inputs, even when some sources of information are unreliable or unavailable. To estimate the surrounding drivable space, the proposed method exploits semantic information of sensed objects and AD domain knowledge (e.g. an observed vehicle is most likely driving on a road, there are roads near traffic lights, etc.) to construct a probabilistic road model from the ego point of view. The sensed objects that are the inputs to our algorithm are high-level landmarks resulting from the fusion of sensor data from multiple sensors and other sources of information (i.e. a map). Since the method estimates the drivable space solely from already fused sensor inputs, it can be easily incorporated in the architecture of a typical AV, as shown in Figure 1. Introducing this component in the architecture of the system ensures the availability of a minimal road model, which can be used for accurate estimation of the ego position with respect to the reference line of the lane it is currently occupying. In this work, this lane-relative position estimation is referred to as *in-lane* localization. Using the reconstructed drivable space for accurate in-lane localization allows for the continued automated operation of the vehicle even under challenging conditions.

This paper is structured as follows. The remainder of this section is dedicated to introduce the research questions and highlight the contribution of this work. Section II presents an overview of related techniques and required background knowledge. Section III elaborates on the method proposed in

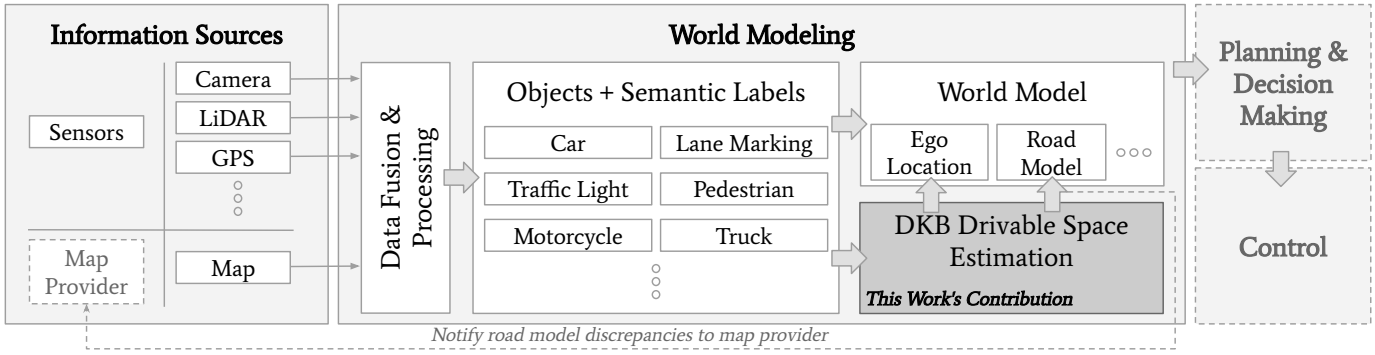


Fig. 1. Typical architecture of an AV (simplified) including our proposed Domain Knowledge-based (DKB) drivable space estimation. Raw input data is processed to construct a world model, which is used to plan and execute the most suitable course of action. Our proposed DKB drivable space estimation ensures availability of a road model and ego localization within it, and in future work it can be used to notify the map provider of outdated road information immediately. Note that the components depicted with a dashed line are currently not part of the pipeline in our experiments.

this work. Section IV describes in detail how the experiments are carried out, and the different performance metrics used to evaluate the proposed method. Section V is dedicated to present and discuss the experiment results. Finally, Sections VI and VII conclude on the main findings and propose interesting research directions for future work. Additionally, several appendices are available, which provide supplementary material (e.g. results of all our experiments) and detailed explanations (e.g. motivation for choice of simulator).

A. Research Questions

This work aims to develop a method that is able to combine data from multiple sensors and sources of information with AD domain knowledge to estimate the drivable space around the ego and provide in-lane localization. Furthermore, this method should be robust to sensor failures and adverse conditions to ensure availability of the drivable space. To that end, we investigate the following:

1) How can sensor data of different types and qualities from the ego vehicle be incorporated with AD domain knowledge to reconstruct the drivable space?
2) How can the ego vehicle be localized in the reconstructed drivable space?
3) How does incorporating AD domain knowledge improve robustness of drivable space reconstruction and ego localization towards missing or unreliable information sources?

B. Contribution

The contribution of our work is threefold:

1) Building a road model not only from geometric features of detected landmarks, but exploiting semantic information from high-level sensed objects and AD domain knowledge to ensure availability of a minimal drivable space which is consistent with the current driving situation.
2) Accurate estimation of the ego position with respect to the reference lane of the lane it is currently occupying (i.e. in-lane localization), using the proposed drivable space reconstruction.
3) Enhanced robustness towards missing or unreliable information sources by not linking the position of the ego vehicle to an existing map for localization, but building the map online in a Simultaneous Localization and Mapping (SLAM) framework, thus ensuring availability of a minimal road model that can be used for in-lane localization.

II. Background & Related Work

This section provides an overview of techniques commonly employed for drivable space modeling in AD. Section II-A describes different space representations used by mobile robots to construct a world model. Next, Section II-B elaborates on techniques used specifically for lane-level model generation for AD applications. Finally, in Section II-B, an overview of SLAM is presented, which is a method that allows a robot to navigate in unknown environments, and therefore it is robust to situations where an up-to-date map of the environment is not available.

A. Drivable space

Drivable (or driving) space is a minimal representation of the world required by an AV to have an understanding of its surroundings and plan the most suitable course of action. For AD applications, this minimal representation must accurately capture all the relevant elements that allow a vehicle to operate safely, and being able to capture as much of this information as possible in an efficient manner has been a research focus in recent years. For example, the authors of [16] distinguish between three types of drivable space: *grid*, *feature* and *topological* space.

Grid maps segment the entire surrounding space into a grid, and each grid cell encodes the probability of that cell being occupied. This space representation was initially proposed by Elfes in 1987 for robotics research [17], and it is widely used in AV applications. Current research using uniform grids typically considers a cell size of 20 cm [18]–[20] since it is small enough to capture all the relevant elements in the driving environment. The main disadvantage of this space representation is its heavy storage and computational burden, although alternative representations addressing this issue exist, such as quadratic trees [21].

On the other hand, *feature* maps do not cover the entire space, and only some objects of the environment are captured, which are typically represented by their poses and shapes. Feature maps are often used in SLAM [22]–[25]. In AD applications that use feature maps, the features represent traffic elements, such as lane markings, other vehicles, or pedestrians.

Topological maps focus on the relationship between elements in the map, rather than their exact positions. For instance, from element A we can reach elements B and D, but not C. A purely topological space is not suitable for AD, since it does not capture the detailed geometric information of the environment required by an AV, and extracting this geometric information from a topological representation remains a challenge. Topological representation is most commonly used to capture the connectivity between road and lane segments in navigation maps, or as an intermediate representation in approaches that generate a grid or feature space, as is the case in graph-based SLAM [26], [27].

B. Maps & Drivable Space Models for Automated Driving

An overview of drivable space estimation methods specifically designed for AD applications is provided in this section. Section II-B1 reviews the most common sensors equipped in an AV that are used for this task. Section II-B2 provides an overview of how sensor data is exploited to construct detailed lane-level road models. Finally, Section II-B3 describes how state of the art models enrich static information from road models with all available information about the driving environment.

B.1 Sensors for Lane-Level Localization & Mapping

A typical AV may be equipped with several sensors that extract information of its environment to generate a model of the drivable space and additionally localize the vehicle in it. The authors of [10] differentiate two categories of sensors: *position* and *perception* sensors.

Position sensors include systems such as Global Navigation Satellite System (GNSS) or Inertial Navigation System (INS). GPS is a common type of GNSS which is cheap and easy to equip in any vehicle, however, it is not sufficiently accurate for AD applications. For instance, the U-blox EVK-6T has meter-level accuracy [28]. On the other hand, INS sensors are accurate (e.g. NovAtel SPAN-CPT or OXTS RT3000 are accurate to centimeter level [29], [30]), but they are too expensive to equip in series-production passenger vehicles.

Perception sensors include systems such as lasers scanner, cameras and radars. Laser scanners (i.e. LiDARs) are able to capture detailed measurements of the physical objects in the environment, but these sensors are expensive and generate vast amounts of raw data that need to be processed efficiently to take advantage of them in real-time. On the other hand, cameras are low-cost and have received much attention in the computer vision community for many tasks in recent years, including lane extraction [31]. However, cameras are sensitive to weather conditions, and depending on the setup suffer from some additional disadvantages: single camera systems do not provide any depth information, stereo camera systems have limited field of view and multi-camera systems are computationally heavy [10]. Finally, radars are widely used sensors for AD, since they are low-cost and additionally provide accurate object detection without being affected by weather conditions like cameras and LiDARs. However, their lower resolution in object detection compared to LiDARs does not make them a desirable technology for detailed lane-level mapping.

B.2 Road Geometry Extraction for Lane-Level Mapping

Depending on the sources of information considered, the various approaches to lane-level road geometry extraction can be classified in three categories: *trajectory-based*, *point cloud-based* and *vision-based* [10].

Trajectory-based methods extract lane centerlines from position sensors, such as single GPS trajectories [6], or multiple crowdsourced GPS trajectories [7], [9], [32]. These methods are computationally cheap, but have the disadvantage of reaching only meter-level accuracy.

Alternatively, *point cloud-based* methods extract lane boundaries instead of lane centerlines from laser sensor data. These methods either (i) extract lane boundaries directly from point cloud data (e.g. setting reflectivity thresholds [33], [34] or using deep learning approaches such as convolutional neural networks [35]); or (ii) first generate a georeferenced feature image and apply a feature extractor such as Hough transform [33].

Vision-based methods extract lane boundaries from camera data using either (i) *feature-based* methods, which exploit lane marking features such as color or width [36]; (ii) *model-based* methods, which focus on the mathematical model of lanes [37]; or (iii) *deep learning* approaches, which have become popular in recent years due to their high lane detection accuracy in complex scenarios [38].

Once the geometry of lanes is extracted, a lane-level road network is modeled. Automation of lane-level road network modeling from geometric features is a challenging task, and it is currently a research focus [10], [39], [40]. Several possibilities for lane modeling are available, including simple straight lines, arc curves, or more elaborated types of curves such as splines or clothoids. Straight lines require little computation and are the preferred approximation when the road segment is nearly straight. Arc curves are commonly used in circle-like intersections, such as roundabouts, due to low computational requirements compared to other curves. More elaborated curves offer higher flexibility for irregular curves, but it is often costly to create them.

Standards exist providing guidelines for describing elements of the map accurately for autonomous vehicles. Some examples of such standards are OpenDRIVE [41], RNDF [42], and NDS [43] or its reduced non-commercial release Open Lane Model (OLM) [44]. These standards are followed to create what is now known as HD maps. As opposed to traditional navigation maps where road-level details suffice, HD maps include information at the centimeter scale, and capture details

such as lane boundaries, steepness of the road, and much more. Despite their accuracy, HD maps typically only capture static information, and fail to model the highly dynamic environment in which self driving vehicles operate, hence HD maps are a required input but not the only source of information required by a fully automated vehicle. This topic has been a research focus in recent years, in order to efficiently deal with a rapidly changing environment, while still maintaining the level of detail required for AD.

B.3 Enhancing Static Maps: Layered Models

To overcome the limitations posed by purely static maps and create a more representative model of the environment, numerous models extending base static maps have been proposed. These models are significantly different from each other, however, they have one common feature: their elements are grouped into layers, typically in increasing level of complexity. For instance, HERE, a company that provides mapping and location services, groups this information in three main layers [12]: a base *mapping* layer contains detailed road geometry information (i.e. the HD map), a *dynamic* layer captures short-term changes in the environment, and an *analytics layer* captures how humans typically behave in any given road segment. Lyft is a ride-sharing company investing in AD research, and their maps are organized into 5 layers [13]: *base*, *geometric*, *semantic*, *map priors*, and *real-time* layers. The authors of [14] go one step further and model the driving environment using seven layers: *road*, *traffic*, *road-lane*, *lane*, *map feature*, *dynamic* and *intelligent decision support*. Another good example of layered maps for AD are Local Dynamic Maps (LDMs), resulting from the SAFESPOT project [15], which model four layers: *permanent static*, *transient static*, *transient dynamic*, and *highly dynamic*. LDMs haven been widely used in later research [45].

These layered models capture the environment with a high level of detail, however, they are heavily reliant on being able to retrieve accurate environment information required for such models. Upon sensor failure or any other complications (e.g. retrieval of an up to date map due to connectivity delays, or adverse weather conditions complicating the detection of nearby landmarks), the reconstructed world model might not be consistent with the current driving situation, and the AV cannot continue to operate safely. Thus, there is still a need for models that allow robust reconstruction of the drivable space even when they are presented with outdated or incomplete information about the environment.

C. Simultaneous Localization and Mapping

To robustly localize a mobile robot in a map, which could be (partially) unknown, SLAM-based approaches are commonly used. This section provides an overview of the foundation of SLAM, which assumes the map is unknown a priori and it is estimated purely from sensor measurements.

SLAM solves the problem of constructing a map of an unknown environment at the same time as estimating the location of a mobile robot within the constructed map. This problem is especially challenging since localizing the robot requires an accurate representation of the map, and to construct an accurate map, a good estimate of the robot's location must be available. SLAM has received much attention in the last few decades, and a wide variety of techniques exist to address this problem. The different techniques can be classified into *filtering* and *smoothing* approaches.

Filtering approaches solve what is commonly referred to as the *online SLAM* problem: only the map and most recent robot pose is estimated, as opposed to the *full SLAM* problem, where estimates for both the entire robot path and the map are obtained. Examples of filtering techniques are: Kalman filter (KF), extended Kalman filter (EKF), unscented Kalman filter (UKF), information filter (IF), or particle filter (PF) [46]. Filtering techniques are not the focus of this work, thus the curious reader is referred to [46] for a comprehensive introduction.

On the other hand, smoothing approaches address the *full SLAM* problem, i.e. the full robot trajectory and map is estimated. Smoothing techniques used to be more common in an *offline* setting, i.e. the map and trajectory estimation is performed in batch, with all the previously measured data. However, advances in the SLAM community allowed for efficient incremental smoothing approaches [47] [48] that can be run *online* while still estimating the entire robot trajectory.

C.1 Formulation of the (full) SLAM Problem

The SLAM problem is typically formulated probabilistically to account for the uncertainty associated with the different variables involved in the process. To solve the SLAM problem, one seeks to estimate a posterior probability, p, given by

$$p\left(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{x}_0\right) \tag{1}$$

where $\mathbf{x}_{1:T} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$ denotes a sequence of robot states up to the most recent time T as it moves in an unknown map $\mathbf{m}$; and $\mathbf{u}_{1:T} = \{\mathbf{u}_1, ..., \mathbf{u}_T\}$ and $\mathbf{z}_{1:T} = \{\mathbf{z}_1, ..., \mathbf{z}_T\}$ denote T odometry and perceptions of the environment (i.e. measurements) acquired in the process. Additionally, $\mathbf{x}_0$ represents an initial estimate of the robot state required to link the robot to the map.

Directly computing p in (1) is not tractable due to the high dimensionality of the problem. Instead, the underlying temporal and spatial structure of the SLAM process is exploited to conveniently model it using different types of probabilistic graphical models (PGMs). Two PGMs commonly used to model the SLAM process are (1) Dynamic Bayesian Networks (DBNs) [49], which highlight the temporal structure of the SLAM process and are typically used to model filtering approaches, and (2) factor graphs [50], which instead highlight the spatial structure and allow the joint probability distribution given by (1) to be decomposed as the product of several single factors, given by

$$\begin{aligned} &p\left(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{x}_0\right) \\ &\qquad = p(\mathbf{x}_0) \prod_{t=1}^{T} p\left(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m}\right), \end{aligned} \tag{2}$$

where $p(\mathbf{x}_0)$ is a prior on the initial robot state, and $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_t)$ and $p(\mathbf{z}_t \mid \mathbf{x}_t, \mathbf{m})$, are given by the robot motion and sensor measurement models for any time $1 \leq t \leq T$.

Such a factorized expression can be reformulated as a nonlinear least squares optimization problem, for which efficient iterative techniques exist to approximate an optimal solution [47], [51].

C.2 SLAM as a Least Squares Optimization Problem

The SLAM problem can also be presented in the so-called graph-based formulation as initially proposed by Lu and Milios in 1997 [52]. SLAM algorithms based on this formulation consist of two main components as shown in Figure 2. First, a front-end processes sensor data to construct

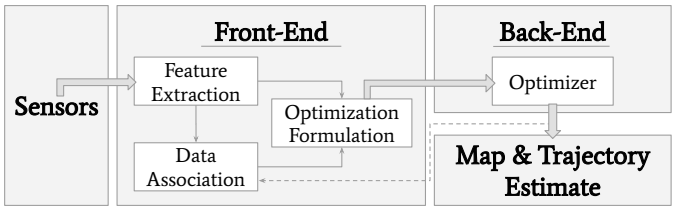


Fig. 2. Architecture of a typical optimization-based SLAM system. The front-end processes data from multiple sensors to extract landmarks and establish data associations, while the back-end receives this information and estimates the most likely map and robot trajectory.

a graph $\mathbf{G}$ consisting of $|\mathbf{N}|$ nodes and $|\mathbf{E}|$ edges, where a node $\mathbf{n} \in \mathbf{N}$ represents either robot or landmark poses, and an edge $\mathbf{e}_{ij} \in \mathbf{E}$ between the i^{th} and j^{th} nodes of the graph, $\mathbf{n}_i$ and $\mathbf{n}_j$, represent an expected relative spatial constraint, $\hat{\mathbf{z}}_{ij}$, which is given by a sensor measurement $\mathbf{z}$ or a control input $\mathbf{u}$. Once the graph representing the optimization problem is built, graph-based SLAM algorithms rely on a back-end optimizer to find the configuration of nodes $\mathbf{N}^*$ that is most consistent with the provided observations (edges). That is,

$$\mathbf{N}^* = \underset{\mathbf{N}}{\operatorname{argmin}} \sum_{\mathbf{e}_{ij} \in \mathbf{E}} \varepsilon_{ij}^T \mathbf{\Omega}_{ij} \varepsilon_{ij}, \tag{3}$$

where ε_{ij} denotes the deviation between the expected and actual measurement of the i^{th} and j^{th} nodes, $\mathbf{n}_i$ and $\mathbf{n}_j$, defined as

$$\varepsilon_{ij} = \varepsilon(\mathbf{n}_i, \mathbf{n}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}; \tag{4}$$

and $\mathbf{\Omega}_{ij}$ denotes the information matrix of the constraint between nodes $\mathbf{n}_i$ and $\mathbf{n}_j$, which can be intuitively seen as the certainty on the constraint. The higher the values in the information matrix, the more confident we are about the constraint, and therefore ε_{ij} has a higher influence during the optimization.

This formulation did not become popular in its early years due to the complexity of the optimization problem. However, advances in sparse linear algebra and iterative error minimization methods provided efficient tools for solving graph-based SLAM, and in recent years has been researched extensively.

An illustrative example of how to construct a graph representing the optimization problem is provided in Figure 3. In this example, two vehicle states, $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, and a landmark, $\mathbf{m}_1$ are converted into three nodes of the graph, $\mathbf{n}_i$, $\mathbf{n}_j$, and $\mathbf{n}_k$, which can be related to each other by sensor measurements or other available information. For instance, after applying

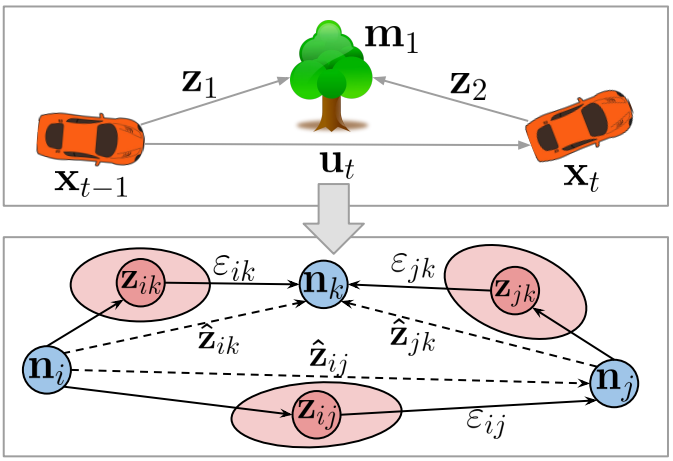


Fig. 3. Example of how a graph-based problem can be formulated from a simple setting in which an AV observes a landmark at different times after applying some control.

control $\mathbf{u}_t$ to the vehicle with state $\mathbf{x}_{t-1}$, it is expected that $\mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{u}_t$. This expectation is introduced as a constraint of our graph in the shape of a virtual measurement $\hat{\mathbf{z}}_{ij}$ relating $\mathbf{n}_i$ and $\mathbf{n}_j$, and associated information matrix $\mathbf{\Omega}_{ij}$ to represent how certain one is about this constraint. However, there might be some error with respect to this expected measurement, ε_{ij}. Minimizing the sum of squared errors, as defined in (3), allows estimation of a map and robot trajectories that are most consistent with the specified constraints.

C.3 (Robust) Data Association

A critical component in SLAM algorithms is *data association*, which is the process of determining if an observed landmark has already been observed in the past. More specifically, determining whether two measurements $\mathbf{z}_k$ and $\mathbf{z}_{k'}$ originated from observing the same landmark $\mathbf{m}_j$. This process of determining which observations originated from which landmarks is also commonly referred to as *correspondence* estimation, and is typically part of the front-end in the architecture of any SLAM algorithm, as illustrated in Figure 2. This is a challenging task because of the inherent uncertainty associated with the robot trajectory and sensor measurements, and a single erroneous association can yield a poor quality estimate of the map and robot poses. Thus, the problem of establishing correct data associations has been studied extensively in the SLAM community, and many different approaches to data association exist nowadays.

In its simplest form, individual measurements can be processed independently to establish data associations through Maximum Likelihood (ML) estimation [53], sometimes referred to in literature as Nearest Neighbor [54]. Through individual compatibility tests [55], unlikely associations are discarded to only estimate over the most likely ones.

Alternatively, it is possible to process measurements in batch to establish data associations. Sequential Compatibility Nearest Neighbor [56] is similar to the methods previously described, but it assumes mutual exclusion of measurements (i.e. no two measurements in the same batch correspond to the same landmark). Joint maximum likelihood seeks to find the maximal set of data associations that maximizes the product of the likelihoods [55]. Joint compatibility (JC) branch and bound methods refer to specific search methods to find the ML data association sets that are jointly compatible [55], [56]. Other approaches deal with the ambiguity of data associations by considering all reasonable data associations (i.e. multiple hypothesis tracking), instead of only choosing the most likely one. The effectiveness of this approach is visible in FastSLAM [57], a filtering method founded on the idea of multiple hypothesis tracking. Further alternatives consider making the right data associations more important than doing anything at all, and therefore delay the decision until sufficient evidence is available, as is the case in [58], which additionally allows to revise past data associations.

All the methods presented in this section so far reduce the probability of establishing false positive data associations between different landmarks, but they can still occur and yield a poor estimation of the map and robot trajectory. In optimization-based SLAM algorithms, this would amount to constructing a graph, as described in the previous section, with the wrong topology, containing edges that are outliers. An interesting approach to address this issue is proposed in [59], introducing in the optimization a new variable for each data association constraint in the graph, which determines if the edge is a potential outlier. The main drawback of this method is the increased computational burden introduced by adding a new variable for each data association. The authors of [60] generalize this idea by proposing a closed form solution that dynamically scales the information matrix in (3) to diminish the effects of wrong data associations, significantly increasing convergence speed without sacrificing accuracy.

III. Proposed Method

To construct a road model and estimate the host vehicle position within it, in a way that is resilient to erroneous information in the inputs, this section proposes a novel method for drivable space modeling as a probabilistic graph that accounts for the uncertainty of the different sources of information.

As shown in Figure 4, the core of the method is to exploit object semantic information and domain knowledge-based assumptions (e.g. an observed vehicle is most likely driving on a road) to hypothesize if the space surrounding detected objects is drivable. These hypotheses are referred to as *drivability projections* or simply *projections*. Next, the projections are used to formulate a graph-based SLAM problem and obtain the most likely configuration of the drivable space. Finally, the individual projections are merged to obtain our estimation of the drivable space, which can be used as a prior on the drivable space for estimation in the future.

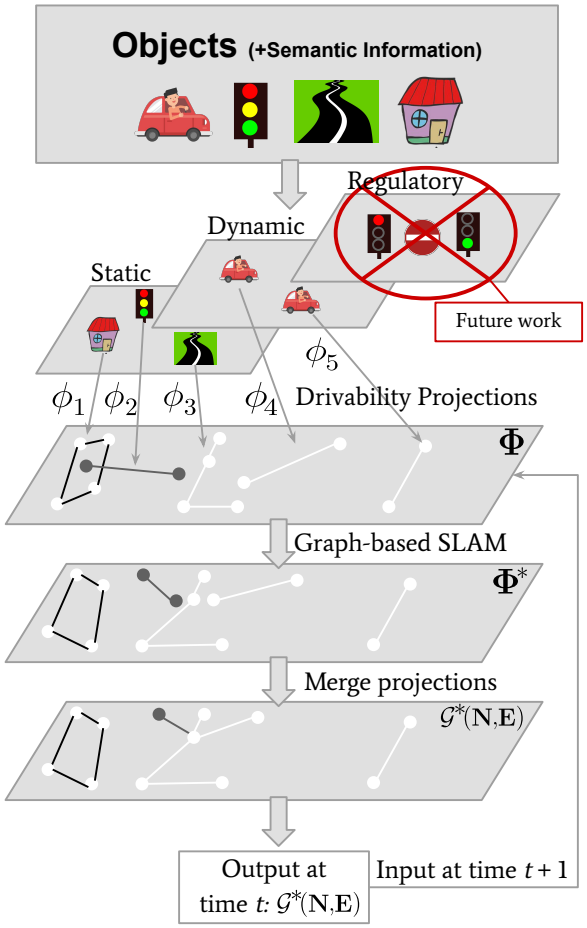


Fig. 4. Overview of the proposed method. Drivability projections are extracted from input objects with semantic labels. Then, an optimal spatial configuration is achieved via SLAM. Finally, the projections are merged to output a drivable space estimation at time t. Projection colors indicate probability of drivability of a road segment and probability of existence of a node, ranging from 0 (black) to 1 (white).

A. Drivable Space

The drivable space is modeled as an undirected graph, $\mathcal{G}(\mathbf{N}, \mathbf{E})$, with $|\mathbf{N}|$ nodes and $|\mathbf{E}|$ edges. Any node $\mathbf{n} \in \mathbf{N}$ represents a point in 2D Euclidean space and is characterized by several attributes, as given by

$$\mathbf{n} = (x, y, \kappa, \tau, p^e), \tag{5}$$

where x and y denote the node's longitudinal and lateral positions, $\kappa \in \{0, 1\}$ indicates if $\mathbf{n}$ is a *knot* separating two road segments, τ represents a *relative importance* used to weight $\mathbf{n}$ when merged with other nodes, and $p^e \in (0, 1]$ represents its *probability of existence*. If p^e is zero for any node, such node is not modeled in the drivable space. Furthermore, an edge $\mathbf{e}_{ij} \in \mathbf{E}$ between nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ encodes the probability that there is a drivable surface between $\mathbf{n}_i$ and $\mathbf{n}_j$, *probability of drivability*, p^d_{ij}

$$\mathbf{e}_{ij} = (\mathbf{n}_i, \mathbf{n}_j, p^d_{ij}). \tag{6}$$

Thus, the drivable space can be characterized by $\mathbf{N}$ and $\mathbf{E}$, with each of their elements defined as in (5) and (6).

To construct this drivable space model, let $\mathbb{O}$ denote the set of all possible objects (e.g. other vehicles, detected camera-based lane model, etc.) that are available to the ego vehicle, and $\mathbb{D}$ the possible drivable spaces representing the environment around the ego. The aim is to develop a robust method which from a limited number of objects, $O \subseteq \mathbb{O}$, can construct the drivable space, $\mathcal{G}^*(\mathbf{N}, \mathbf{E}) \in \mathbb{D}$, that most accurately represents the real drivable space. For example, the drivable space derived from objects detected by an in-vehicle camera sensor will be limited and incomplete. Hence, by combining multiple sources of information, shortcomings of one source of information are compensated by others and a better representation of the drivable space can be recovered.

For ease of readability, this work uses some non-standard terminology and notation, which are introduced in the sections where they are first used. An overview of all relevant terminology and notation is provided in Appendix A for clarity.

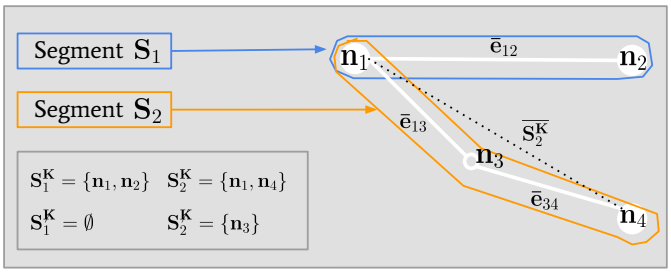


Fig. 5. Illustrative example of terminology and notation used in this work. Filled nodes indicate *knots* (i.e. $\mathbf{n}_1$, $\mathbf{n}_2$ and $\mathbf{n}_4$), while hollow nodes (i.e. $\mathbf{n}_1$) are used to preserve geometric information.

When referring to a property $\psi \in \{x, y, \kappa, \tau, p^e\}$ of the i^{th} node of the drivable space, it is denoted as $\mathbf{n}_i^{\psi}$, with one exception in which two properties are referred simultaneously, $\mathbf{n}_i^{xy}$ to denote the node coordinates. Furthermore, $\bar{\mathbf{e}}_{ij}$ denotes the line segment connecting points $\mathbf{n}_i^{xy}$ and $\mathbf{n}_j^{xy}$, while $\vec{\mathbf{e}}_{ij}$ represents the vector from $\mathbf{n}_i^{xy}$ to $\mathbf{n}_j^{xy}$. A *segment* $\mathbf{S}$ is defined as a subgraph of the drivable space with $|\mathbf{S^N}|$ nodes and $|\mathbf{S^E}|$ edges, which represents a maximal path between two *knots* $\mathbf{n}_i$ and $\mathbf{n}_j$ where $\forall_{\mathbf{n}_p \in S^{\mathbf{N}} \wedge i \neq p \wedge j \neq p}(\mathbf{n}_p^{\kappa} = 0)$. The two knots of a segment $\mathbf{S}$ are referred to as $\mathbf{S^K}$, and all intermediate *non-knots* $\mathbf{S^{\not K}}$. Finally, for any $\mathbf{S}$, a straight segment between its two knots $\mathbf{n}_1, \mathbf{n}_2 \in \mathbf{S^K}, \mathbf{n}_1 \neq \mathbf{n}_2$ is denoted $\overline{\mathbf{S^K}}$, and the vector from $\mathbf{n}_1$ to $\mathbf{n}_2$ is given by $\vec{\mathbf{S}}^{\mathbf{K}}$. A simplified example illustrating this notation is shown in Figure 5.

B. Drivability Projections

The first step in our method is to exploit an object's semantic information to extract projections about the drivability of its surroundings. To that end, *projection functions* $f^p : \mathbb{O} \mapsto \mathbb{D}^a$, $a \geq 1$, are defined to map each input onto one or more drivability projections $\mathbf{\Phi} = \{\phi_1, ..., \phi_a\}$.

A projection ϕ has the same shape as the drivable space we seek to recover. For ease of notation, let $\phi^{\mathbf{N}}$, $\phi^{\mathbf{E}}$ and $\phi^{\mathbf{S}}$ denote the nodes, edges and segments in ϕ. Although the format of projections is the same as the drivable space, we distinguish between two types of projections: drivable and non-drivable. The sets of all drivable and non-drivable projections are denoted as $\mathbb{D}^+$ and $\mathbb{D}^-$, respectively. The following conditions are imposed to each type of projection:

$$\phi \in \mathbb{D}^+ \implies \forall_{\mathbf{e}_{ij} \in \phi^{\mathbf{E}}} \; p_{ij}^d \in (0, 1], \text{ and} \tag{7}$$

$$\phi \in \mathbb{D}^- \implies \forall_{\mathbf{e}_{ij} \in \phi^{\mathbf{E}}} \; p_{ij}^d = 0. \tag{8}$$

Projection functions are defined for the five types of objects that are considered as input to our algorithm: buildings, vehicles, traffic lights, a lane-level road network, and the ego vehicle's lane. More details on how these objects are obtained are provided in Section IV. The remainder of this section is dedicated to elaborate on the two types of projection functions that are defined: *domain knowledge-based* and *empirically extracted*.

The main difference between these two projections, is that for projections $\phi \in \mathbb{D}^+$, $p_{ij}^d = 1$ for all $\mathbf{e}_{ij} \in \phi^{\mathbf{N}}$ if ϕ is domain knowledge-based, while $p_{ij}^d \in (0, 1]$ for all $\mathbf{e}_{ij} \in \phi^{\mathbf{N}}$ if ϕ is empirically extracted, and the exact value of p_{ij}^d is determined, as the name indicates, empirically.[1]

B.1 Domain Knowledge-based Projections

Domain knowledge-based projections result from assumptions that can be made about an object given its semantic information (e.g. the type of object it is). These projections are defined for buildings, vehicles, roads and the ego lane, as shown in Figure 6.

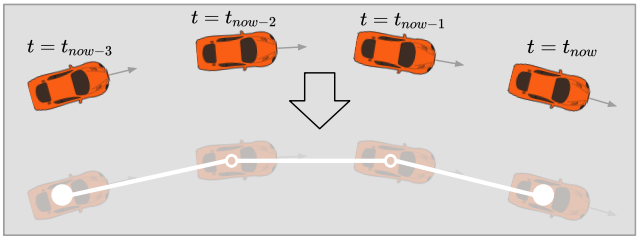
a) Vehicles: If a vehicle is observed at time t on location $\mathbf{v}_t = (x_t, y_t)$, it is reasonable to assume that $\mathbf{v}_t$ is part of a road and therefore drivable. Furthermore, if the next time the same vehicle is observed on $\mathbf{v}_{t+\Delta t}$, it is reasonable to assume that a vehicle can drive from $\mathbf{v}_t$ to $\mathbf{v}_{t+\Delta t}$, provided that Δt is not too large. Thus, for each vehicle, its observed trajectory allows us to define a drivable projection $\phi \in \mathbb{D}^+$, with one node for each observed vehicle position, and an edge between nodes resulting from consecutive positions (Figure 6a).

b) Buildings: If a building is observed, clearly the space it occupies is not drivable. Thus, a building's projection function generates one $\phi \in \mathbb{D}^-$. The nodes and edges of building projections indicate the building perimeter (Figure 6b).

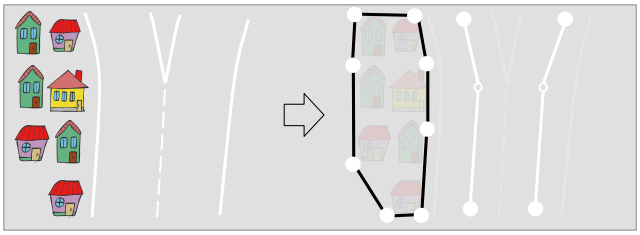
c) Roads: If information is available about the road network (e.g. from a map), this knowledge is translated into a drivable projection (Figure 6b). Since the road network extracted from the simulator is also defined as an undirected graph, its nodes and edges are directly considered in the projection.

d) Ego lane: If information about the ego lane is available, a drivable projection is extracted from the subset of nodes and edges of the road network that are part of the lane the ego vehicle is currently occupying (Figure 6c).

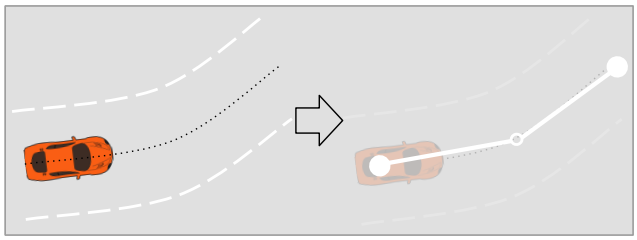
[1]There is a mismatch in the current implementation, which also assigns $p_{ij}^d = 1$ to empirically extracted projections (as can be seen in Figures 8b and 10). However, this difference does not affect classification results in our experiments, since estimation is performed for one timestep and the classification threshold is not higher than the empirically extracted probability.



(a) Example of projection extracted from a detected vehicle and its observed past trajectory.



(b) Example of projection extracted from buildings and road lanes.



(c) Example of projection extracted from ego lane.

Fig. 6. Examples of domain-knowledge based projections extracted from buildings, lanes, and vehicles.

B.2 Empirically Extracted Projections

Some domain-knowledge assumptions, as is the case with traffic lights, are not enough to fully define the projection of an object. If a traffic light is observed, it is reasonable to assume that there are roads nearby that traffic light. However, it is not possible to determine exactly the location or shape of those roads. Nonetheless, traffic lights are located at junctions and it is likely that the roads around them have some patterns.

Traffic lights generate two projections, one drivable and one non-drivable. The non-drivable projection represents the perimeter of the traffic light pole, while the drivable one represents an average drivability around a traffic light and is extracted as follows.

Given one traffic light $\mathbf{l} = (x, y, \theta)$ on point (x, y) and with heading angle θ degrees, all road segments in a 60x60m area centered at $\mathbf{l}^{xy}$ are extracted. Next, the traffic light and all surrounding road segments are rotated $-\theta$ degrees around $\mathbf{l}^{xy}$ to ensure all the lights are oriented equally. The space is discretized into a grid with cell resolution 3.5m (the average lane width) and each cell assigned a value of 1 if it intersects with a road segment, or 0 if it does not. This process is illustrated in Figure 7 for one traffic light. After performing this process with all the traffic lights, the cells from all the grids are averaged to determine the probability

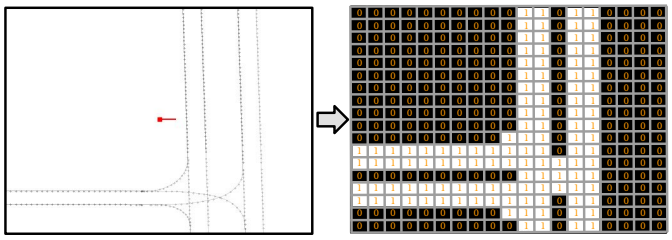
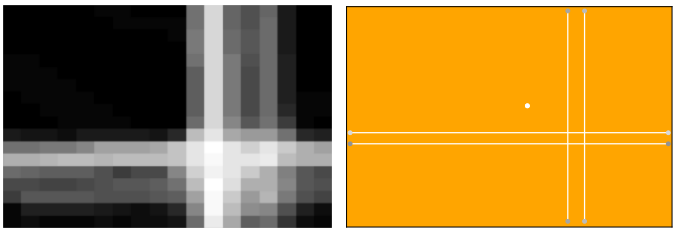


Fig. 7. Discretization of the space nearby a traffic light. Cells intersecting with road segments are assigned a value of 1 and are colored white, and the rest a 0 and are colored black.



(a) Drivability grid representing the probability of encountering a road in the space around a traffic light.

(b) Projection resulting from the averaged drivability grid.

Fig. 8. Projection extraction from average drivability grid of the space surrounding traffic lights.

of encountering a road in that area (Figure 8a). Finally, the averaged grid is traversed vertically and horizontally, to extract the drivable projection for a traffic light, and only traversals with a minimum threshold probability are considered. The resulting projection for a single detected traffic light is shown in Figure 8b.

If multiple traffic lights are observed simultaneously, as is the case in intersections, the individual projections from all the observed traffic lights would add several redundant elements inconsistent with the shape of a road, as illustrated in Figure 9a. To prevent this redundancy, when multiple traffic lights are observed at the same time, the projection extraction procedure is modified as follows.

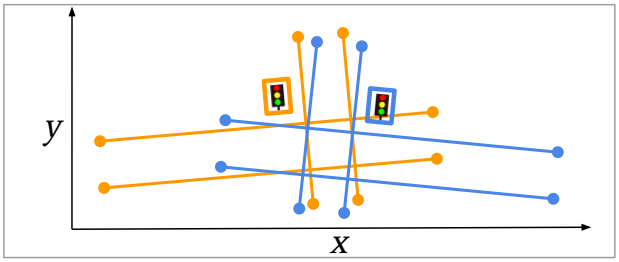
Let $\mathbf{L} = \{\mathbf{l}_1, ...\mathbf{l}_n\}$ denote the set of detected traffic lights, and $\mathbf{\Phi_L} = \{\phi_1, ..., \phi_n\}$ the drivable projections resulting from all traffic lights. To ensure a consistent orientation of the segments, each $\phi_i \in \mathbf{\Phi_L}$ is first rotated around the traffic light from which it originated, $\mathbf{l}_i$, by an angle $\mathrm{r}(\phi_i)$, given by

$$\mathrm{r}(\phi_i) = \frac{1}{|\mathbf{L}|} \sum_{\mathbf{l}_j \in \mathbf{L}} \mathbf{l}_j^\theta - \mathbf{l}_i^\theta. \tag{9}$$

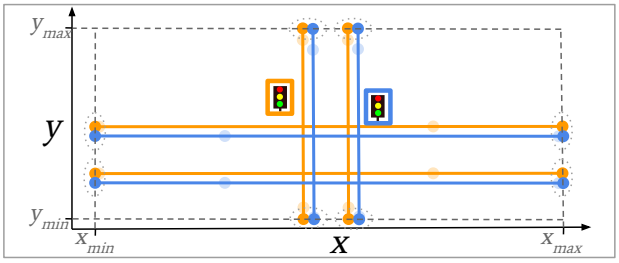
Next, the projection segments are divided, depending on their orientation, into $\mathbf{V}$ and $\mathbf{H}$, the set of vertical and horizontal segments, given by

$$\mathbf{V} = \{\mathbf{S} \in \bigcup_{\phi \in \mathbf{\Phi_L}} \phi^{\mathbf{S}} | 45 \leq \theta_{\mathrm{u}}(\vec{\mathbf{S}}^{\mathbf{K}}, \vec{\mathbf{x}}) \leq 135\} \tag{10}$$

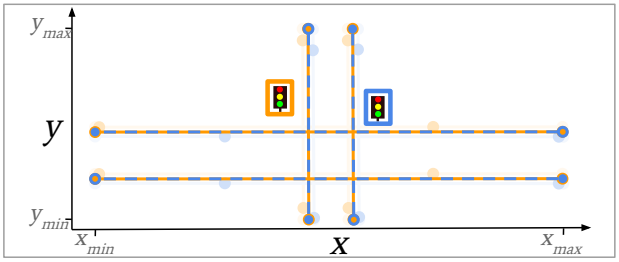
$$\mathbf{H} = \{\mathbf{S} \in \bigcup_{\phi \in \mathbf{\Phi_L}} \phi^{\mathbf{S}} | \, \theta_{\mathrm{u}}(\vec{\mathbf{S}}^{\mathbf{K}}, \vec{\mathbf{x}}) < 45 \vee 135 \leq \theta_{\mathrm{u}}(\vec{\mathbf{S}}^{\mathbf{K}}, \vec{\mathbf{x}}))\}, \tag{11}$$



(a) Projections resulting from two traffic lights at an intersection.



(b) Projections are rotated around the traffic light to match the average angle, and segments are extended to intersect with $x_{max}, x_{min}, y_{max}$, and x_{min}.



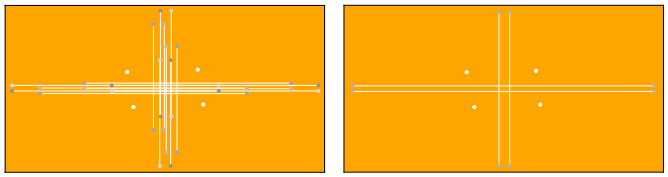
(c) Nearby segments are combined and the resulting segments are introduced at the average node coordinates of merged nodes

Fig. 9. Illustrative example showing how multiple traffic light projections are modified. For clarity, blue and orange illustrate the projections originating from different traffic lights, and there is no relation between the color and the probabilities encoded in their nodes and edges.

where θ_{u} denotes the smallest unsigned angle between two vectors, and $\vec{\mathbf{x}}$ represents the unit vector in the direction of the x-axis. Furthermore, all segments are extended to intersect with the vertical and horizontal lines corresponding to the maximum and minimum lateral and longitudinal coordinate values of segment *knots*, x_{max}, x_{min}, y_{max} and y_{min}. Segments $\mathbf{S} \in \mathbf{V}$ are extended to intersect with the horizontal lines defined by $y = y_{max}$ and $y = y_{min}$, while segments $\mathbf{S} \in \mathbf{H}$ are extended to intersect with the vertical lines given by $x = x_{max}$ and $x = x_{min}$, as shown in Figure 9b. Finally, the *knots* of segments corresponding to the same lane are merged (Figure 9c), and the new node properties are calculated as follows. Let $\mathbf{n}_i$ and $\mathbf{n}_j$ denote two knot nodes to be merged. The new node, $\mathbf{n}_f$, is given by

$$\mathbf{n}_f = \left(\frac{\mathbf{n}_i^x + \mathbf{n}_j^x}{2}, \frac{\mathbf{n}_i^y + \mathbf{n}_j^y}{2}, \mathbf{n}_i^\kappa, \mathbf{n}_i^\tau, \max(\mathbf{n}_i^{p^e}, \mathbf{n}_i^{p^j}) \right). \quad (12)$$

An example of projections resulting from multiple traffic lights at an intersection in our simulations is shown in Figure 10. Figure 10a illustrates what the drivable space would looks like immediately after introducing projections from four different traffic lights. Figure 10b shows the drivable space resulting from merging the projections of all the sensed traffic light projections.



(a) Individual projections resulting from four traffic lights

(b) Result of merging the individual traffic light projections

Fig. 10. Projections generated from four traffic light at an intersection (left) and resulting simplified projection after merging them (right).

C. Projection Alignment via Graph-based SLAM

To obtain the most likely spatial configuration of projections, $\mathbf{\Phi}^*$, nodes from $\mathbb{D}^-$ projections are treated as landmarks, and nodes from $\mathbb{D}^+$ projections as possible robot paths in order to formulate a Graph-based SLAM optimization problem. To that end, a new graph $\mathbf{G}$ is constructed with nodes originating from the projections and edges encoding spatial constraints between them, as detailed in this section. The least squares problem encoded in $\mathbf{G}$ is solved using Gauss-Newton to find the configuration of nodes that minimizes the errors specified by the edges.

C.1 Nodes

The nodes considered for the error minimization problem, $\mathbf{G^N}$, result from all the projection nodes, and some additional ones resulting from pairwise projection alignments, given by

$$\mathbf{G^N} = \underbrace{\left(\bigcup_{\phi \in \mathbf{\Phi}} \phi^{\mathbf{N}} \right)}_{\text{All projection nodes}} \cup \underbrace{\left(\bigcup_{\substack{\phi_i, \phi_j \in \mathbf{\Phi} \\ i \neq j}} h(\phi_i, \phi_j) \right)}_{\substack{\text{New nodes from pairwise} \\ \text{projection alignments}}}, \quad (13)$$

with $h(\phi_i, \phi_j)$ defined as

$$h(\phi_i, \phi_j) = \bigcup_{\mathbf{n} \in \phi_i^{\mathbf{N}}} \{ g(\mathbf{n}, \phi_j) \mid \mathbf{n}^\kappa = 1 \wedge P_1(\mathbf{n}, \phi_j) \wedge P_2(\mathbf{n}, \phi_j) \} \quad (14)$$

and $g(\mathbf{n}, \phi_j)$ denoting a function that returns the geometric projection of $\mathbf{n}^{xy}$ onto ϕ_j, as illustrated in Figure 11.

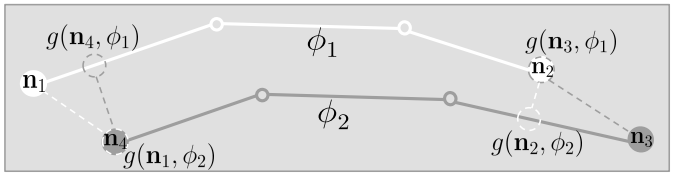


Fig. 11. Two segments and projections of their knots onto the other segment. The projected node is indicated by the hollow circle with a dashed border.

Furthermore, the binary predicates $P_1, P_2 : \mathbf{G^N} \times \mathbf{\Phi} \mapsto \{0, 1\}$ are Boolean-valued function to evaluate certain conditions, given by

$$P_1(\mathbf{n}, \phi) = \begin{cases} 1 & \text{if } ||g(\mathbf{n}, \phi) - \mathbf{n}^{xy}|| \leq 1.75\text{m} \\ 0 & \text{otherwise, and} \end{cases} \quad (15)$$

$$P_2(\mathbf{n}, \phi) = \begin{cases} 1 & \text{if } \neg\exists_{\mathbf{n}' \in \phi^{\mathbf{N}}} \left(||\mathbf{n}'^{xy} - g(\mathbf{n}, \phi)|| \leq 0.2\text{m} \right) \\ 0 & \text{otherwise,} \end{cases} \quad (16)$$

where $||\mathbf{v}||$ denotes the magnitude of vector $\mathbf{v}$. Additionally, 1.75m is chosen since it is half of a typical lane width, and a 0.2m resolution is considered sufficient detail in current research [18]–[20].

C.2 Edges

An edge or constraint $\mathbf{e}_{ij} \in \mathbf{G^E}$ between nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ represents a spatial constraint between them, and is given by

$$\mathbf{e}_{ij} = (\mathbf{n}_i, \mathbf{n}_j, \hat{\mathbf{z}}_{ij}, \mathbf{\Omega}_{ij}, \rho), \quad (17)$$

where $\mathbf{n}_i, \mathbf{n}_j \in \mathbf{G^N}$, $\hat{\mathbf{z}}_{ij}$ denotes an expected measurement between $\mathbf{n}_i$ and $\mathbf{n}_j$, i.e. a relative transformation that makes the two nodes overlap; $\mathbf{\Omega}_{ij}$ is the information matrix representing how confident we are about $\hat{\mathbf{z}}_{ij}$; and ρ is an optional kernel function used to weigh the mismatch between the expected and actual relative configuration of the nodes, which is typically used to diminish the effect of outliers and make the optimization more robust. To that end, (3) is replaced by

$$\mathbf{G^N}^* = \underset{\mathbf{G^N}}{\operatorname{argmin}} \sum_{\mathbf{e}_{ij} \in \mathbf{G^E}} \mathbf{F}(\mathbf{e}_{ij}), \text{ and} \quad (18)$$

$$\mathbf{F}(\mathbf{e}_{ij}) = \begin{cases} \rho\left(\varepsilon_{ij}^T \mathbf{\Omega}_{ij} \varepsilon_{ij}\right) & \text{if } \rho \text{ is specified} \\ \varepsilon_{ij}^T \mathbf{\Omega}_{ij} \varepsilon_{ij} & \text{otherwise.} \end{cases} \quad (19)$$

Constraints between nodes are established based on the semantic information of the input object from which they originated. The remainder of this section explains the types of constraints that are introduced, and between which nodes.

a) Pose prior: Independently of the type of object a node $\mathbf{n}$ originated from, there is a current belief regarding its location $\mathbf{n}^{xy}$, which is used to anchor $\mathbf{n}$ to this point. This is the only unary constraint introduced, i.e. specifying a constraint for a single node. All the remaining constraints establish a relative measurement between two nodes. Prior constraints are illustrated in Figure 12a.

b) Odometry: If nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ in the graph originated from the ego vehicle location at two consecutive timesteps, $\mathbf{v}_t$ and $\mathbf{v}_{t+\Delta t}$, an odometry measurement is available to relate the two nodes, $\hat{\mathbf{z}}_{ij} = \mathbf{u}_t$. Odometry constraints are illustrated in Figure 12b.

c) Motion models: Many models exist to describe the motion of road users. Common vehicle motion models are Constant Velocity (CV) and Constant Acceleration and Turn Rate (CTRA) [61]. For simplicity, we assume the existence of a motion model $\mathcal{M}$, which given a vehicle's position at time t and prediction horizon Δt, retrieves a feasible vehicle position

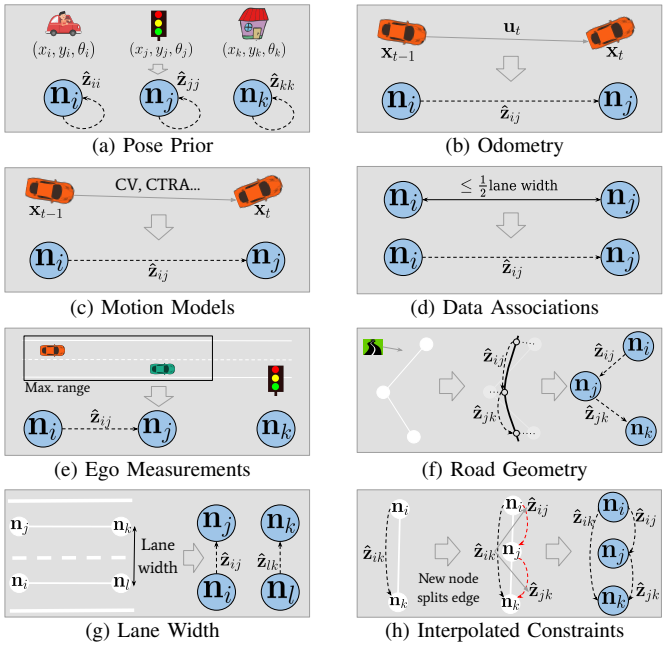


Fig. 12. Constraints used to construct the graph representing the graph-based SLAM problem.

at time $t + \Delta t$, $\hat{\mathbf{v}}_{t+\Delta t} = \mathcal{M}(\mathbf{v}_t, \Delta t)$, and relate two nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ originating from consecutive vehicle positions $\mathbf{v}_t$ and $\mathbf{v}_{t+\Delta t}$ by $\hat{\mathbf{z}}_{ij} = \hat{\mathbf{v}}_{t+\Delta t} - \mathbf{v}_t$. Motion model constraints are illustrated in Figure 12c.

d) Data association: Data Association (DA) constraints are established between any pair of *knots*, as long as they both originate from a drivable or non-drivable projection, and provided that they originate from different projections. That is, a node $\mathbf{n} \in \phi^{\mathbf{N}}$ with $\phi \in \mathbb{D}^s$ may only associated with a node $\mathbf{n}' \in \phi'^{\mathbf{N}}$ with $\phi' \in \mathbb{D}^{s'}$ if $s = s'$, $\phi \neq \phi'$ and $\mathbf{n}^\kappa = \mathbf{n}'^\kappa = 1$. Furthermore, DA constraints are established when the operation $g(\mathbf{n}, \phi)$, as defined in the previous section, is performed. When $P_1(\mathbf{n}, \phi)$ holds, $\mathbf{n}$ is associated with one of the following: (i) a new node which is inserted in ϕ at $g(\mathbf{n}, \phi)$ if $\neg P_2(\mathbf{n}, \phi)$, or (ii) the closest node to $g(\mathbf{n}, \phi)$ otherwise. For these constraints, ρ is as specified in the Dynamic Covariance Scaling (DCS) approach [60] to disable potential false positive associations during the optimization. DA constraints are illustrated in Figure 12d.

e) Measurements: When an object is within visibility range of the ego vehicle, measurements indicating the relative position of this object with respect to the ego are available. Such measurements are directly translated into a constraint between the object and the ego location at the time the measurement was obtained. Measurement constraints are illustrated in Figure 12e.

f) Road Geometry: Clothoids are a type of curve commonly used to model roads [62], [63], since their linear change of curvature with arc length allows comfortable steering in curved road segments. If information is available about the road network, *knots* are used to construct a clothoid curve as in [64], and the rest of the nodes are constrained to diminish deviations from such curve, as illustrated in Figure 12f.

g) Lane Width: Nodes from adjacent lanes should be constrained to keep their distance and avoid different lanes erroneously merging. To that end, constraints are introduced between nodes of adjacent lanes to maintain a separation of 3.5 meters. Currently, lane width constraints are only introduced between nodes of traffic light projections, and introducing them between nodes originating from the road network and ego lane projections remains for future work (see Appendix E). Lane width constraints are illustrated in Figure 12g.

h) Interpolated constraints: As stated earlier, when performing the operation $g(\mathbf{n}, \phi)$, a new node $\mathbf{n}_n$ could be introduced in $\phi^{\mathbf{N}}$, splitting an edge $\mathbf{e}_{ij}$ between nodes $\mathbf{n}_i$ and $\mathbf{n}_j$. When this occurs and a constraint $\hat{\mathbf{z}}_{ij}$ already exists relating $\mathbf{n}_i$ and $\mathbf{n}_j$, two new constraints $\hat{\mathbf{z}}_{in}$ and $\hat{\mathbf{z}}_{nj}$ resulting from a simple linear interpolation of $\hat{\mathbf{z}}_{ij}$ at the point $\mathbf{n}_n^{xy}$. Interpolated constraints are illustrated in Figure 12h.

C.3 Optimization

The problem of finding the configuration of nodes that is most consistent with the specified constraints has been thoroughly studied in the field of robotics, and several open-source tools are available implementing efficient algorithms to solve the error minimization problem encoded in the graph. An example of such tools is g2o [65], a general-purpose optimization library commonly used for SLAM.

The optimization itself is not the focus of our work, so this task is delegated to g2o and the optimal spatial configuration of the drivable space is achieved using the Gauss-Newton algorithm, limited to a maximum of 20 iterations. This particular choice of algorithm and its parameters is motivated by [60], where the same setting is used to evaluate the DCS approach.

C.4 Drivable & Non-drivable Projection Intersections

After the most likely configuration of nodes is achieved, it is possible to encounter segments from $\mathbb{D}^+$ projections intersecting with segments from $\mathbb{D}^-$ projections. These situations might occur due to obstacles on the road, or outdated information of the road network. When $\bar{\mathbf{e}}_{ij}$ from a drivable projection and $\bar{\mathbf{e}}_{kl}$ from a non-drivable one intersect at point $\mathbf{n}_b^{xy} = (x_b, y_b)$, $\mathbf{e}_{ij}$ is split into $\mathbf{e}_{ib'}$ and $\mathbf{e}_{b''j}$ with $\mathbf{n}_{b'}^{xy}$ and $\mathbf{n}_{b''}^{xy}$, given by

$$\mathbf{n}_{b'}^{xy} = \mathbf{n}_b^{xy} + \min(1.75, ||\vec{\mathbf{e}}_{ib}||) \cdot \frac{\vec{\mathbf{e}}_{bi}}{||\vec{\mathbf{e}}_{bi}||}, \tag{20}$$

$$\mathbf{n}_{b''}^{xy} = \mathbf{n}_b^{xy} + \min(1.75, ||\vec{\mathbf{e}}_{bj}||) \cdot \frac{\vec{\mathbf{e}}_{bj}}{||\vec{\mathbf{e}}_{bj}||}. \tag{21}$$

The value 1.75m is arbitrarily chosen in order allow a safety separation between a drivable and non-drivable projection. This operation is illustrated in Figure 13.

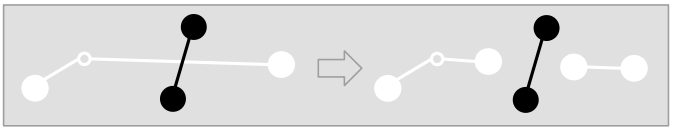


Fig. 13. Result of a drivable (white) and a non-drivable (black) segment intersecting after optimization. The drivable segment is split at the intersection.

D. Unification of Projections

The last step towards achieving our final estimate of the drivable space is to merge all the optimized individual projections, $\mathbf{\Phi}^*$, into one unified projection, which is considered the drivable space, $\mathcal{G}^*(\mathbf{N}, \mathbf{E})$.

To obtain $\mathcal{G}^*(\mathbf{N}, \mathbf{E})$, a procedure $f^m : \mathbb{D}^{|\mathbf{\Phi}|} \mapsto \mathbb{D}$ is defined, consisting of two steps. First, segments are merged by merging their *knots*, after which the geometry, as approximated by *non-knots*, is updated, and finally the probabilities of *existence* and *drivability*. This procedure is detailed next.

D.1 Merging Segment Knots

The first step towards obtaining a unified drivable space is to determine which nodes of the different projections should be merged. To that end, another graph is created with all *knots* in $\mathbf{G}$ and edges between nodes that contain a valid data association after optimization in order to extract $\mathcal{C}$, the set of connected components through valid data associations. A data association is considered valid if it satisfies two conditions. The first condition to consider a data association valid is that its information matrix is not scaled down during the optimization, that is $s = 1$ as given by the dynamic covariance scaling technique from [60], which scales the information matrix of a constraint by a factor s^2, with $s \in [0, 1]$. The second condition to consider an association valid is that the distance between the nodes it connects is not larger than 1.75m (half of a typical lane width). This situation is illustrated in Figure 14a.

Next, the nodes of each connected component $\mathbf{C} \in \mathcal{C}$ are merged and replaced by a node $\mathbf{n}_f$ with coordinates given by

$$\mathbf{n}_f^{xy} = \sum_{\mathbf{n} \in \mathbf{C}} w(\mathbf{n}) \cdot \mathbf{n}^{xy}, \tag{22}$$

where $w(\mathbf{n})$ is a weighting function that accounts for the confidence we have that this node exists, and its relative importance $\mathbf{n}^\tau$, given by

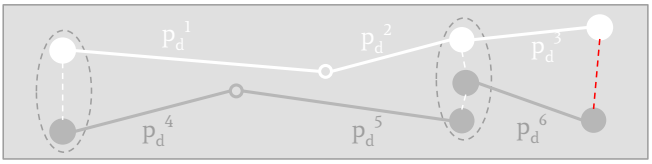
$$w(\mathbf{n}) = \frac{\mathbf{n}^{p_e} \cdot \mathbf{n}^\tau}{\sum_{\mathbf{n}' \in \mathbf{C}} \mathbf{n}'^{p_e} \cdot \mathbf{n}'^\tau}. \tag{23}$$

Furthermore, the new node's *probability of existence*, $\mathbf{n}_f^{p_e}$, is determined with the addition theorem for multiple non-exclusive events [66], which by De Morgan's law and under independence assumption can be calculated as

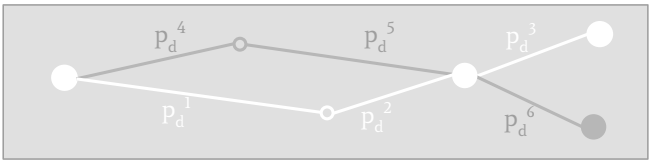
$$\mathbf{n}_f^{p_e} = 1 - \prod_{\mathbf{n} \in \mathbf{C}} (1 - \mathbf{n}^{p_e}). \tag{24}$$

Once the old *knots* of a segment to be merged, $\mathbf{S'^K} = \{\mathbf{n}_{o_1}, \mathbf{n}_{o_2}\}$, are merged resulting in the new *knots*, $\mathbf{S^K} = \{\mathbf{n}_{f_1}, \mathbf{n}_{f_2}\}$, the segment geometry is updated by applying an affine transformation $\mathbf{A}$ to every *non-knot* $\mathbf{n} \in \mathbf{S'^K}$, where $\mathbf{A}$ is defined as

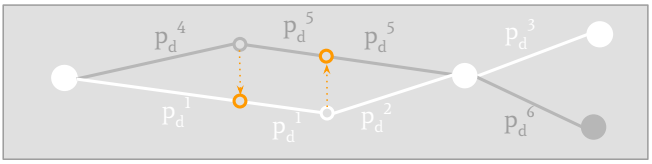
$$\mathbf{A} = \begin{bmatrix} s_x \cos\theta & -s_y \sin\theta & t_x s_x \cos\theta - t_y s_y \sin\theta + t'_x \\ s_x \sin\theta & s_y \cos\theta & t_x s_x \sin\theta + t_y s_y \cos\theta + t'_y \\ 0 & 0 & 1 \end{bmatrix} \tag{25}$$



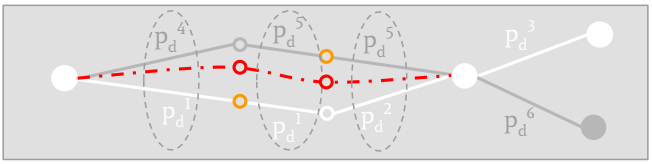
(a) First, the nodes that will be merged are retrieved by finding the connected components (dashed ellipses) though valid data associations (white dashed line). Invalid data (red dashed line) are discarded.



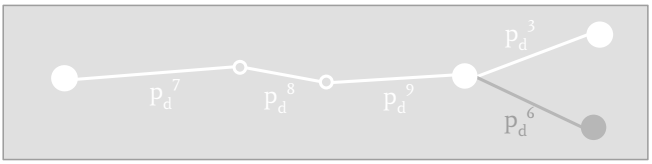
(b) Segment knots to be merged are translated to their final coordinates and non-knot nodes are updated accordingly. The resulting geometry does not resemble a road.



(c) New nodes are introduced in the segments to be merged to ensure there is not a single dominant segment during regression.



(d) All the resulting segment nodes are used to find a polynomial passing through the segment knots and minimizing deviation from non-knots. The polynomial is simplified by line segments and the resulting probability of drivability is computed with (24).



(e) Final segment after the merging operation.

Fig. 14. Illustrative example of two segments being merged.

and is the result of the composition of four transformation matrices that make $\overline{\mathbf{S}'^{\mathbf{K}}}$ fully overlap with $\overline{\mathbf{S}^{\mathbf{K}}}$: a translation given by vector $(t'_x, t'_y) = \mathbf{n}_{o_1}^{xy}$; a rotation by angle $\theta = \theta_s(\mathbf{S}'^{\mathbf{K}}, \vec{\mathbf{S}}^{\mathbf{K}})$, where $\theta_s(\mathbf{v}_1, \mathbf{v}_2)$ denotes the signed angle from vector $\mathbf{v}_1$ to vector $\mathbf{v}_2$; a scaling by factor (s_x, s_y), given by

$$(s_x, s_y) = \left(\frac{\mathbf{n}_{f_2}^x - \mathbf{n}_{f_1}^x}{\mathbf{n}_{o_2}^x - \mathbf{n}_{o_1}^x}, \frac{\mathbf{n}_{f_2}^y - \mathbf{n}_{f_1}^y}{\mathbf{n}_{o_2}^y - \mathbf{n}_{o_1}^y} \right); \tag{26}$$

and a final translation given by vector $(t_x, t_y) = \mathbf{n}_{f_1}^{xy}$. After performing these operations, the segments of the drivable space still do not resemble the geometry of the road due to various edges branching out, as illustrated in Figure 14b.

D.2 Merging Segment Non-knots and Edges

To unify edges of the segments to be merged, we find a third degree polynomial curve passing through the segment *knots* and minimizing deviation from all other nodes. Before finding this curve, new nodes are introduced in the segments to be merged to ensure that the contribution of all segments to the final geometry of the merged segment is equal (Figure 14c). Once the number of nodes in each segment is equal, polynomial regression is performed with all segment nodes as follows.

Let $\mathbf{S}_i$ and $\mathbf{S}_j$ be two segments to be merged, and therefore after merging their nodes as previously explained, $\mathbf{S}_i^{\mathbf{K}} = S_j^{\mathbf{K}} = \{\mathbf{n}_k, \mathbf{n}_l\}$. We seek to find a curve passing through $\mathbf{n}_k^{xy}$ and $\mathbf{n}_l^{xy}$, and minimizing deviation from $\mathbf{S}_i^{\mathbf{N}} \cup S_j^{\mathbf{N}}$, as illustrated in Figure 14d. The curve, $f(x)$, is given by

$$f(x) = \beta_3 x^3 + \beta_2 x^2 + \beta_1 x + \beta_0 \tag{27}$$

and the optimal parameters $\beta = \{\beta_0, \beta_1, \beta_2, \beta_3\}$ are obtained by non-linear least squares optimization. This optimization is similar to the one performed to align the nodes of the drivable space (Section III-C), but the constraints are significantly simpler. Thus, there is no need for an advanced library such as g2o and we resort to SciPy [67] instead, simply setting the certainty of the *knots* three orders of magnitude higher than the rest to ensure the optimized parameters represent a curve passing through (or extremely close to) $\mathbf{n}_k$ and $\mathbf{n}_l$. All other parameters are not modified, therefore the method employed for optimization is the default: Levenberg–Marquardt [68]. To aid in the convergence of the algorithm, β_1 is initialized as the slope of $\overline{\mathbf{S}_i^{\mathbf{K}}}$ and $\overline{\mathbf{S}_j^{\mathbf{K}}}$, given by

$$\beta_1 = \frac{\mathbf{n}_l^y - \mathbf{n}_k^y}{\mathbf{n}_l^x - \mathbf{n}_k^x}. \tag{28}$$

Finally, the regression line is approximated by a polyline, inserting nodes at the points resulting from geometry projection of the nodes used for regression, and edges connecting consecutive nodes. The probabilities of existence and drivability of nodes and edges is once again given by (24). The final result of the merging operation is illustrated in Figure 14e.

IV. Experiments

Several simulation tools were compared to assess the performance of the proposed method, and the CARLA driving simulator [69] was chosen for this task. More details about the comparison of the simulation tools is available in Appendix C. This section is dedicated to present the testing scenarios, how the required data is extracted from the simulator, and finally specify the experiments carried out to assess the method.

A. Use cases

The experiments take place in CARLA's "Town03", which features an urban environment, and all the experiments are carried out in four scenarios. The ego's point of view in each scenario is shown in Figure 16.

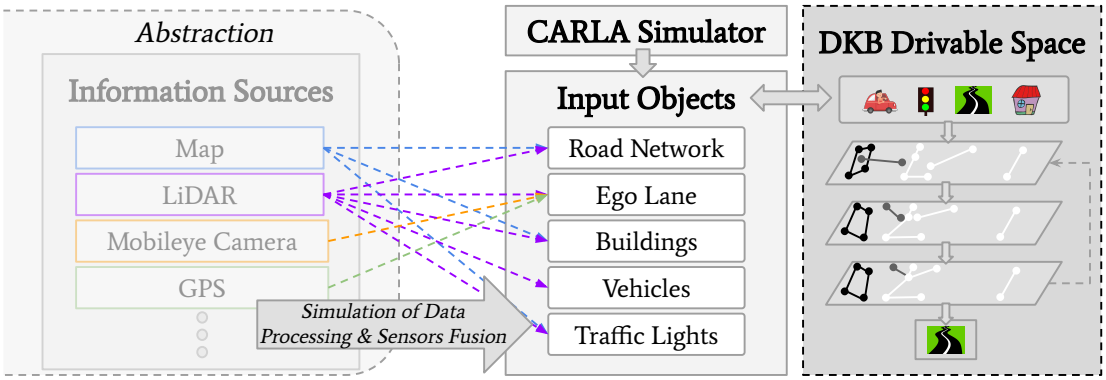


Fig. 15. Data extraction process. We abstract away from the specific source of information of different inputs, and extract such inputs from the CARLA driving simulator directly.

(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 16. Ego point of view in the four testing scenarios.

a) Scenario 1: A street with two 1-lane roads. Road visibility is partially occluded due to a curved road segment (Figure 16a).

b) Scenario 2: A street with two 2-lane roads. Road visibility is partially occluded due to a curved road segment (Figure 16b).

c) Scenario 3: A T-intersection. The road ahead is not visible due to an elevation difference (Figure 16c).

d) Scenario 4: A X-intersection. The road ahead is not visible due to an elevation difference (Figure 16d).

B. Data Acquisition

The input to our algorithm is a collection of processed objects resulting form the sensor fusion of raw data acquired by several sensors or sources of information. One of such input objects could therefore originate from multiple sources of information, and we abstract from this process and directly extract the inputs from CARLA, as illustrated in Figure 15. The extraction process for the different objects is summarized in the remainder of this section.

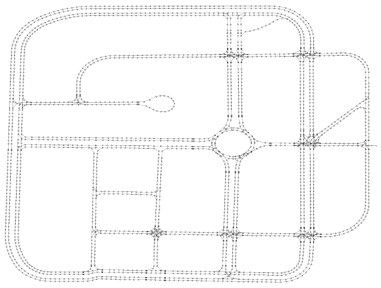
a) Traffic Lights: The position and orientation of all traffic lights is available through the Python API that CARLA provides. Thus, ground truth for these objects is easily retrieved.

b) Vehicles: Using CARLA's Python API, spawning vehicles and accessing their properties at any time in the simulation is straightforward. Ground truth poses of all vehicles are sampled at 20Hz for the duration of the entire simulation.

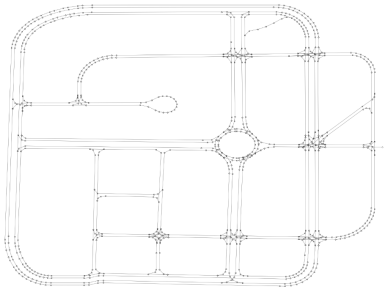
c) Buildings: Areas of the simulation environment where buildings are located are specified manually. Unfortunately, properties of many static objects already present in the map cannot be easily accessed through the Python API.

d) Road Network: A detailed lane-level road network of the simulation map can be easily extracted from CARLA. This map is is processed to re-format it in a similar fashion as our previously described projections, since the format provided by CARLA is not suitable for our purposes. This process is illustrated in Figure 17, and summarized as follows. First, the road topology is extracted from CARLA, consisting of nodes 5m apart, and edges connecting consecutive nodes of the same or subsequent lanes (Figure 17a). Next, redundant nodes from straight road segments are removed applying Ramer-Douglas-Peucker [70], [71] (Figure 17b). Finally, a simple rule-based algorithm is applied to detect the nodes separating straight and curved segments and classify them as either *knots* or *non-knots* (Figure 17c).

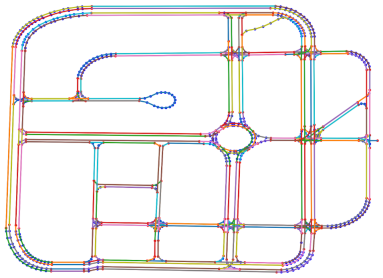
e) Ego Lane: Once we have obtained all the lanes in the simulation map, the ego lane is simply extracted at run-time based on the ego's location.



(a) Evenly spaced (5m) road network directly extracted from CARLA



(b) Simplified road network using the Ramer-Douglas-Peucker algorithm



(c) Segments extracted from the simplified road network. Red nodes indicate *knots*, and blue nodes *non-knots*. Differently colored lines are used to identify different segments.

Fig. 17. Road network extraction process.

C. Experiment Setup

To demonstrate the robustness of the proposed method towards different input qualities, different levels of additive Gaussian noise [72] with zero mean and varying variances are introduced to the coordinates of the input objects and constraints between them before the optimization described in Section III-C.

The values we use for the variances of the noise introduced to each input are based on a recent study testing the robustness of tracking algorithms towards noise [73] and a relative ordering of expected noise of the different inputs to our algorithm (more details are provided in Appendix B). Based on these values, for each type of input a base variance value is defined, σ^2_{base}, which is multiplied by a factor $s \in \{0, 0.3, ..., 2.7, 3\}$, where $s \leq 1$ represent reasonable (based on the variances used in [73], and $s > 1$ are introduced to test our proposed method with extreme levels of noise.

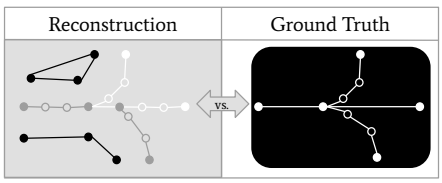
To account for the randomness of the artificial noise sampling, each experiment is carried out five times, and the mean absolute error (MAE) and standard deviation (unless it is zero) of the various performance metrics is reported.

D. Evaluation Metrics

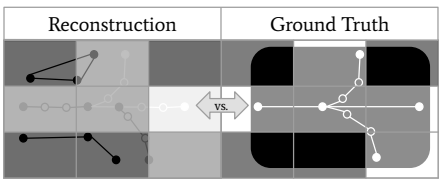
Systematic evaluation of our proposed method is not straightforward. The drivable space format of the novel method proposed poses a challenge: direct comparison with the ground truth is not possible. As shown in Figure 18a, the graphs representing the ground truth road model and the reconstructed space are considerably different in topology despite representing a similar drivable space geometrically. In the ground truth graph, all the space that is not modeled as drivable is implicitly considered non-drivable. On the other hand, the proposed method models non-drivable areas explicitly (black edges in Figure 18a), and the lack of graph elements in an area means that not enough information is available to conclude anything about its drivability. Furthermore, the topology and number of elements in the graphs representing similar road networks can be significantly different (white and gray nodes and edges in Figure 18a).

Considering these differences, it is not possible to apply standard graph similarity metrics between the reconstructed and ground truth drivable spaces without developing an additional procedure to make their graphs comparable. For instance, Graph Edit Distance (GED) [74] and SimRank [75] are two metrics commonly used to measure (dis)similarity between graphs and their elements. Applying GED to the graphs of Figure 18a would misleadingly indicate a high disparity between them, since several non-drivable and extraneous nodes and edges would need to be removed from the reconstructed drivable space to arrive to an isomorphism of the ground truth graph. Furthermore, SimRank is simply not applicable since it provides pairwise similarities between graph elements, but the overall graph similitude remains unclear.

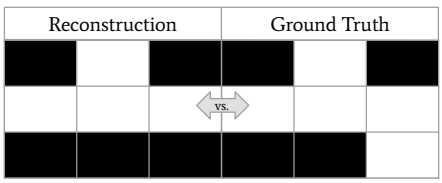
To quantify the performance of the proposed drivable space three criteria are analyzed: drivable space classification, distance to lane center (DTLC) error, and quality of the topology.



(a) Reconstructed drivable space (left) and ground truth (right). Despite representing similar drivable spaces, the two graphs differ in number of nodes and location of edges, making it challenging to compare.



(b) Discretization of the space into a grid. Each cell is assigned an average drivability according to the segments intersecting the cell, taking into account lane width.



(c) Each cell is classified into drivable (white) and not drivable (black). The reconstructed and ground truth drivable spaces can now be compared by means of standard classification metrics.

Fig. 18. Example of drivability grid extraction from drivable space.

Drivable Space Classification

To assess if the reconstructed graph represents a valid drivable space, the entire continuous space is discretized into a uniform grid of cell resolution 20cm, which is commonly considered sufficient detail in literature [18]–[20], and intersecting edges representing road segments (considering a lane width of 3.5m) are used to compute an average drivability for each cell (Figure 18b). Finally, each cell can be classified into drivable or non-drivable given a probability threshold, and deviations from the true drivable space can now be quantified by means of standard classification metrics such as accuracy (acc.), precision (pr.), recall (re.) and F_1 score [76]. This approach can effectively assess if the reconstructed drivable space can be used to determine the drivability of the space near the ego vehicle, however, it does not consider if the topology of the recovered road network is correct.

Topology Quality

The graphs representing drivable space reconstructions are subject to visual inspection and analyzed for imperfections. Depending on their quality, the reconstructions are assigned one of the following labels:

1) *Correct*: The topology of the recovered drivable space does not contain any error.
2) *Correct near Ego:* The topology of the recovered drivable space contains some errors, but these are not located in the ego lane 30-35 meters in front of the ego vehicle. This safety distance is chosen, as it is considered a safe stopping distance at maximum urban speeds [77].
3) *Incorrect:* The topology of the recovered drivable space contains some error in the ego lane 30-35 meters in front of the ego vehicle.

DTLC Error

The last criterion used to assess the proposed method is accuracy of the estimated Distance To Lane Center (DTLC), as illustrated in Figure 19. Let $DTLC$ denote the true distance to lane center, and $\widehat{DTLC}$ denote the estimated DTLC using the recovered drivable space and ego location. Then the DTLC error, ε_{DTLC} is given by

$$\varepsilon_{DTLC} = DTLC - \widehat{DTLC}. \tag{29}$$

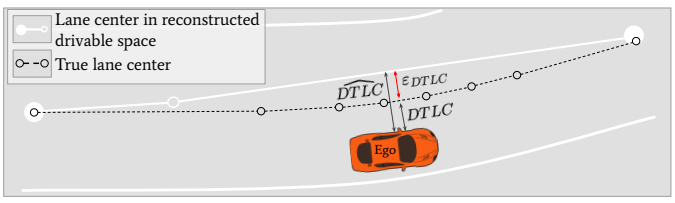


Fig. 19. Illustrative example of DTLC error achieved when using the proposed drivable space for in-lane localization.

E. Benchmark Method

For similar reasons that make the proposed method challenging to evaluate, comparing it to existing approaches is also not straightforward. At this stage, our focus is on ensuring the availability of a road model and ego (in-lane) localization which is sufficiently accurate to operate autonomously, as opposed to most research that focuses on (i) accurate road modeling [30], [78]–[80], or (ii) incorporating semantic information with raw sensor data directly to improve landmark detection and ensure enough inputs are available for construction of the world model [81]–[83]. Our approach does not improve the data processing component or produce the most accurate map. Instead, our aim is to reconstruct a minimal representation of the world (i.e. the drivable space) required for AD applications solely from object semantic information and domain knowledge, to ensure robustness towards situations where inputs are extremely scarce, missing, or wrong.

All these differences in the main purpose, assumed inputs, and output format prevent comparison between our proposed and existing methods, since this comparison is either not possible or it would be significantly biased and favor one of the methods. As such, our method is compared with the performance achieved in drivable space reconstruction and ego in-lane localization when using the ground truth road network and ego location under different levels of noise and missing inputs, as explained in Section IV-C.

V. Results

The experiment results are presented in two main sections. First, the quality of the reconstructed drivable space is evaluated without any positional noise, and the effect of missing different inputs is analyzed. Second, robustness towards noise in the inputs is evaluated. In both sections, first the topology quality concluded from visual inspection of the reconstructions is presented. Then, the different performance indicators (i.e. DTLC error, accuracy, precision, recall and F_1) are evaluated. Note that in this section, only the final estimation of the drivable space is shown. For a detailed step-by-step example illustrating intermediate stages of the method, the reader is referred to Appendix D. Furthermore, example reconstructions and drivability classifications are shown only when all inputs are present and when the road input is missing. All experiment results, including results with other missing inputs, can be found in Appendix F.

A. Noise-free

The quality of the topology achieved in the drivable space reconstructions is summarized in Table I, and examples of reconstructions with all inputs present and when missing the road network are shown in in Figures 20 and 21.

For ease of readability, let $T_q(s, m)$ denote the percentage of reconstructions considered to have topology quality $q \in Q$, with $Q = \{\text{C}, \text{CE}, \text{I}\}$, for scenario $s \in S$, with $S = \{1, 2, 3, 4\}$, when missing input $m \in M$, with M denoting the set of possible missing inputs, including "None" to indicate that all inputs are present. Additionally, when referring to a missing

input only its first letter is used (e.g. N denotes None, B denotes Buildings, etc.). Furthermore, the change in performance when missing an input is given by

$$\Delta T_q(s,m) = T_q(s,m) - T_q(s,\mathrm{N}). \tag{30}$$

Table I shows that $T_q(1,m) = T_q(2,m)$, $\forall_{q\in Q, m\in M}$ and $T_q(3,m) = T_q(4,m)$, $\forall_{q\in Q, m\in M \wedge m\neq \mathrm{R}}$, indicating that the topology quality achieved by reconstructions in intersection-free scenarios is identical, and in scenarios with intersections it is similar. When no noise is introduced to the inputs, $T_{\mathrm{C}}(s,\mathrm{N}) = 100\%$, for $s \in \{1,2\}$, indicating perfect topology recovery in intersection-free scenarios. When $s \in \{3,4\}$, $T_{\mathrm{CE}}(s,\mathrm{N}) = 100\%$, so despite containing some imperfections, the recovered topology ensures the ego can continue operating autonomously within the lane it is currently occupying.

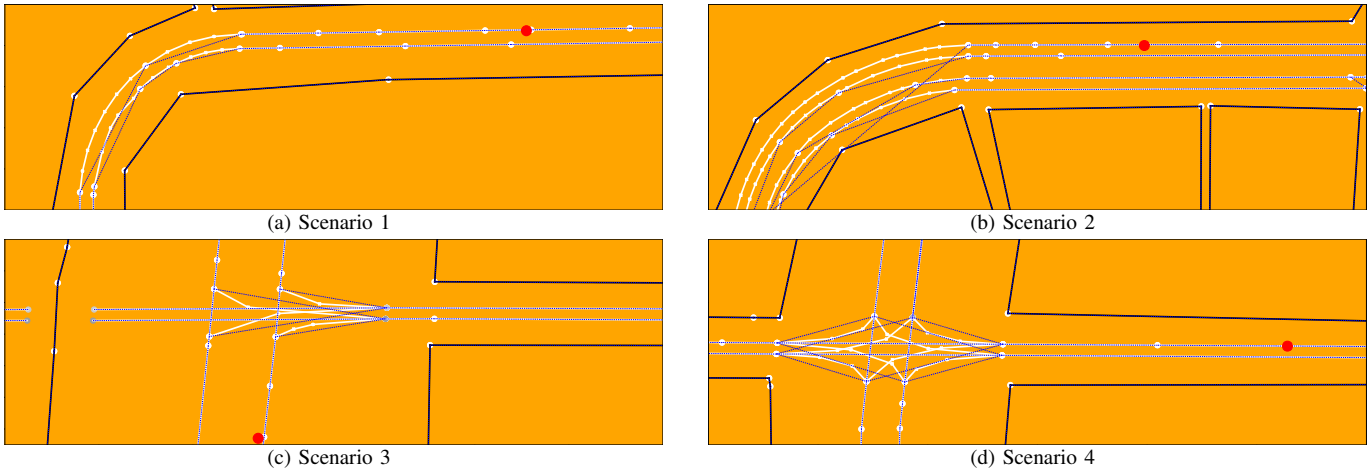
Furthermore, $\Delta T_q(s,m) = 0\ \forall_{q\in Q, s\in S, m\in\{\mathrm{B,M,V}\}}$, so missing these inputs does not have any effect on the quality of the topology. On the other hand, $\Delta T_{\mathrm{C}}(s,\mathrm{R}) = -100\%$ and $\Delta T_{\mathrm{CE}}(s,\mathrm{R}) = 100\%$ for $s \in \{1,2\}$, while $\Delta T_{\mathrm{C}}(4,\mathrm{R}) = -100\%$ and $\Delta T_{\mathrm{I}}(4,\mathrm{R}) = 100\%$, from which we can conclude

TABLE I
DRIVABLE SPACE TOPOLOGY QUALITY UNDER NO NOISE IN THE INPUTS

Missing Quality [%] / Scenario	None C	None CE	None I	Buildings C	Buildings CE	Buildings I	Mobileye C	Mobileye CE	Mobileye I	Vehicles C	Vehicles CE	Vehicles I	Road C	Road CE	Road I	Traf. Lights C	Traf. Lights CE	Traf. Lights I
1	100	-	-	100	-	-	100	-	-	100	-	-	0	100	-	n/a	n/a	n/a
2	100	-	-	100	-	-	100	-	-	100	-	-	0	100	-	n/a	n/a	n/a
3	-	100	-	-	100	-	-	100	-	-	100	-	-	100	-	100	0	-
4	-	100	-	-	100	-	-	100	-	-	100	-	-	0	100	100	0	-

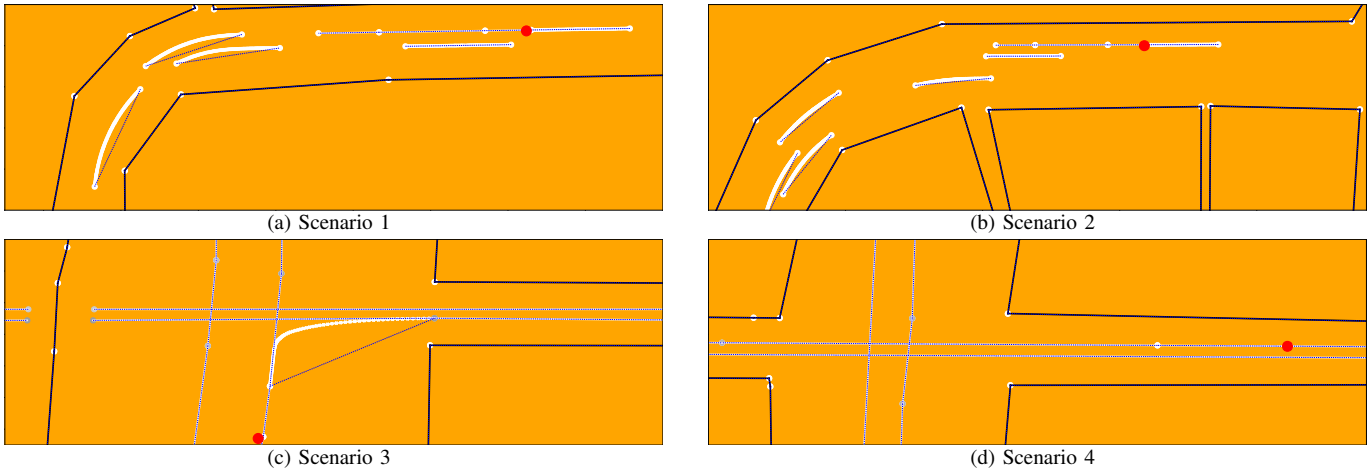
C - Correct | CE - Correct near ego | I - Incorrect | Red indicates an increase when an input is missing, and blue a decrease | "-" - Value is always zero independently of missing input | 0 - Value decreased to zero when an input is missing.

that the road network input has a significant impact in the quality of the reconstructions. However, even without any prior knowledge about the road network, $T_{\mathrm{CE}}(s,\mathrm{R}) = 100\%$ for $s \in \{1,2,3\}$, showing that the drivable space can at least recover the ego lane successfully. This is especially interesting for $s = 3$, where the correct topology of the ego lane is recovered (Figure 21c) after observing traffic lights and a vehicle performing a right turn at an intersection, even if the intersection itself is not visible from the ego's point of view



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 20. Drivable space reconstructions achieved when all inputs are present. The color of nodes and edges represents their probability of existence and drivability, ranging from black (probability of 0) to white (probability of 1). Blue dashed lines connect the knots of a segment. The ego position estimate within the drivable space is denoted by a red circle.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4

Fig. 21. Drivable space reconstructions achieved when the road network input is missing.

TABLE II
DTLC AND CLASSIFICATION METRICS ERROR UNDER NO NOISE IN THE INPUTS

Missing Input Scenario	DTLC MAE [cm]						Acc. MAE [$\times 10^4$]						Pr. MAE [$\times 10^4$]						Re. MAE [$\times 10^4$]						F_1 MAE [$\times 10^4$]					
	N	B	M	V	R	T	N	B	M	V	R	T	N	B	M	V	R	T	N	B	M	V	R	T	N	B	M	V	R	T
1	0	0	0.01	0	0.01	-	11	11	11	0	654	-	41	41	41	0	28	-	37	37	37	0	4456	-	39	39	39	0	2874	-
2	3.37	3.37	1.22	3.37	6.06	-	8	8	7	2	1730	-	23	23	18	10	99	-	12	12	12	0	7775	-	18	18	15	5	6366	-
3	3.1	3.1	7.67	3.1	8.95	7.68	201	304	203	198	285	7	680	1004	683	676	778	12	20	20	28	16	242	16	361	537	366	357	517	14
4	0.06	0.06	0.31	0.08	0.15	0	4	4	4	3	198	0	8	8	8	8	215	0	9	9	9	6	661	0	8	8	8	7	443	0

N - None | B - Buildings | M - Mobileye | V - Vehicles | R - Road Network | T - Traffic Lights | Red indicates an increase when an input is missing, and blue a decrease.

(Figure 16c). Unlike in scenario 3, $T_{\mathrm{I}}(4,\mathrm{R}) = 100\%$, since the drivable space fails to capture the allowed right turn at the intersection, despite correctly capturing the straight path the ego may follow (Figure 21d).

Finally, projections resulting from traffic lights introduce topological issues when all other inputs are present, $\Delta T_{\mathrm{C}}(s,\mathrm{R}) = 100\%$ for $s \in \{3,4\}$, and are most beneficial when the road network input is missing since they allow road discovery in intersection scenarios (Figures 21c and 21d).

We now investigate the different performance metrics, $PM = \{\text{DTLC, Acc., Pr., Re, } \mathrm{F}_1\}$ of drivable space classification and ego (in-lane) localization defined previously. Similarly to the notation defined earlier to refer to the quality of the topology in the reconstructions, let $i(s,m)$ denote the value of metric $i \in PM$ for scenario $s \in S$ and missing input $m \in M$. Furthermore, let $\varepsilon_i(s,m)$ denote the error in metric $i \in PM$ for scenario $s \in S$ and missing input $m \in M$. For classification metrics, $\varepsilon_i(s,m)$ is given by

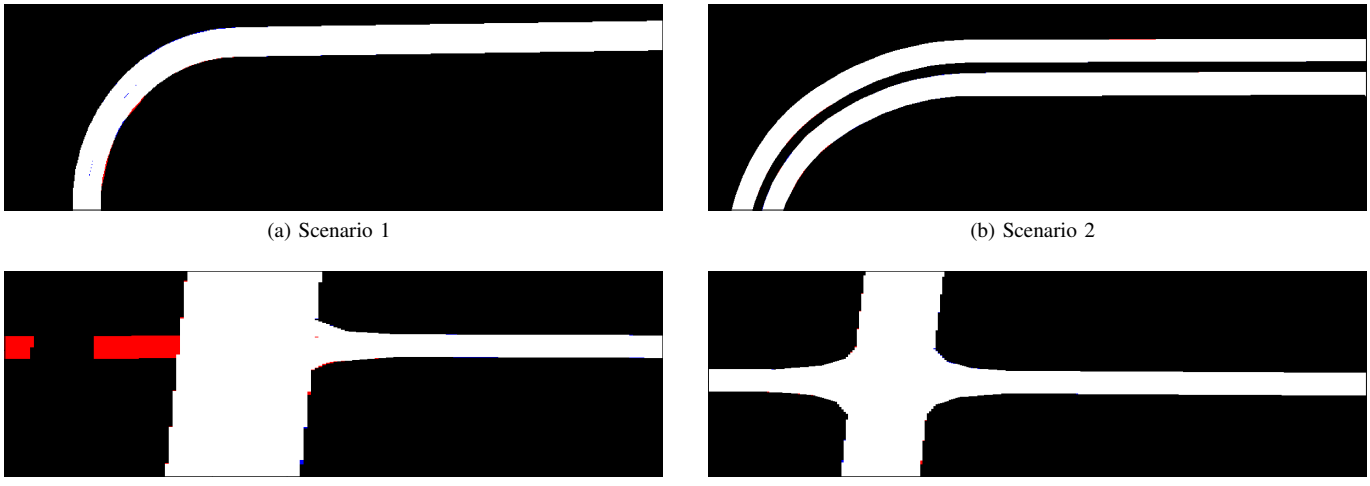
$$\varepsilon_i(s,m) = 1 - i(s,m). \tag{31}$$

Finally, $\Delta\varepsilon_i(s,m)$ denotes the change in metric $i \in PM$ given in scenario $s \in S$ when input $m \in M$ is missing.

Table II presents an overview of the error achieved in drivable space classification and ego in-lane localization when different inputs are missing. An example illustrating drivable space classification when all inputs are present is shown in Figure 22, and Figure 23 shows a classification when the road input is missing. As shown in Table II, $\varepsilon_{\mathrm{DTLC}}(s,\mathrm{N}) \leq 0.06$ cm for $s = \{1,4\}$, indicating an almost perfect estimation of the ego DTLC for these scenarios. However, despite this ideal setting without any noise and all the inputs, $3.31 \leq \varepsilon_{\mathrm{DTLC}}(s,\mathrm{N}) \leq 3.37$ cm for $s \in \{2,3\}$. This small error is introduced when merging segments from different projections in the last step of the algorithm. An example of this situation is given when the projection from a vehicle trajectory not driving in the center of the lane is merged with the projection resulting from the road network. In this instance, the true lane center is slightly displaced towards the trajectory after the merging operation.

The classification metrics when all inputs are present show an almost perfect classification of the drivable space except for scenario 3, in which two wrongly introduced roads cause a portion of the space to be wrongly classified as drivable (Figure 22c), having the highest impact on precision and subsequently F_1, with $\varepsilon_{\mathrm{pr.}}(3,\mathrm{N}) = 0.068$ and $\varepsilon_{F_1}(3,\mathrm{N}) = 0.0361$.

Examining the effect that missing different inputs have on the performance metrics, one can easily observe that some inputs are more vital than others. For instance, $\Delta T_I(s,\mathrm{B}) = 0$ $\forall_{i \in PM, s \in S \wedge s \neq 3}$, so buildings do not have a high impact in most cases. When $s = 3 \wedge i \neq$ DTLC, there is a small perfor-



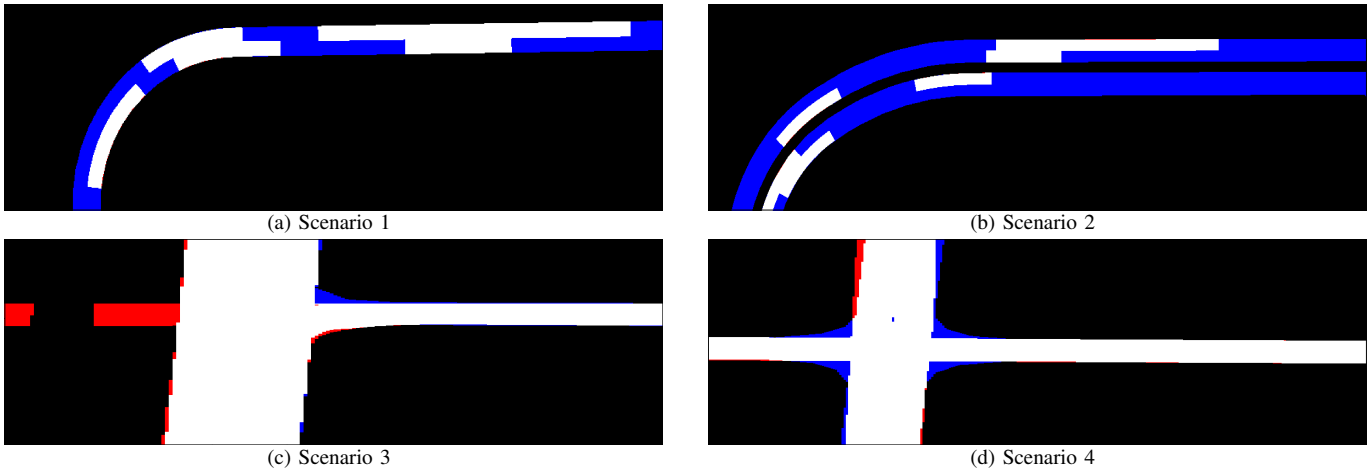
(a) Scenario 1

(b) Scenario 2

(c) Scenario 3

(d) Scenario 4

Fig. 22. Drivable space classification when all inputs are present. White - true positives | Black - true negatives | Red - false positives | Blue - false negatives.



(a) Scenario 1 (b) Scenario 2

(c) Scenario 3 (d) Scenario 4

Fig. 23. Drivable space classification when all inputs are present. White - true positives | Black - true negatives | Red - false positives | Blue - false negatives.

mance decrease, since the building projection partially erases the road segments wrongly introduced by the traffic light projections. Furthermore, observed vehicle trajectories mainly affect classification metrics, and $\Delta\varepsilon_{\text{DTLC}}(s, V) \leq 0.02$ cm for all $s \in S$, since the trajectories in the testing scenarios do not overlap with the ego position at the time of estimation.

On the other hand, missing inputs such as the road network or traffic lights can have a more significant impact in performance metrics. Scenarios 1 and 2 do not contain any traffic lights, and when the road network input is missing, discovery of drivable space (and corresponding classification metrics) is heavily dependant on the number of dynamic objects on the road that can be observed at the time of estimation. As can be seen in Figures 23a and 23b, when the road input is missing the drivable space reconstruction fails to identify much of the drivable space if few vehicles are on the road, resulting in many false positives and having the highest impact on recall with $\varepsilon_{\text{re.}}(1, \text{R}) = 0.4456$ and $\varepsilon_{\text{re.}}(2, \text{R}) = 0.7775$. However, even without any information about the road network, accuracy of the ego's DTLC estimate does not suffer considerably, with only $\Delta\varepsilon_{\text{DTLC}}(1, \text{R}) = 0.01$ cm and $\Delta\varepsilon_{\text{DTLC}}(2, \text{R}) = 2.69$ cm.

When considering testing scenarios 3 and 4, similar behavior is observed. However, due to the traffic light projections in these settings, the classification metrics do not suffer such a significant drop, and most of the drivable space is classified correctly (Figures 23c and 23d), although $\Delta\varepsilon_{\text{DTLC}}(1, \text{R}) = 5.85$ cm resulting in a total $\varepsilon_{\text{DTLC}}(1, \text{R}) = 8.95$ cm, the highest DTLC error in all four testing scenarios.

These results show that missing inputs can lead to errors in the drivable space reconstruction, having a significant influence on the road topology reconstruction at intersections. However, even without any prior information about the road network, it is possible to construct a road model which is sufficiently accurate to achieve a maximum DTLC error of under 9 cm, ensuring the ego vehicle can continue operating safely.

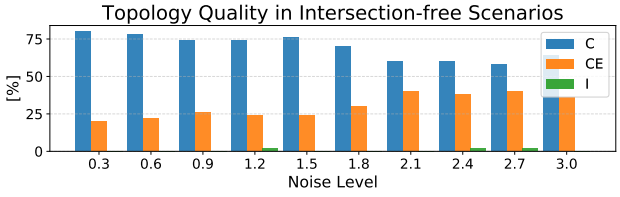
B. Noise in Inputs

The effect of adding different noise levels to the inputs is now investigated to assess the robustness of the method towards inputs of different qualities. Table III shows the topology quality achieved for all scenarios when different inputs are missing and noise is injected to the remaining inputs. Similarly to the noise-free experiments, the topology quality presents a similar patter in scenarios with and without intersections, i.e. $T_{\text{C}}(s, m) > T_{\text{CE}}(s, m) \wedge T_{\text{C}}(s, m) > T_{\text{I}}(s, m)\ \forall_{s=\{1,2\}, m \in M}$.

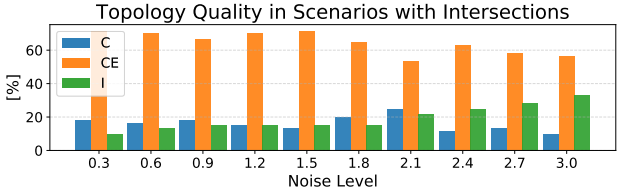
TABLE III
DRIVABLE SPACE TOPOLOGY QUALITY WITH NOISE IN THE INPUTS

Missing	None			Buildings			Mobileye			Vehicles			Road			Traf. Lights		
Quality [%]	C	CE	I	C	CE	I	C	CE	I	C	CE	I	C	CE	I	C	CE	I
Scenario																		
1	76	24	-	74	26	-	84	14	2	100	0	-	0	100	-	n/a	n/a	n/a
2	90	8	2	94	6	0	76	22	2	100	0	0	0	100	0	n/a	n/a	n/a
3	-	74	26	-	82	18	-	70	30	-	98	2	-	72	28	82	8	10
4	26	72	2	18	80	2	14	82	4	18	82	0	0	0	100	36	56	8

C - Correct | CE - Correct near ego | I - Incorrect | Red indicates an increase when an input is missing, and blue a decrease | "-" - Value is always zero independently of missing input | 0 - Value decreased to zero when an input is missing.



(a) Scenarios without intersections



(b) Scenarios with intersections

Fig. 24. Topology quality in the reconstructions with noisy inputs.

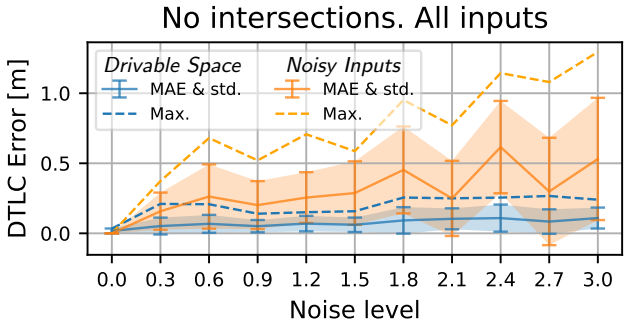
On the other hand, $T_{\text{CE}}(s,m) > T_{\text{C}}(s,m) \wedge T_{\text{CE}}(s,m) > T_{\text{I}}(s,m)\ \forall_{s=\{3,4\},m\in M\wedge m\neq \text{T}}$. Following this observation, the effect of different levels of noise is investigated separately in scenarios with and without intersections, as shown in Figure 24. Figure 24a shows that when $s \in \{1,2\}$, the percentage of reconstructions with perfect topology slightly decreases linearly from 75% in the lowest noise setting to around 60% with the highest noise levels. Despite this decrease, a majority of the reconstructions can recover correctly at least the ego lane, with only a small fraction of the reconstructions resulting in incorrect topology.

On the other hand, when $s \in \{3,4\}$, a majority of the reconstructions contain imperfections, but the topology in the drivable space is sufficiently accurate to recover the ego lane, as shown in Figure 24b. However, a significant fraction of the experiments yield an incorrect topology, reaching 30% when introducing the highest levels of noise.

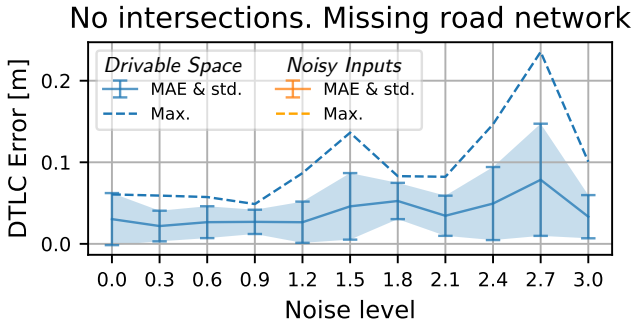
Furthermore, vehicle trajectories and traffic lights seem to introduce some topological errors. The issues with vehicle trajectories are especially noticeable when $s = \{1,2\}$, where $10 \leq \Delta T_C(s,\text{V}) \leq 24\%$, and in scenario 3, where $\Delta T_I(3,\text{V}) = -24\%$. The issues with traffic lights can be seen by $\Delta T_C(3,\text{T}) = 82\%$, where the extraneous road segments of Figure 20c are never introduced by the traffic lights.

When considering the different performance metrics in the noise-free scenarios, we have already seen that missing some of the inputs does not have a significant influence in the performance achieved in some of the testing scenarios. For instance, buildings only affect scenario 3, while traffic lights are not relevant at all for scenarios 1 and 2 where there are none. For simplicity, figures reporting the performance of noisy and missing inputs simultaneously are omitted except for the input with the highest influence in performance: the road network. Furthermore, as observed previously, when the road network input is missing there is a different trend in performance in testing scenarios with and without intersections, i.e. performance of drivable space classification drops significantly for the intersection-free scenarios, while in those with intersections this effect is not that visible due to the traffic light projections. As such, the robustness towards noise in the inputs is presented for scenarios with and without intersections separately, and only DTLC error and one summary classification metric (F1) is included in this section. All experiment results, including all evaluation metrics can be found in Appendix F.

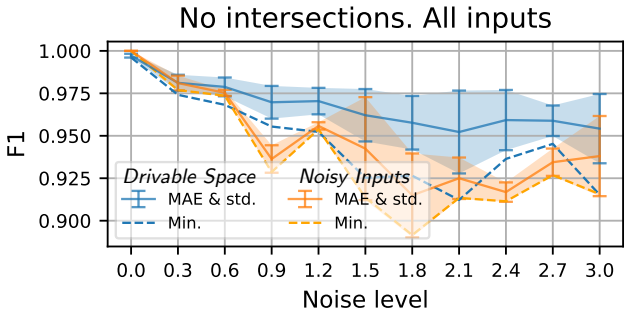
The performance using our proposed drivable space format (blue) is compared with the performance achieved considering the noisy inputs directly (orange). In both types of scenarios, with and without intersections, using the proposed method there is a significant performance improvement in DTLC error when noise is injected to the inputs, as can be seen in Figures 25a and 26a. While employing the noisy road network and ego position estimate for localization, the error promptly increases, achieving a DTLC MAE of over 0.5 meters and great variance,



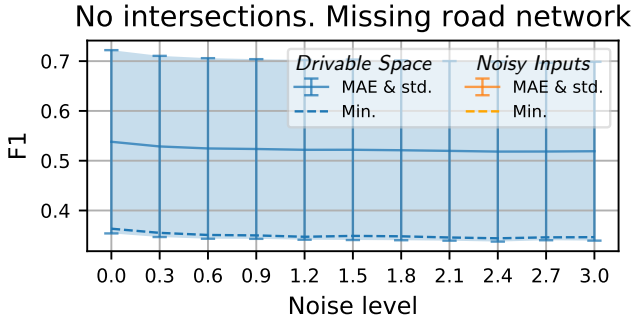
(a) DTLC error of scenarios 1 and 2 when all inputs are available



(b) DTLC error of scenarios 1 and 2 missing the road network input

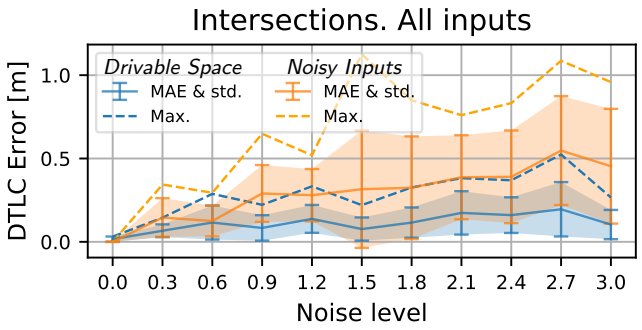


(c) F1 of scenarios 1 and 2 when all inputs are available

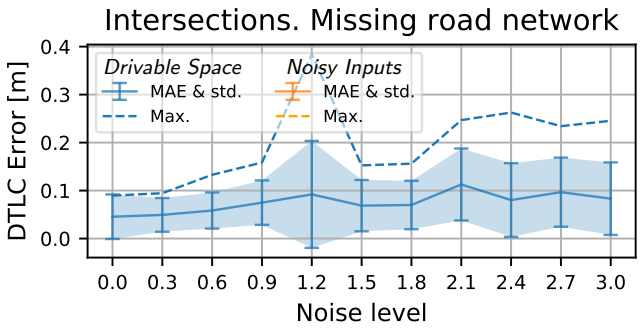


(d) F1 of scenarios 1 and 2 missing the road network input

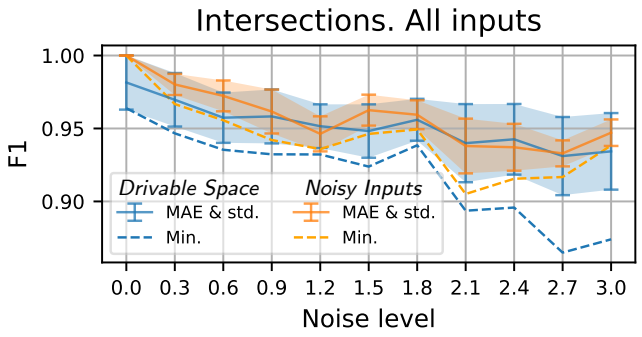
Fig. 25. DTLC error and F1 score achieved under different levels of noise in scenarios without intersections. The dashed line indicates the mean performance metric and the vertical bar the standard deviation. *Drivable Space* (blue) indicates the performance achieved using the proposed method, as opposed to *Noisy Inputs* (orange) which represents the performance achieved using the noisy inputs directly.



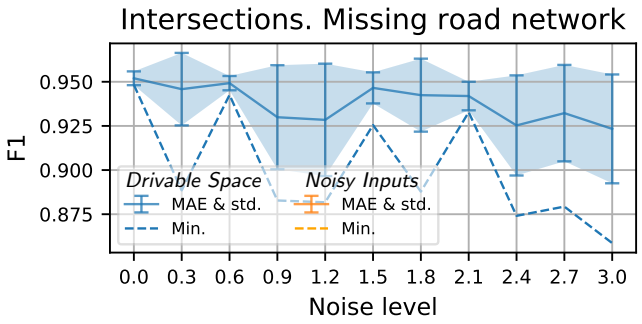
(a) DTLC error of scenarios 3 and 4 when all inputs are available



(b) DTLC error of scenarios 3 and 4 missing the road network input



(c) F1 of scenarios 3 and 4 when all inputs are available



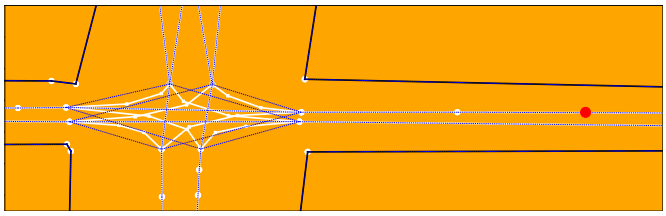
(d) F1 of scenarios 3 and 4 missing the road network input

Fig. 26. DTLC error and F1 score achieved under different levels of noise in scenarios with intersections.

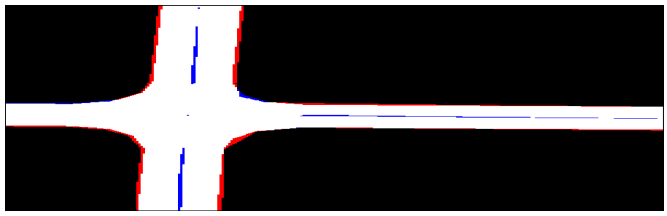
and a maximum DTLC error of over 1 meter. Furthermore, if information about the road network is unavailable, localization becomes impossible. On the other hand, the proposed drivable space method allows maintaining the DTLC MAE under 0.31 meters and significantly lower variance in all the testing scenarios, even under the highest noise levels and without a prior road network. Interestingly, the maximum DTLC error is recorded in testing scenario 3 when all inputs are present, achieving 0.52 meters in the experiments with one of the highest levels of noise (Figure26a), and removing the noisy road network improves DTLC error to a maximum of 0.38 meters (Figure 26b).

When investigating the performance in drivable space classification, there is a significant difference between scenarios with and without intersections. As summarized in Figure 25c, the classification performance in scenarios 1 and 2 is superior using our proposed method. However, in Figure 26c one can observe that in intersection scenarios the proposed drivable space no longer achieves a superior performance. This drop in performance can be explained by the inability of the proposed method to recover the correct topology in intersection scenarios. An example of this situation was already presented in Figure 20c, and a similar case can be seen in Figure 27, which showcases an incorrect reconstruction affecting the final classification performance. Nonetheless, it is still advantageous that this classification is available even without any prior knowledge of the road model.

Despite an imperfect reconstruction in some instances of the intersection scenarios, the proposed method is able to discover much of the drivable space, and even without any prior knowledge of the road network and other inputs being highly unreliable, it is able to reconstruct a road model that can be used to localize the ego within its lane accurately.



(a) Scenario 4 - Drivable Space



(b) Scenario 4 - Drivability Grid

Fig. 27. Example of wrong topology in drivable space reconstruction and resulting drivability grid of scenario 4 under noise level of 1.2.

C. Limitations

The main shortcoming of our proposed drivable space estimation is that it is brittle in intersection scenarios due to its inability to correctly recover the topology of the intersection. This inability to recover the topology is partially caused by empirically extracted projections such as traffic lights, which do not capture an average road connectivity and geometry. Traffic light projections were significantly simplified during the extraction process, modeled as simple straight lines, and their geometry cannot be easily modified in the current procedure. Despite allowing discovery of roads at intersections when the road network is not available, they can introduce some issues when it is, as shown by the reconstructions of scenarios 3 and 4.

Furthermore, only subjecting the graph nodes to the optimization process leads to some issues when deciding what segments to merge, and resulting in an incorrect reconstruction, such as the example shown in Figure 27.

Finally, at the moment estimation is only being performed once, but this estimate can be used as a prior in the shape of another input in the drivable space estimation of a future time-step, and increase the confidence we have in parts of the drivable space that are supported by new sensor data, while decreasing the confidence we have in those parts that are not supported by other inputs.

VI. Conclusion

For automated vehicles to operate safely, it is required that an accurate representation of the world is available at all times, and that the vehicle can be accurately localized within this representation of the world. This world modeling and localization process must be resilient towards any kind of complication, including sensor failure, adverse weather conditions, or outdated information of the inputs. To that end, we investigated how to incorporate AD domain knowledge with ego sensor data and introduced the foundations of a novel method to model the drivable space around the ego vehicle and provide in-lane localization accurate enough to allow for the safe and uninterrupted operation of an automated vehicle.

Our proposed method for drivable space modeling exploits semantic labels of sensed high-level landmarks to estimate the drivability of the space around such landmarks. These estimations are referred to as projections, and they are modeled as a probabilistic graph, which can be extracted from both domain knowledge-based assumptions and empirical data. Next, the nodes of the projections and its nodes are used to formulate a SLAM optimization problem and achieve the most likely configuration of its elements, providing a final estimation of the drivable space.

The method provides accurate in-lane localization, and is robust towards unreliable and missing inputs. Even if there is no prior information available about the road network, the proposed method achieved a maximum DTLC error of 0.09 meters if all other inputs are noise-free, and 0.38 meters when the other inputs are extremely unreliable. However, topology recovery is brittle in intersection scenarios, and further research is required for these cases.

Despite some limitations, the proposed drivable space estimation method already shows several benefits in some situations, and has been designed in a way that can be easily integrated in the software architecture of a typical automated vehicle in the future.

VII. Future Work

Several interesting research directions are available to improve the proposed method.

a) Extension to multiple time-steps: At the moment estimation of the drivable space is being performed for one time instant, but the proposed method allows to receive such an estimate as prior in the next timestep, and it is processed just as another drivability projection. Modeling a decay of the nodes' probability of existence with time would allow incorrectly introduced elements to be automatically removed if there is not new evidence supporting their existence.

b) Various drivability projections improvements: Drivability projections that are extracted empirically would greatly benefit from machine learning procedures, which can generalize from vast amounts of data in an automated manner and better capture the average road geometry around detected objects. For domain knowledge-based projections of dynamic objects, currently only past and present information (i.e. observed trajectories) is incorporated in the projections. However, prediction models are available to anticipate these dynamic elements' intentions [84] and incorporate their predicted motions in the projection extraction procedure.

c) Projections from regulatory elements: Many regulatory elements commonly found on the road also carry information about the existence of nearby roads. For instance, a sign indicating a mandatory right turn could be used to extract a projection capturing such road segment. Furthermore, the state of a traffic light or other restricting regulatory elements such as the direction of traffic could be used to determine the availability of a drivable surface: there might be a drivable surface ahead, but it is currently not drivable, or not at all drivable for the ego vehicle. This distinction between drivability and availability is currently not supported.

d) Subject edges to the optimization process: Currently only graph nodes are used to build the graph that is subject to optimization, and road segment geometry is estimated by simple polynomial regression according to the configuration of nodes after the optimization. If the geometry of edges were parameterized and introduced in the optimization process, more advanced constraints could be introduced (e.g. maximum curvature or geometric continuity). Furthermore, by introducing an edge element in the optimization, it would allow to establish data association constraints between edges directly and alleviate some topology issues introduced when merging only one segment knot.

Additionally, various implementation-specific improvements are possible. An overview of these improvements is provided in Appendix E.

REFERENCES

[1] M. M. Morando, Q. Tian, L. T. Truong, and H. L. Vu, "Studying the Safety Impact of Autonomous Vehicles Using Simulation-Based Surrogate Safety Measures," *Journal of Advanced Transportation*, 2018.

[2] J. Ploeg, A. F. A. Serrarens, and G. J. Heijenk, "Connect & Drive: design and evaluation of cooperative adaptive cruise control for congestion reduction," *Journal of Modern Transportation*, vol. 19, no. 3, p. 207–213, 2011.

[3] D. Milakis, B. Van Arem, and B. Van Wee, "Policy and society related implications of automated driving: A review of literature and directions for future research," *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 2017.

[4] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016," Tech. Rep., 2018.

[5] O. Pink and C. Stiller, "Automated map generation from aerial images for precise vehicle localization," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2010.

[6] T. Zhang, S. Arrigoni, M. Garozzo, D. g. Yang, and F. Cheli, "A lane-level road network model with global continuity," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 32–50, 2016.

[7] X. Xie, K. B. Y. Wong, H. Aghajan, P. Veelaert, and W. Philips, "Inferring directed road networks from GPS traces by track alignment," *ISPRS International Journal of Geo-Information*, 2015.

[8] X. Xie, K. B.-Y. Wong, H. Aghajan, P. Veelaert, and W. Philips, "Road network inference through multiple track alignment," *Transportation Research Part C: Emerging Technologies*, vol. 72, pp. 93–108, 2016.

[9] X. Yang, L. Tang, L. Niu, X. Zhang, and Q. Li, "Generating lane-based intersection maps from crowdsourcing big trace data," *Transportation Research Part C: Emerging Technologies*, 2018.

[10] L. Zheng, B. Li, B. Yang, H. Song, and Z. Lu, "Lane-level road network generation techniques for lane-level maps of autonomous vehicles: A survey," *Sustainability (Switzerland)*, vol. 11, no. 16, pp. 1–19, 2019.

[11] W.-C. Ma, I. Tartavull, I. A. Bârsan, S. Wang, M. Bai, G. Mattyus, N. Homayounfar, S. K. Lakshmikanth, A. Pokrovsky, and R. Urtasun, "Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization," 2019. [Online]. Available: http://arxiv.org/abs/1908.03274

[12] I. Delaney, "Hd live map explained," Available online: https://360.here.com/2016/01/07/hd-live-map-explained/, accessed on 2020-06-25.

[13] K. Chellapilla, "Rethinking Maps for Self-Driving," Available online: https://medium.com/lyftlevel5/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6, accessed on 2020-06-25.

[14] K. Jiang, D. Yang, C. Liu, T. Zhang, and Z. Xiao, "A Flexible Multi-Layer Map Model Designed for Lane-Level Route Planning in Autonomous Vehicles," *Engineering*, 2019.

[15] "Safespot final report," Available online: http://www.safespot-eu.org/documents/D8.1.1_Final_Report_-_Public_v1.0.pdf, accessed on 2020-06-25.

[16] D. Yang, X. Jiao, K. Jiang, and Z. Cao, "Driving Space for Autonomous Vehicles," *Automotive Innovation*, vol. 2, no. 4, pp. 241–253, 2019. [Online]. Available: https://doi.org/10.1007/s42154-019-00081-1

[17] A. Elfes, "Sonar-Based Real-World Mapping and Navigation," *IEEE Journal on Robotics and Automation*, 1987.

[18] H. Mouhagir, R. Talj, V. Cherfaoui, F. Aioun, and F. Guillemard, "Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation," in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2016.

[19] S. Hoermann, P. Henzler, M. Bach, and K. DIetmayer, "Object Detection on Dynamic Occupancy Grid Maps Using Deep Learning and Automatic Label Generation," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018.

[20] R. V. Carneiro, R. C. Nascimento, R. Guidolini, V. B. Cardoso, T. Oliveira-Santos, C. Badue, and A. F. De Souza, "Mapping Road Lanes Using Laser Remission and Deep Neural Networks," in *Proceedings of the International Joint Conference on Neural Networks*, 2018.

[21] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, 1974.

[22] M. W. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, 2001.

[23] J. Guivant, E. Nebot, and S. Baiker, "Localization and map building using laser range sensors in outdoor applications," *Journal of Robotic Systems*, 2000.

[24] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): part I The Essential Algorithms," *Robotics & Automation Magazine*, vol. 2, pp. 99–110, 2006.

[25] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[26] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, 2010.

[27] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," in *International Journal of Robotics Research*, 2006.

[28] C. Kim, S. Cho, M. Sunwoo, and K. Jo, "Crowd-sourced mapping of new feature layer for high-definition map," *Sensors (Switzerland)*, 2018.

[29] H. Kaartinen, J. Hyyppä, A. Kukko, A. Jaakkola, and H. Hyyppä, "Benchmarking the performance of mobile laser scanning systems using a permanent test field," *Sensors (Switzerland)*, 2012.

[30] G. P. Gwon, W. S. Hur, S. W. Kim, and S. W. Seo, "Generation of a Precise and Efficient Lane-Level Road Map for Intelligent Vehicle Systems," *IEEE Transactions on Vehicular Technology*, 2017.

[31] C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, P. Furgale, T. Sattler, and M. Pollefeys, "3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection," *Image and Vision Computing*, 2017.

[32] X. Xie, K. B. Y. Wong, H. Aghajan, P. Veelaert, and W. Philips, "Road network inference through multiple track alignment," *Transportation Research Part C: Emerging Technologies*, 2016.

[33] B. Yang, L. Fang, Q. Li, and J. Li, "Automated extraction of road markings from mobile lidar point clouds," *Photogrammetric Engineering and Remote Sensing*, 2012.

[34] M. Soilán, B. Riveiro, J. Martínez-Sánchez, and P. Arias, "Segmentation and classification of road markings using MLS data," *ISPRS Journal of Photogrammetry and Remote Sensing*, 2017.

[35] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[36] T. Youjin, C. Wei, L. Xingguang, and C. Lei, "A robust lane detection method based on vanishing point estimation," in *Procedia Computer Science*, 2018.

[37] Y. Xing, C. Lv, H. Wang, D. Cao, and E. Velenis, "Dynamic integration and online evaluation of vision-based lane detection algorithms," *IET Intelligent Transport Systems*, 2019.

[38] X. Zhang, W. Yang, X. Tang, and J. Liu, "A fast learning method for accurate and robust lane detection using two-stage feature extraction with YOLO v3," *Sensors (Switzerland)*, 2018.

[39] C. Guo, J. I. Meguro, Y. Kojima, and T. Naito, "Automatic lane-level map generation for advanced driver assistance systems using low-cost sensors," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2014.

[40] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[41] "Opendrive specification," Available online: https://www.asam.net/standards/detail/opendrive/, accessed on 2020-06-25.

[42] "Rndf," Available online: https://www.grandchallenge.org/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf, accessed on 2020-06-25.

[43] "Nds," Available online: https://nds-association.org/, accessed on 2020-06-25.

[44] "Open lane model," Available online: http://www.openlanemodel.org/, accessed on 2020-06-25.

[45] T. Eiter, H. Füreder, F. Kasslatter, J. X. Parreira, and P. Schneider, "Towards a Semantically Enriched Local Dynamic Map," *International Journal of Intelligent Transportation Systems Research*, 2019.

[46] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[47] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," in *International Journal of Robotics Research*, 2006.

[48] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, 2008.

[49] T. L. Dean and K. Kanazawa, "Probabilistic temporal reasoning." in *AAAI*, 1988, pp. 524–529.

[50] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, 2001.

[51] J. Dong and Z. Lv, "miniSAM: A flexible factor graph non-linear least squares optimization framework," *CoRR*, vol. abs/1909.00903, 2019. [Online]. Available: http://arxiv.org/abs/1909.00903

[52] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, 1997.

[53] D. Avitzour, "A Maximum Likelihood Approach to Data Association," *IEEE Transactions on Aerospace and Electronic Systems*, 1992.

[54] Y. Bar-Shalom, T. E. Fortmann, and P. G. Cable, " Tracking and Data Association ," *The Journal of the Acoustical Society of America*, 1990.

[55] T. Bailey, "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments," *Philosophy*, 2002.

[56] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Transactions on Robotics and Automation*, 2001.

[57] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "Fastslam: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, 2004.

[58] D. Hähnel, S. Thrun, B. Wegbreit, and W. Burgard, "Towards lazy data association in SLAM," *Springer Tracts in Advanced Robotics*, 2005.

[59] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[60] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013.

[61] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, no. 1, 2008.

[62] A. Kobryń, *Transition curves for highway geometric design*. Springer, 2017, vol. 14.

[63] J. McCrae and K. Singh, "Sketching piecewise clothoid curves," *Computers and Graphics (Pergamon)*, 2009.

[64] E. Bertolazzi and M. Frego, "G1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, 2015.

[65] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011.

[66] R. Bartoszynski and M. Niewiadomska-Bugaj, *Probability and Statistical Inference*, 2007.

[67] P. Virtanen *et al.*, "SciPy 1.0: fundamental algorithms for scientific computing in Python," *Nature Methods*, 2020.

[68] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, 1944.

[69] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[70] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, 1972.

[71] D. H. Douglas and T. K. Peucker, "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, 1973.

[72] A. K. Boyat and B. K. Joshi, "A Review Paper : Noise Models in Digital Image Processing," *Signal & Image Processing : An International Journal*, 2015.

[73] M. Fiaz, A. Mahmood, and S. K. Jung, "Tracking noisy targets: A review of recent object tracking approaches," *arXiv preprint arXiv:1802.03098*, 2018.

[74] A. Sanfeliu, A. Sanfeliu, and K. S. Fu, "A Distance Measure Between Attributed Relational Graphs for Pattern Recognition," *IEEE Transactions on Systems, Man and Cybernetics*, 1983.

[75] G. Jeh and J. Widom, "SimRank: A measure of structural-context similarity," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

[76] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[77] G. B. D. for Transport, G. B. D. . V. S. Agency, Driver, and V. S. Agency, *The Official Highway Code*. Stationery Office, 2015.

[78] A. Joshi and M. R. James, "Generation of accurate lane-level maps from coarse prior maps and lidar," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 19–29, 2015.

[79] W. Jang, J. An, S. Lee, M. Cho, M. Sun, and E. Kim, "Road Lane Semantic Segmentation for High Definition Map," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018.

[80] A. Chen, A. Ramanandan, and J. A. Farrell, "High-precision lane-level road map building for vehicle navigation," in *Record - IEEE PLANS, Position Location and Navigation Symposium*, 2010.

[81] J. Zhao, Y. Huang, X. He, S. Zhang, C. Ye, T. Feng, and L. Xiong, "Visual semantic landmark-based robust mapping and localization for autonomous indoor parking," *Sensors (Switzerland)*, 2019.

[82] X. Chen, A. Milioto, E. Palazzolo, P. Giguere, J. Behley, and C. Stachniss, "SuMa ++ : Efficient LiDAR-based Semantic SLAM," *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pp. 4530–4537, 2019. [Online]. Available: https://github.com/PRBonn/semantic_suma/

[83] J. Thomas, J. Tatsch, W. Van Ekeren, R. Rojas, and A. Knoll, "Semantic grid-based road model estimation for autonomous driving," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019.

[84] M. Muñoz Sánchez, E. Silvas, D. Pogosov, and D. C. Mocanu, "A Hybrid Framework Combining Vehicle System Knowledge with Machine Learning Methods for Improved Highway Trajectory Prediction," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2020.

[85] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[86] K. Tong, Z. Ajanovic, and G. Stettinger, "Overview of tools supporting planning for automated driving," *arXiv preprint arXiv:2003.04081*, 2020.

[87] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," 2019.

[88] C. Pilz, G. Steinbauer, M. Schratter, and D. Watzenig, "Development of a scenario simulation platform to support autonomous driving verification," in *2019 8th IEEE International Conference on Connected Vehicles and Expo, ICCVE 2019 - Proceedings*, 2019.

[89] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," 2018.

[90] "Deepdrive," Available online: https://deepdrive.voyage.auto/, accessed on 2020-06-25.

[91] "Lgsvl simulator," Available online: https://www.lgsvlsimulator.com/, accessed on 2020-06-25.

[92] "Cognata," Available online: https://www.cognata.com/, accessed on 2020-06-25.

[93] "Udacity open sources its self-driving car simulator for anyone to use," Available online: https://techcrunch.com/2017/02/08/udacity-open-sources-its-self-driving-car-simulator-for-anyone-to-use/, accessed on 2020-06-25.

[94] V. Méndez, H. Catalán, J. R. Rosell, J. Arnó, R. Sanz, and A. Tarquis, "SIMLIDAR - Simulation of LIDAR performance in artificially simulated orchards," *Biosystems Engineering*, 2012.

[95] S. Bechtold and B. Höfle, "Helios: a Multi-Purpose LIDAR Simulation Framework for Research, Planning and Training of Laser Scanning Operations with Airborne, Ground-Based Mobile and Stationary Platforms," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2016.

[96] J. O. Woods and J. A. Christian, "GLIDAR: An open GL-based, real-time, and open source 3d sensor simulator for testing computer vision algorithms," *Journal of Imaging*, 2016.

[97] "Carcraft," Available online: https://www.engadget.com/2017-09-11-waymo-self-driving-car-simulator-intersection.html, accessed on 2020-06-25.

[98] "Drive constellation," Available online: https://www.nvidia.com/en-us/self-driving-cars/drive-constellation/, accessed on 2020-06-25.

PUBLICATIONS

At the time of writing of this thesis, the author has contributed in the following articles:

1. [Accepted] M. Muñoz Sánchez, E. Silvas, D. Pogosov, D. C. Mocanu. "A Hybrid Framework Combining Vehicle System Knowledge with Machine Learning Methods for Improved Highway Trajectory Prediction", in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2020.

2. [In preparation] M. Muñoz Sánchez, D. Pogosov, E. Silvas, D. C. Mocanu. "Domain Knowledge-based Drivable Space Estimation". Submission conference or journal not yet determined, 2020.

APPENDIX

A. Overview of Notation

For ease of readability, other than standard notation, non-standard terminology and notation has been introduced in this work to refer to specific concepts of our implementation. This terminology is introduced in-text throughout the report when it is first used. For convenience, this section provides an overview of the notation used in the report, along with a brief description of its meaning.

$\mathbf{x}$: robot state
$\mathbf{x}_t$: robot state at time t
$\mathbf{x}_{1:T}$: robot states at times $t = \{1, ..., T\}$
$\mathbf{m}$: map
$\mathbf{u}$: odometry measurement or control applied to robot (subscripts have the same meaning as for the robot state $\mathbf{x}$)
$\mathbf{z}$: sensor measurement (subscripts have the same meaning as for the robot state $\mathbf{x}$)
$\hat{\mathbf{z}}$: expected relative spatial constraint
$\hat{\mathbf{z}}_{ij}$: expected relative spatial constraint nodes $\mathbf{n}_i$ and $\mathbf{n}_j$
$\mathbf{n}$: node
$\mathbf{n}_i$: ith nodes
$\mathbf{N}$: set of nodes
$\mathbf{N}^*$: set of nodes after optimization
$\mathbf{e}$: edge connecting nodes $\mathbf{n}_i$ and $\mathbf{n}_j$
$\mathbf{e}_{ij}$: edge
$\mathbf{E}$: set of edges
$\mathbf{\Omega}$: information matrix
$\mathbf{\Omega}_{ij}$ information matrix associated with a constraint relating nodes $\mathbf{n}_i$ and $\mathbf{n}_j$
ε_{ij}: deviation between the expected and actual measurement of the i^{th} and j^{th} nodes
$\mathcal{G}$: Graph representing the drivable space
$\mathbf{G}$: Graph encoding the optimization problem (i.e. Graph-based SLAM error minimization)
$\mathbf{n}_i^x$: x-coordinate of the i^{th} node
$\mathbf{n}_i^y$: y-coordinate of the i^{th} node
$\mathbf{n}_i^{xy}$: (x, y) coordinate pair of the i^{th} node
$\mathbf{n}_i^{\kappa}$: whether or not i^{th} is a knot, $\kappa \in \{0, 1\}$
$\mathbf{n}_i^{\tau}$: relative importance of the i^{th} node
$\mathbf{n}_i^{p^e}$: probability of existence of the i^{th} node
$\mathbf{e}^{p^d}$: probability of drivability of edge $\mathbf{e}$
p_{ij}^d: probability of drivability of the edge connecting nodes $\mathbf{n}_i$ and $\mathbf{n}_j$
$\bar{\mathbf{e}}_{ij}$: line segment connecting points $\mathbf{n}_i^{xy}$ and $\mathbf{n}_j^{xy}$
$\vec{\mathbf{e}}_{ij}$: directed vector from point $\mathbf{n}_i^{xy}$ to point $\mathbf{n}_j^{xy}$
$\mathbf{S}$: a *segment* of the drivable space, defined as a subgraph of the drivable space with specific properties (Section III-A).
$\mathbf{S}^{\mathbf{N}}$: set of all nodes of segment $\mathbf{S}$
$\mathbf{S}^{\mathbf{K}}$: set of knot nodes of segment $\mathbf{S}$
$\mathbf{S}^{\not\mathbf{K}}$: set of non-knot nodes of segment $\mathbf{S}$
$\mathbf{S}^{\mathbf{E}}$: set of all edges of segment $\mathbf{S}$
$\overline{\mathbf{S}^{\mathbf{K}}}$: line segment between the (x, y)-coordinates of knots of segment S
$\vec{\mathbf{S}}^{\mathbf{K}}$: directed vector between the (x, y)-coordinates of knots of segment S
ϕ: drivability projection
$\mathbf{\Phi}$: set of drivability projections
$\phi_i^{\mathbf{N}}$: Set of nodes of the i^{th} projection
$\phi_i^{\mathbf{E}}$: Set of edges of the i^{th} projection
$\phi_i^{\mathbf{S}}$: Set of segments of the i^{th} projection
$\mathbb{O}$: Set of all objects that are the input to our algorithm (i.e. high-level landmarks)
$\mathbb{D}$: Set of all drivability projections
$\mathbb{D}^+$: Set of all drivable projections
$\mathbb{D}^-$: Set of all non-drivable projections
$\mathbf{l}$: single traffic light
$\mathbf{L}$: set of traffic lights
$\mathbf{\Phi}_{\mathbf{L}}$: set of drivable projections extracted from traffic lights

$\vec{\mathbf{x}}$: unit vector in the direction of the x-axis.
θ_u: operation returning the smallest unsigned angle between two vectors
$\mathbf{\Phi}^*$: Set of projections after optimization
$h(\phi_i, \phi_j)$: Operation that aligns projections ϕ_i and ϕ_j (Section III-C)
$g(\mathbf{n}, \phi)$: operation returning the geometric projection of $\mathbf{n}^{xy}$ onto ϕ
$\mathbf{c}_{ij}$: an edge/constraint in $\mathbf{G}$ between nodes $\mathbf{n}_i$ and $\mathbf{n}_j$
ρ: (optional) kernel function used to weight a constraint error
$||\vec{\mathbf{e}}||$: magnitude of vector $\vec{\mathbf{e}}$
$\mathcal{G}^*$: Final drivable space estimation after merging the optimized projections $\mathbf{\Phi}^*$
$w(\mathbf{n})$: weight of node $\mathbf{n}$ when being merged with other nodes
$\mathbf{C}$: one connected component of a graph
$\mathcal{C}$: set of all connected components of a graph
DTLC: distance to lane center
$DTLC$: true DTLC
$\widehat{DTLC}$: estimated DTLC
ε_{DTLC}: DTLC error
$T_q(s, m)$: Percentage of topology reconstructions having quality q in testing scenario s when missing input m
$\Delta T_q(s, m)$: Change in percentage of topology reconstructions having quality q in testing scenario s when missing input m with respect to when all inputs are present
$\varepsilon_i(s, m)$: Performance given by metric i in testing scenario s when missing input m
$\Delta\varepsilon_i(s, m)$: Change in performance given by metric i in testing scenario s when missing input m with respect to when all inputs are present

B. Constants & Noise

All the base variance values used for noise sampling and constants used in the implementation are shown in Tables IV and V respectively. As briefly explained in Section IV-C, the sampled noise results from additive Gaussian noise with varying variances for each input, $\sigma^2 = \{0 \cdot \sigma^2_{base}, 0.3 \cdot \sigma^2_{base}, ..., 2.7 \cdot \sigma^2_{base}, 3 \cdot \sigma^2_{base}\}$. This noise is added to the inputs both laterally and longitudinally using the same variance.

In consultation with domain experts, we considered an expected relative noise between the different inputs to our algorithm. The highest level of noise is considered from a map, from which we extract our road network and buildings. Next, the traffic lights, following the sensed vehicles (and their observed trajectories) and lastly the camera-based ego lane from the Mobileye camera. Based on this relative ordering and the study performed in [73], the different base variances are as specified in Table IV.

TABLE IV
VARIANCE VALUES FOR NOISE SAMPLING OF DIFFERENT INPUTS AND CONSTRAINTS

Variable	Value	Description
VAR_ROAD	0.15	Base variance for error sampling added to the road network
VAR_BUILDING	0.15	Base variance for error sampling added to buildings
VAR_TRAF	0.1	Base variance for error sampling added to the traffic lights
VAR_TRAJ	0.05	Base variance for error sampling added to vehicle trajectories
VAR_MOBILEYE	0.01	Base variance for error sampling added to the ego lane
VAR_VEHICLE_MEASUREMENT	0.01	Base variance for error sampling added to ego-vehicle measurement constraints (see Section III-C2)
VAR_ROAD_MEASUREMENT	0.005	Base variance for error sampling added to ego-road measurement constraints (see Section III-C2)
VAR_BUILDING_MEASUREMENT	0.005	Base variance for error sampling added to ego-building measurement constraints (see Section III-C2)
VAR_MOTION_MODEL	0.02	Base variance for error sampling added to motion model constraints (see Section III-C2)
VAR_ODOMETRY	0.001	Base variance for error sampling added to odometry constraints (see Section III-C2)

As indicated in Table V, base drivability values for drivable projections are set to 1, and for non-drivable projections to 0. Importance values are specified for different projections relatively to control how much they contribute when merging different nodes (i.e. prior the highest, next static objects, next vehicle trajectories since vehicles might not be driving on the lane center and finally traffic lights, since elements from the projection are actually not observed).

TABLE V
CONSTANTS BEING USED IN THE IMPLEMENTATION

Variable	Value	Description
DRIVABILITY_TRAJECTORY, DRIVABILITY_TRAFFIC_LIGHT, DRIVABILITY_MOBILEYE, DRIVABILITY_ROAD	1	Base probability of drivability for nodes of drivable projections
DRIVABILITY_BUILDING, DRIVABILITY_TRAFFIC_LIGHT_POLE	0	Base probability of drivability for nodes of non-drivable projections
EXISTENCE_TRAJECTORY, EXISTENCE_BUILDING, EXISTENCE_MOBILEYE, EXISTENCE_ROAD, EXISTENCE_TRAFFIC_LIGHT_POLE	1	Base probability of existence for nodes of projections originating from different inputs
IMPORTANCE_BUILDING, IMPORTANCE_TRAFFIC_LIGHT_POLE, IMPORTANCE_MOBILEYE, IMPORTANCE_ROAD	2	Relative importance of nodes from static object projections
IMPORTANCE_PRIOR	3	Relative importance of nodes from prior on drivable space
IMPORTANCE_TRAFFIC_LIGHT	0.1	Relative importance of nodes from traffic light drivable projections
IMPORTANCE_TRAJECTORY	0.5	Relative importance of nodes from trajectory projections
MOBILEYE_MAX_REACH, LIGHTS_MAX_REACH, BUILDINGS_MAX_REACH, BUILDING_CONSTRAINT_MAX_DISTANCE, ROAD_CONSTRAINT_MAX_DISTANCE, VEHICLE_CONSTRAINT_MAX_DISTANCE	50	Maximum distance from ego to consider inputs and specify ego-input measurement constraints. Typical sensors have a longer range, especially in the case of laser sensors (i.e. LiDAR), which can reach up to several hundred meters [10]. However, in this work the maximum range is kept lower to simulate challenging conditions.

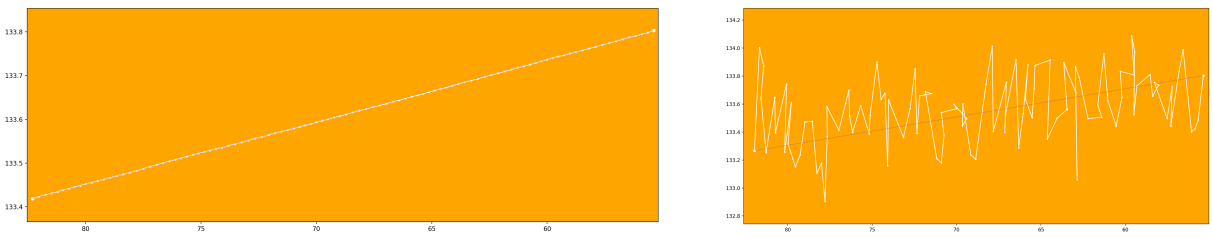


Fig. 28. Example of ground truth vehicle trajectory (left) and noisy trajectory after applying noise level 1 (right)

C. Simulators for Automated Driving

Before new techniques are implemented and tested on the road, they are first evaluated in driving simulators or general-purpose simulators that can be used in this context. For the purpose of this work, several simulators were reviewed according to the following criteria:

a) *Open Source License:* The simulator should be freely available to anyone.
b) *Python scripting:* In order to quickly design experiments and prototypes, the availability of a Python API or the possibility to manipulate the simulation via simple Python scripts is a key feature.
c) *Extensive documentation:* Many open-source projects have the disadvantage of poor documentation or lack of support from the community, which makes it challenging to get started with the tool, and possibly be unable to overcome some implementation issues.
d) *Advanced features:* With advanced features we refer to the availability of sensors typically available in AVs (other than LiDAR), or additional features (e.g. custom map loading) that allow extensions and make the simulator suitable for any specific scenario a researcher might want to recreate.

Table VIIa provides an overview comparing several state of the art simulators according to the specified criteria. For a more detailed comparison of these simulators, among others, the reader is referred to [86]–[88]. Out of the simulators considered, only CARLA, AirSim and LGSVL satisfied the specified requirements, therefore a more extensive comparison of them was performed. As the results summarized in Table VIIb indicate, the three simulators are quite similar in general, with LGSVL being the most demanding in terms of GPU requirements, and CARLA having the most active community at the moment.

TABLE VI
COMPARISON OF SIMULATORS

Simulator	Open Source	Python	Documen-tation	Advanced Features
CARLA [69]	✓	✓	✓	✓
AirSim [89]	✓	✓	✓	✓
DeepDrive [90]	✓	✓	x	✓
LGSVL [91]	✓	✓	✓	✓
Cognata [92]	x	✓	-	✓
Udacity [93]	✓	✓	-	✓
SIMLidar [94]	✓	x	x	x
Helios [95]	✓	x	x	x
GLIDAR [96]	✓	x	x	x
Constellation [97]	x	✓	-	✓
Carcraft [98]	x	-	-	✓

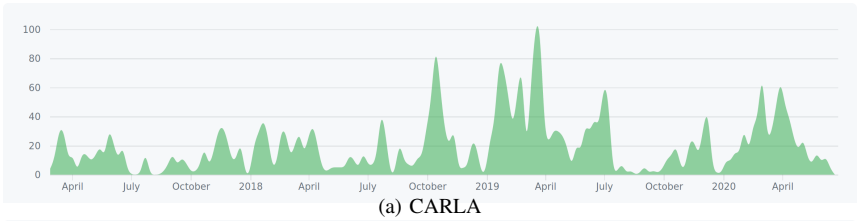
* Legend: ✓ – Yes | x – No | - – Unknown or unclear

(a) Overview of the reviewed simulators according to the specified criteria.

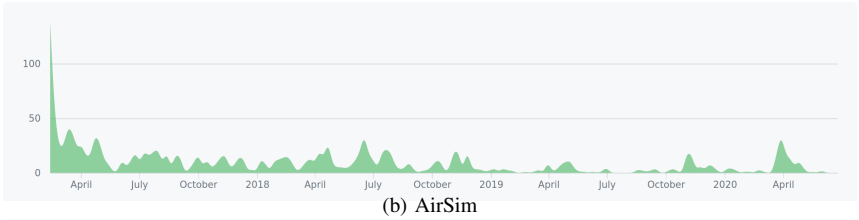
Feature	CARLA	AirSim	LGSVL
Last update*	1	29	33
Update trend**	Steady	Lightly downwards	Downwards
Physics Engine	Unreal	Unreal/Unity	Unity
Supported Platforms	Windows/Linux	Windows/Linux	Windows/Linux
Recommended Platform	Linux	-	Windows
GPU requirements	4GB	4GB	8GB
GPS	✓	✓	✓
IMU	✓	✓	✓
LiDAR	✓	✓	✓
Radar	✓	x	✓
Infrared	x	✓	x
Stereo/color Camera	✓	✓	✓
Depth Camera	✓	✓	✓
Semantic Segmentation	✓	✓	✓

*Days since last commit to the main branch at the time of writing.
**Change in update frequency since creation of the repository (see Figure 29)

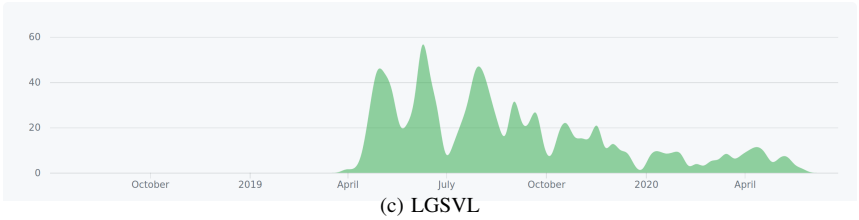
(b) More detailed comparison of CARLA, AirSim and LGSVL.



(a) CARLA



(b) AirSim



(c) LGSVL

Fig. 29. Number of commits to the main branch since the start of the simulator's repository

D. Step-by-step Example Execution

This section provides an example of the drivable space estimation step by step. The scenario used for this example is scenario 2 under maximum level of noise (noise level 3) and all inputs present. Figures 30 to 33 illustrate the execution of the drivable space estimation algorithm step by step and in 34 the final drivability grid classification is shown. In all figures illustrating the algorithm's execution, the left subfigure shows the entire area of the reconstruction, while the right one zooms in to show more detail about the elements nearby the ego vehicle.

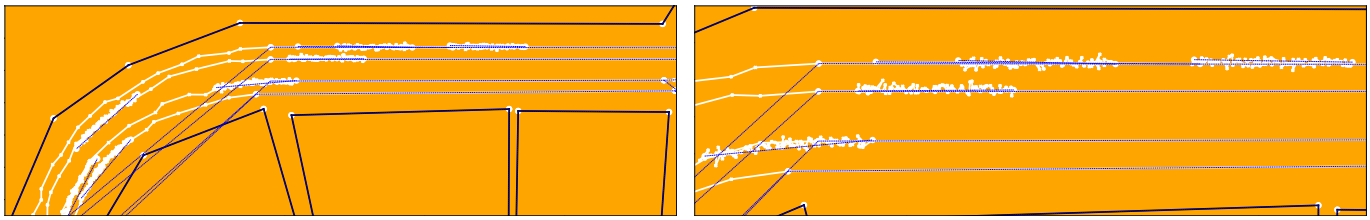


Fig. 30. First step of the algorithm: projections from the noisy inputs are extracted. The color of nodes indicate their probability of existence and the color of the edges their probability of drivability. The probabilities range from zero (black) to one (white). Blue dashed lines indicate a straight line segment between the knots of a projection segment.

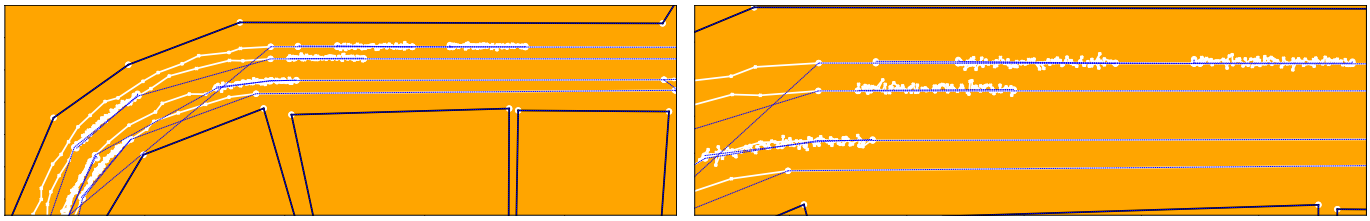


Fig. 31. Projections are aligned by geometric projection of segment knots onto segments of different projections. If the result of the geometric projection is closer than 1.75m (half of a typical lane width), a new node is introduced and data association constraint established between the nodes.

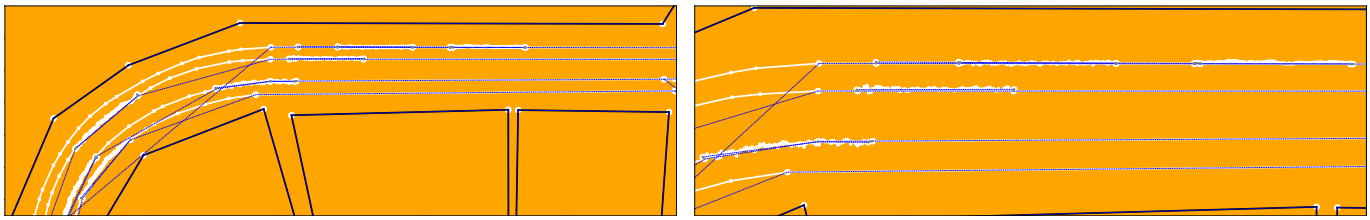


Fig. 32. Several other constraints are established between nodes depending on their semantic labels (e.g. odometry constraints) to obtain the most likely configuration of the drivable space. This figure illustrates the configuration of nodes after optimization is performed. Valid data association constraints between nodes are shown with dashed gray lines.

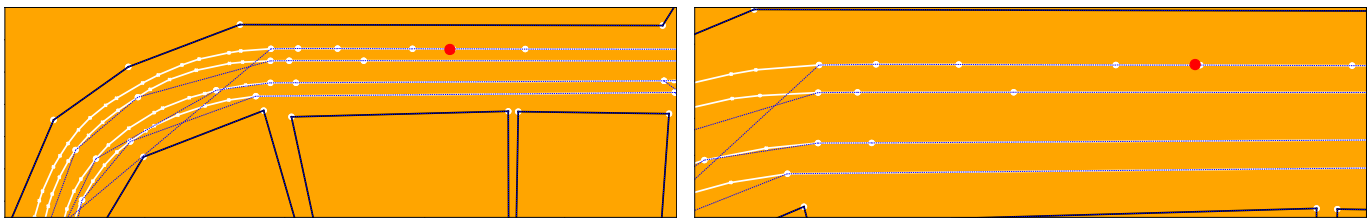


Fig. 33. Finally, after optimization the segments with valid data association constraints between their knots are merged to generate the final drivable space, which can can be used for accurate in-lane localization. The ego position within this space is represented by the red circle. DTLC error achieved in this setting is 0.0641 ± 0.0625m.

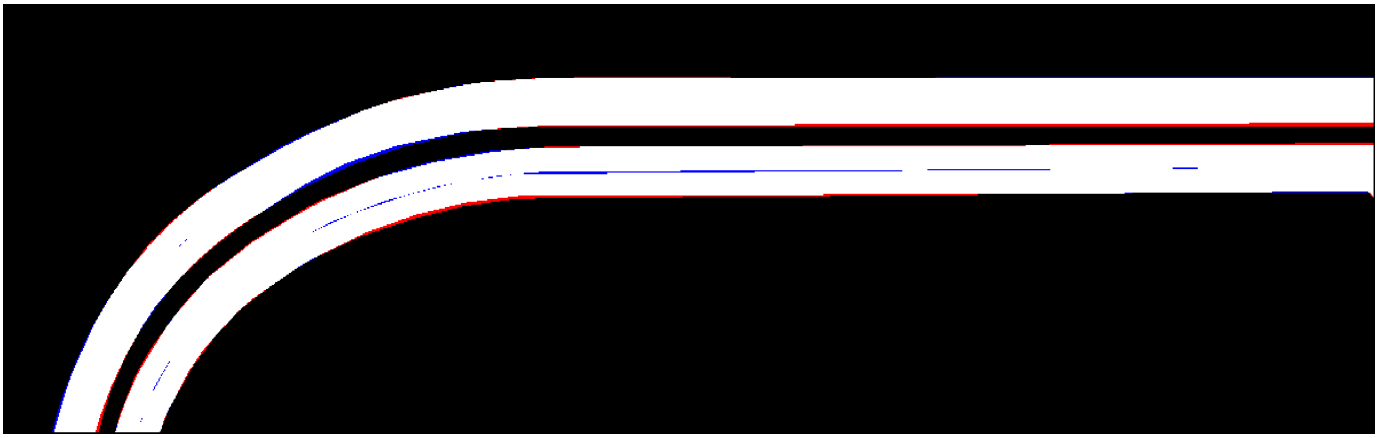


Fig. 34. Classification of drivable space resulting from the drivability grid. White cells are correctly classified as drivable and black cells correctly classified as non-drivable. Red cells indicate wrongly classified cells as drivable, and blue cells wrongly classified cells as non-drivable. The different performance metrics achieved in this reconstruction are as follows. Accuracy: 0.976 ± 0.012, precision: 0.94 ± 0.031, recall: 0.953 ± 0.024, and F1: 0.947 ± 0.027.

E. Future Work: Overview of Implementation Improvements

There are some implementation issues or improvements that have been identified and would considerably enhance the method, both conceptually and in terms of performance. However, they could not be implemented due to time constraints. In this section we provide a brief overview of these improvements.

a) Efficiency: The entire method was coded without taking any care of running time. As such, there is room for massive improvement in this regard in almost every component of the method.

b) Projection alignment order: When aligning different projections, the order in which they are processed is sometimes relevant. Consider the following case: 3 projections (a, b, c) are being aligned in that order. First, knots from a are projected onto b and c, and if close enough new knots are created in b and c. Next, knots of b are projected onto a and c, and it is possible that new knots in a are introduced, which will never be projected onto c because a was already processed. These cases should be considered, and newly introduced knots should be projected from already processed projections.

c) Intersecting edges during alignment: During alignment of projections, intersections of segments should introduce new nodes (see slides "Meeting 2020-06-08" included in the deliverables). There is a faulty implementation of this procedure already in the code, but it had to be discontinued to move forward in the project. Incorporating this procedure will improve topology recovery, but might introduce some unforeseen issues in testing scenarios with intersections.

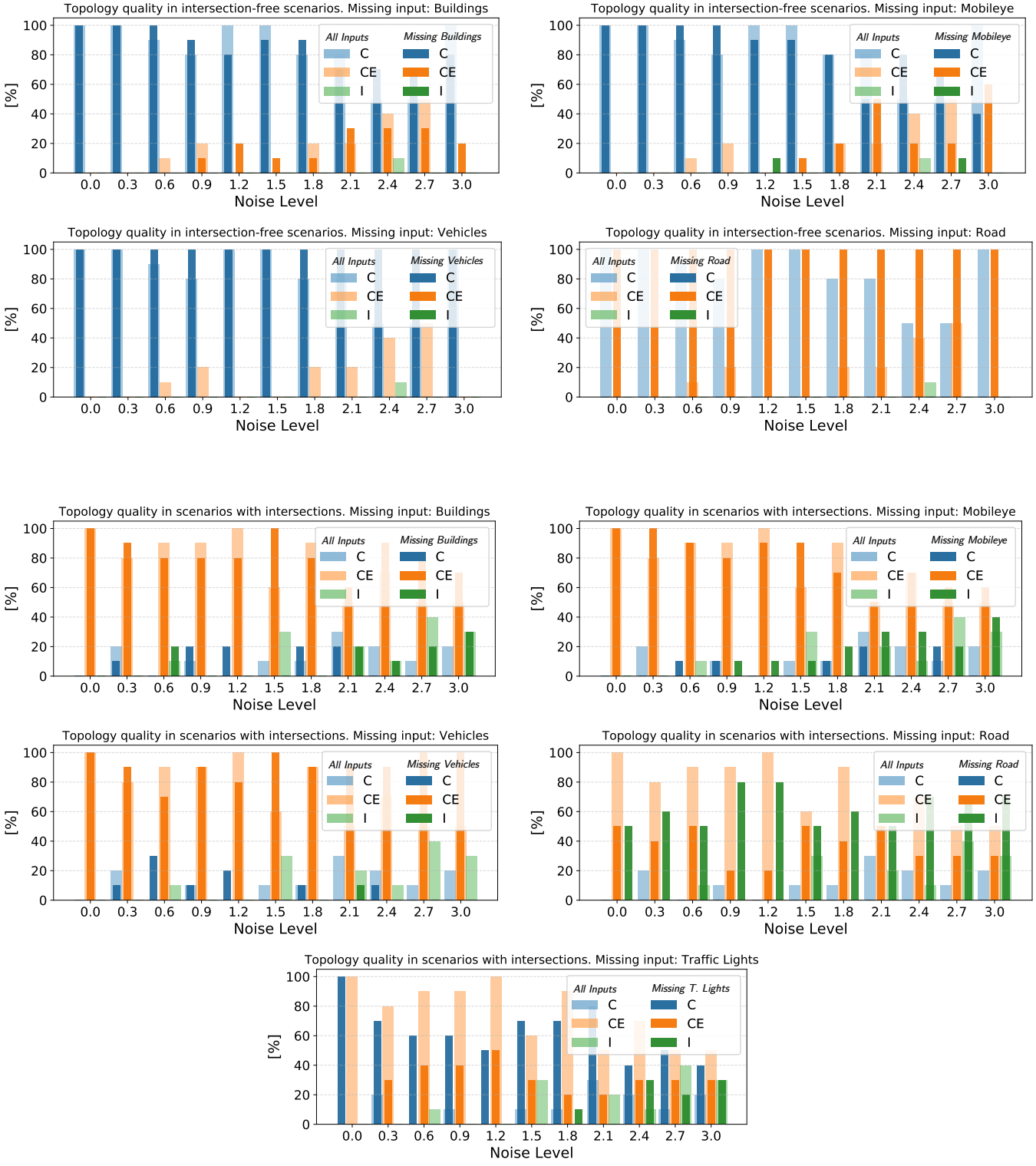
d) Mobileye simulation mismatch with road network: There is a small mismatch in the ego-lane extracted from the simulated Mobileye information and the road network. The entire road network is extracted and simplified as specified in IV-B. As such, nearly straight segments are simplified into a completely straight segment for the road network (which is used as ground truth for the reconstruction), but this is not the case for the Mobileye projections. Thus, sometimes the Mobileye projection introduces a small displacement in the drivable space reconstruction with respect to the ground truth road network (which is the simplified road network).

e) Non-drivable projections merging: Nodes from non-drivable projections are currently being processed in the same way as drivable projections, resulting in some nodes from different inputs being merged when they should not. For instance, nodes from projections of traffic lights (poles) and buildings are merged when they are close enough and slightly displace some non-drivable edges. This is not a significant issue, since it only occurs between non-drivable nodes that are 1.75m meters away, and this space is not sufficient for a road. However, they should be treated differently to avoid displacement of these non drivable areas.

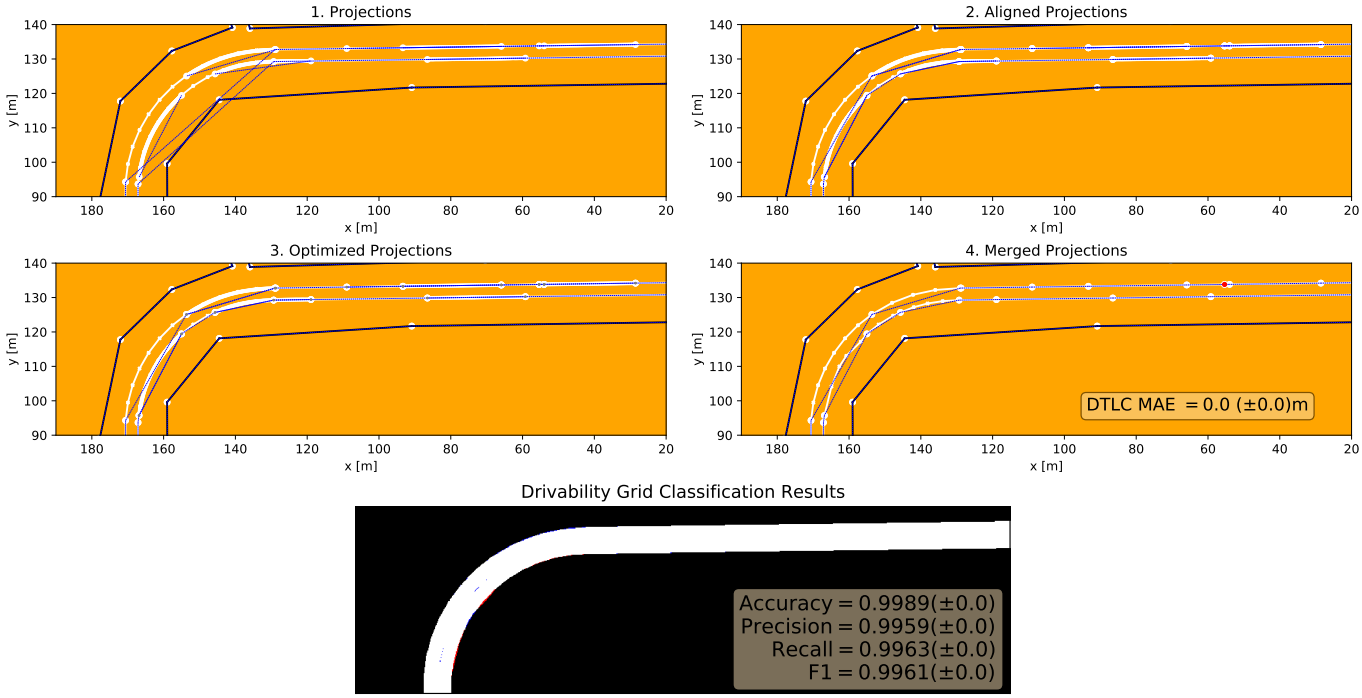
f) Lane width constraints between road network nodes: Currently only nodes of the drivable space originating from traffic lights are constrained to keep a distance of 3.5m between them. These constraints are not established between nodes originating from the road network due to implementation-specific difficulties. The road network extracted from CARLA does not readily provide information about which lanes are adjacent, and relying on the distance between them to decide what lanes to relate is not possible since noise is being injected to the inputs in our simulations before formulating the Graph-based SLAM optimization problem. Thus, it would require extra processing of the road network or modifying the pipeline of our method to inject noise after these constraints are established. Introducing lane width constraints between nodes of adjacent lanes from the road network would significantly improve the quality of the recovered road model.

F. Extended Experiment Results

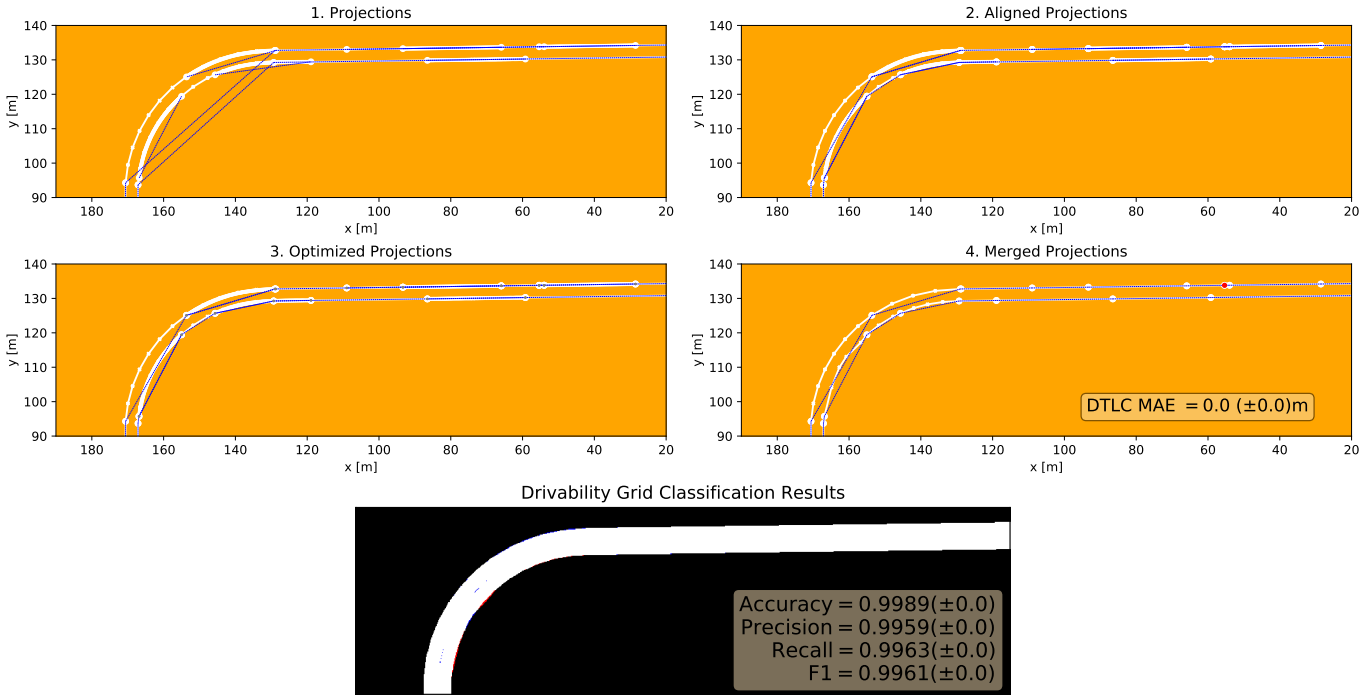
Topology Quality with Missing Inputs and Noise



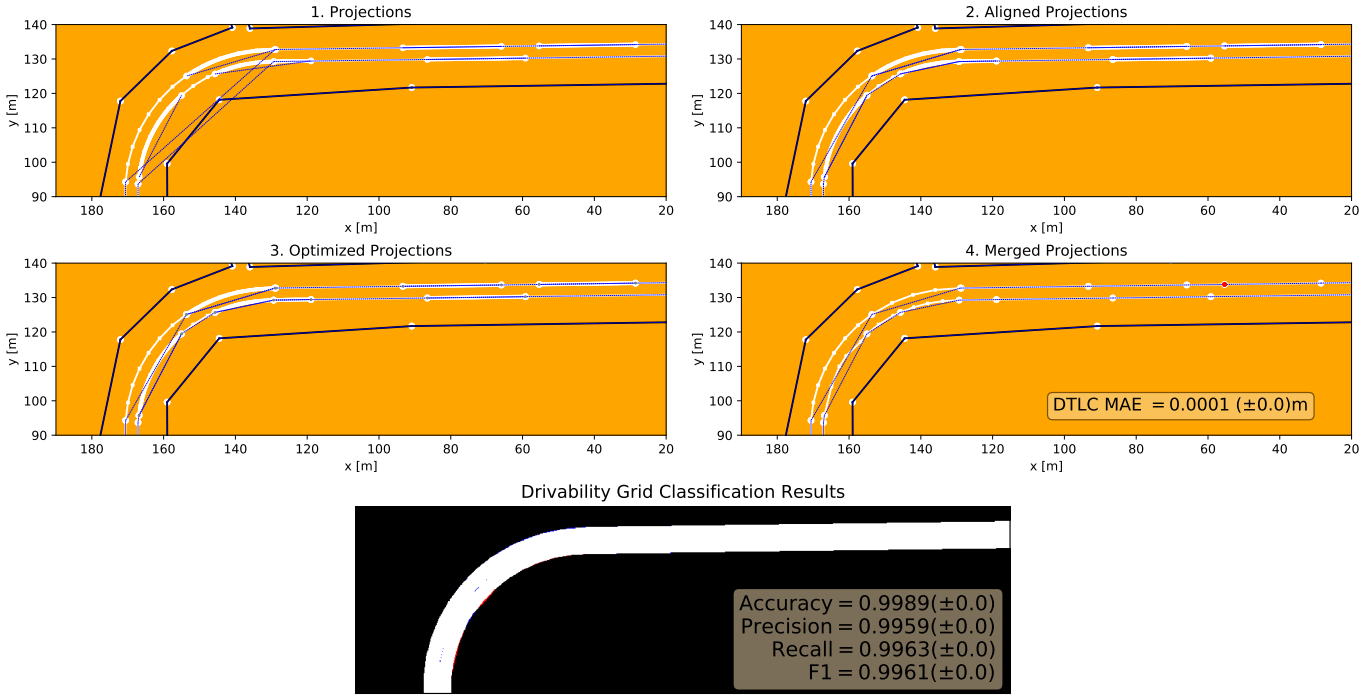
Scenario 3. Noise level 0.0. Missing input: None



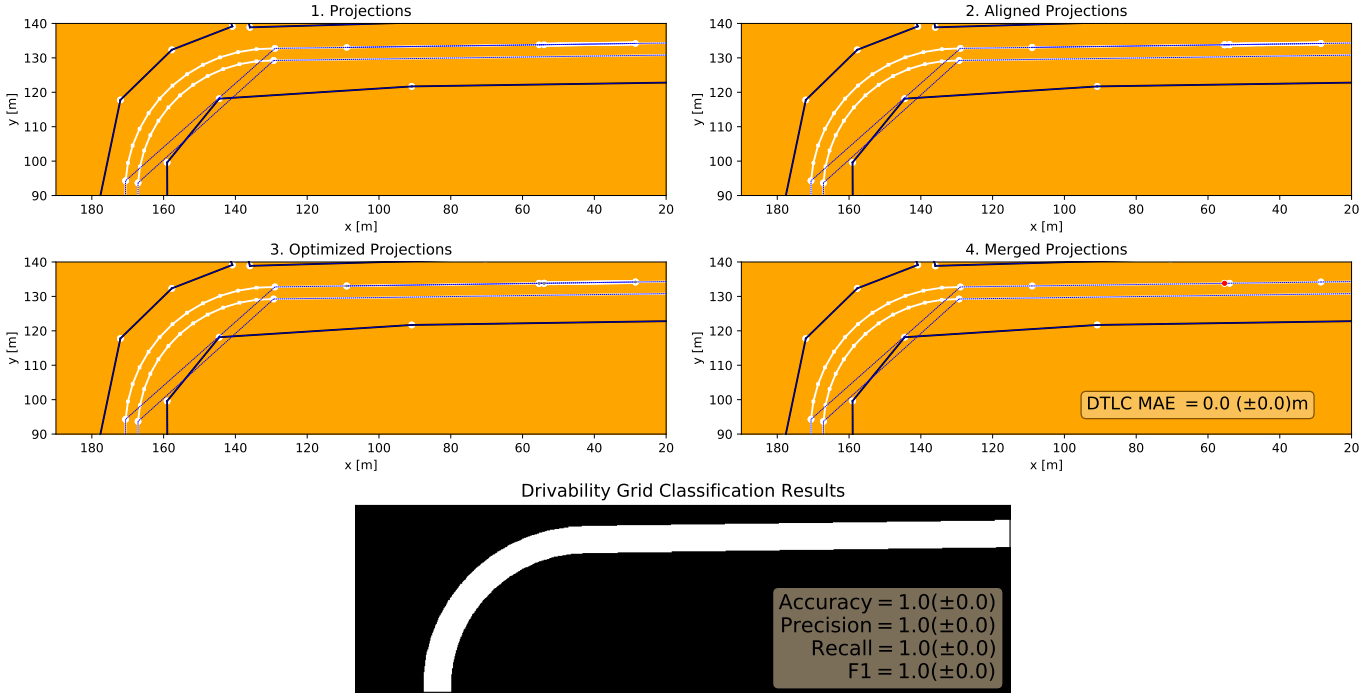
Scenario 3. Noise level 0.0. Missing input: Buildings



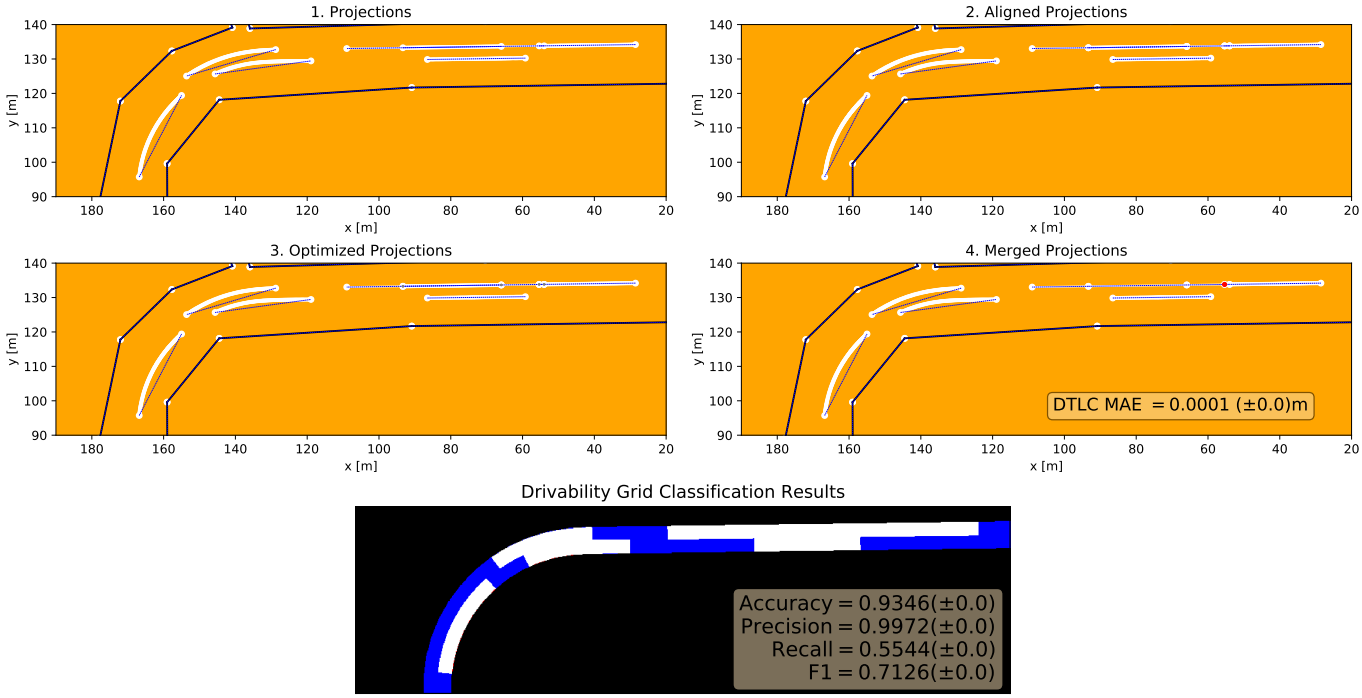
Scenario 3. Noise level 0.0. Missing input: Mobileye



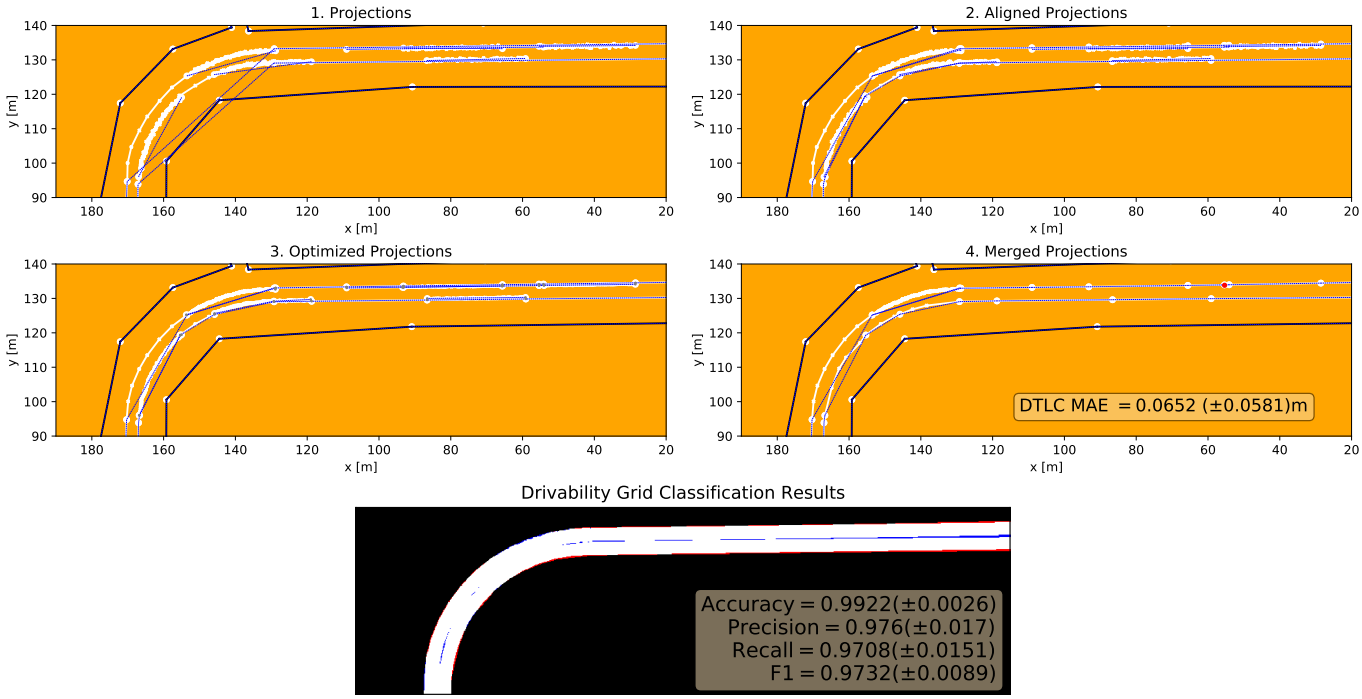
Scenario 3. Noise level 0.0. Missing input: Vehicles



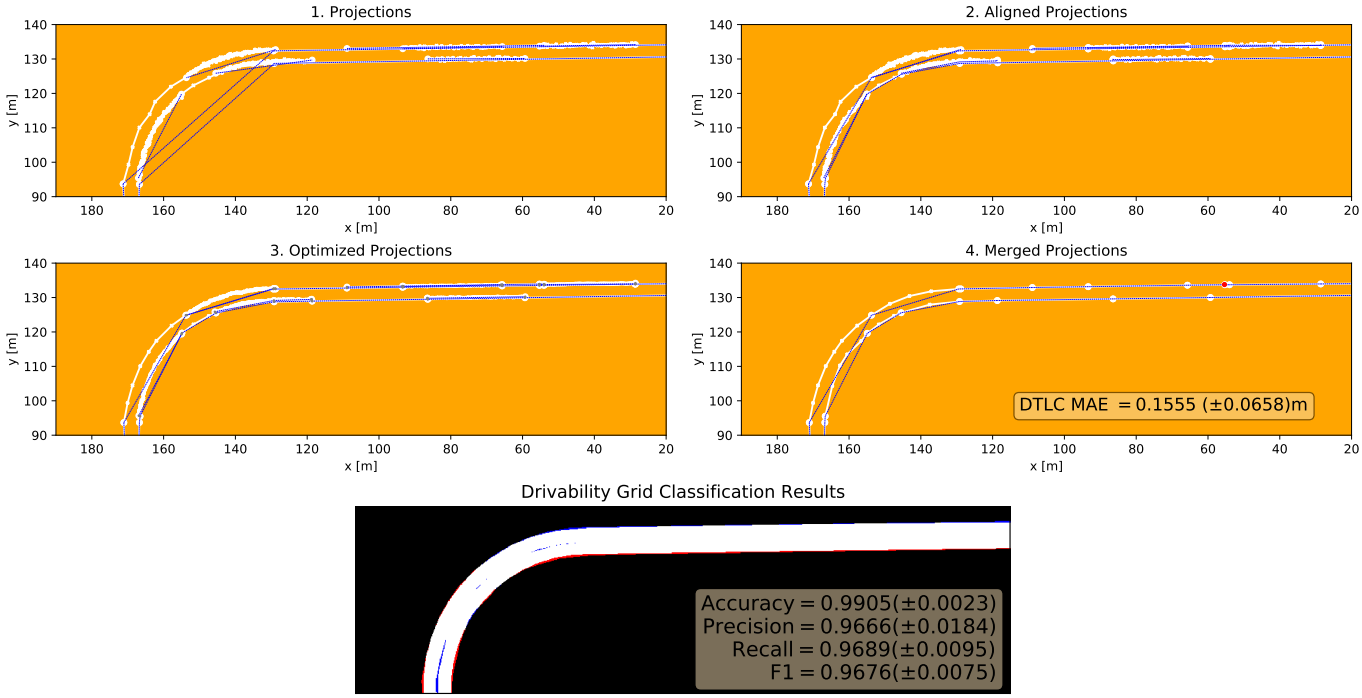
Scenario 3. Noise level 0.0. Missing input: Road network



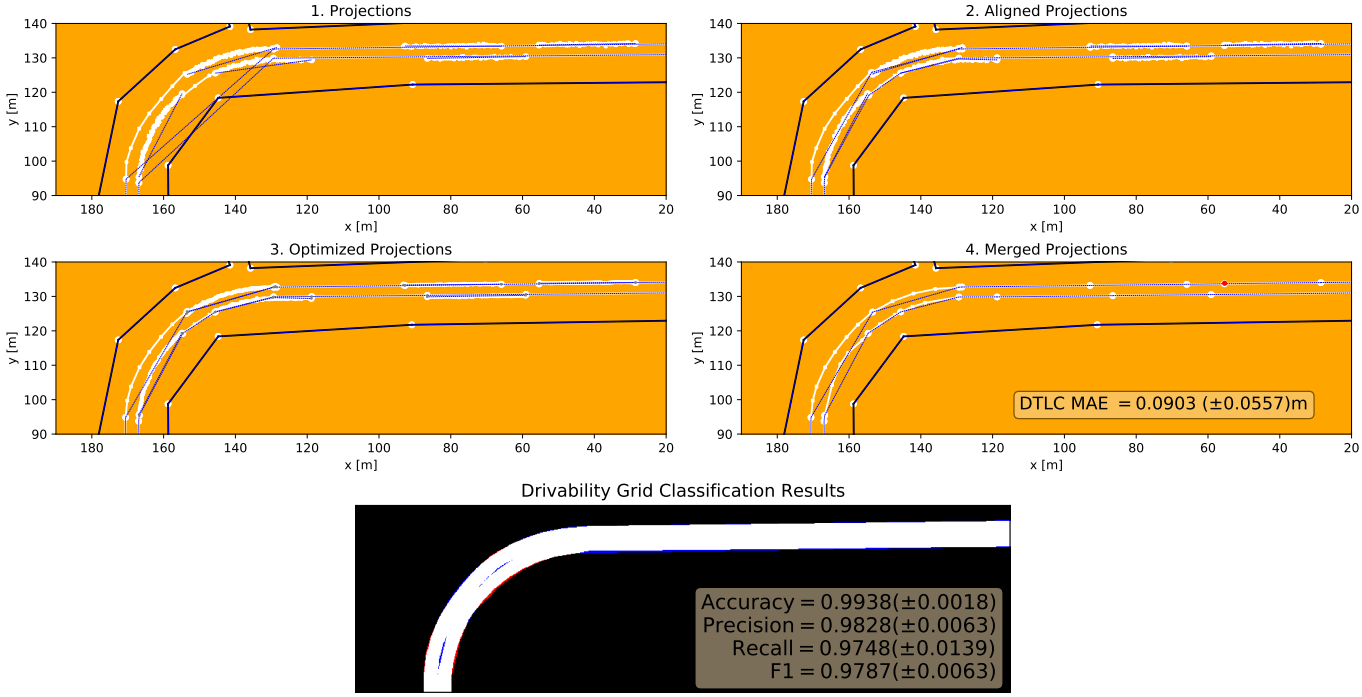
Scenario 3. Noise level 0.9. Missing input: None



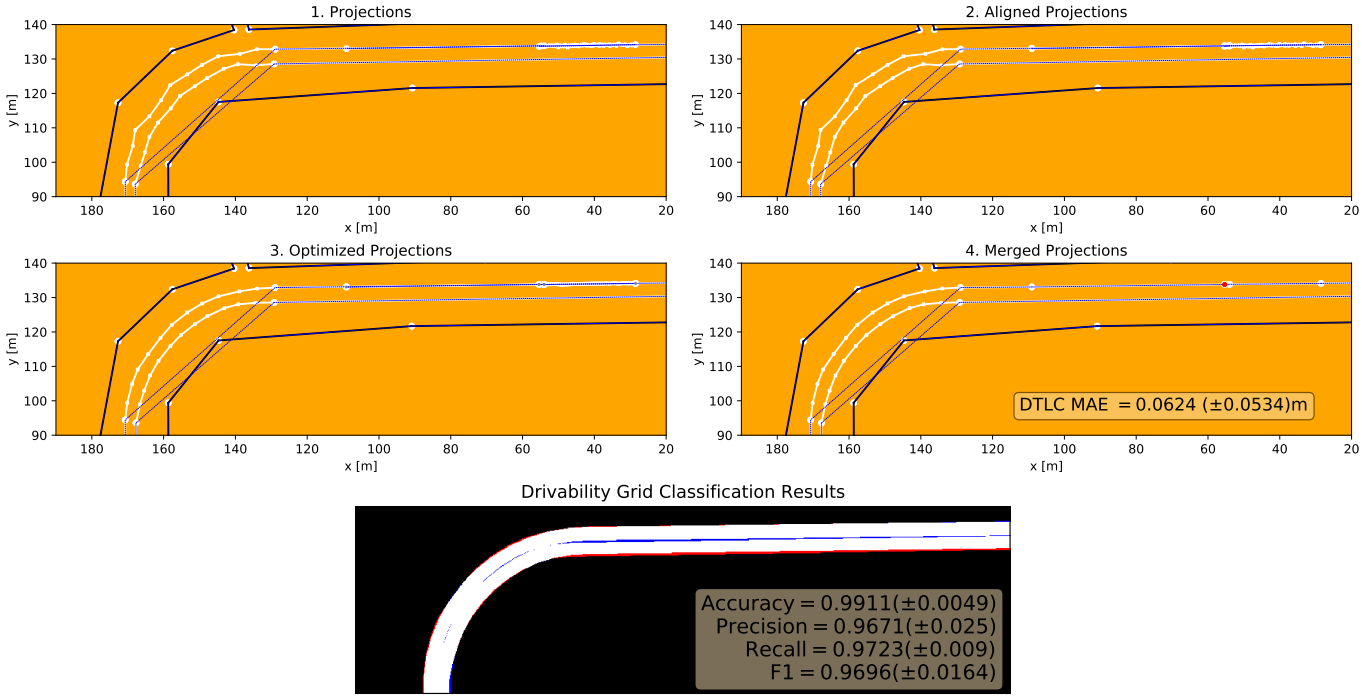
Scenario 3. Noise level 0.9. Missing input: Buildings



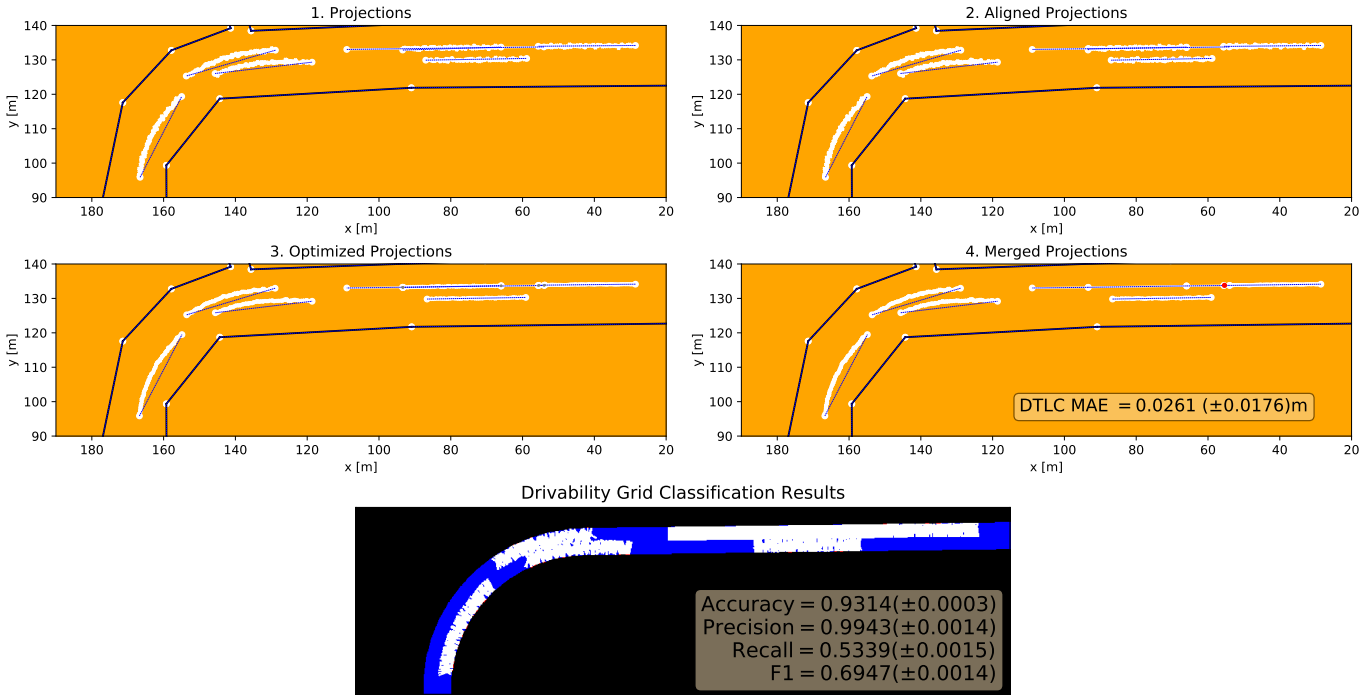
Scenario 3. Noise level 0.9. Missing input: Mobileye



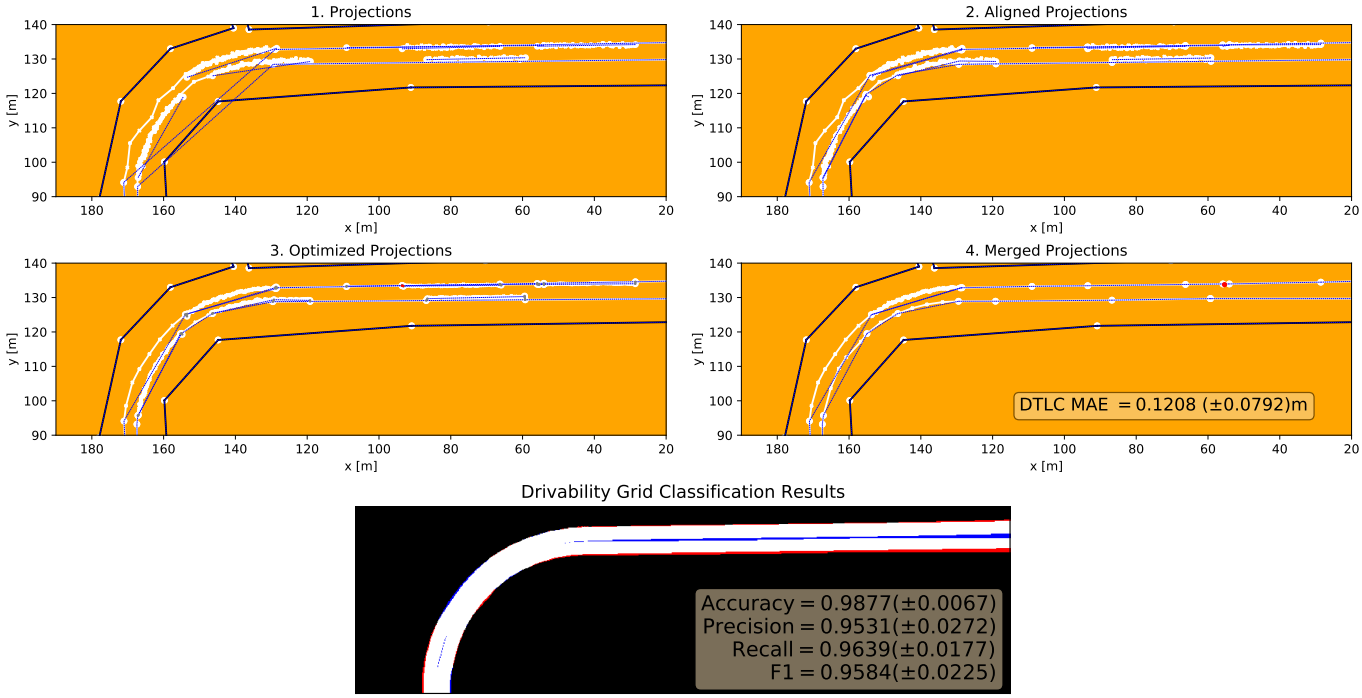
Scenario 3. Noise level 0.9. Missing input: Vehicles



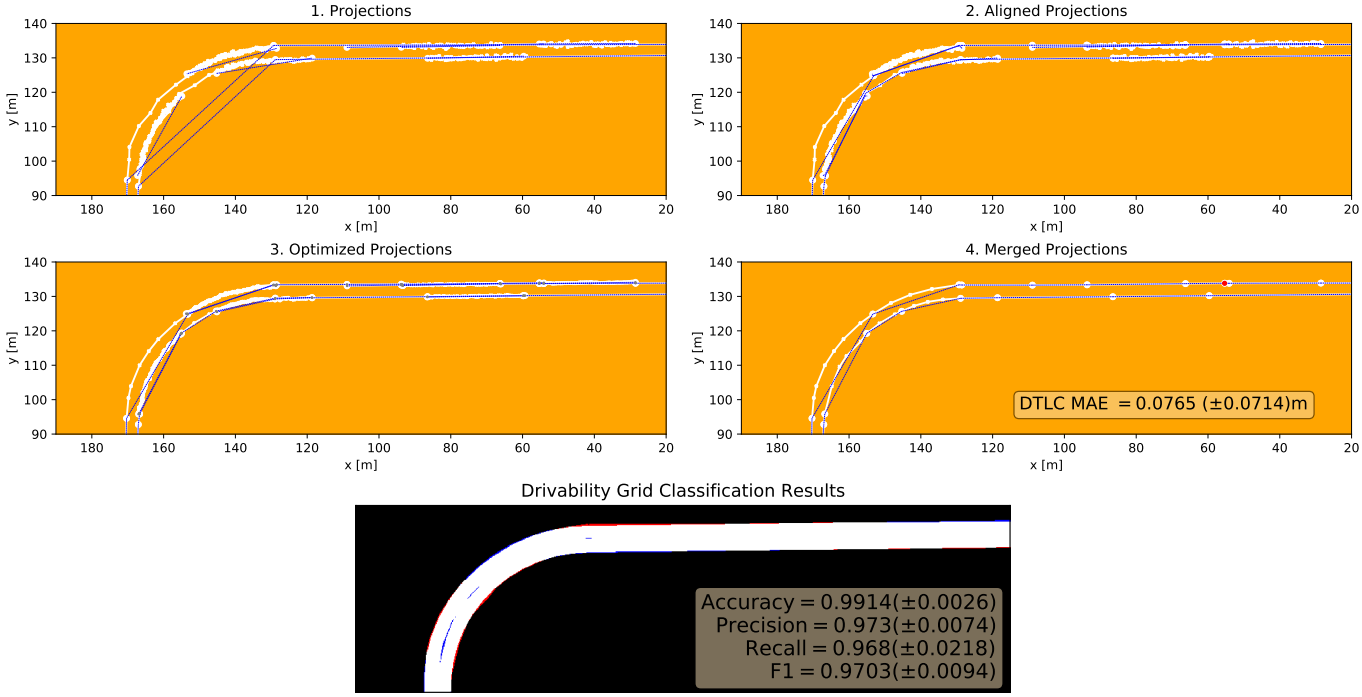
Scenario 3. Noise level 0.9. Missing input: Road network



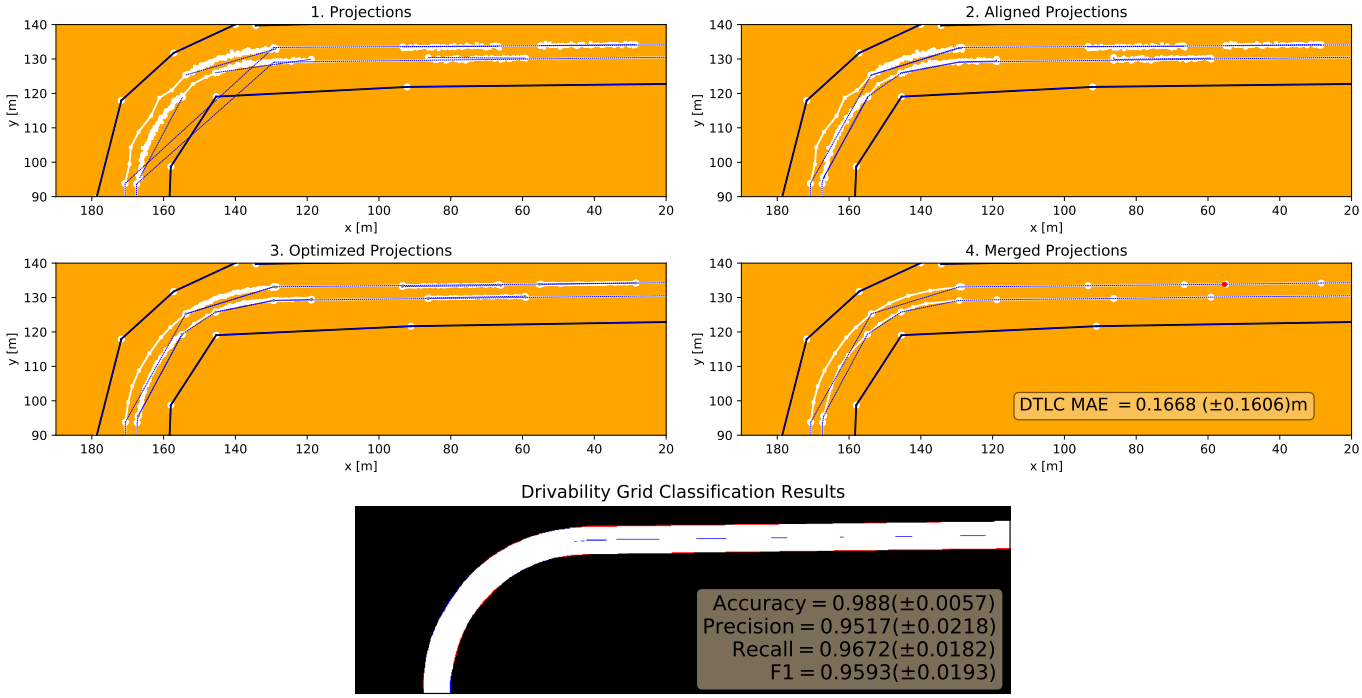
Scenario 3. Noise level 1.8. Missing input: None



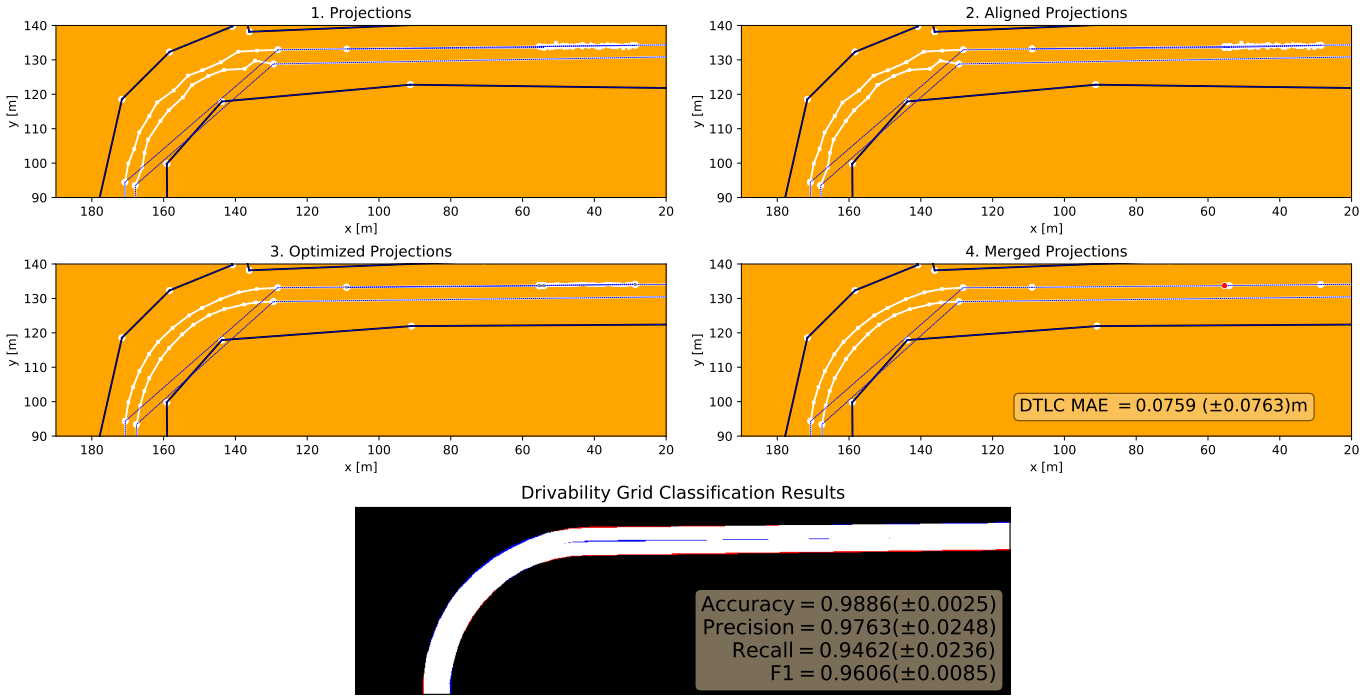
Scenario 3. Noise level 1.8. Missing input: Buildings



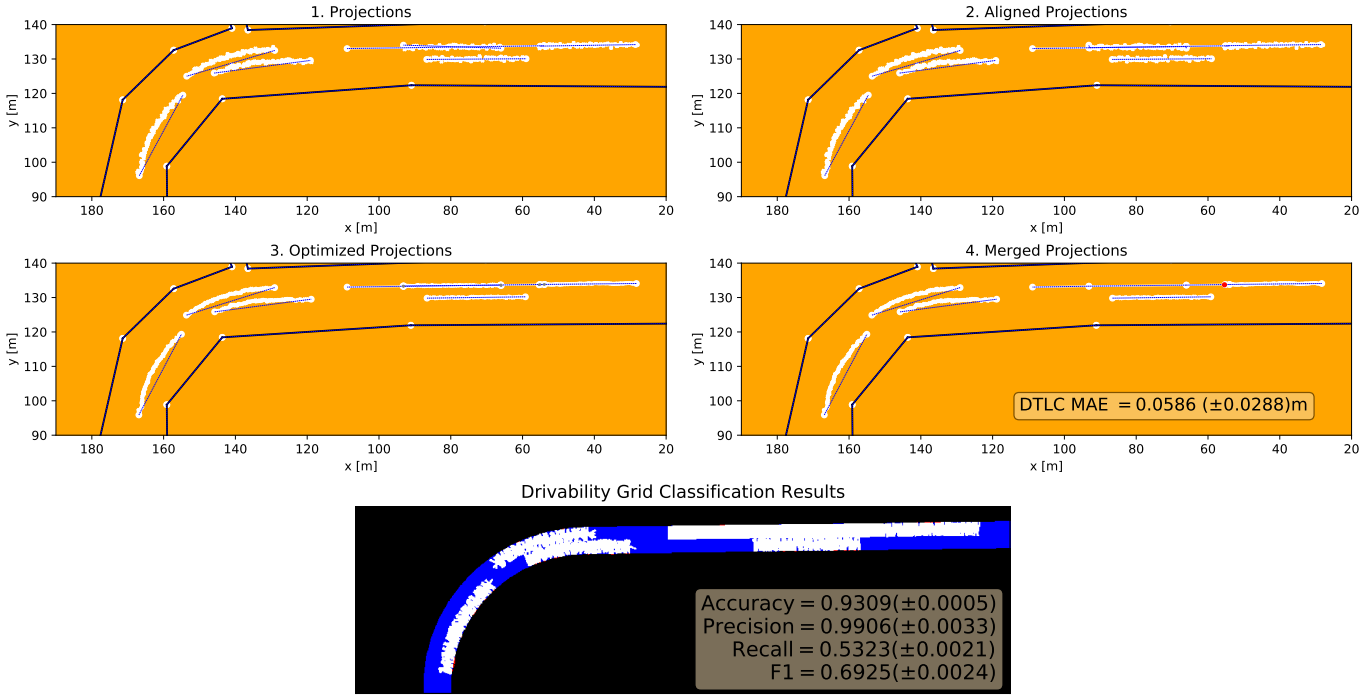
Scenario 3. Noise level 1.8. Missing input: Mobileye



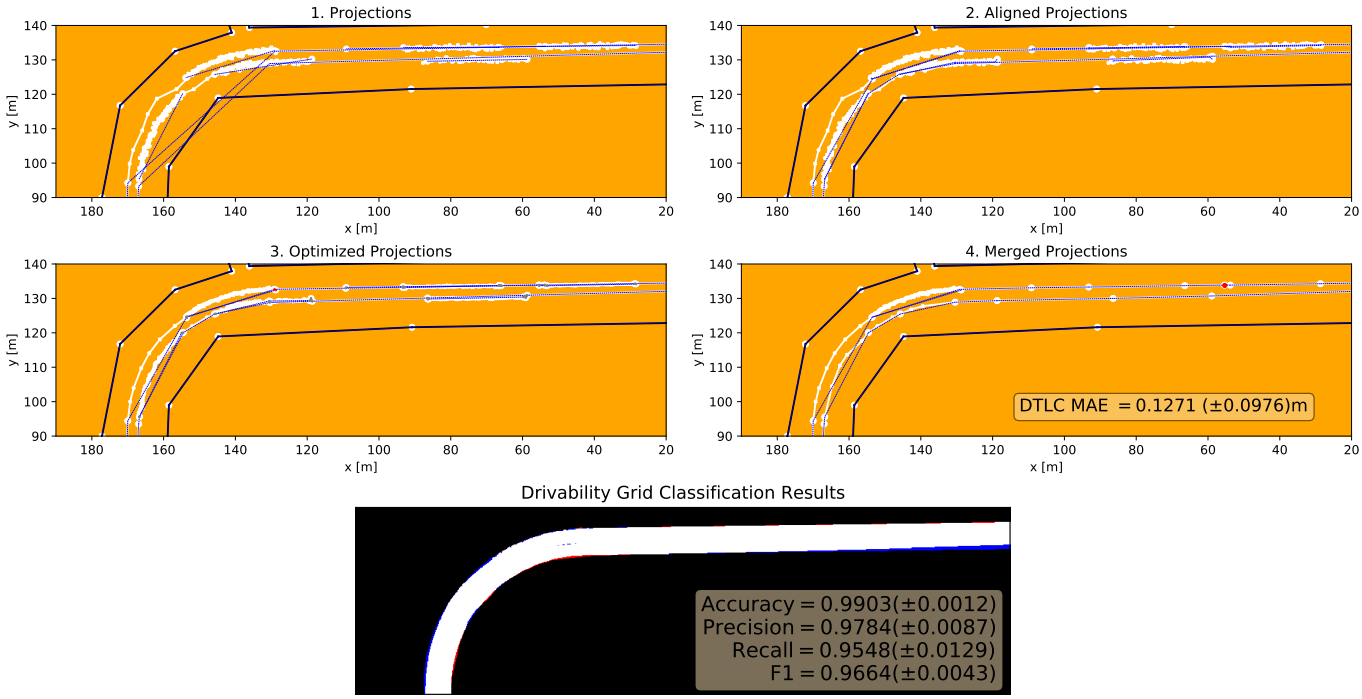
Scenario 3. Noise level 1.8. Missing input: Vehicles



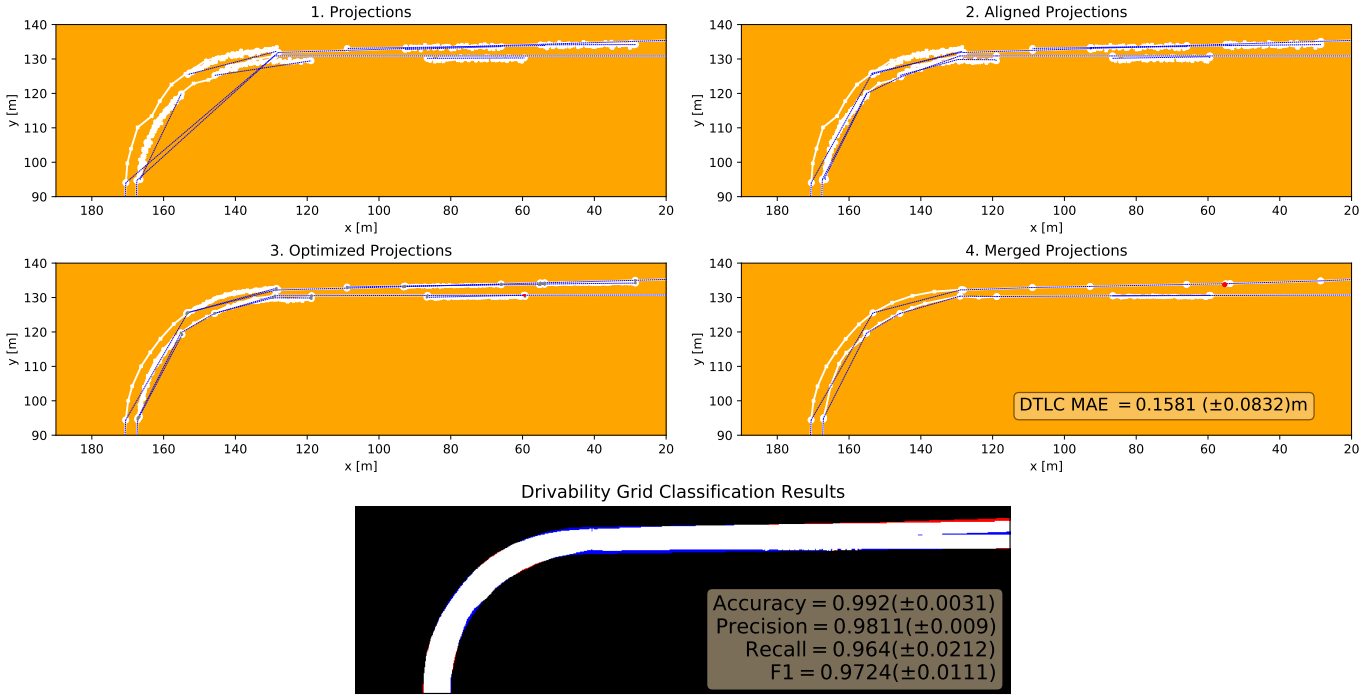
Scenario 3. Noise level 1.8. Missing input: Road network



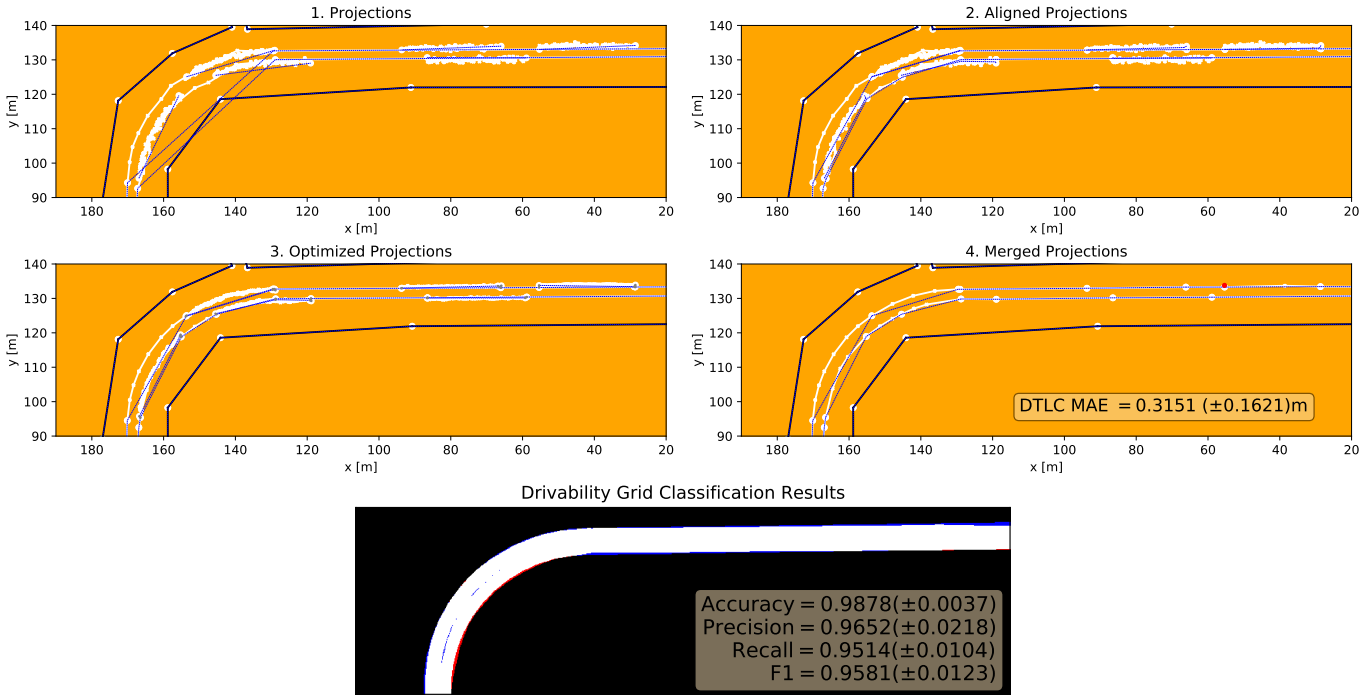
Scenario 3. Noise level 2.7. Missing input: None



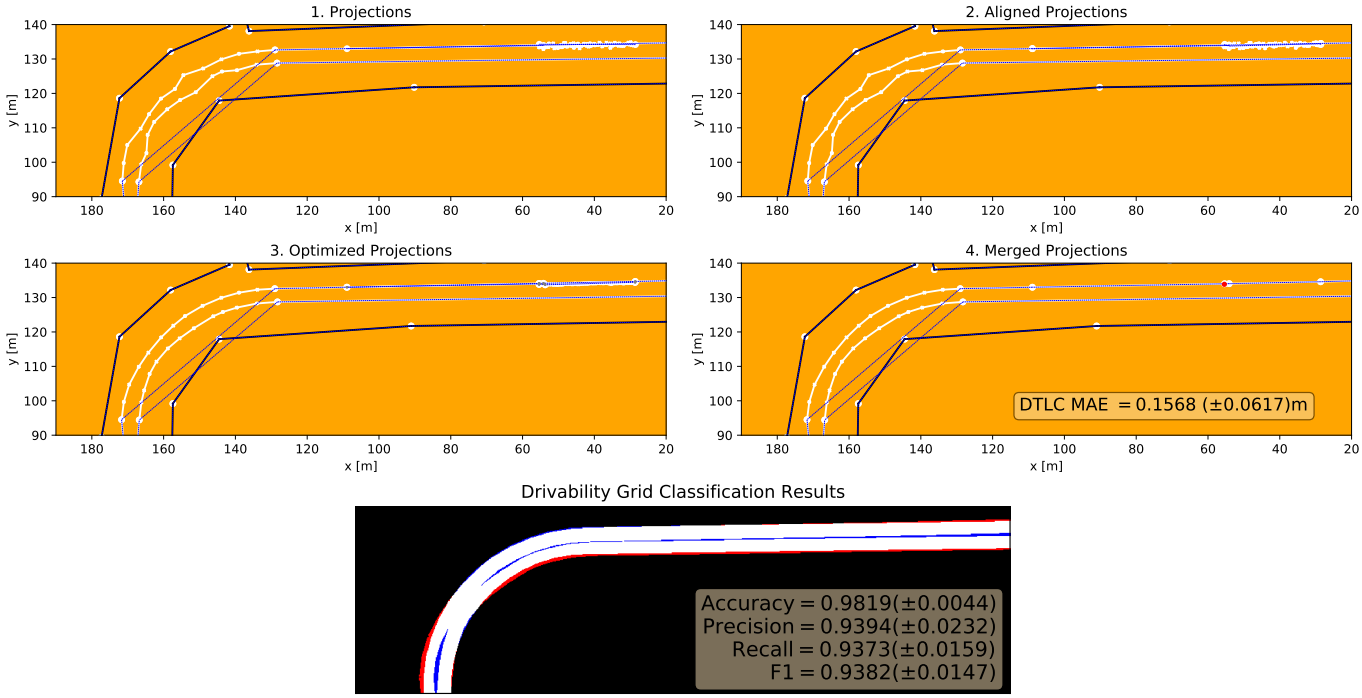
Scenario 3. Noise level 2.7. Missing input: Buildings



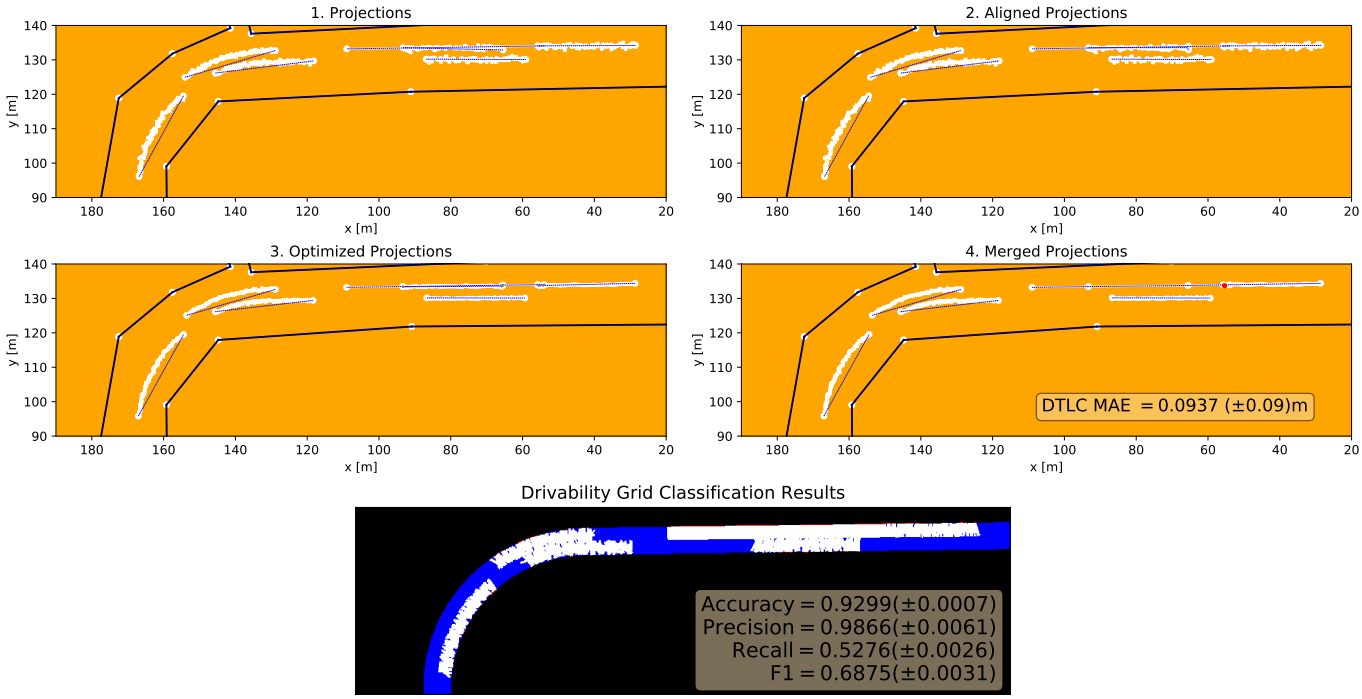
Scenario 3. Noise level 2.7. Missing input: Mobileye



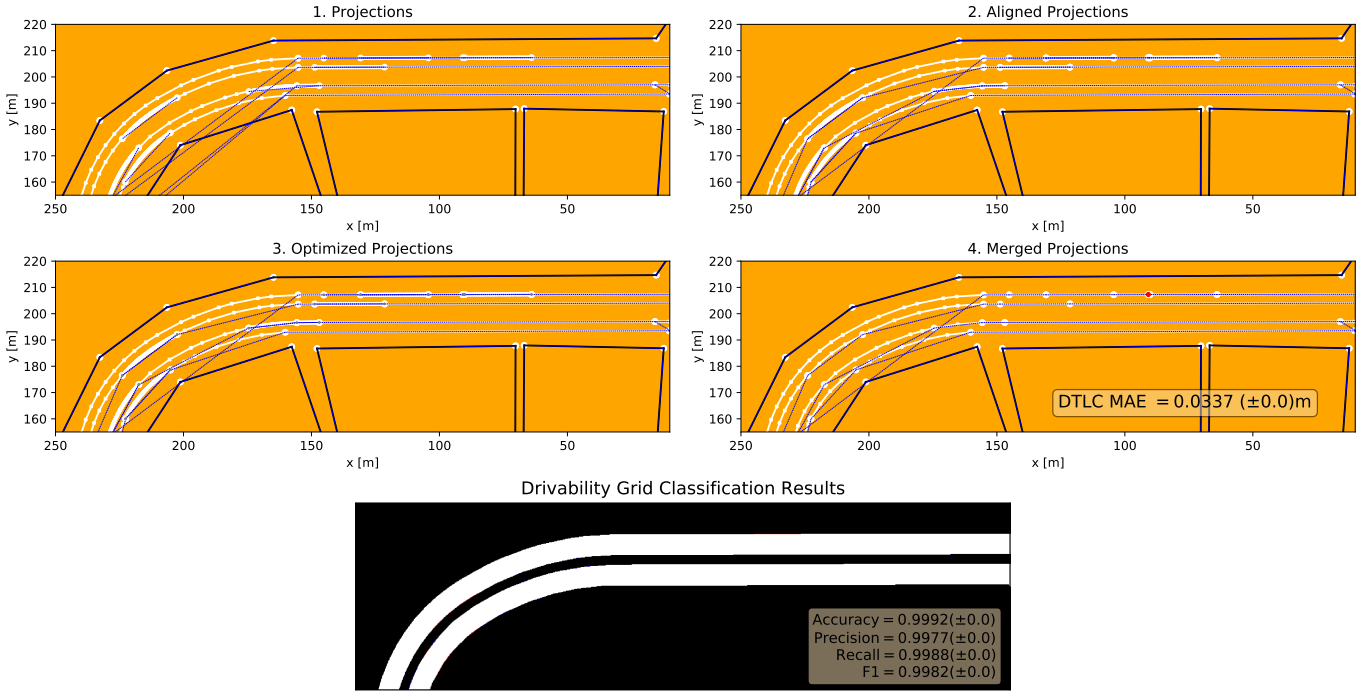
Scenario 3. Noise level 2.7. Missing input: Vehicles



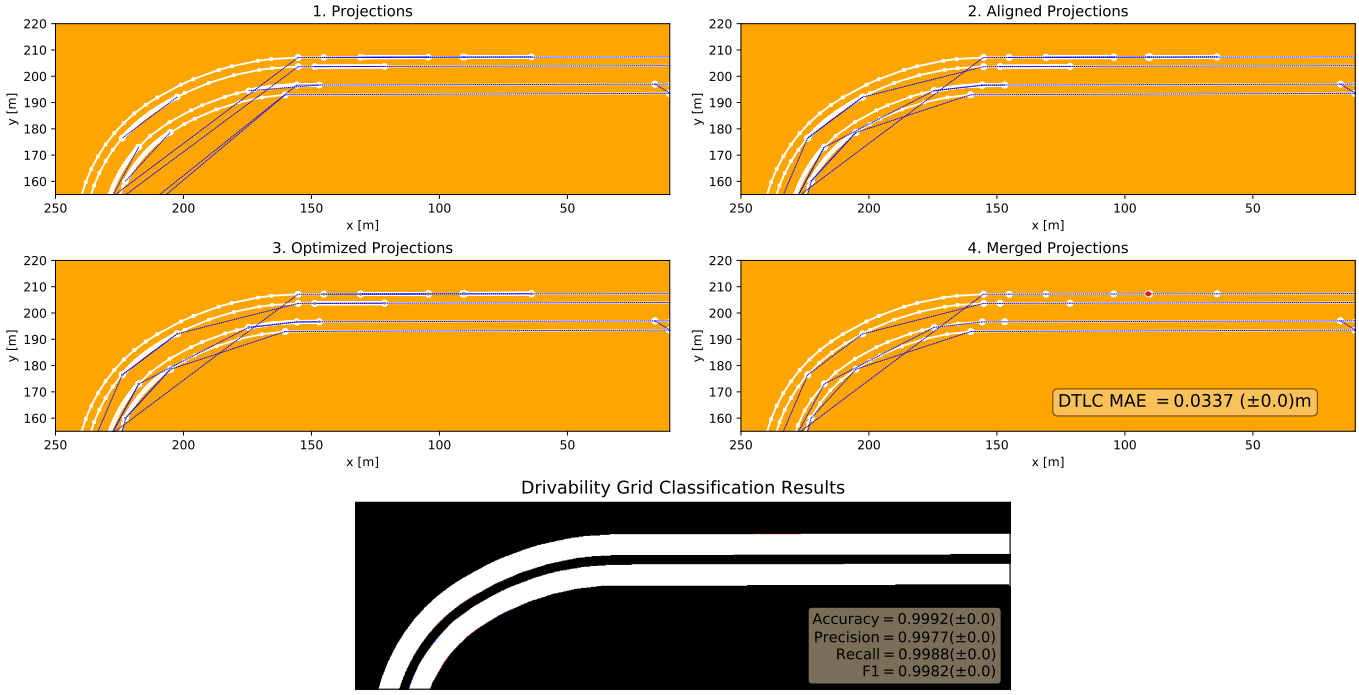
Scenario 3. Noise level 2.7. Missing input: Road network



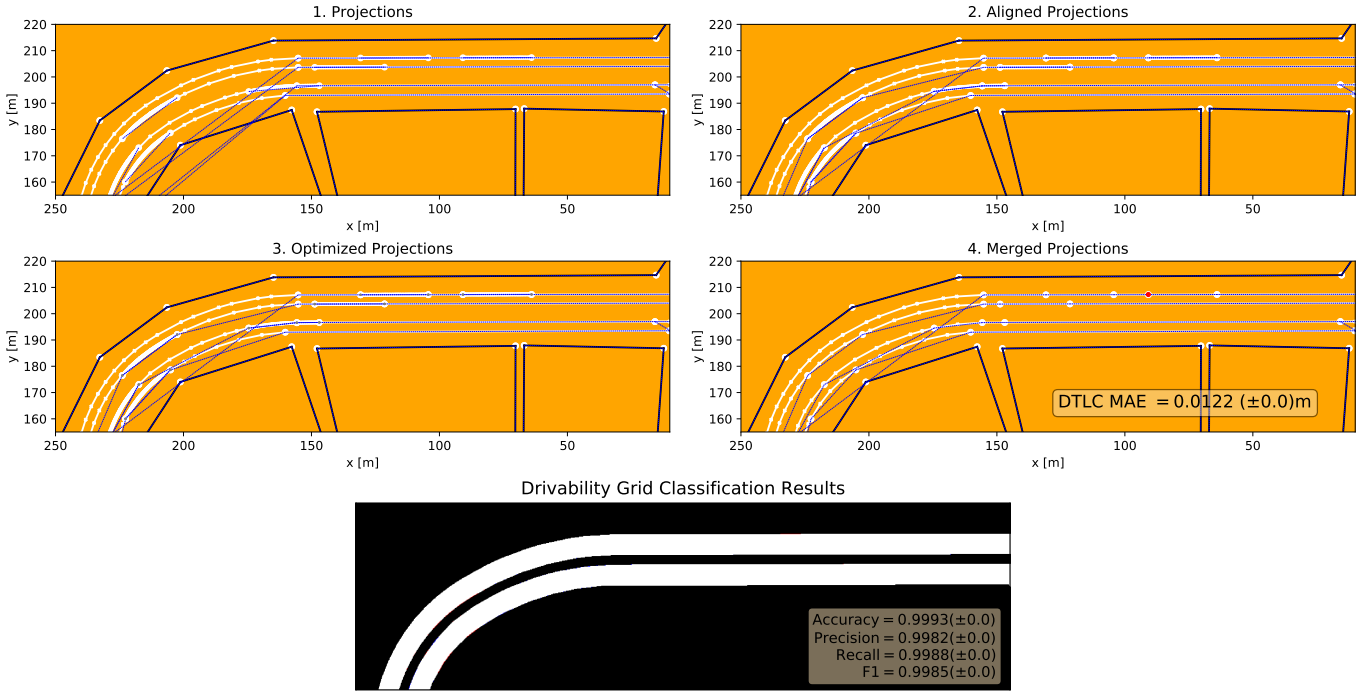
Scenario 4. Noise level 0.0. Missing input: None



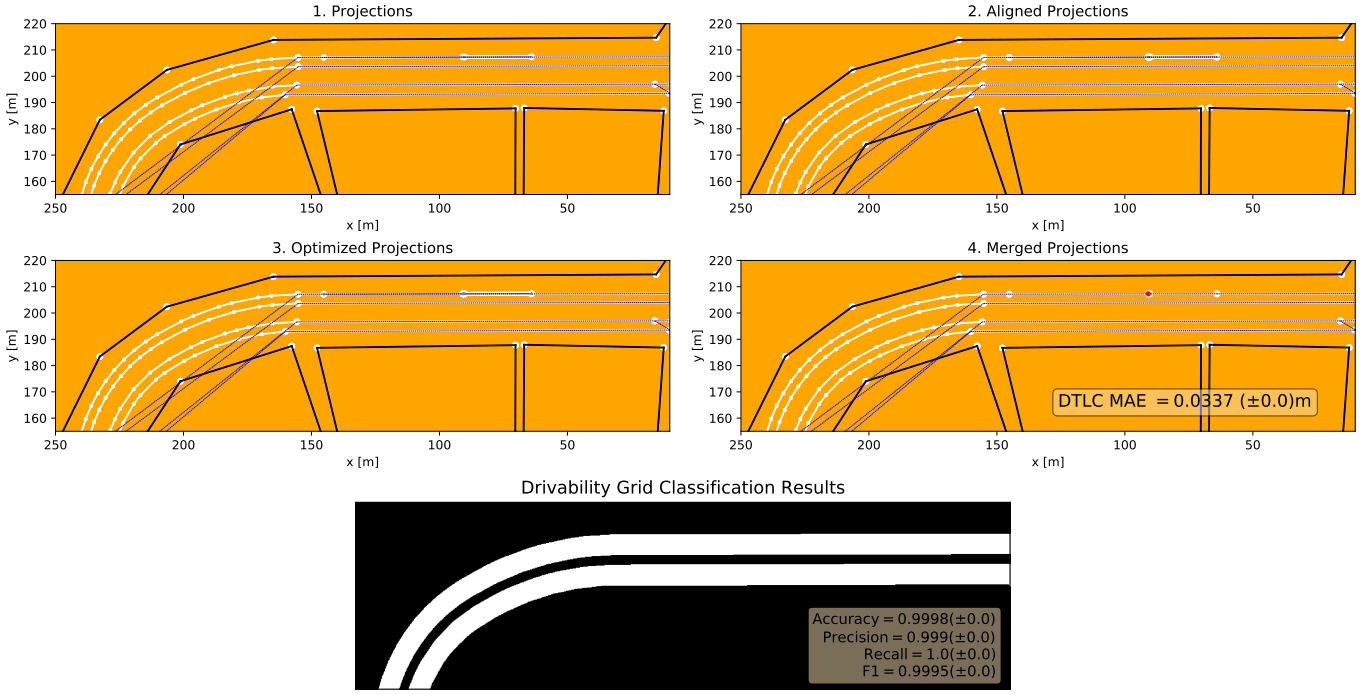
Scenario 4. Noise level 0.0. Missing input: Buildings



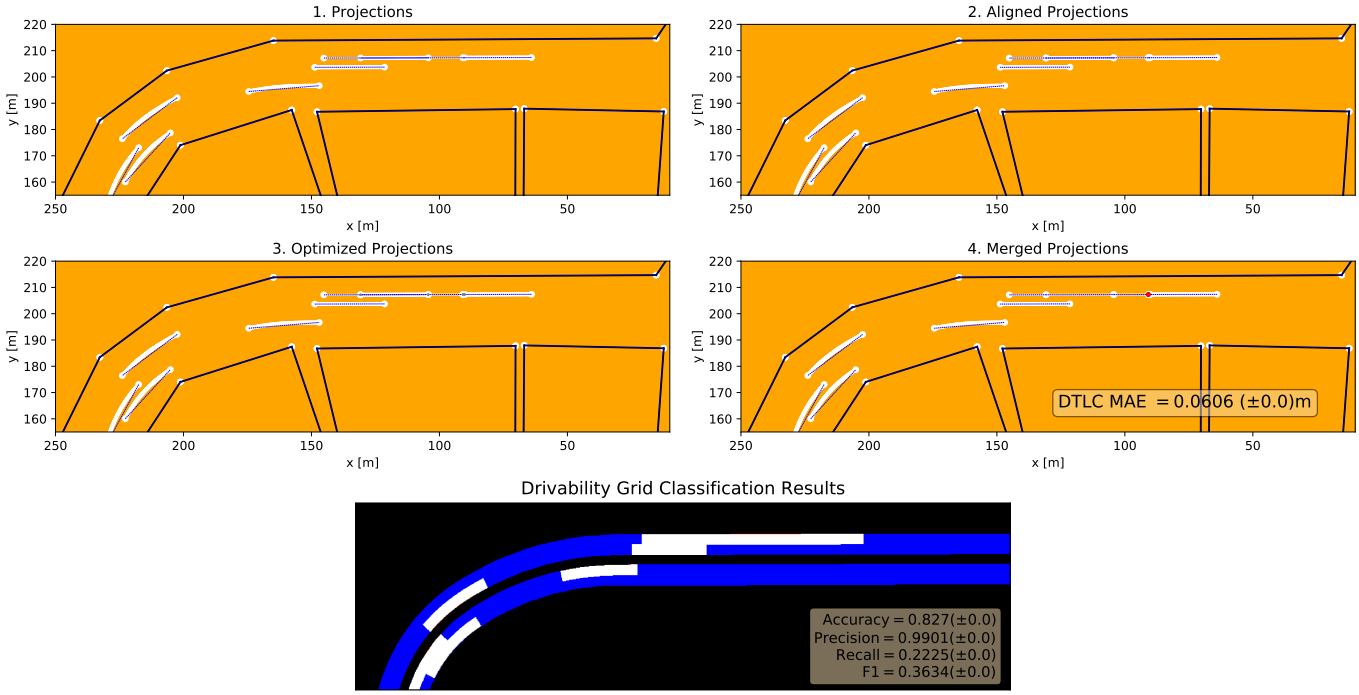
Scenario 4. Noise level 0.0. Missing input: Mobileye



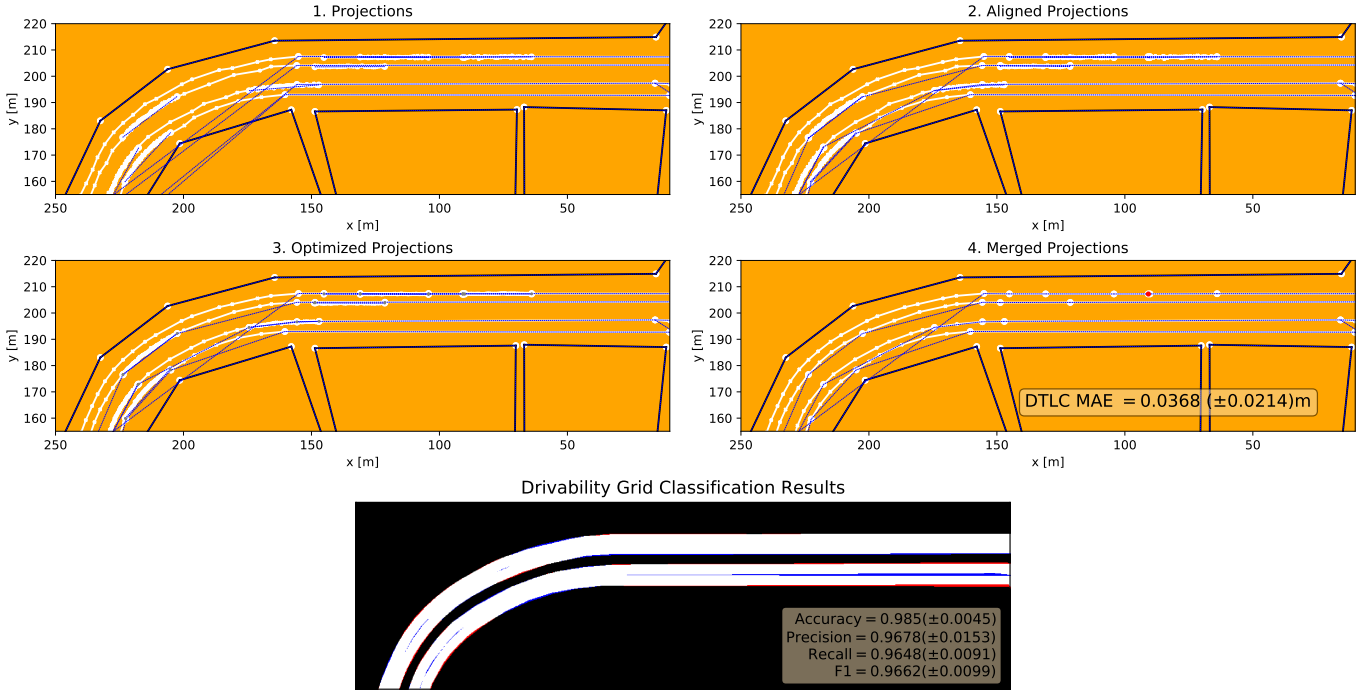
Scenario 4. Noise level 0.0. Missing input: Vehicles



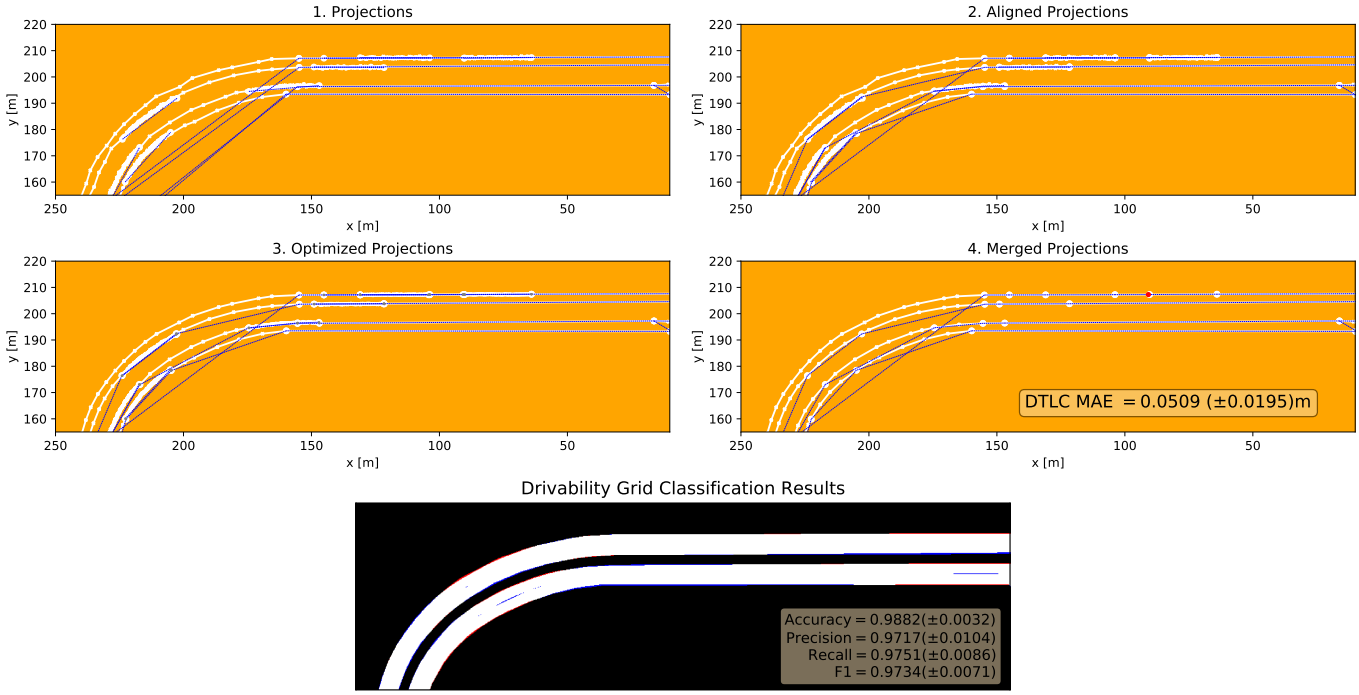
Scenario 4. Noise level 0.0. Missing input: Road network



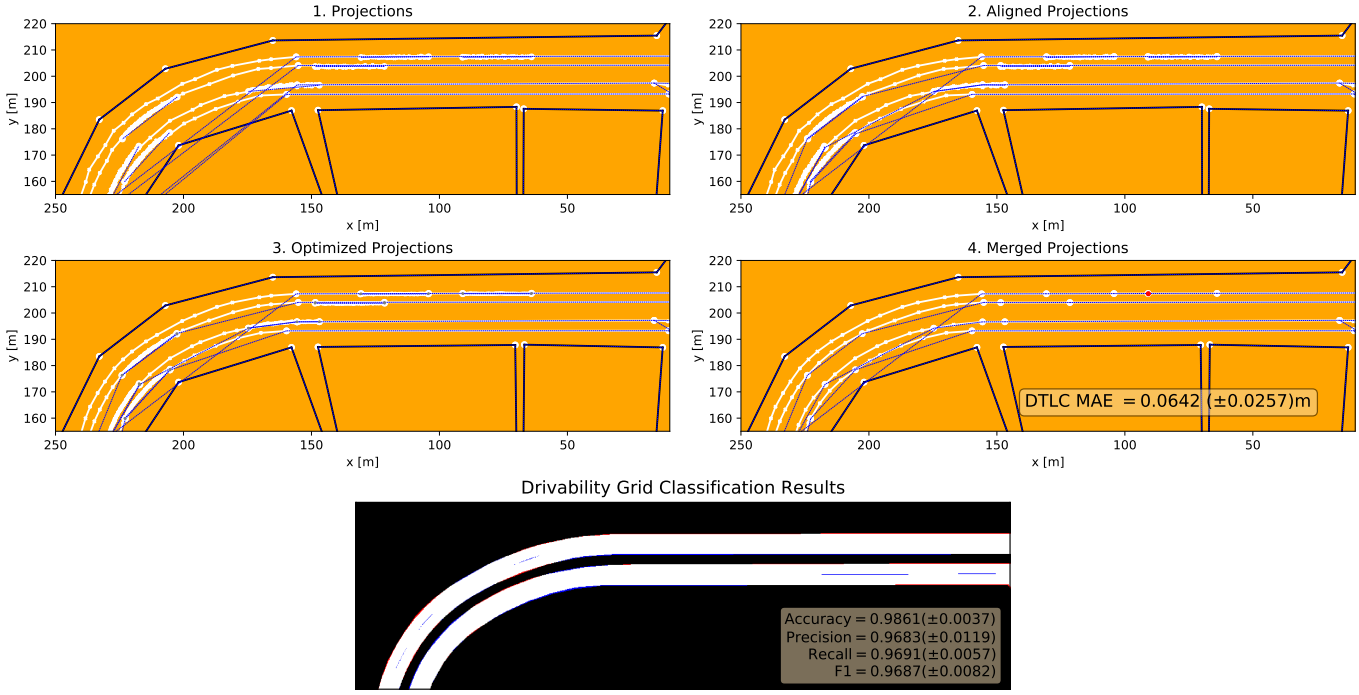
Scenario 4. Noise level 0.9. Missing input: None



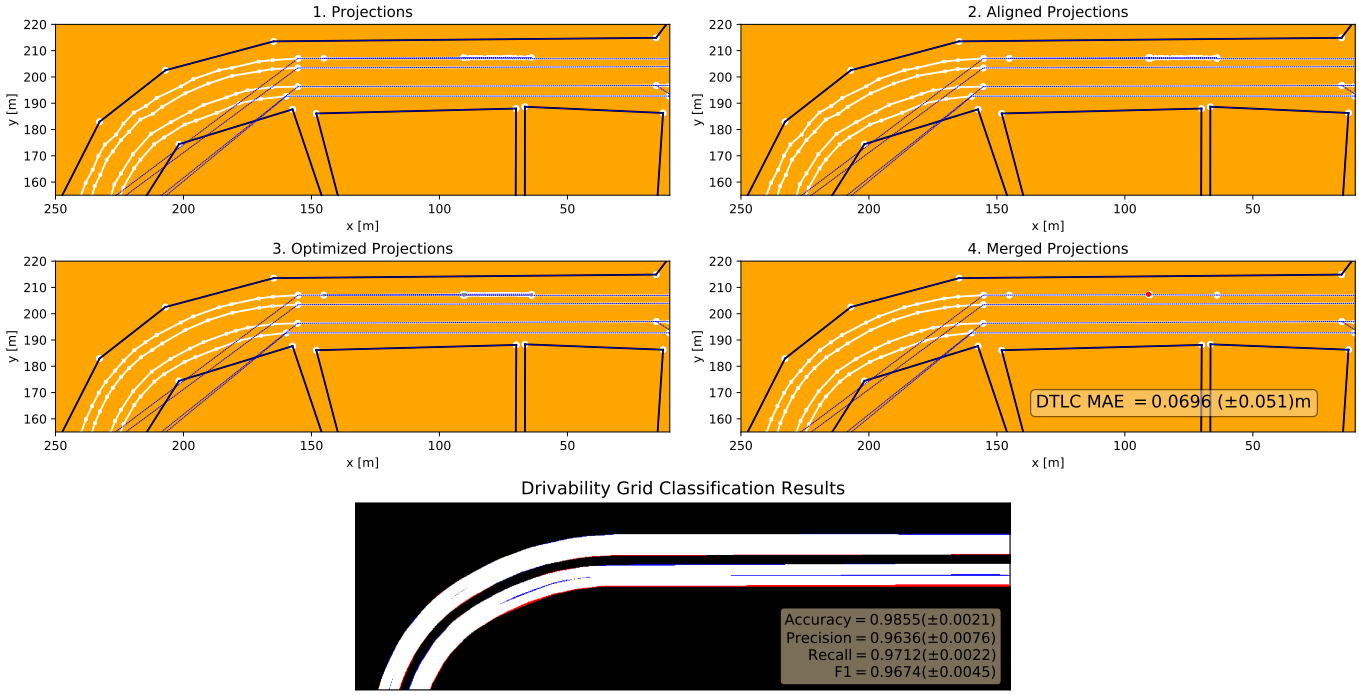
Scenario 4. Noise level 0.9. Missing input: Buildings



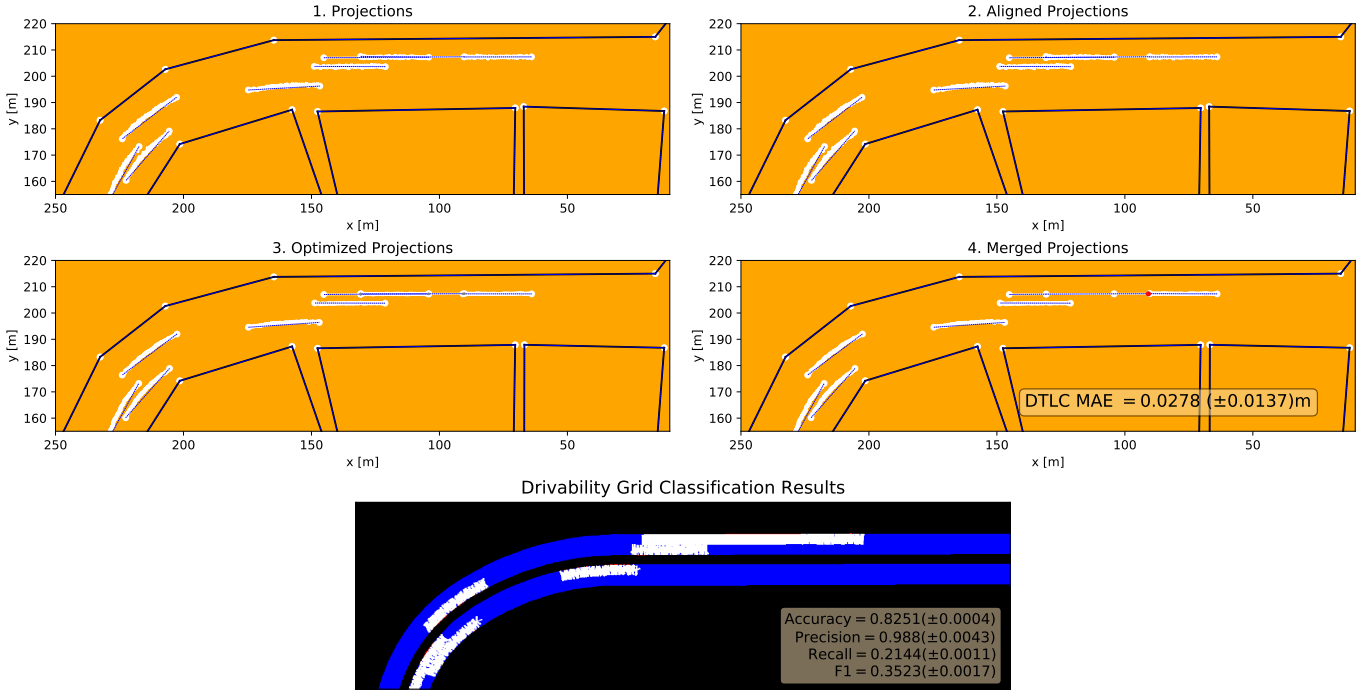
Scenario 4. Noise level 0.9. Missing input: Mobileye



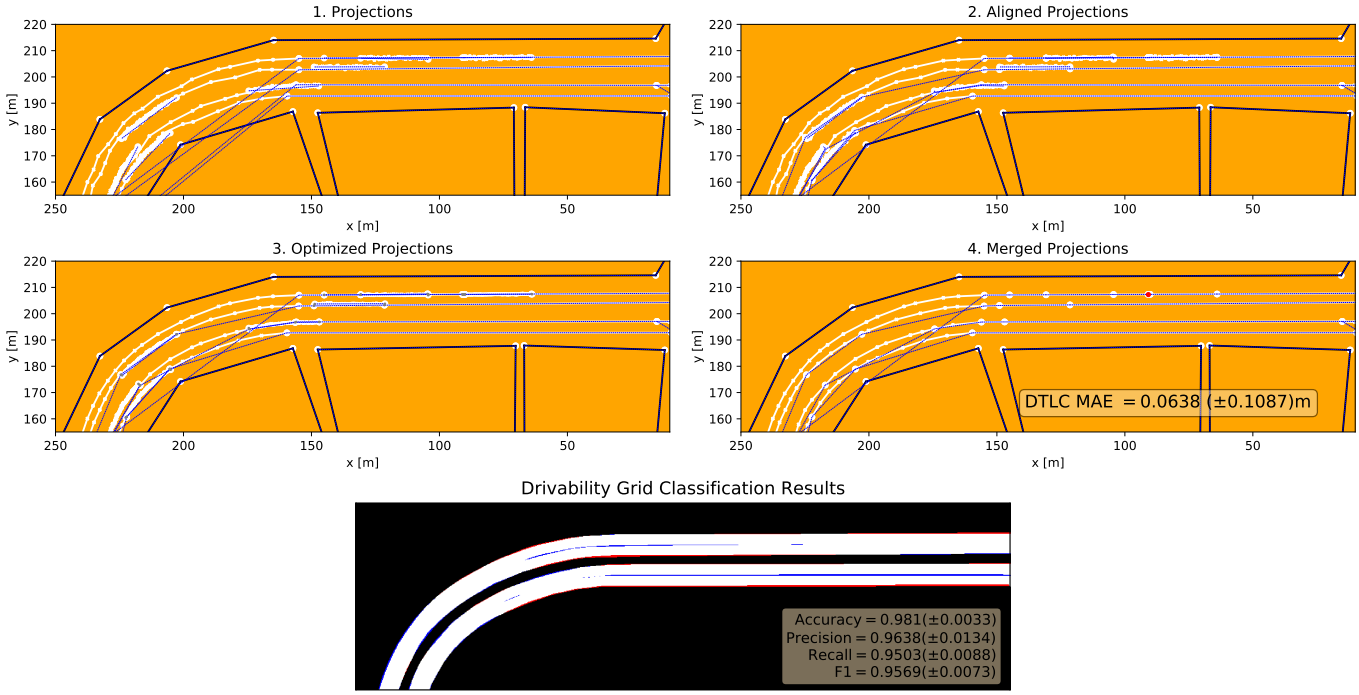
Scenario 4. Noise level 0.9. Missing input: Vehicles



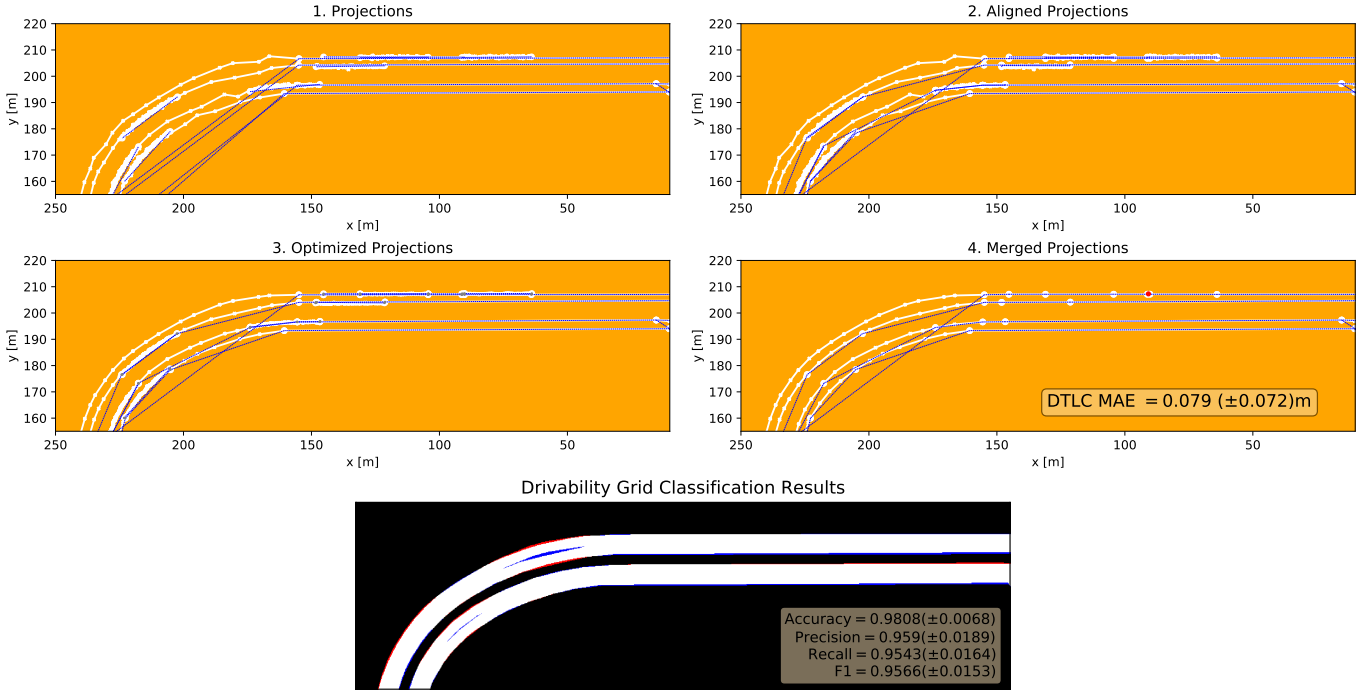
Scenario 4. Noise level 0.9. Missing input: Road network



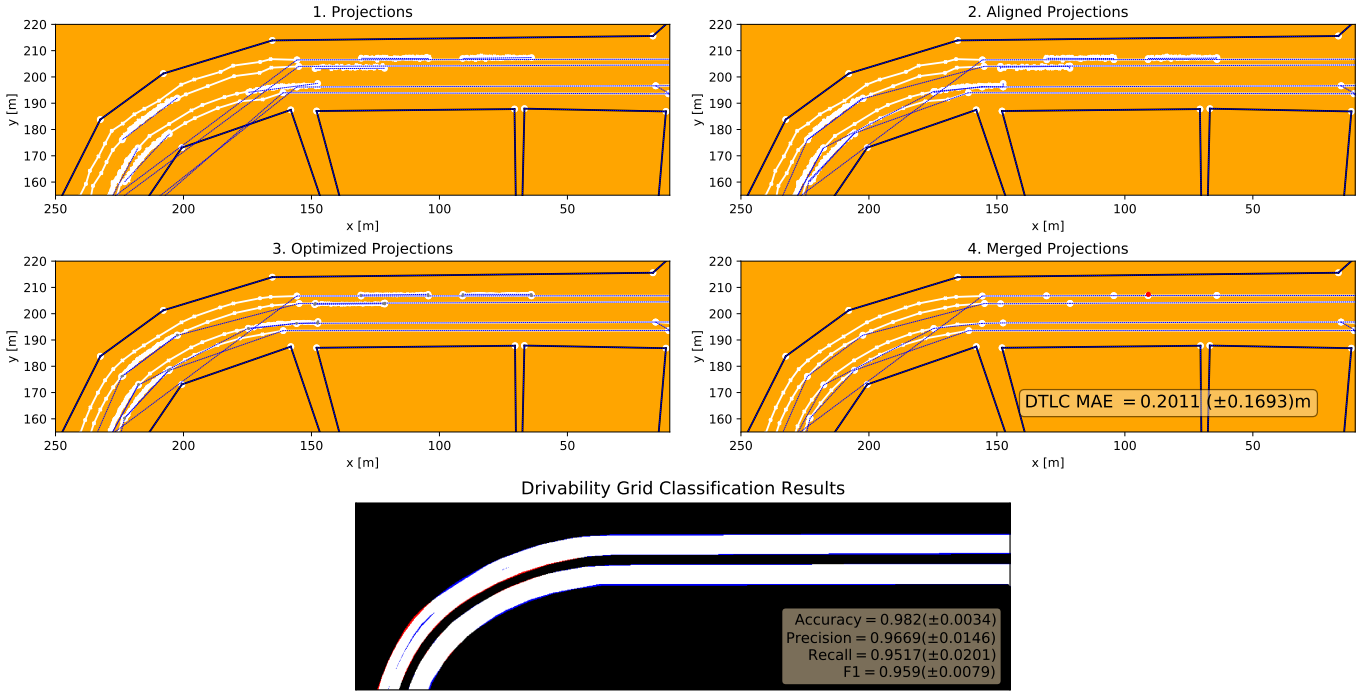
Scenario 4. Noise level 1.8. Missing input: None



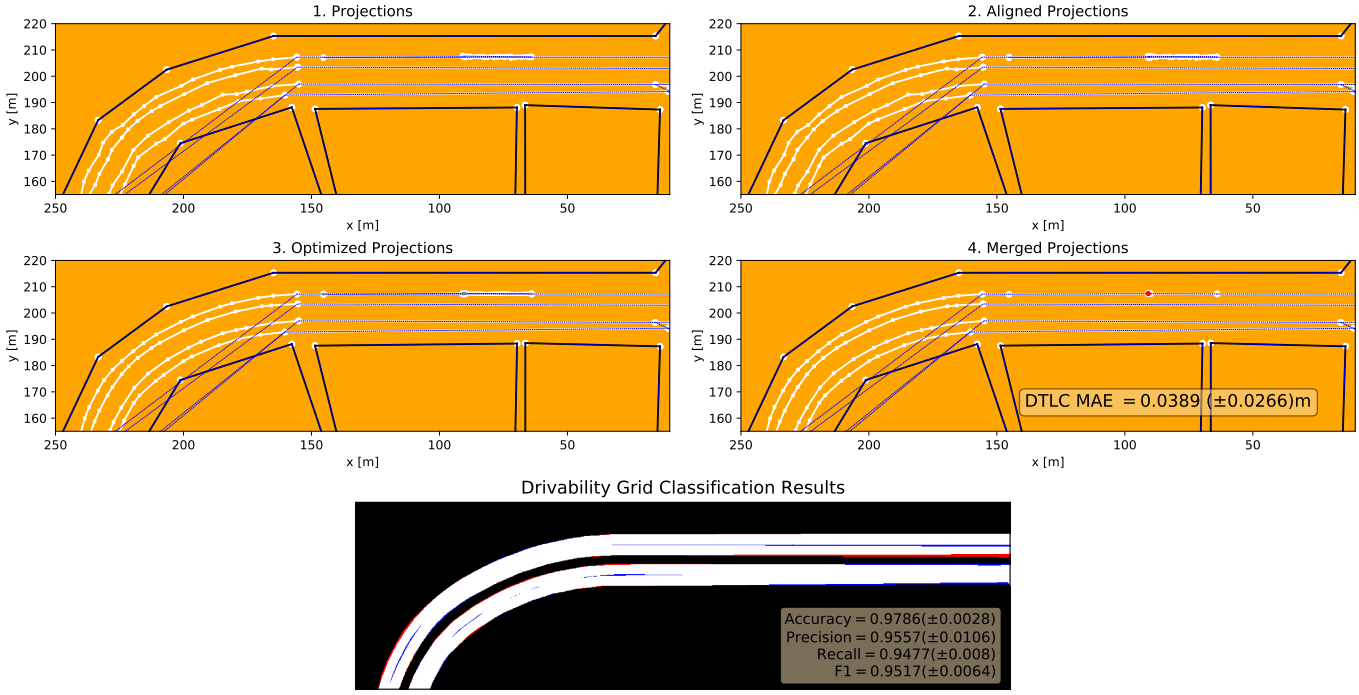
Scenario 4. Noise level 1.8. Missing input: Buildings



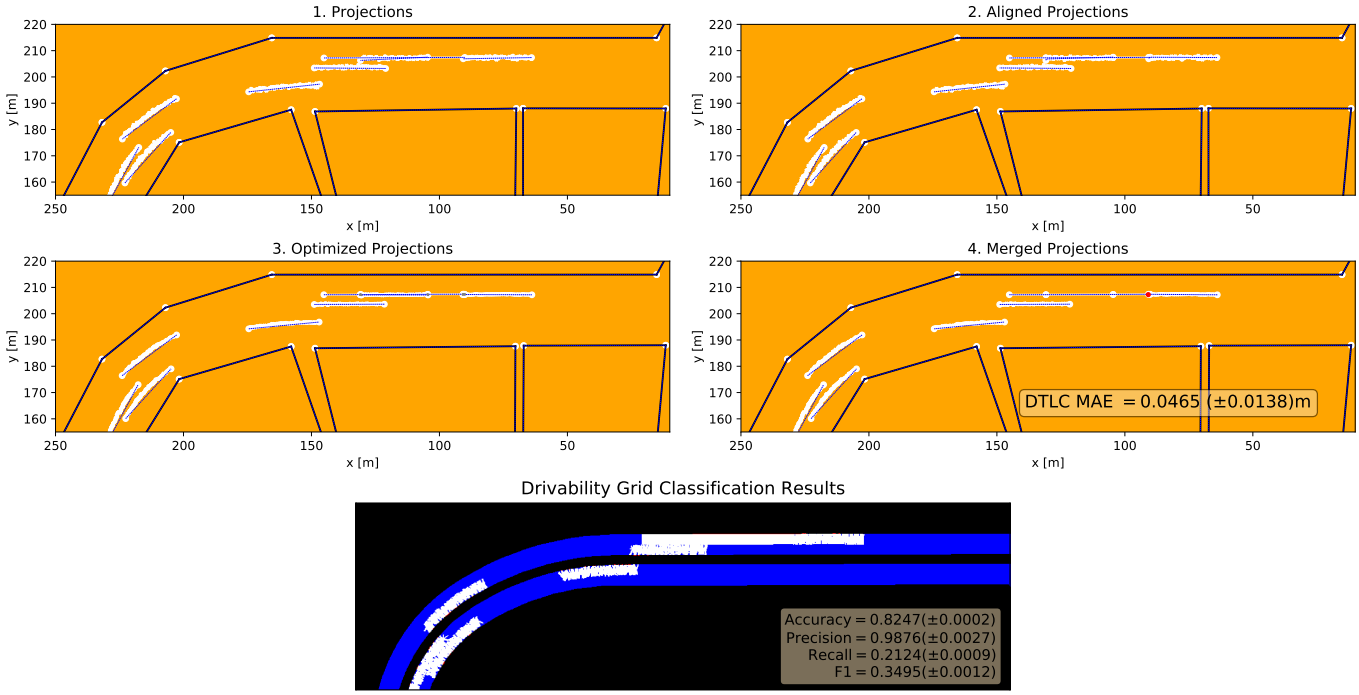
Scenario 4. Noise level 1.8. Missing input: Mobileye



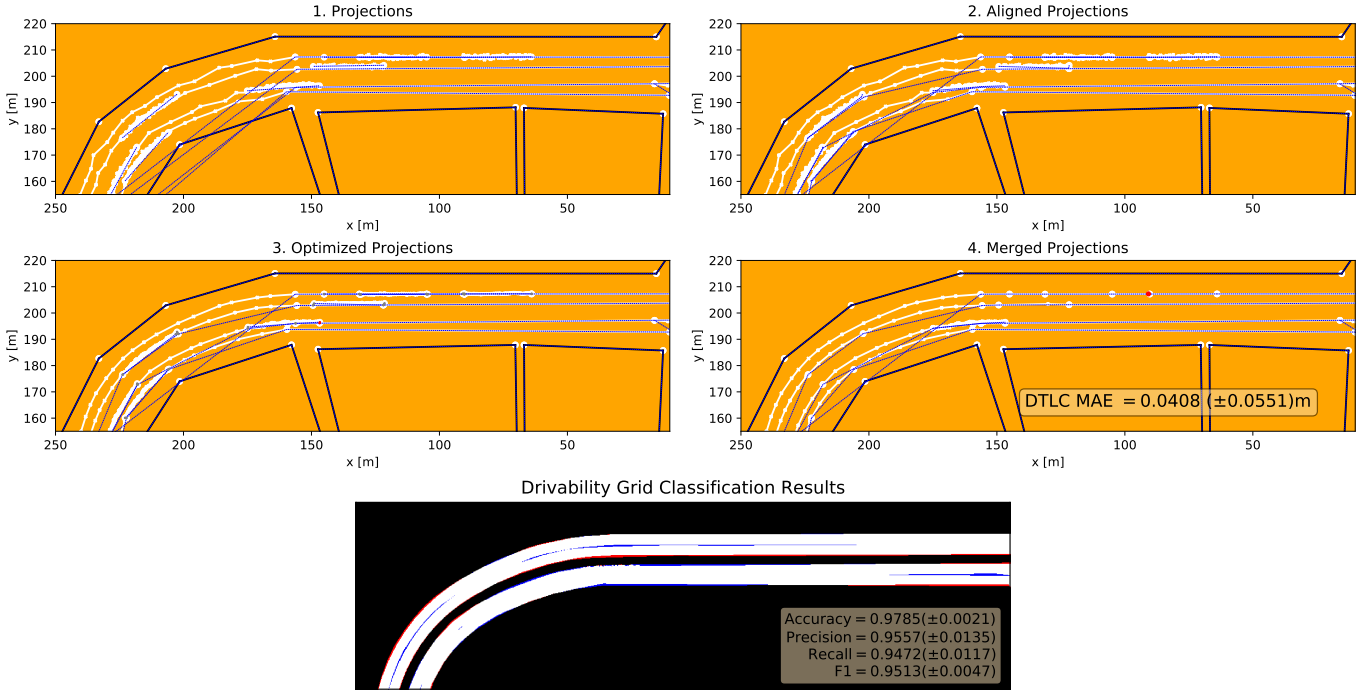
Scenario 4. Noise level 1.8. Missing input: Vehicles



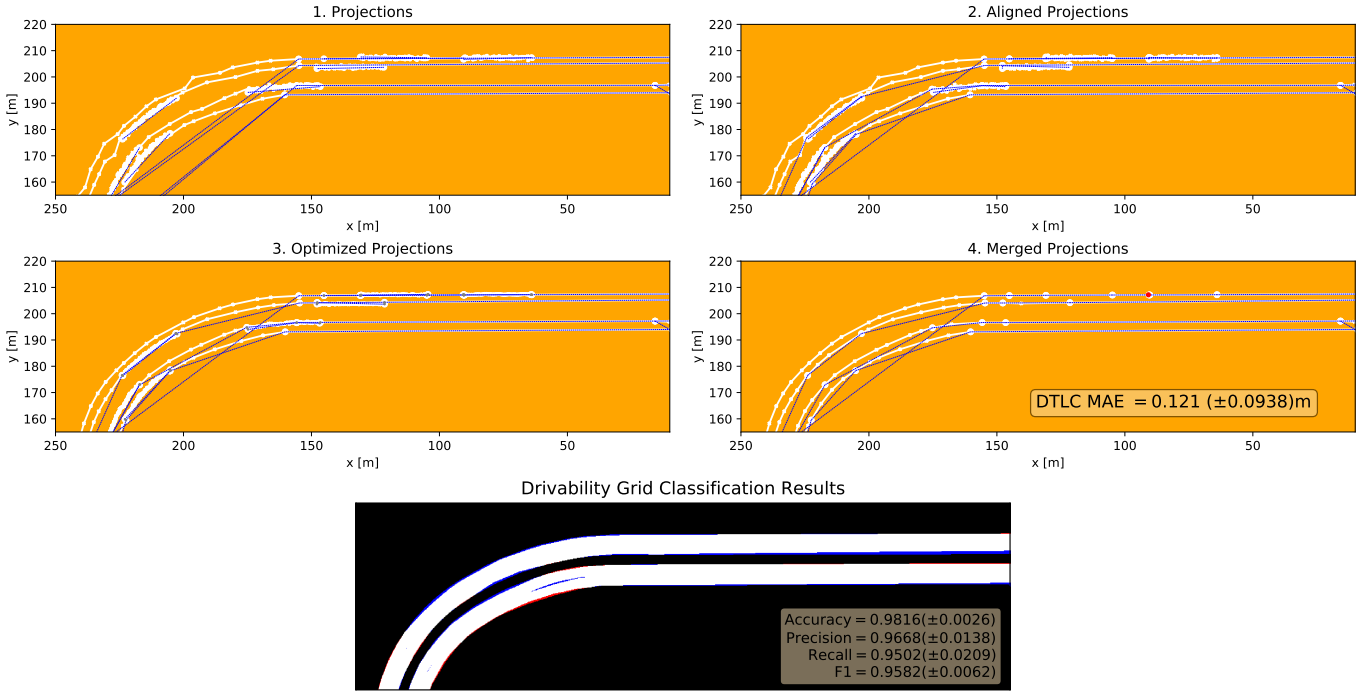
Scenario 4. Noise level 1.8. Missing input: Road network



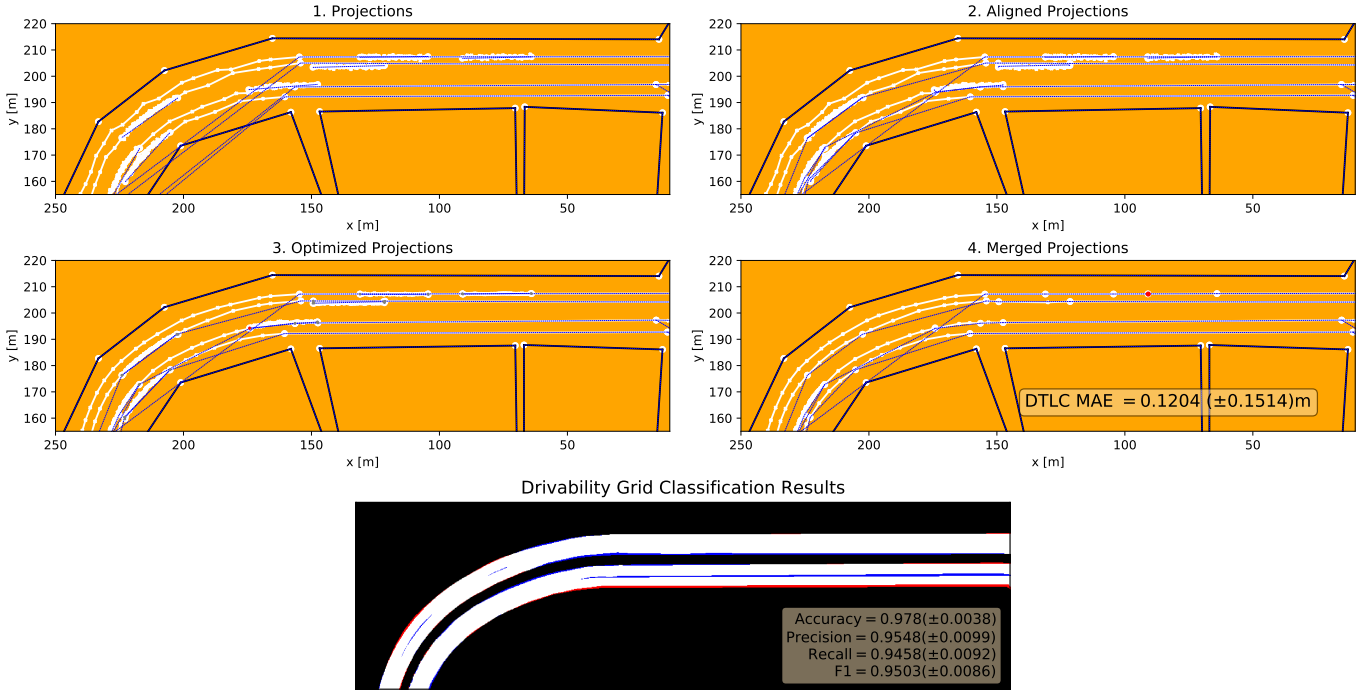
Scenario 4. Noise level 2.7. Missing input: None



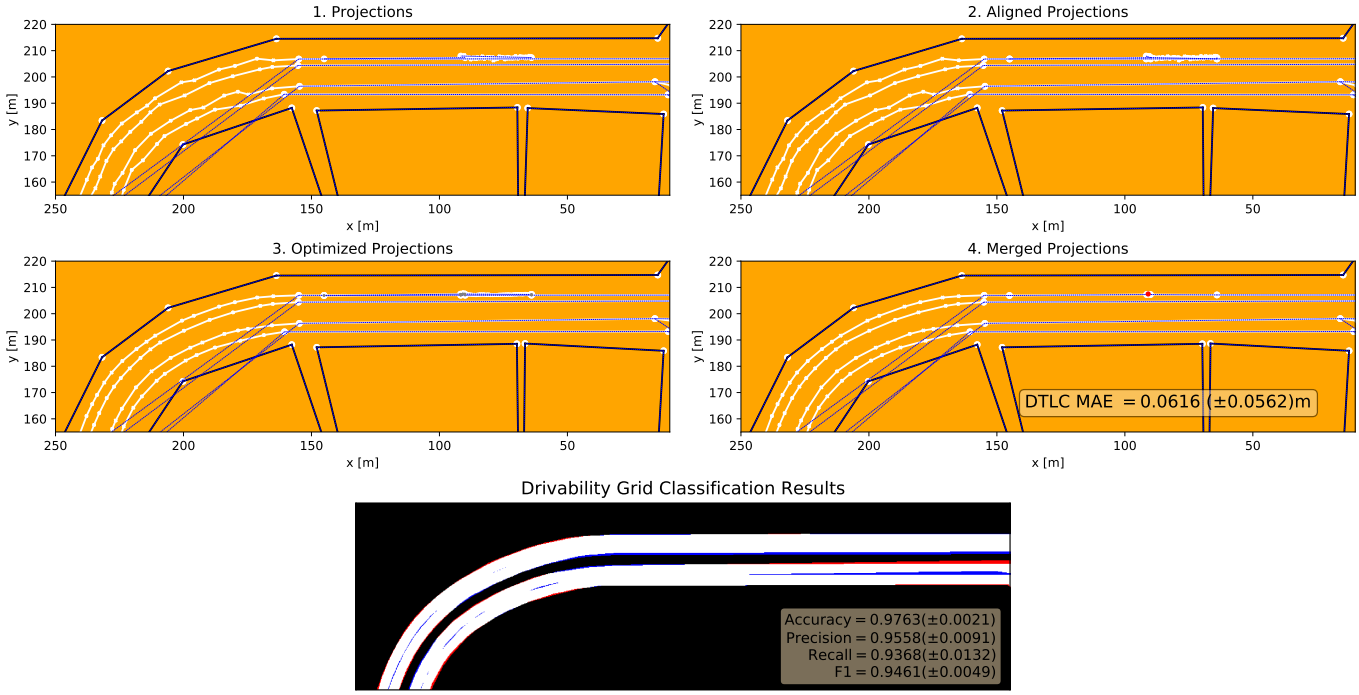
Scenario 4. Noise level 2.7. Missing input: Buildings



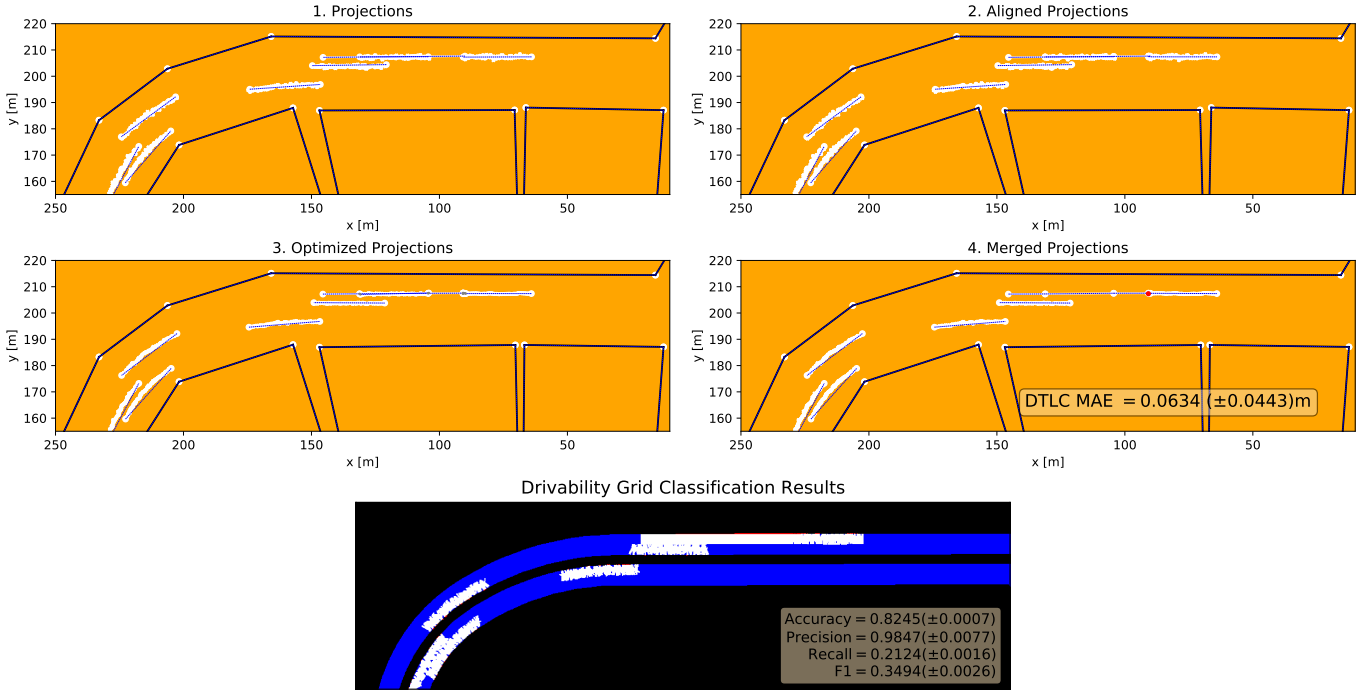
Scenario 4. Noise level 2.7. Missing input: Mobileye



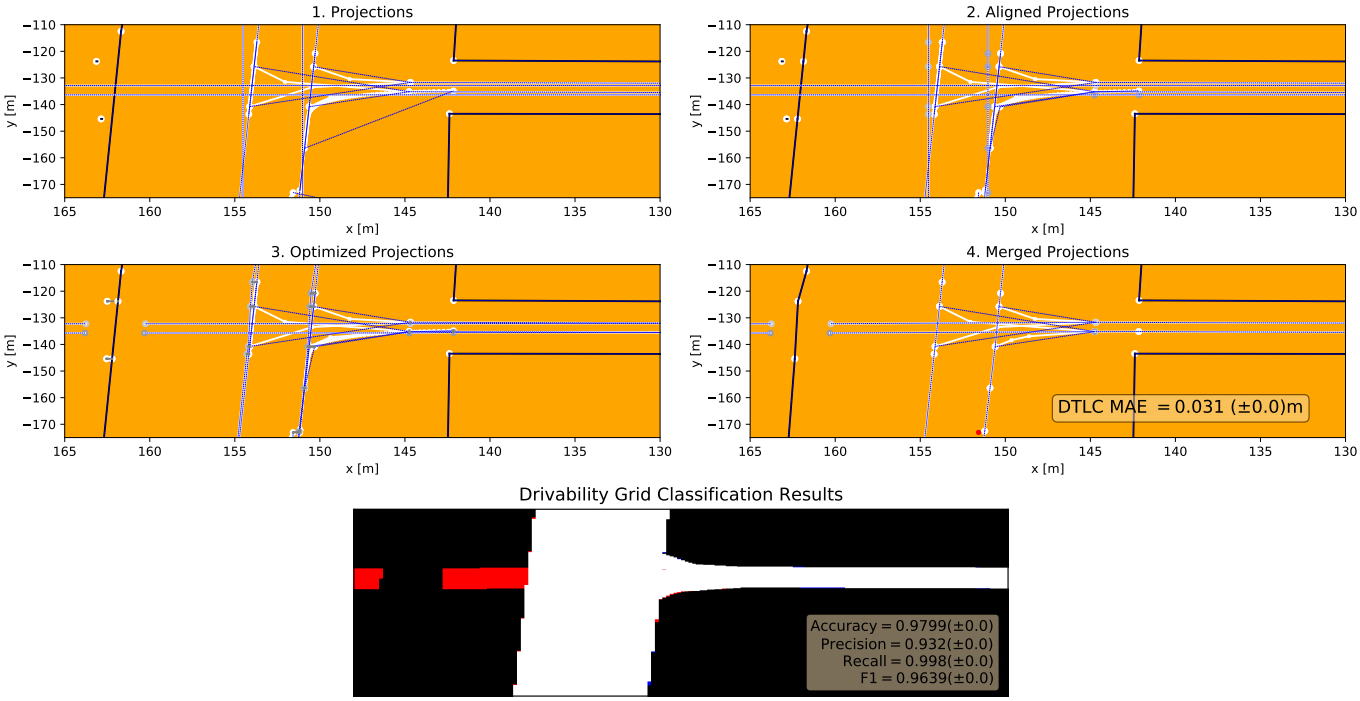
Scenario 4. Noise level 2.7. Missing input: Vehicles



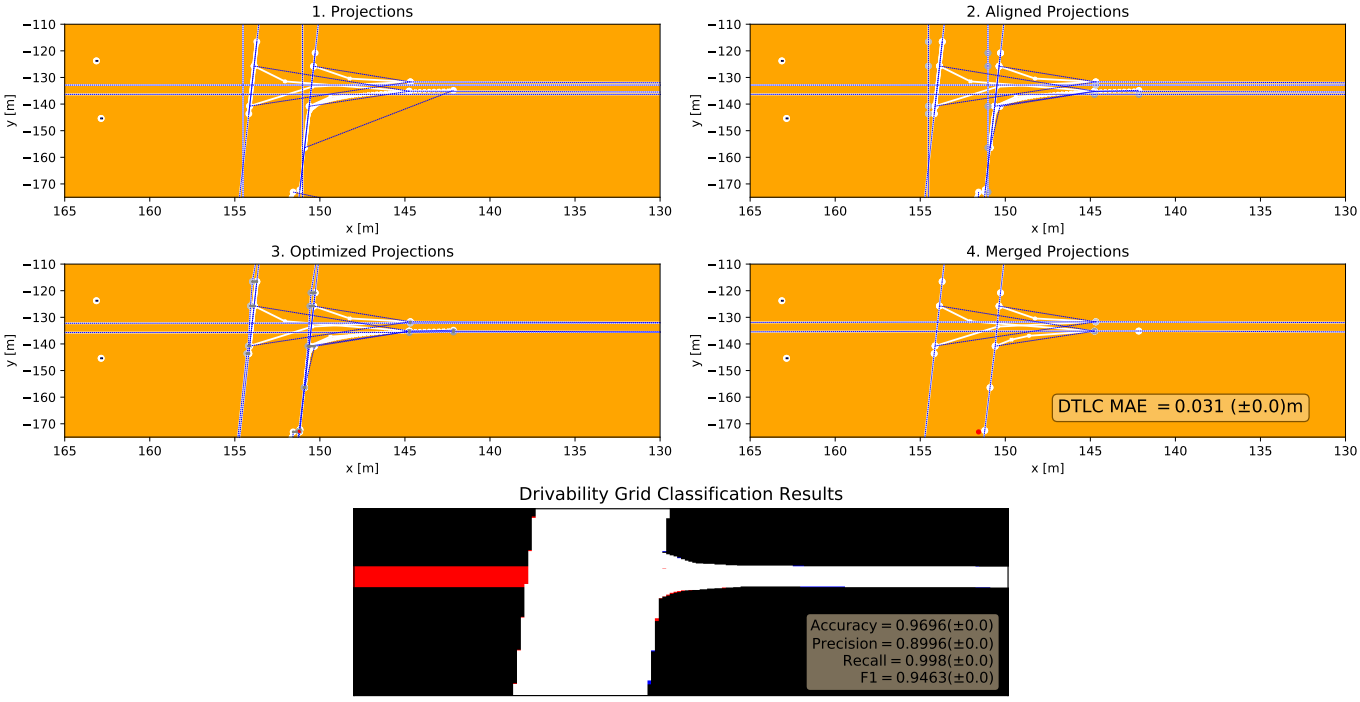
Scenario 4. Noise level 2.7. Missing input: Road network



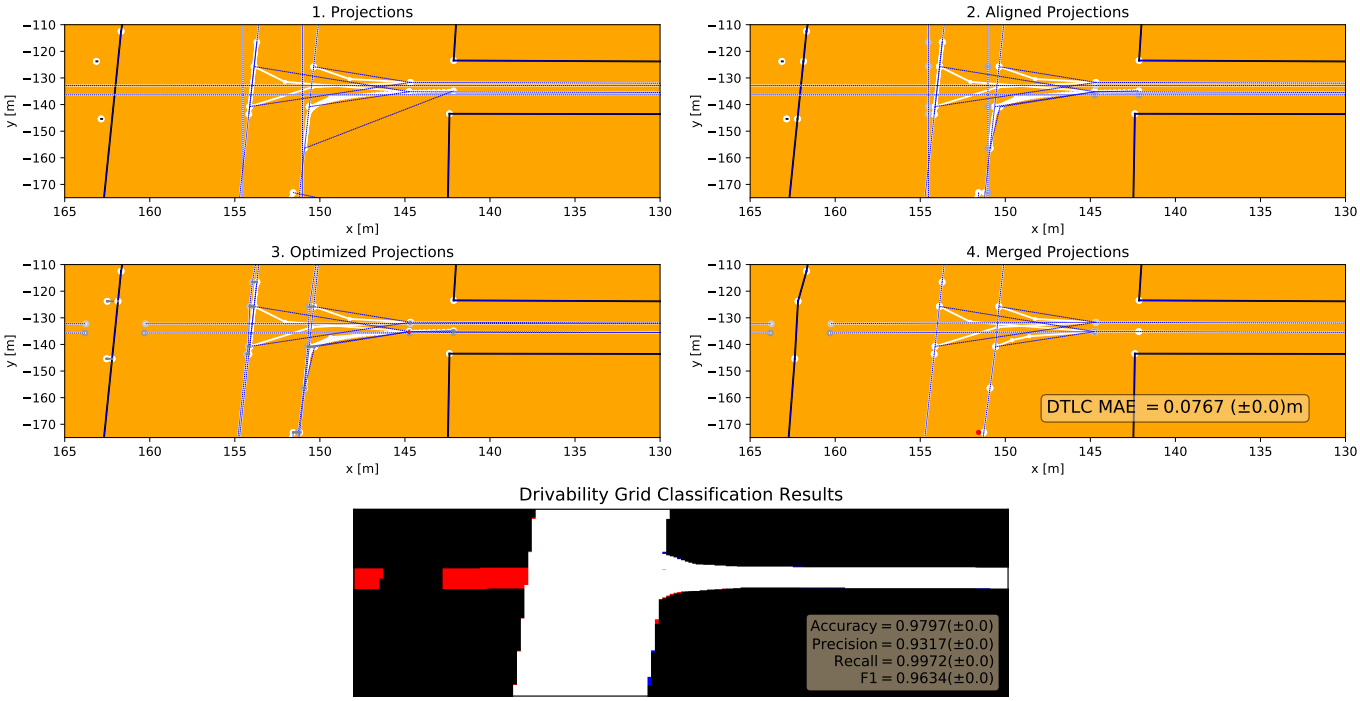
Scenario 5. Noise level 0.0. Missing input: None



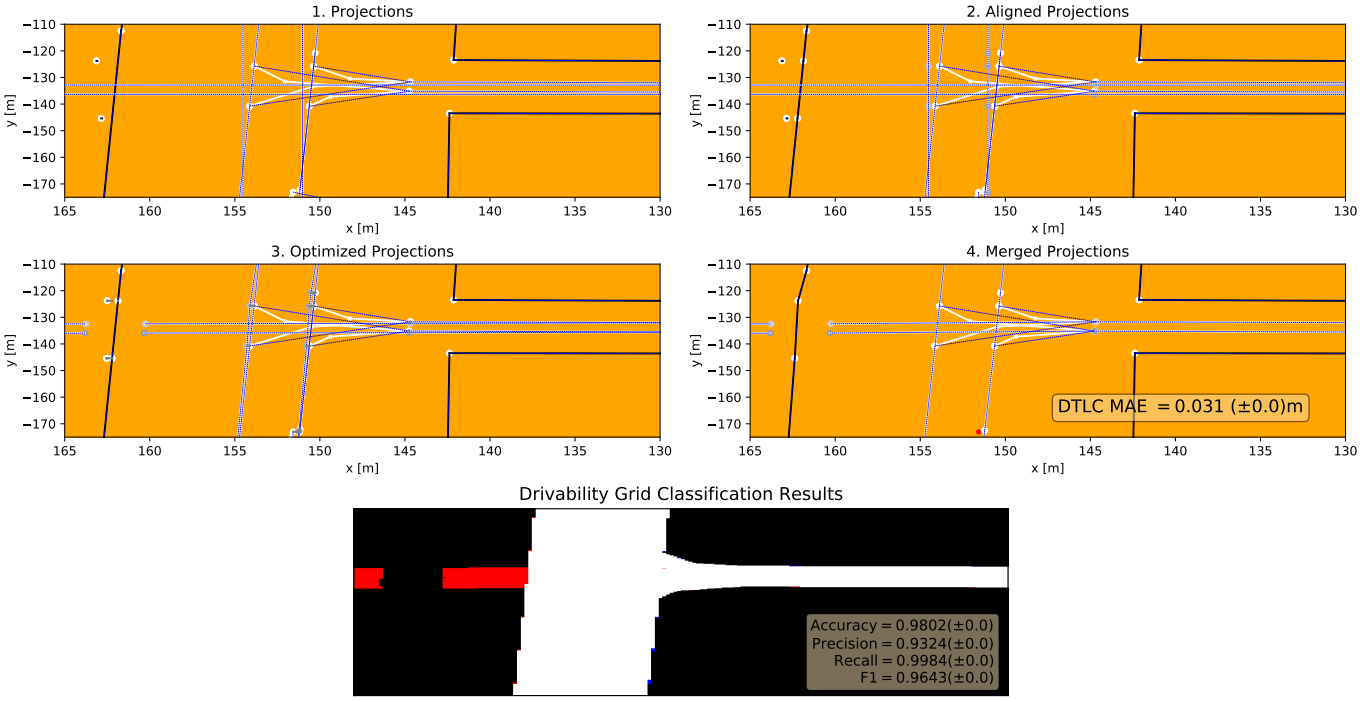
Scenario 5. Noise level 0.0. Missing input: Buildings



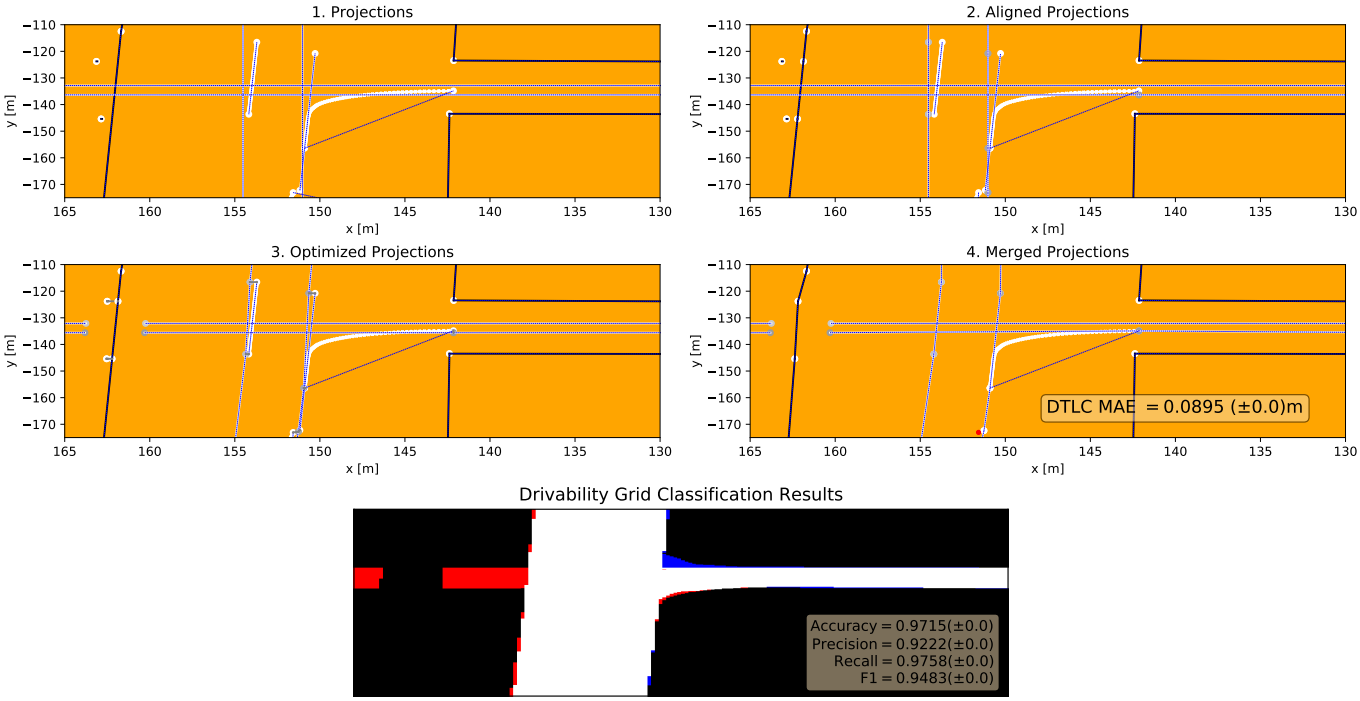
Scenario 5. Noise level 0.0. Missing input: Mobileye



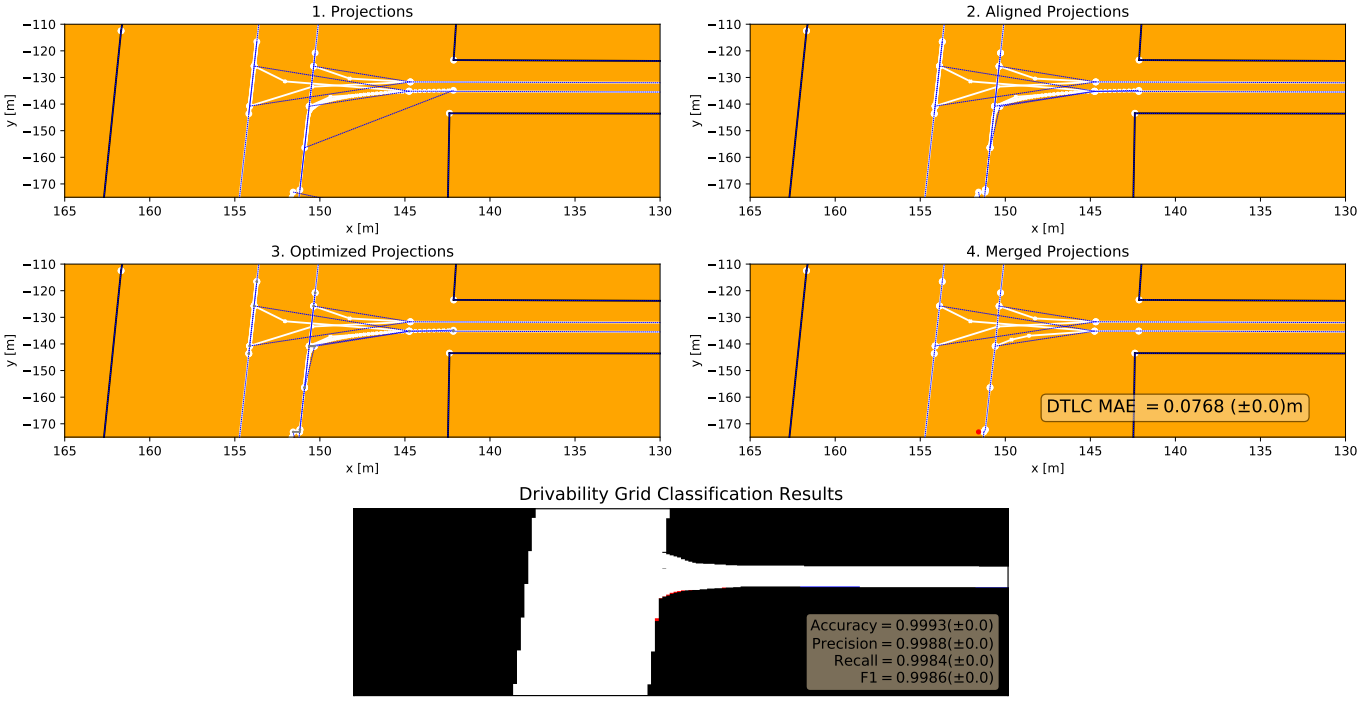
Scenario 5. Noise level 0.0. Missing input: Vehicles



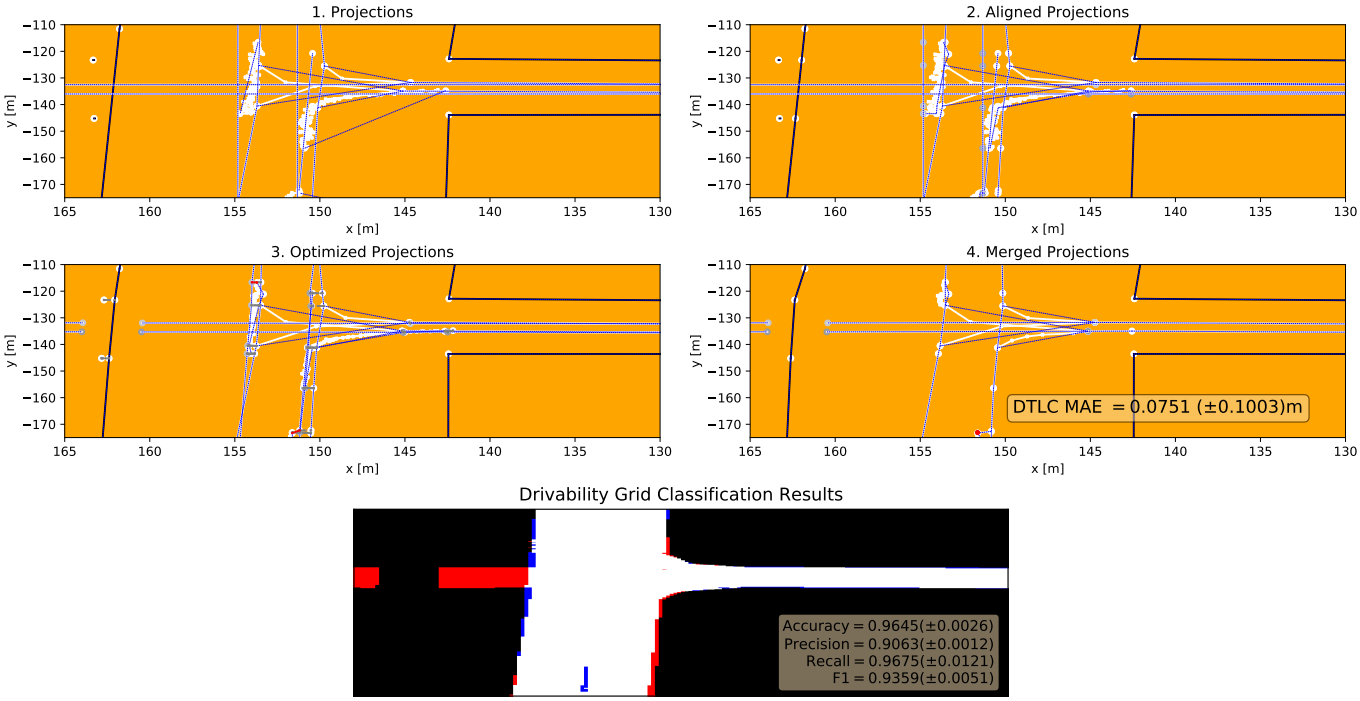
Scenario 5. Noise level 0.0. Missing input: Road network



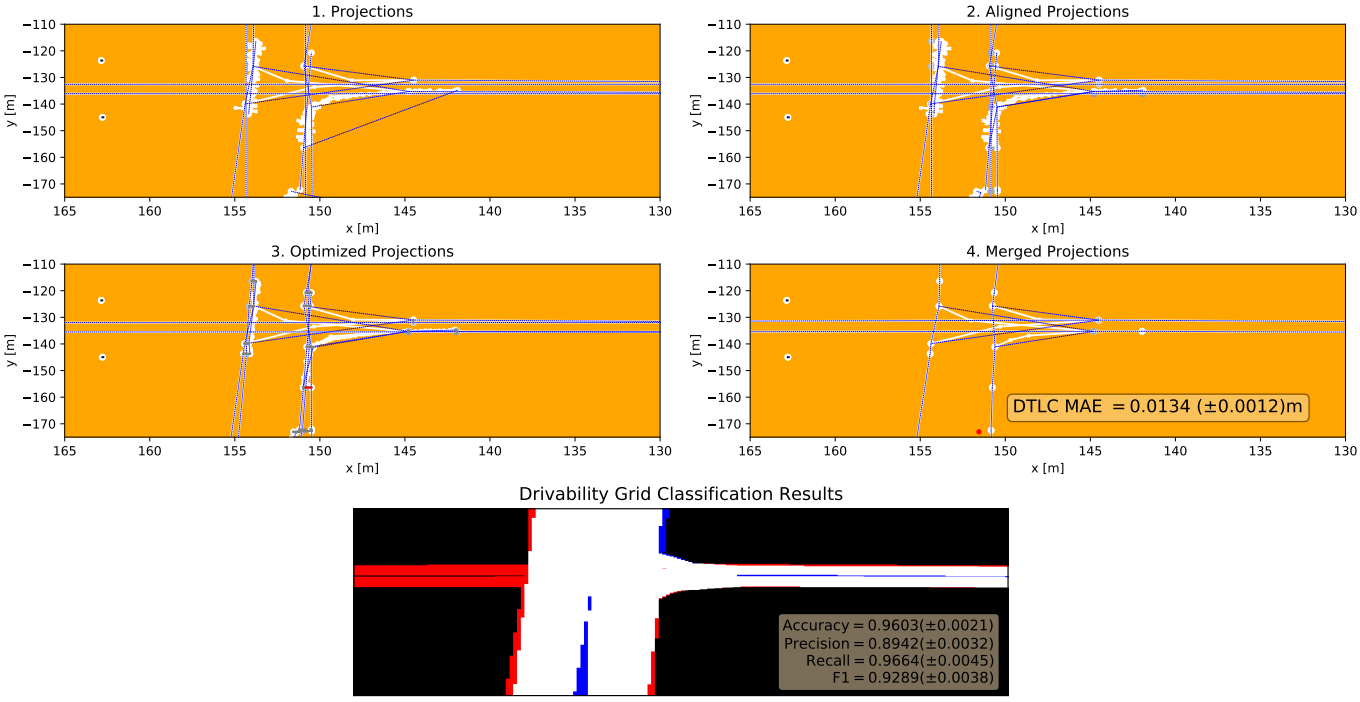
Scenario 5. Noise level 0.0. Missing input: Traffic lights



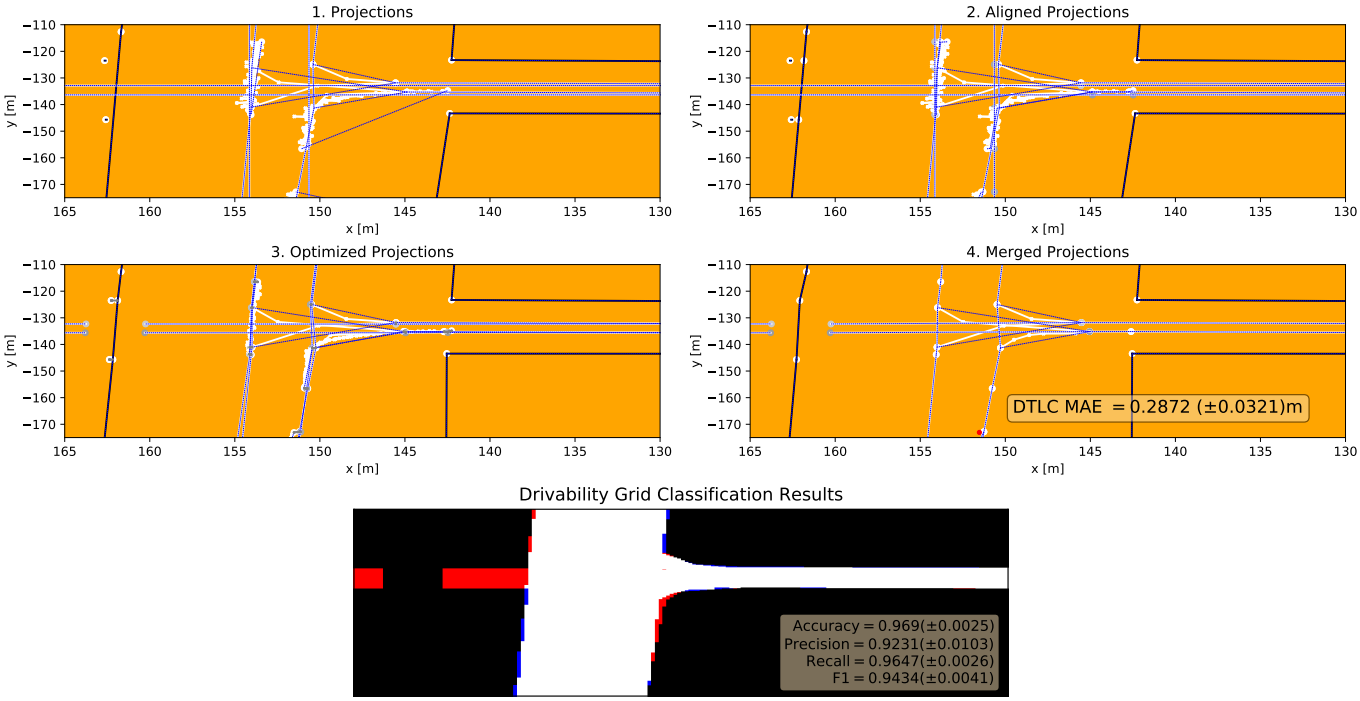
Scenario 5. Noise level 0.9. Missing input: None



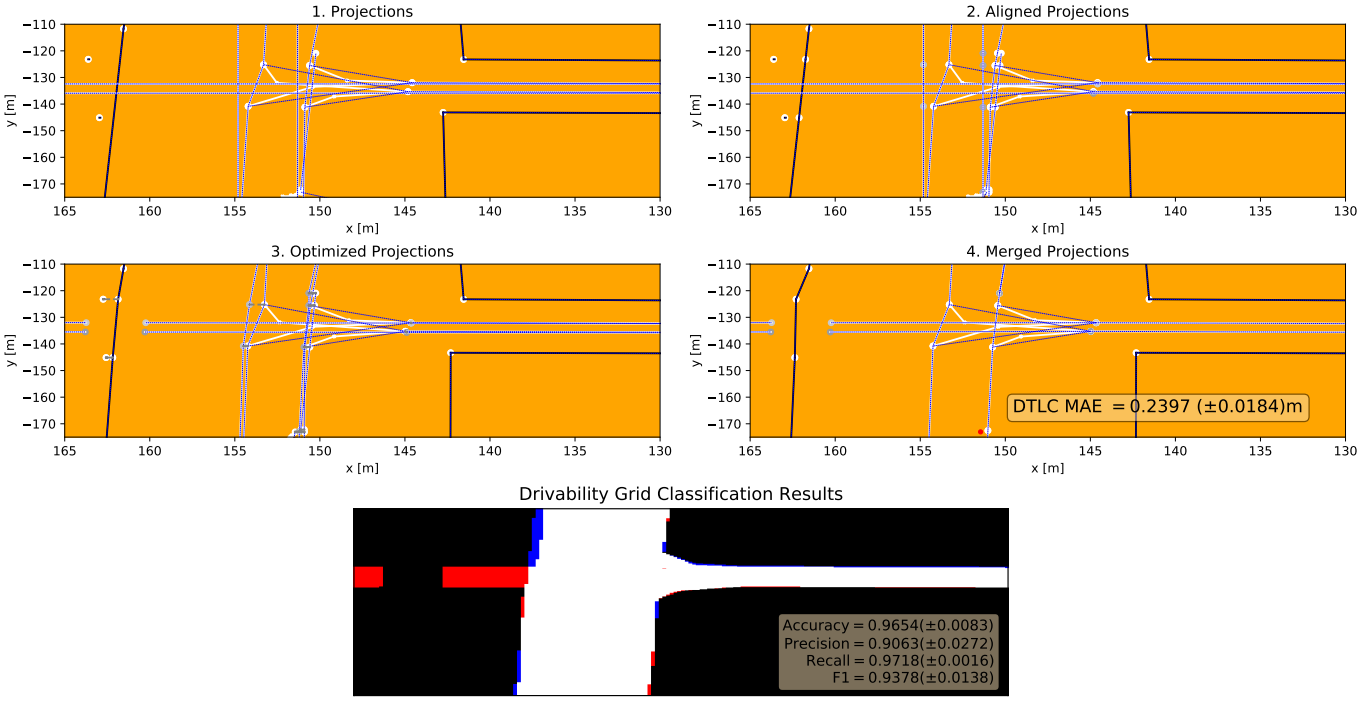
Scenario 5. Noise level 0.9. Missing input: Buildings



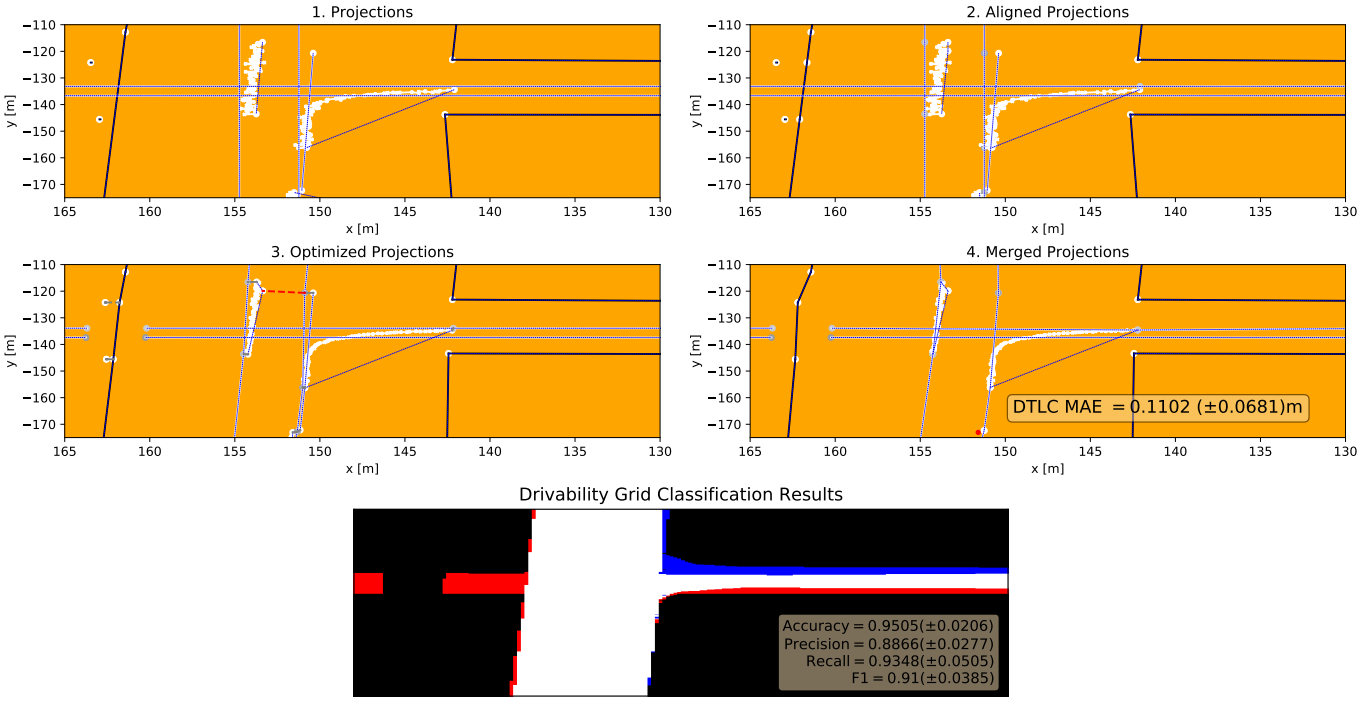
Scenario 5. Noise level 0.9. Missing input: Mobileye



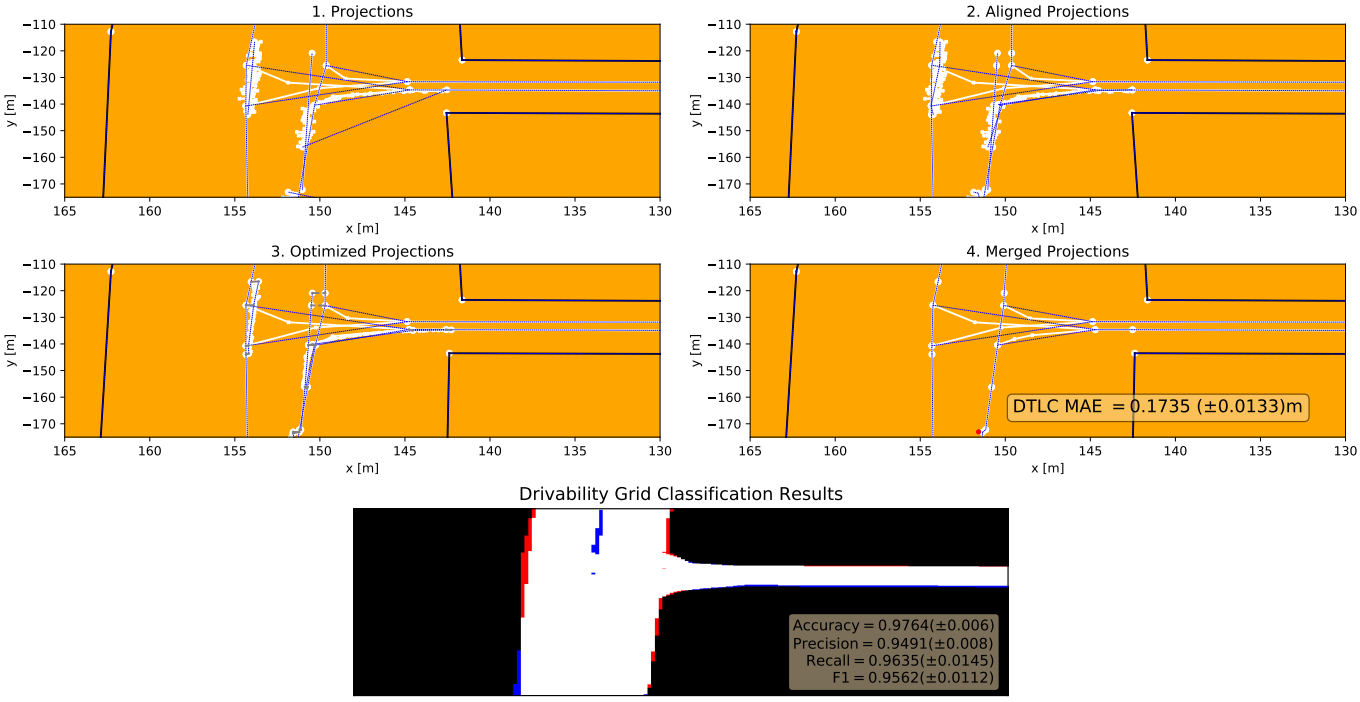
Scenario 5. Noise level 0.9. Missing input: Vehicles



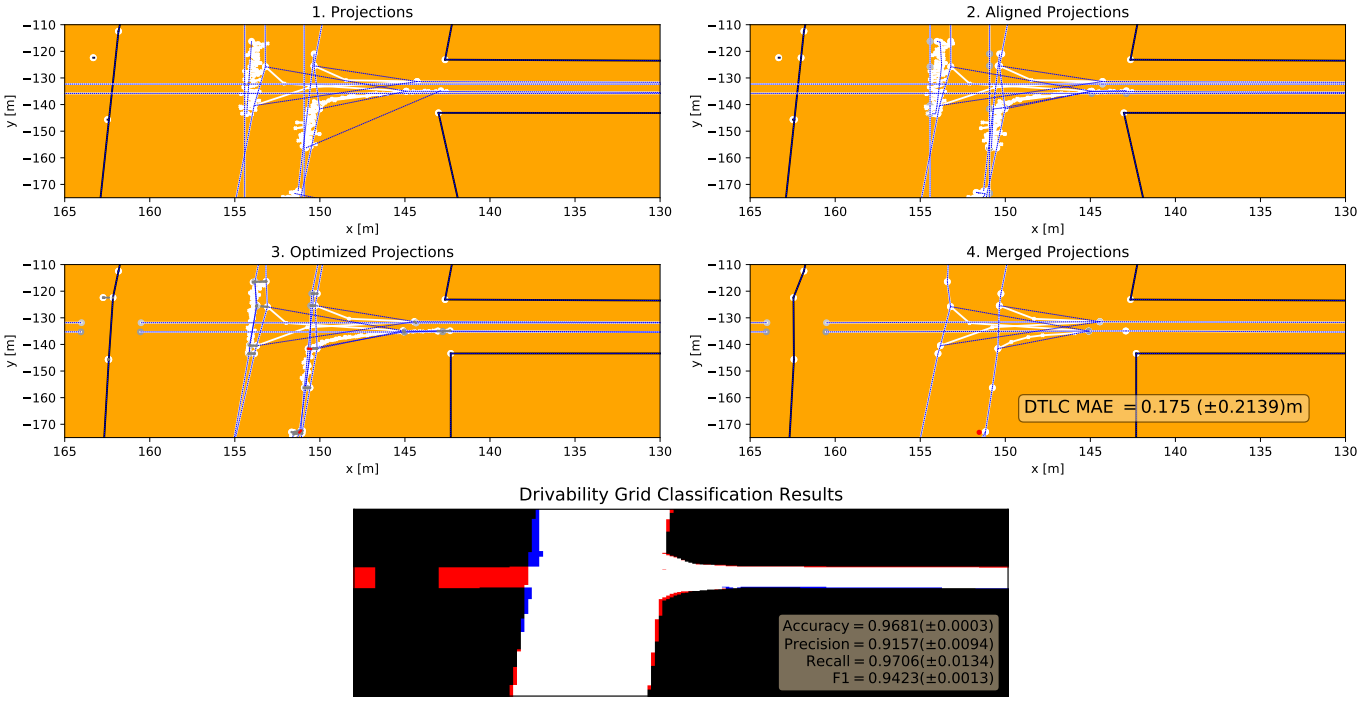
Scenario 5. Noise level 0.9. Missing input: Road network



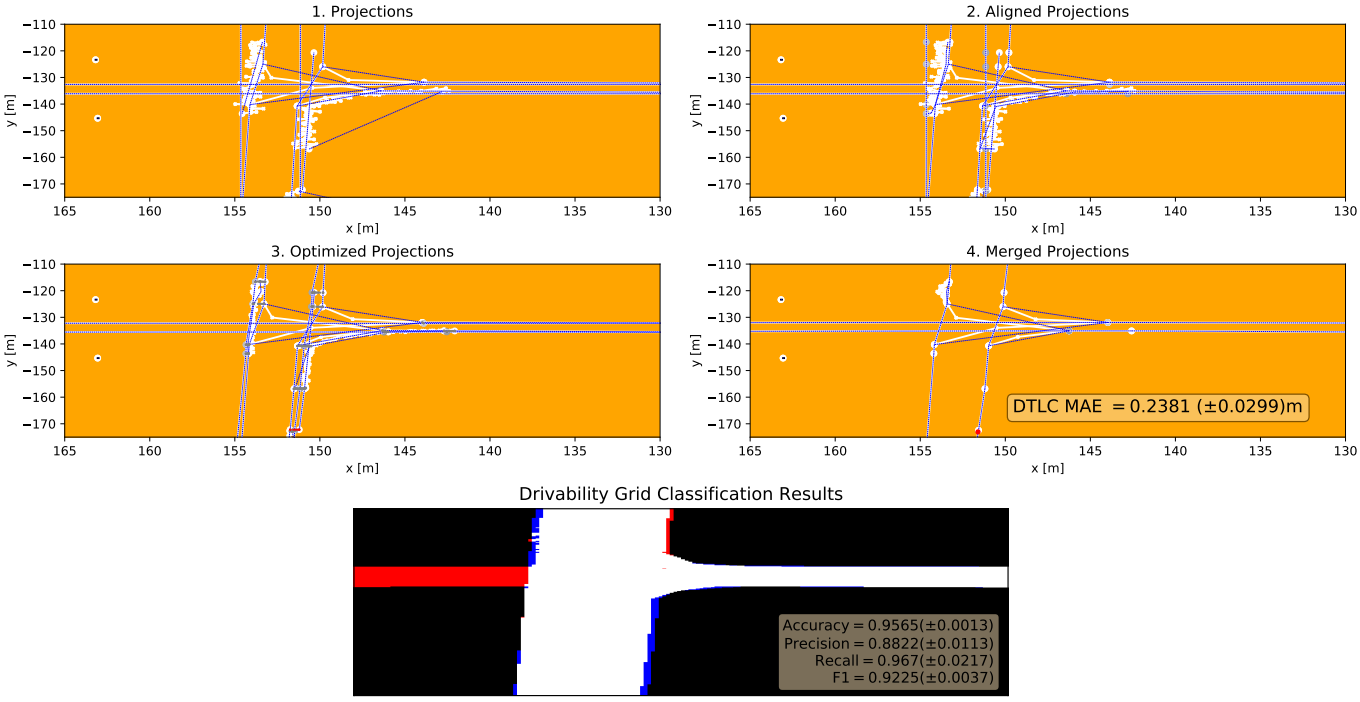
Scenario 5. Noise level 0.9. Missing input: Traffic lights



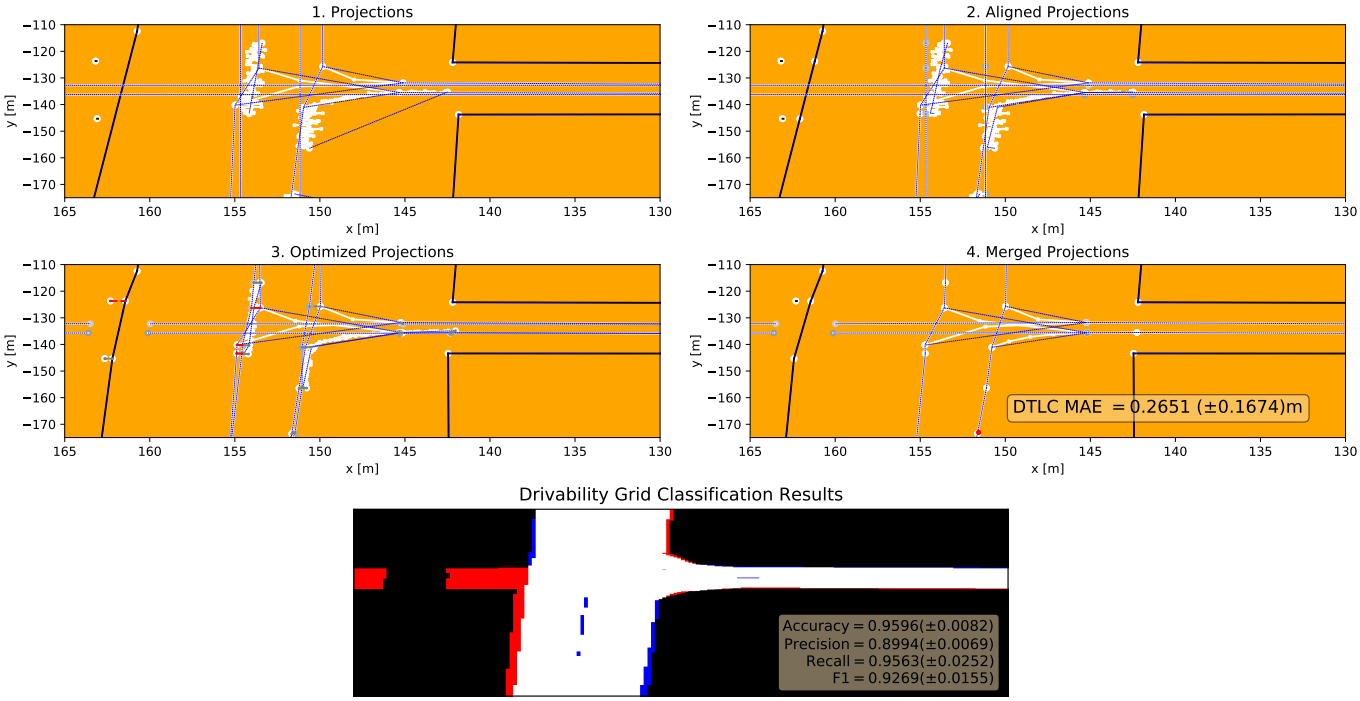
Scenario 5. Noise level 1.8. Missing input: None



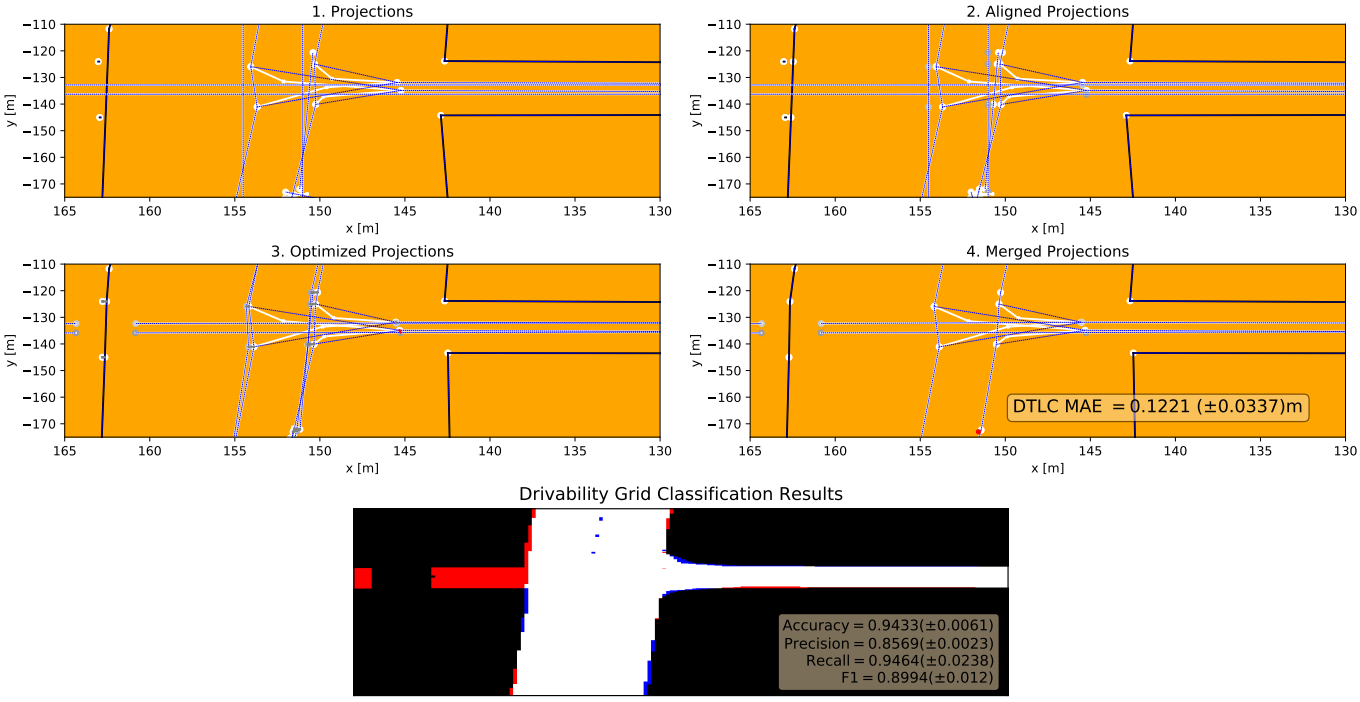
Scenario 5. Noise level 1.8. Missing input: Buildings



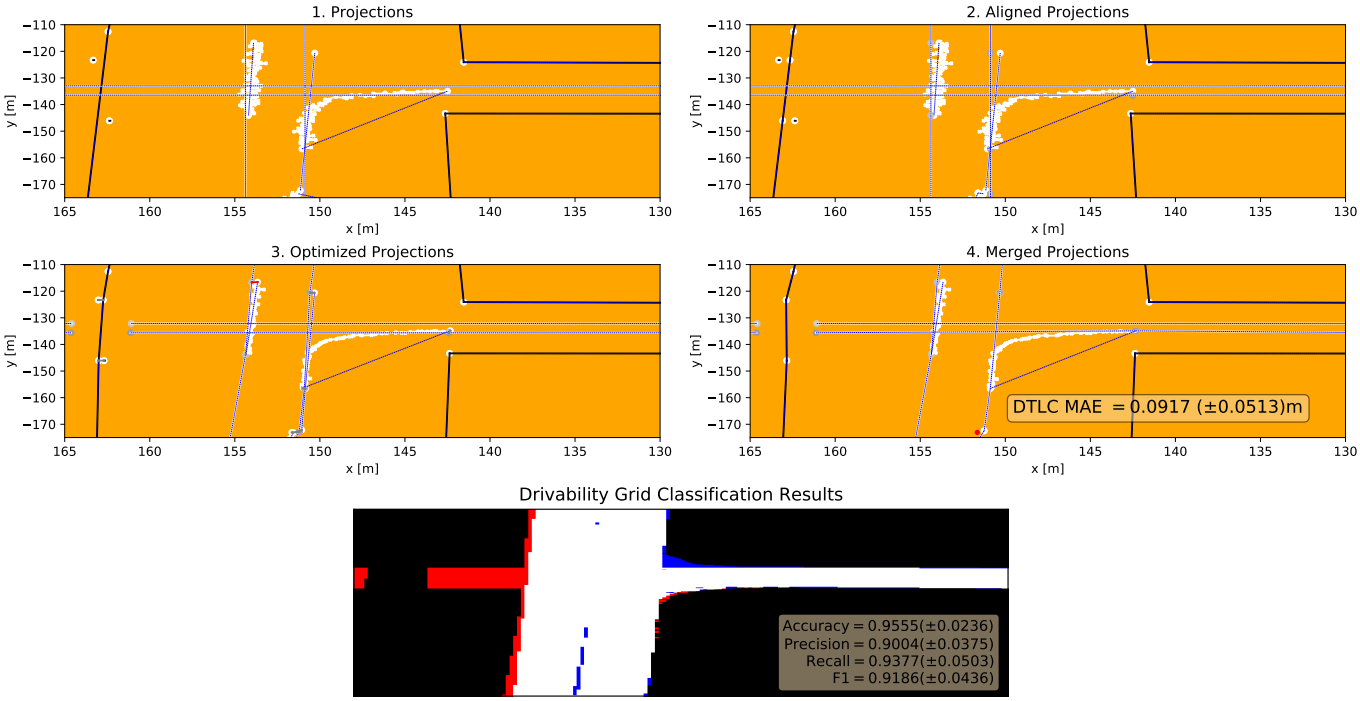
Scenario 5. Noise level 1.8. Missing input: Mobileye



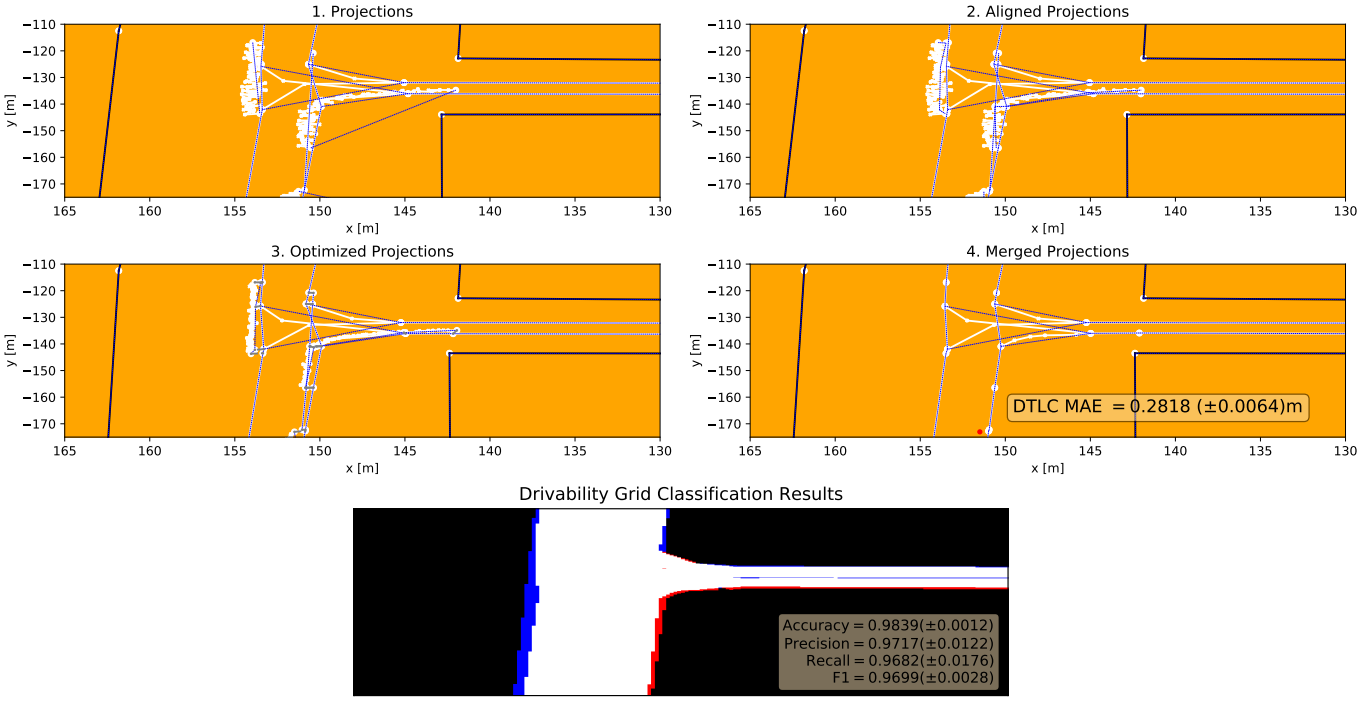
Scenario 5. Noise level 1.8. Missing input: Vehicles



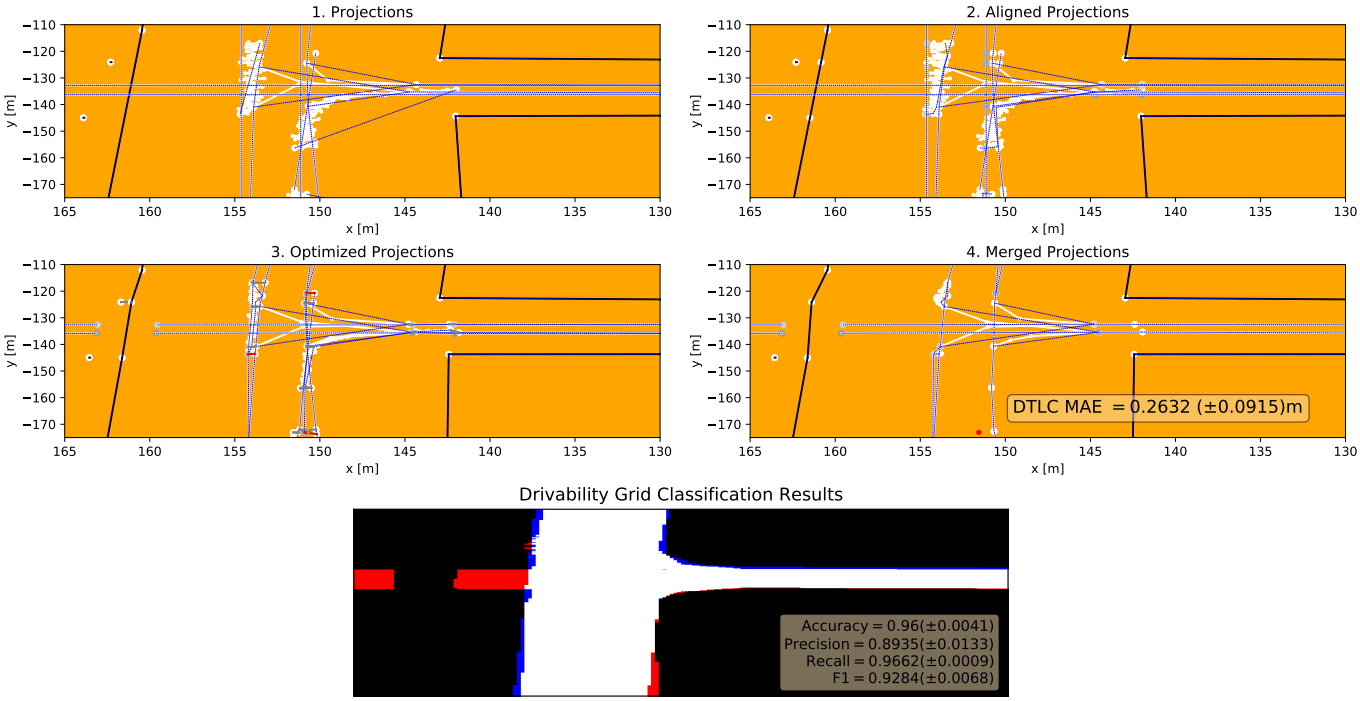
Scenario 5. Noise level 1.8. Missing input: Road network



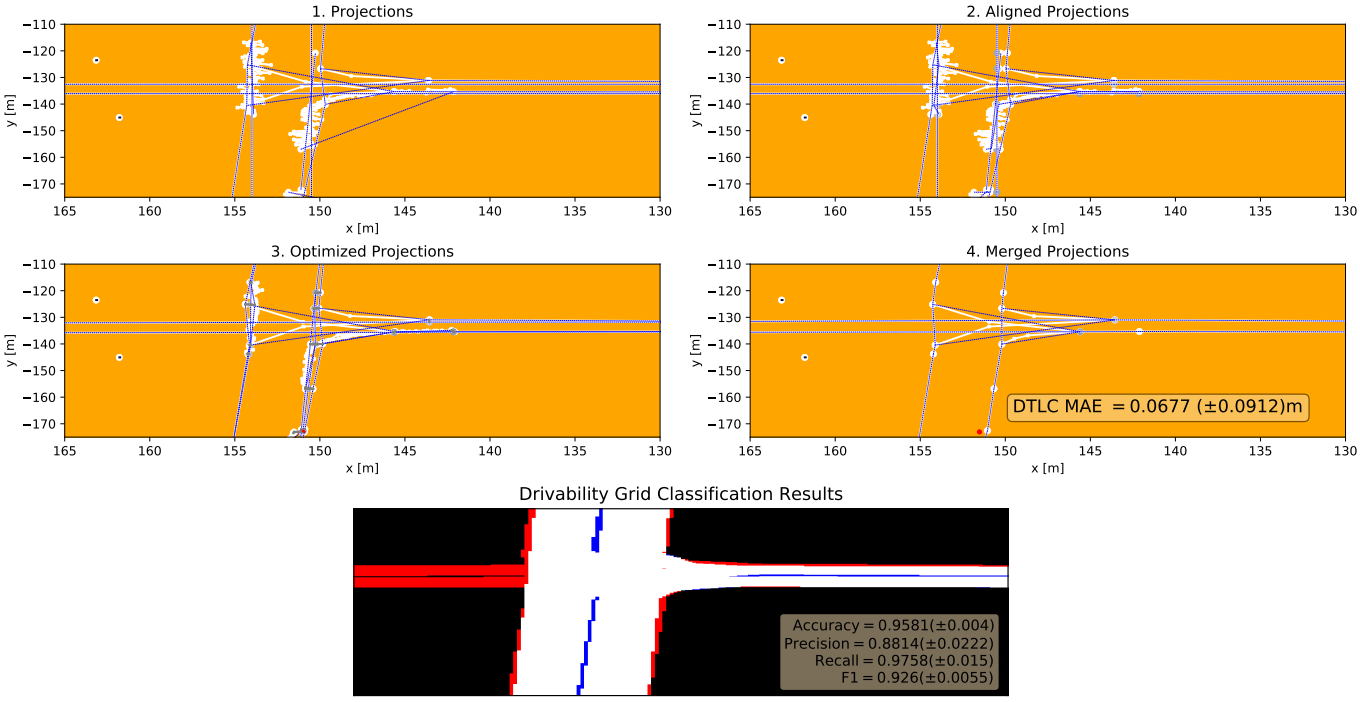
Scenario 5. Noise level 1.8. Missing input: Traffic lights



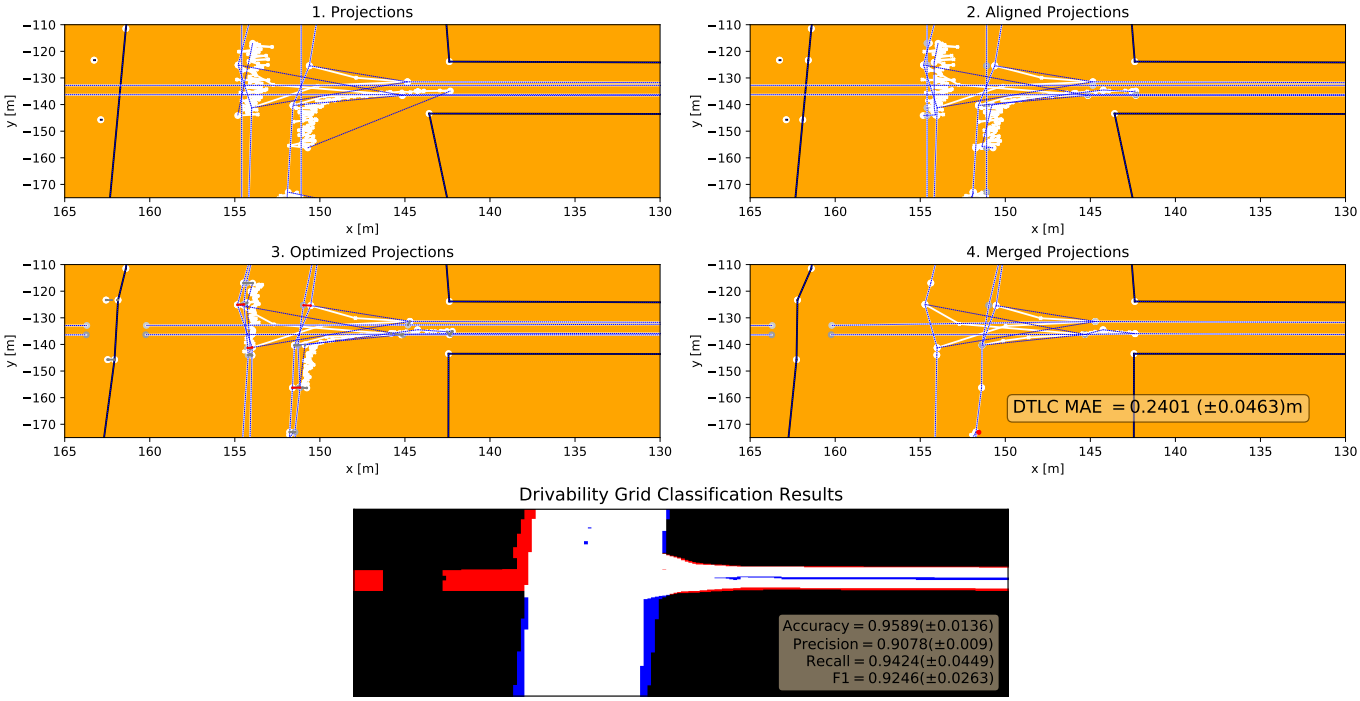
Scenario 5. Noise level 2.7. Missing input: None



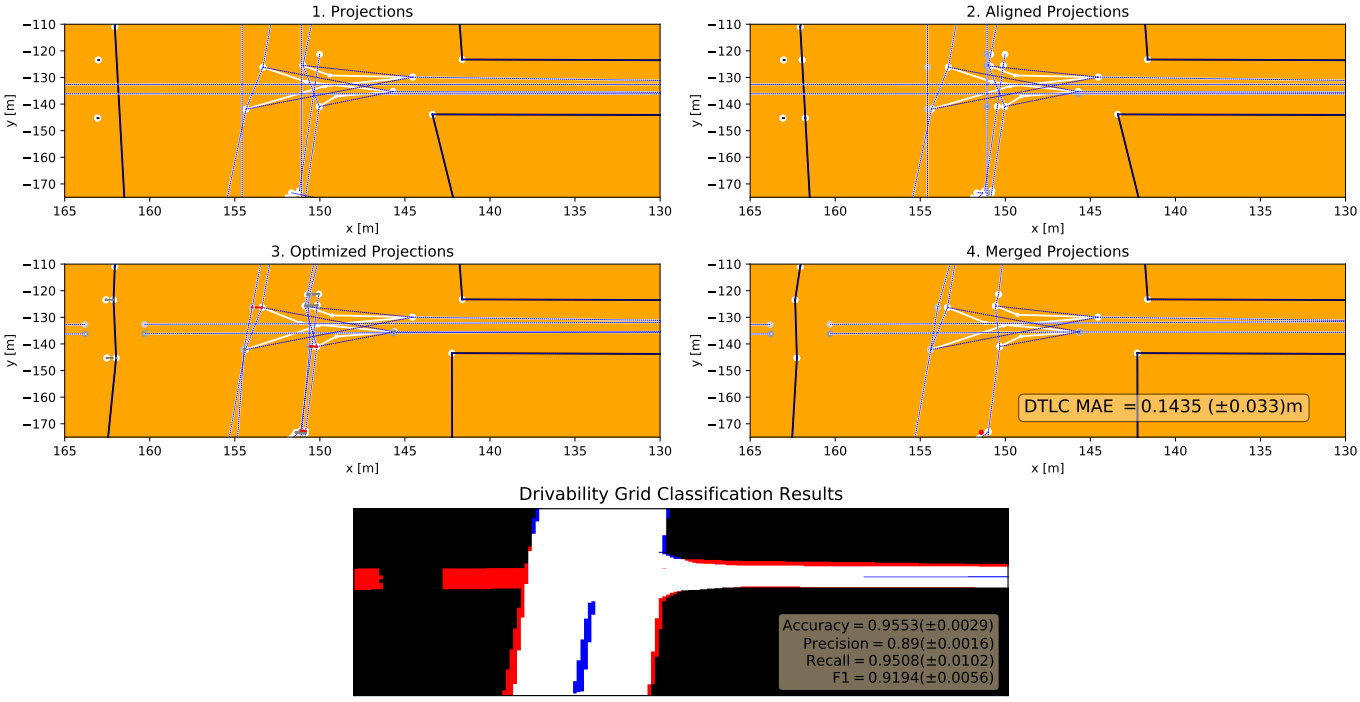
Scenario 5. Noise level 2.7. Missing input: Buildings



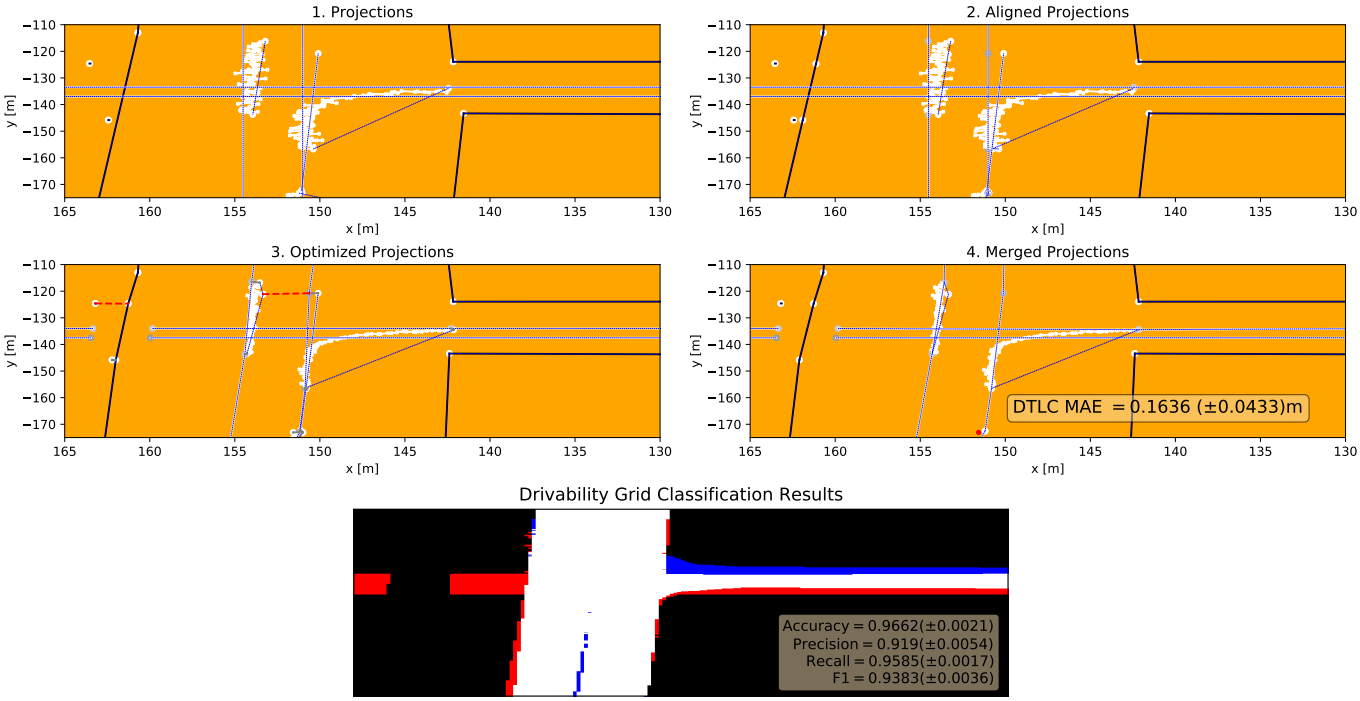
Scenario 5. Noise level 2.7. Missing input: Mobileye



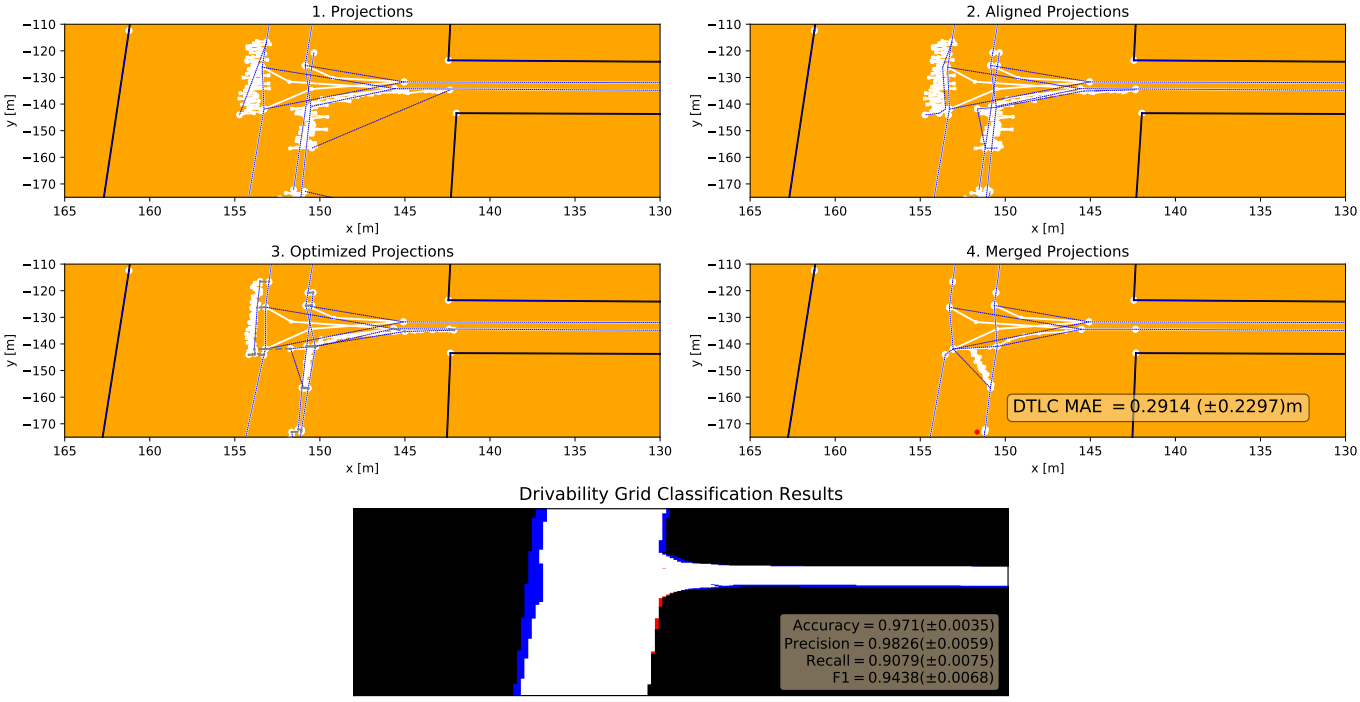
Scenario 5. Noise level 2.7. Missing input: Vehicles



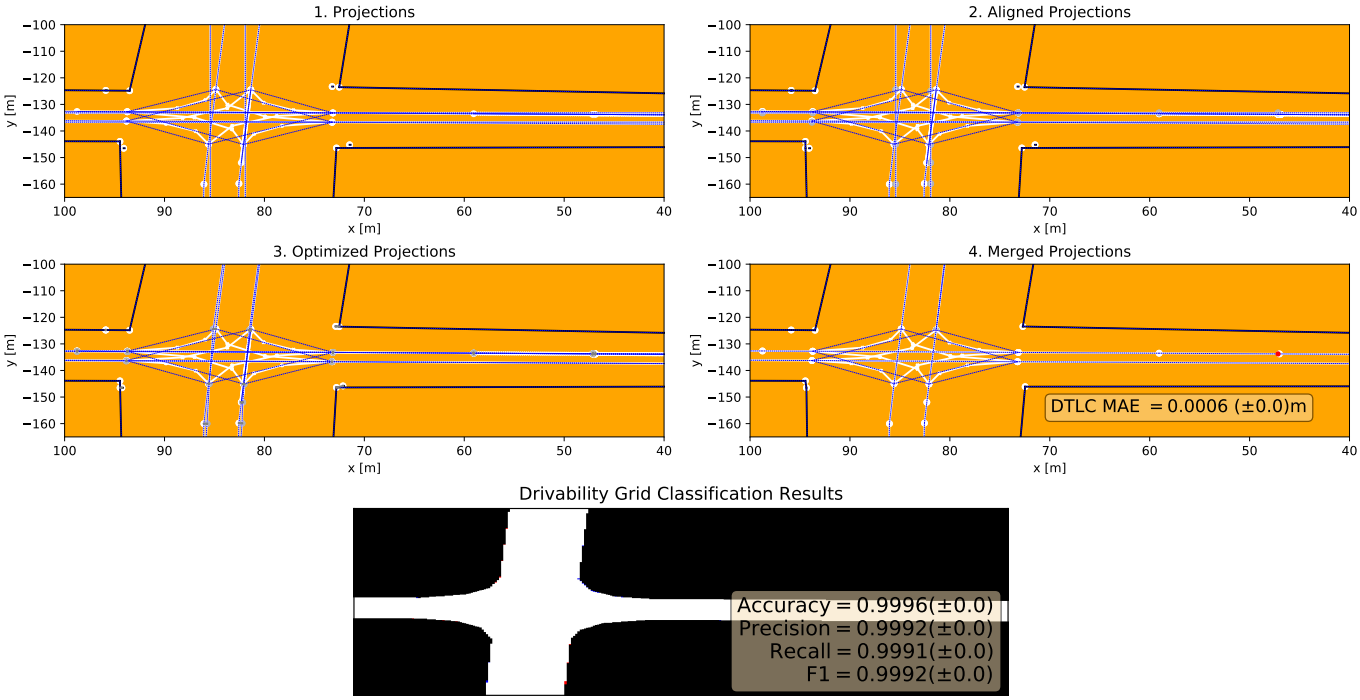
Scenario 5. Noise level 2.7. Missing input: Road network



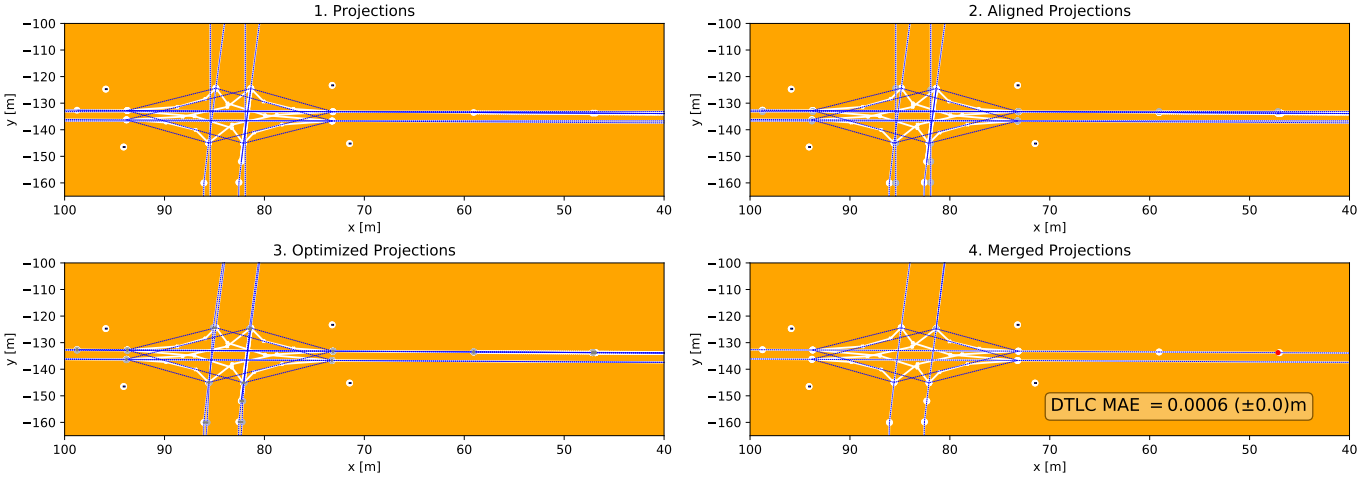
Scenario 5. Noise level 2.7. Missing input: Traffic lights

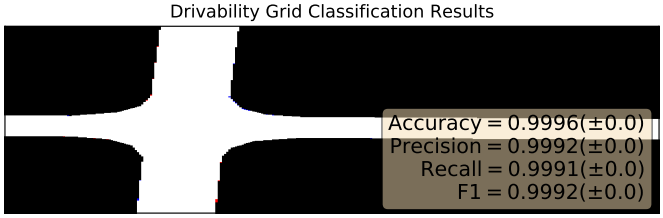


Scenario 6. Noise level 0.0. Missing input: None

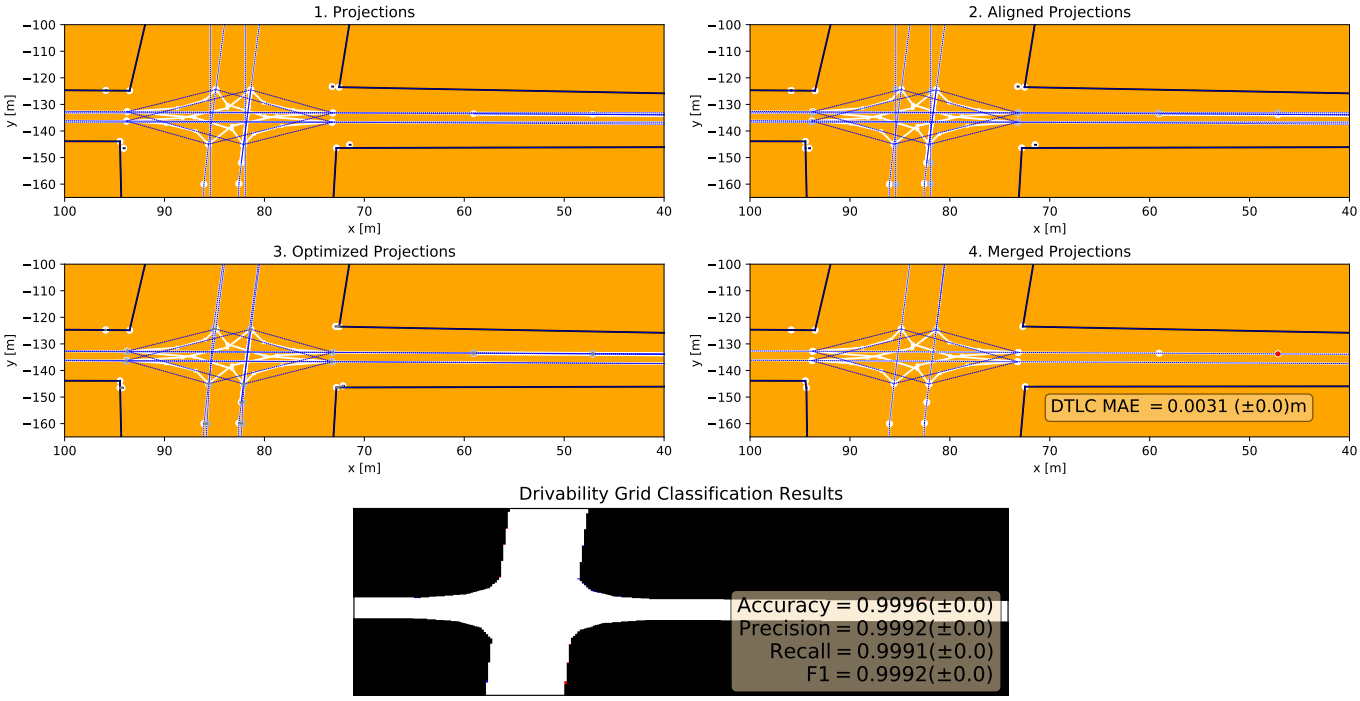


Scenario 6. Noise level 0.0. Missing input: Buildings

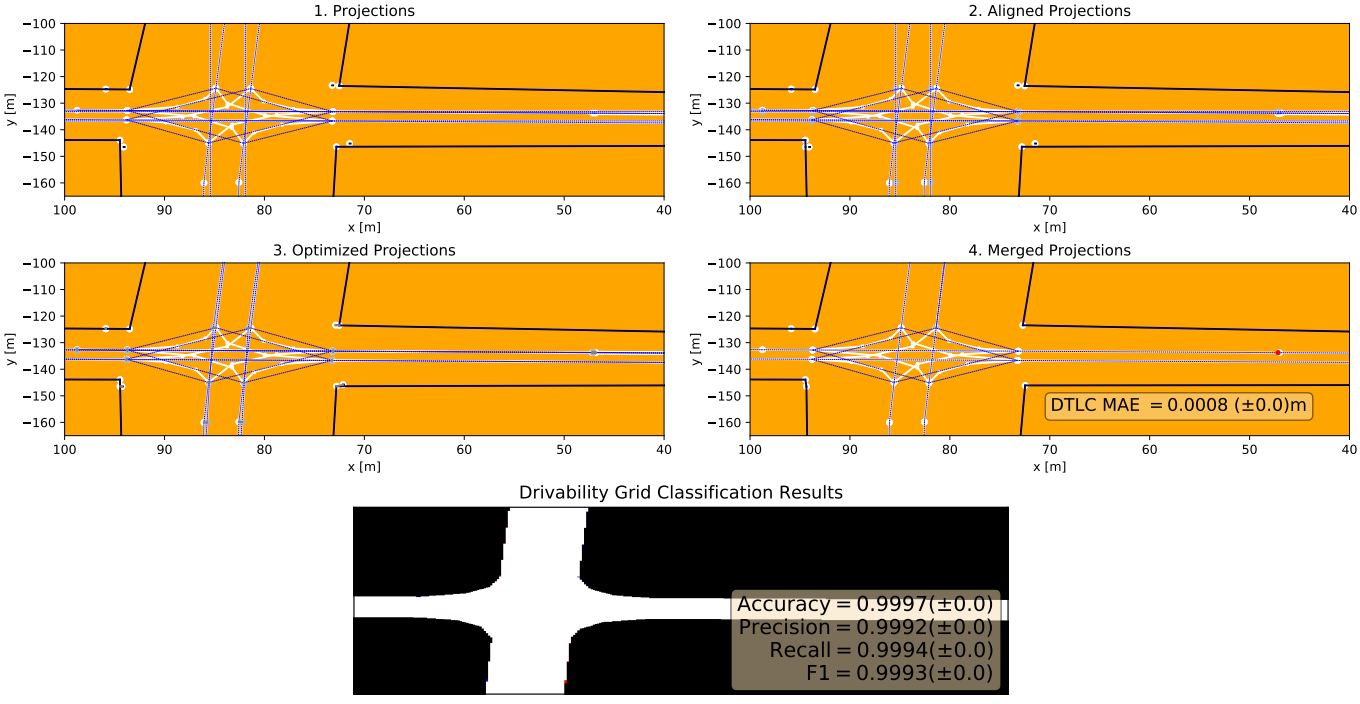




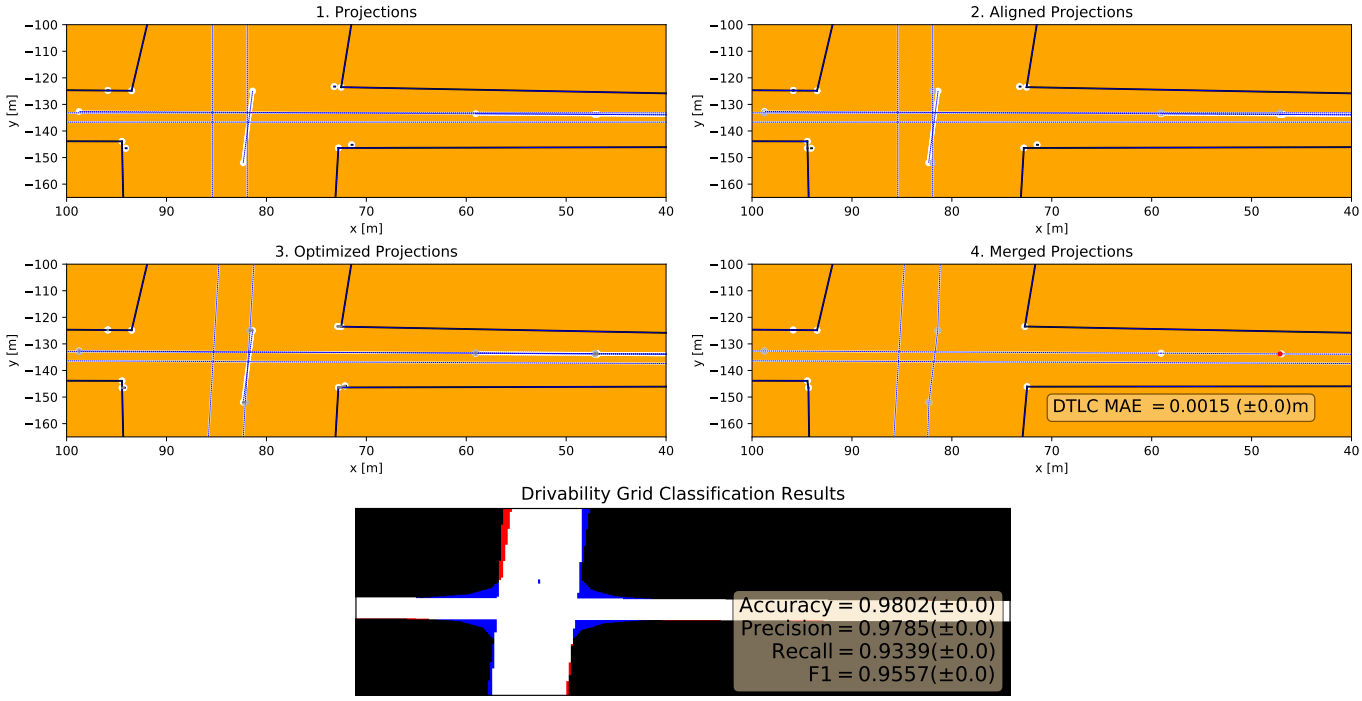
Scenario 6. Noise level 0.0. Missing input: Mobileye



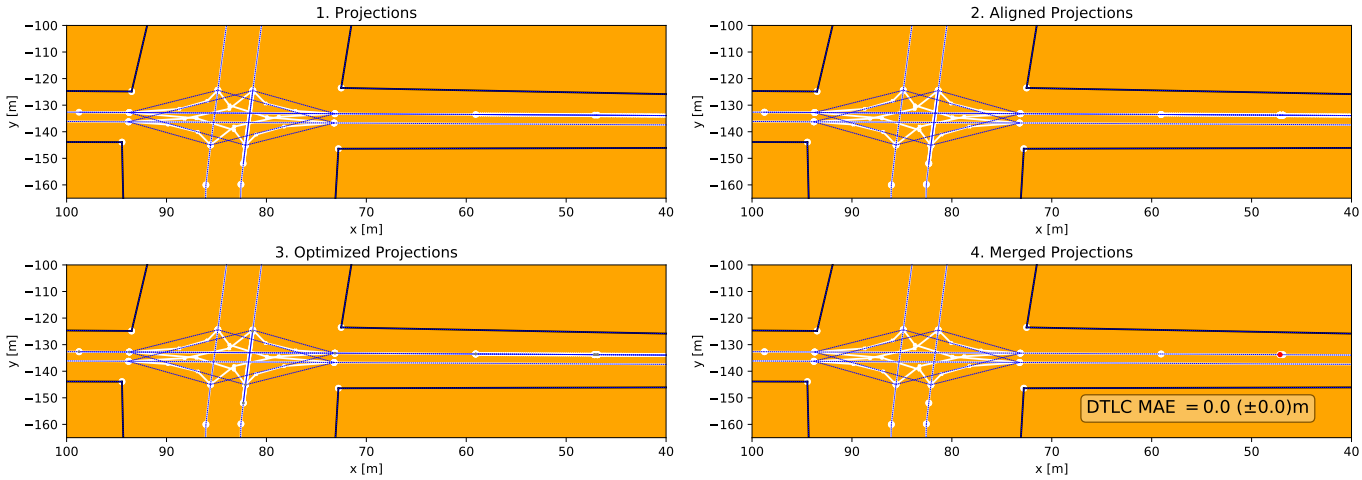
Scenario 6. Noise level 0.0. Missing input: Vehicles

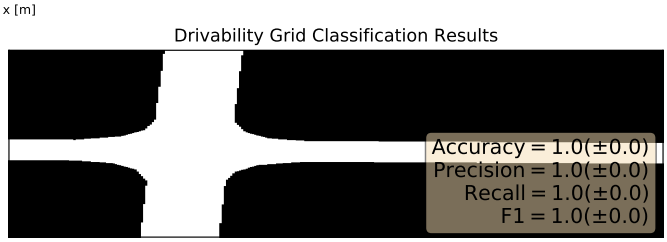


Scenario 6. Noise level 0.0. Missing input: Road network

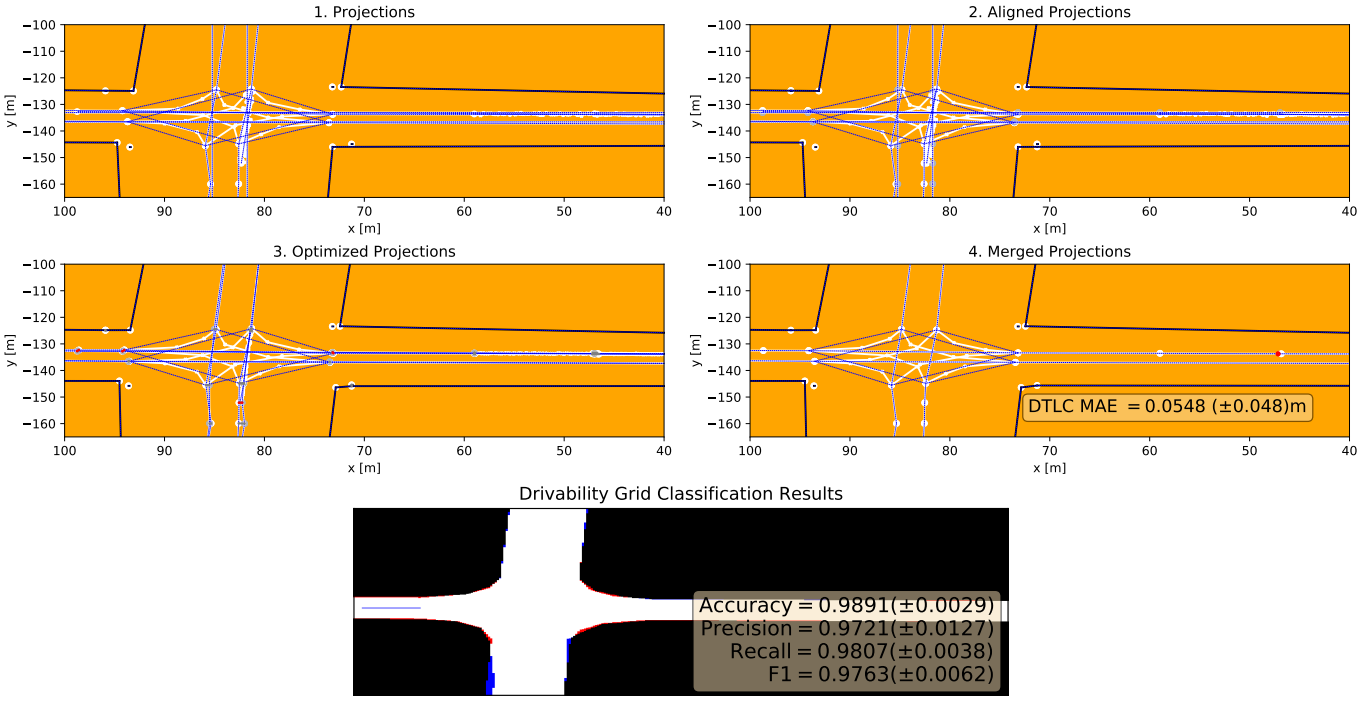


Scenario 6. Noise level 0.0. Missing input: Traffic lights

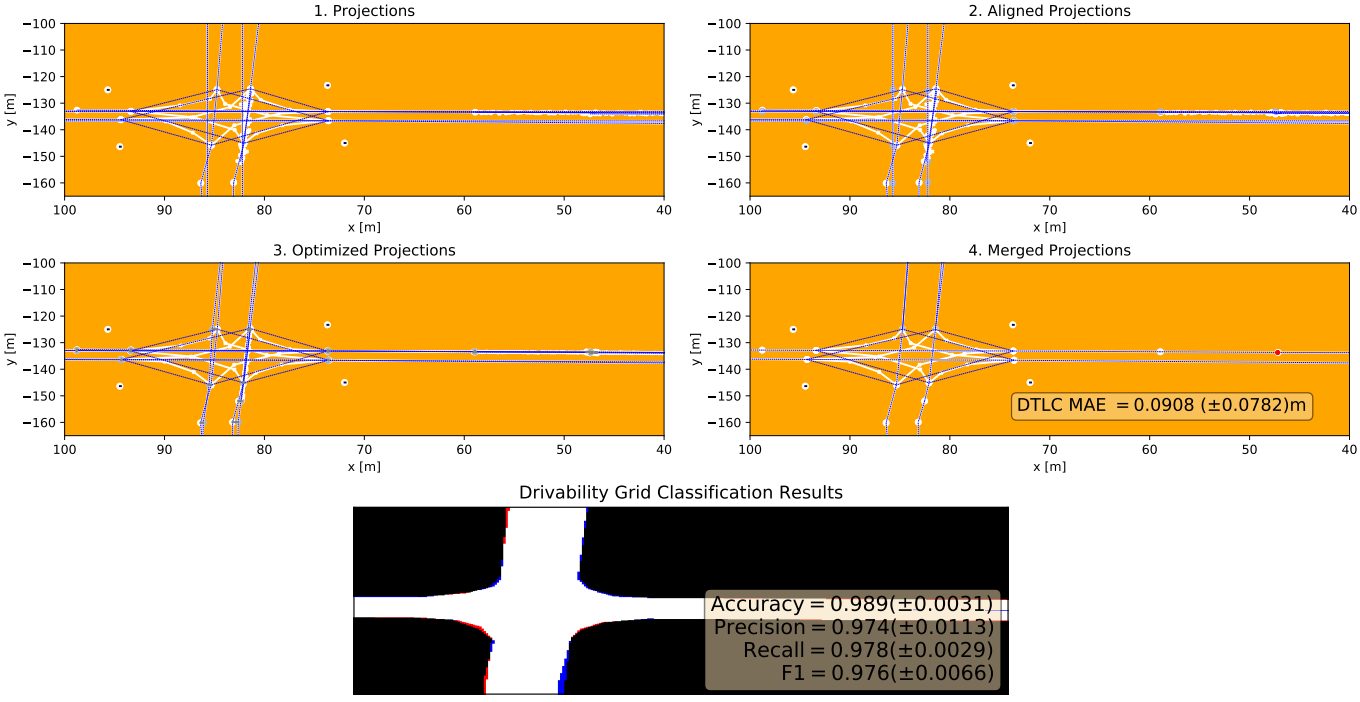




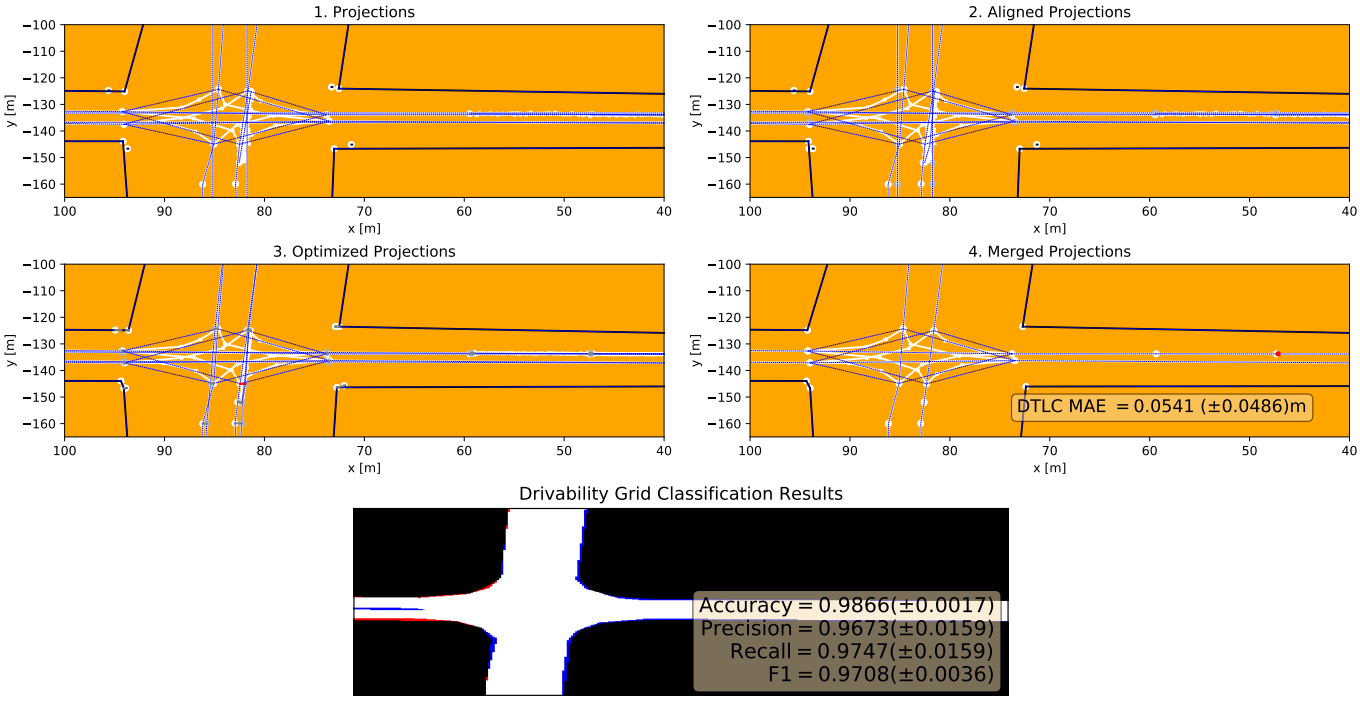
Scenario 6. Noise level 0.9. Missing input: None



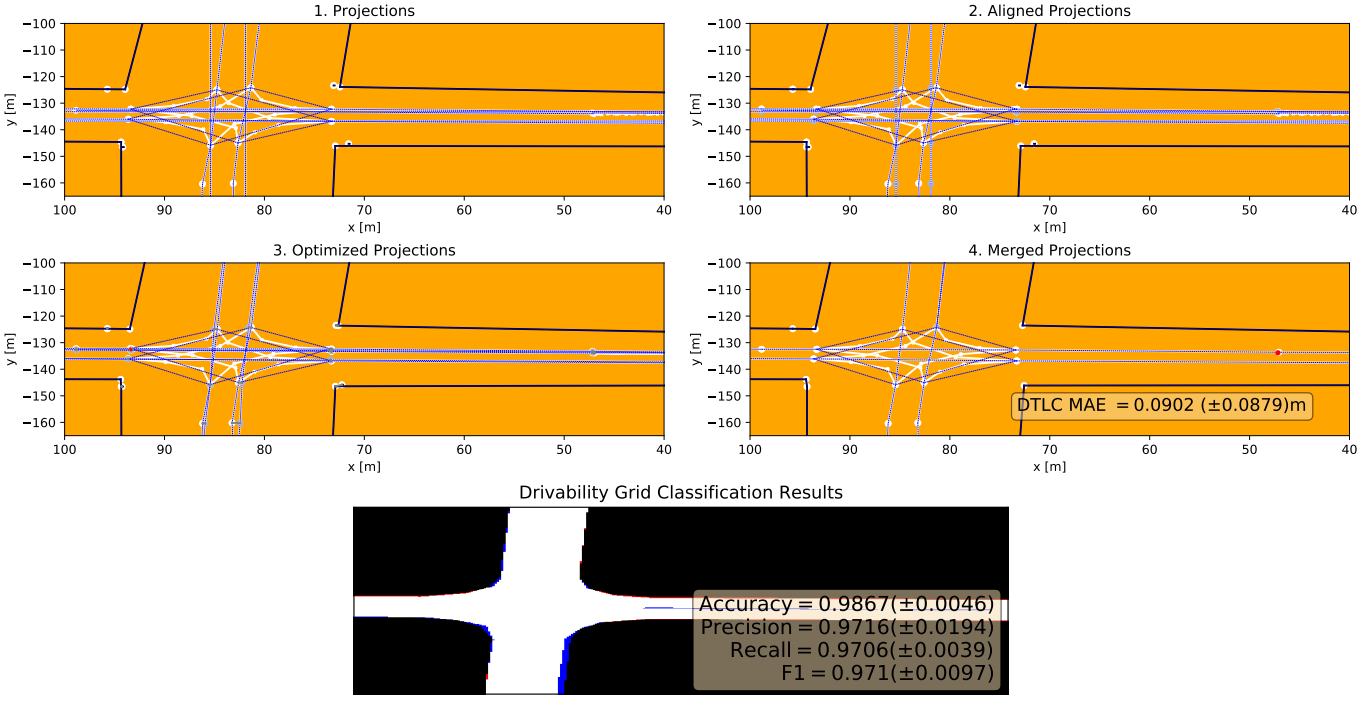
Scenario 6. Noise level 0.9. Missing input: Buildings



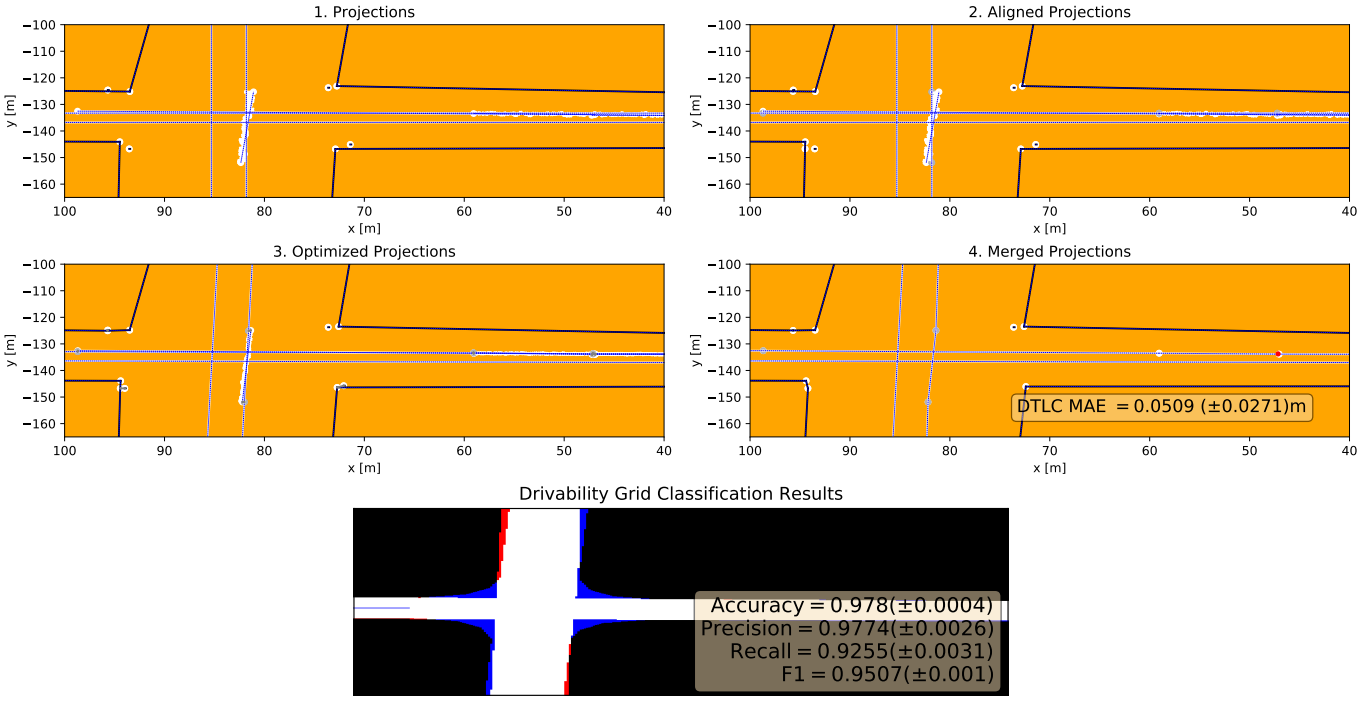
Scenario 6. Noise level 0.9. Missing input: Mobileye



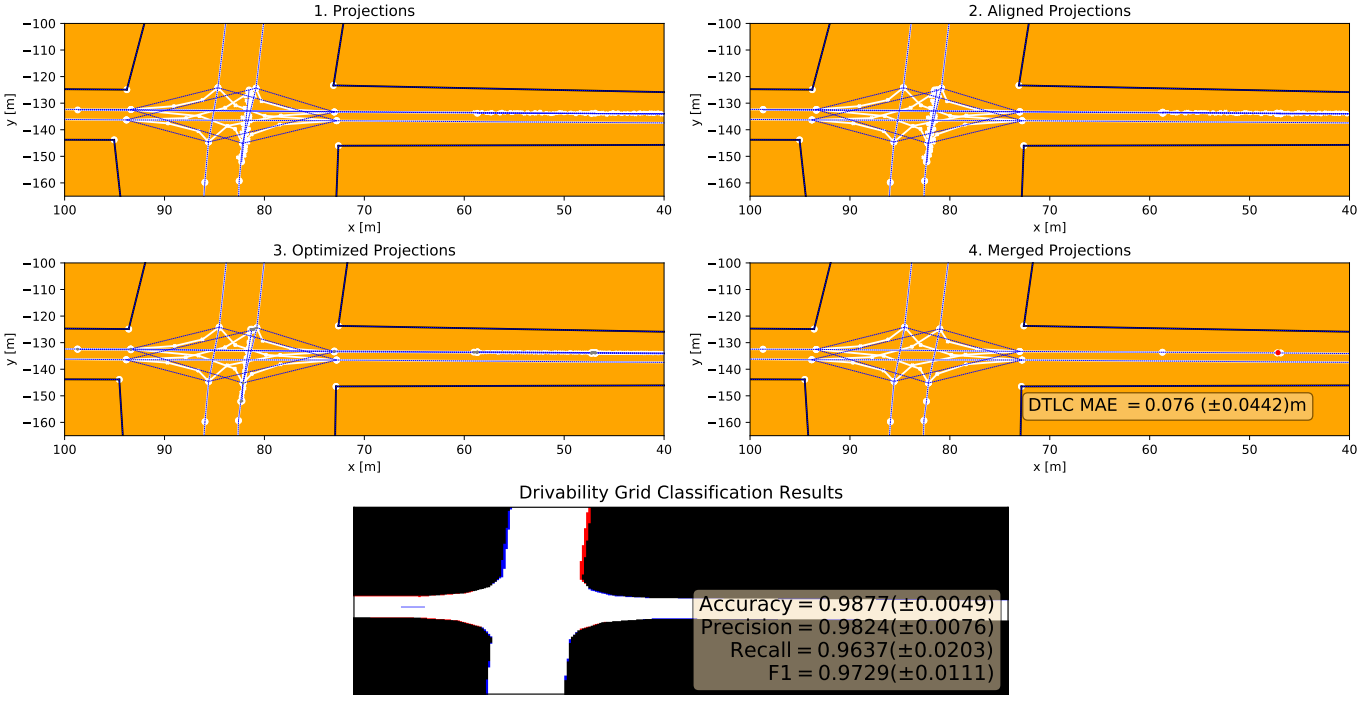
Scenario 6. Noise level 0.9. Missing input: Vehicles



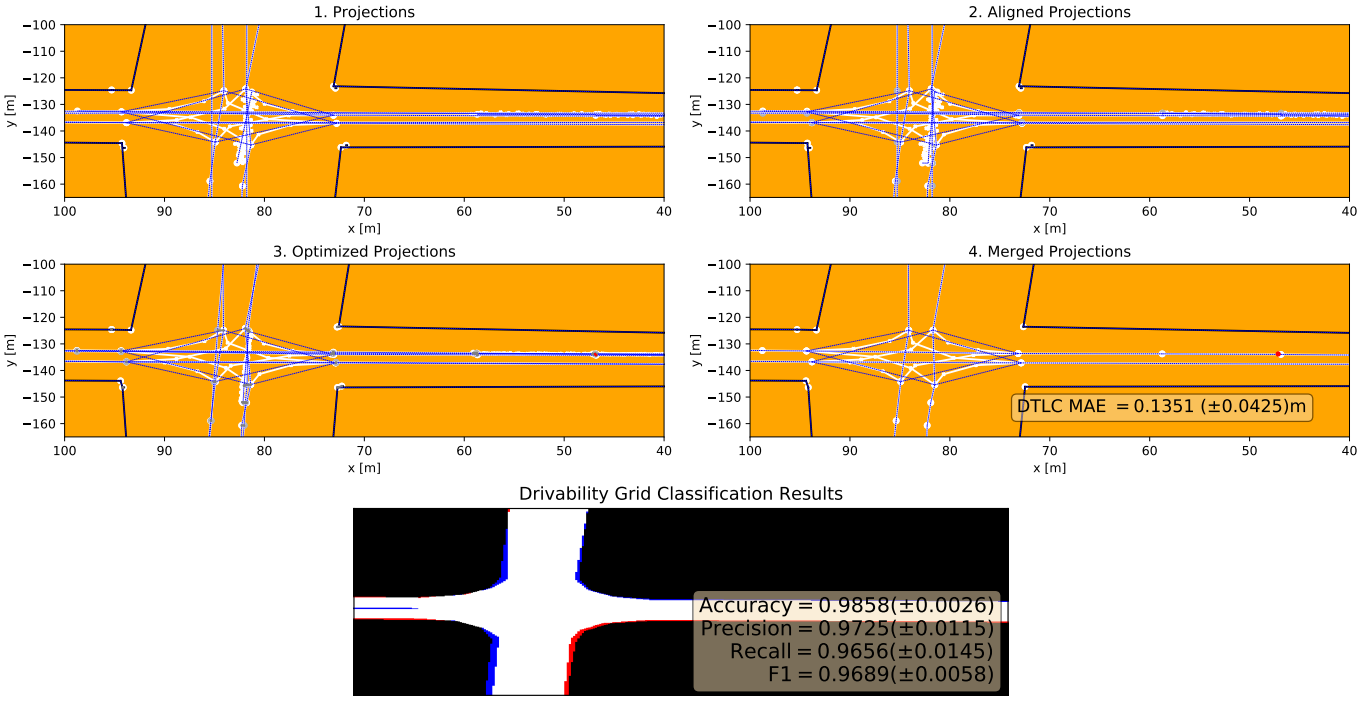
Scenario 6. Noise level 0.9. Missing input: Road network



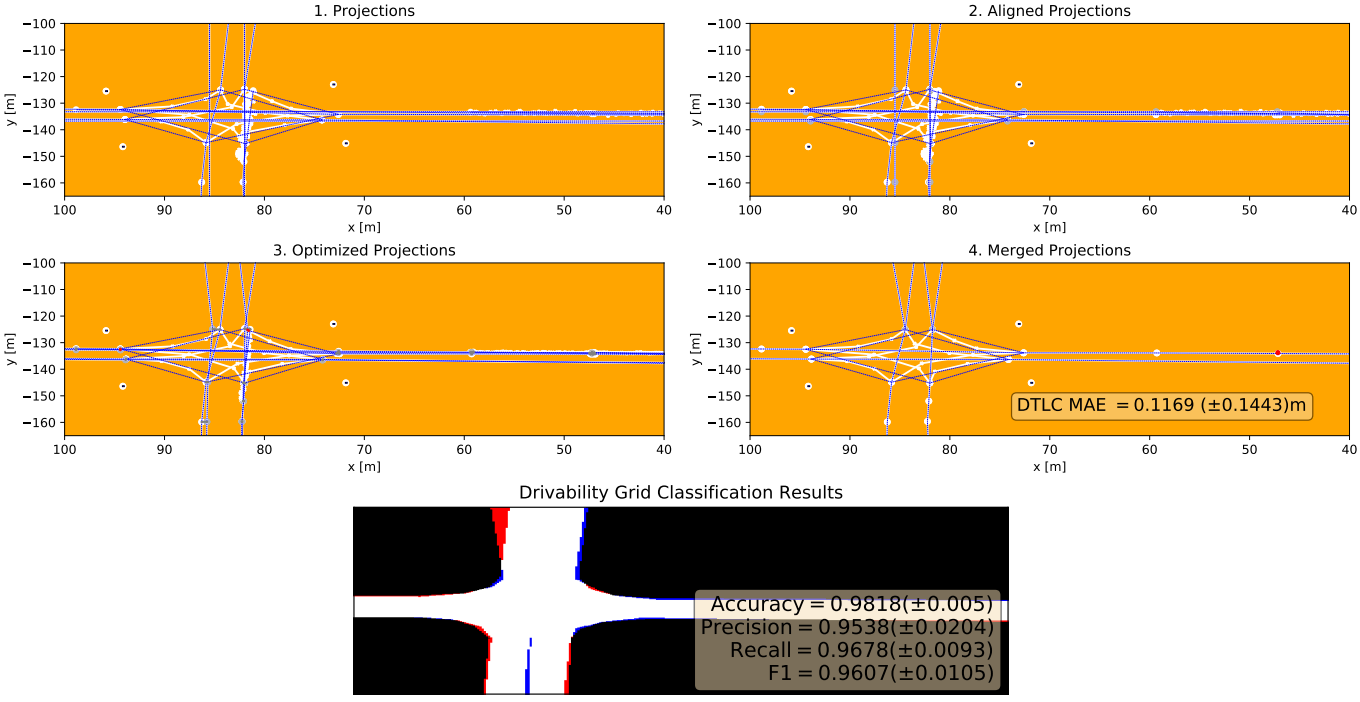
Scenario 6. Noise level 0.9. Missing input: Traffic lights



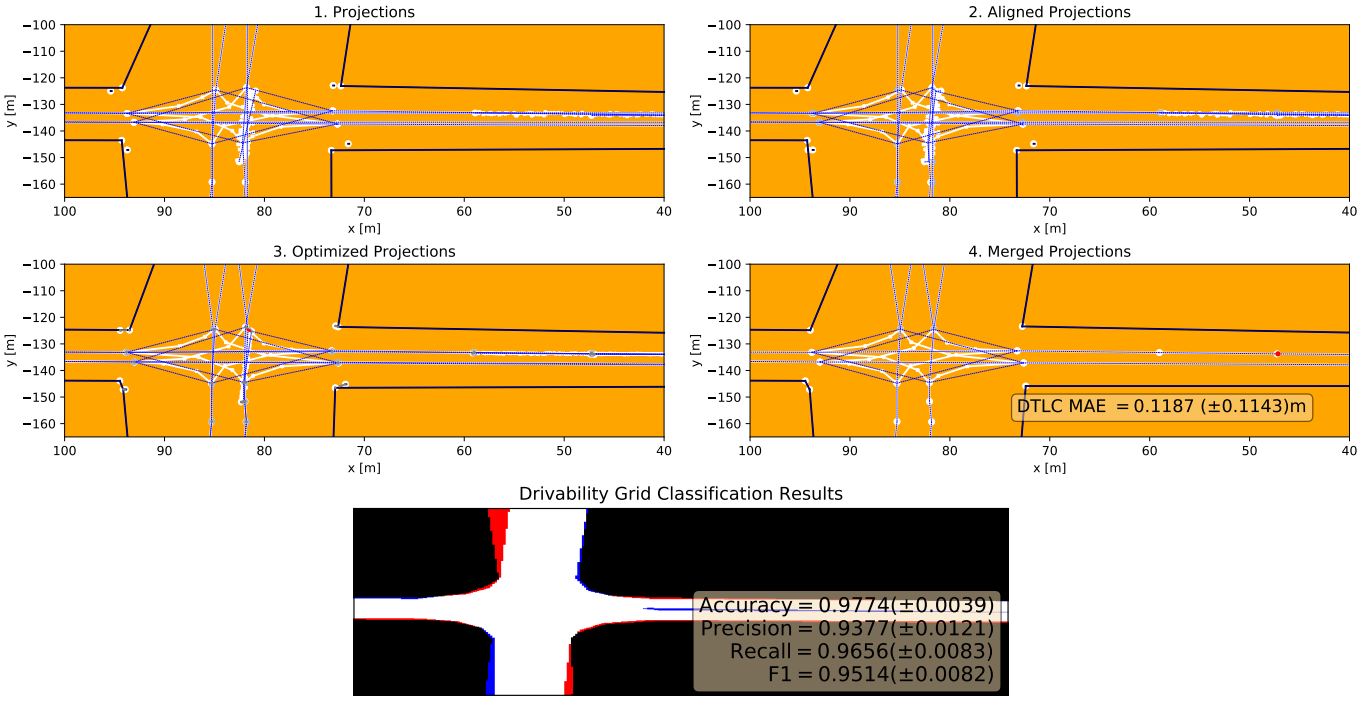
Scenario 6. Noise level 1.8. Missing input: None



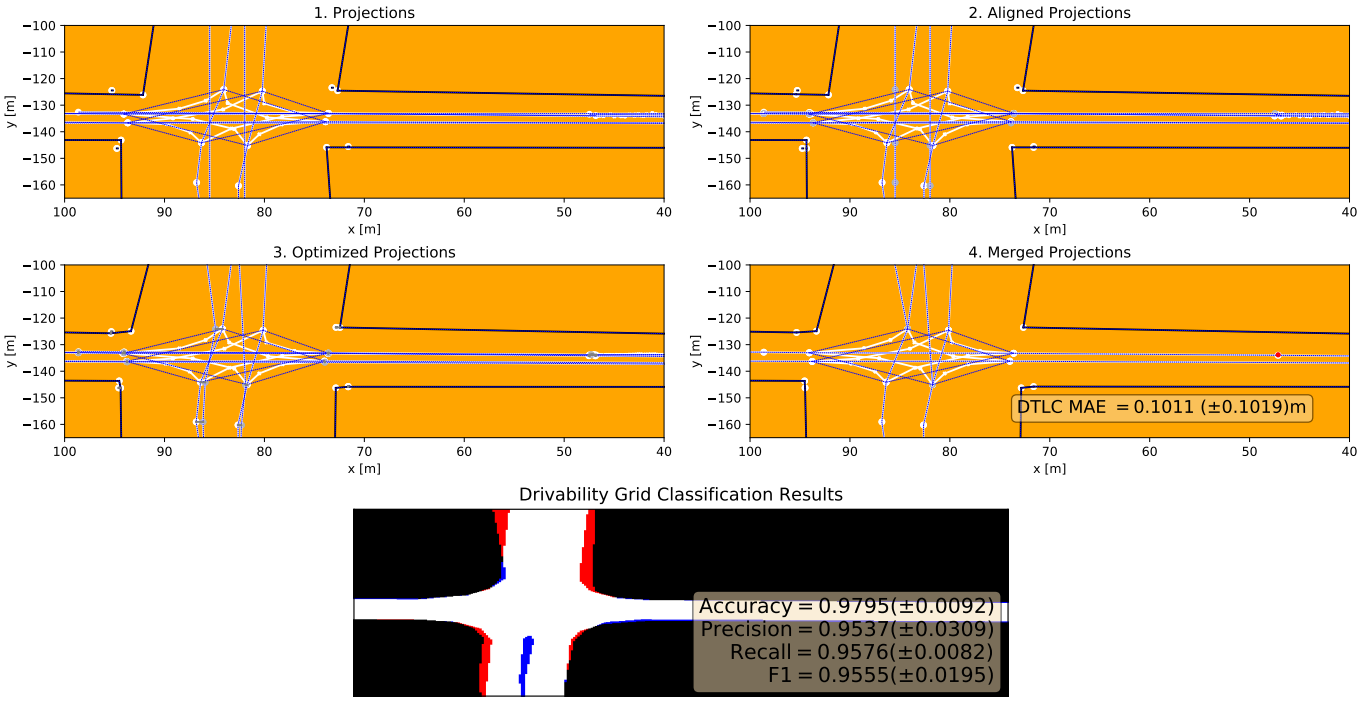
Scenario 6. Noise level 1.8. Missing input: Buildings



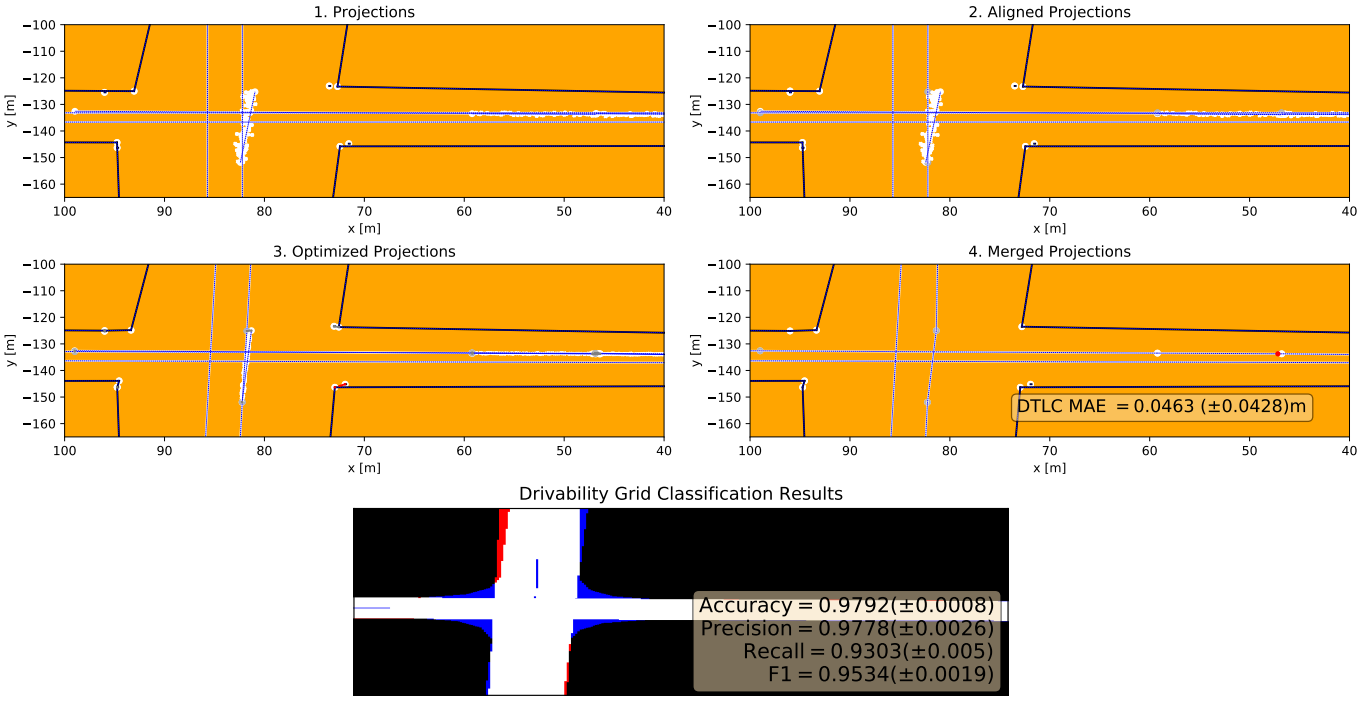
Scenario 6. Noise level 1.8. Missing input: Mobileye



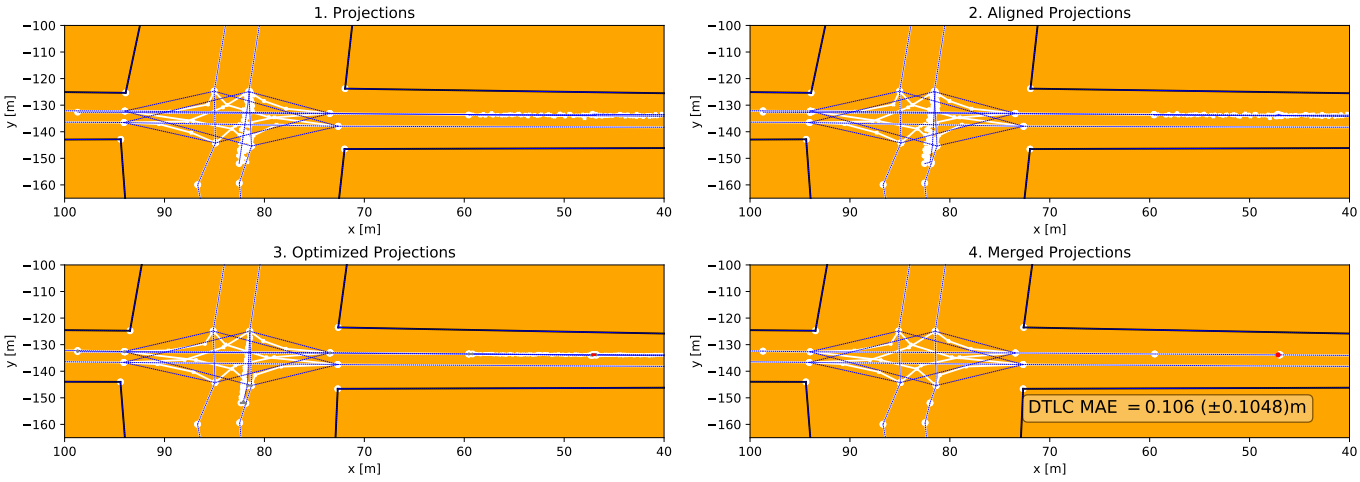
Scenario 6. Noise level 1.8. Missing input: Vehicles

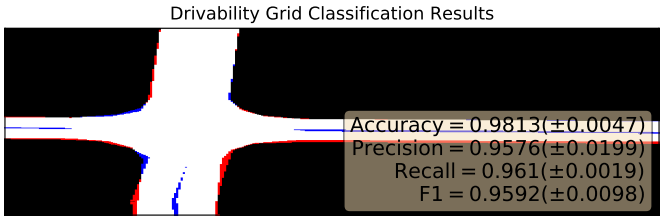


Scenario 6. Noise level 1.8. Missing input: Road network

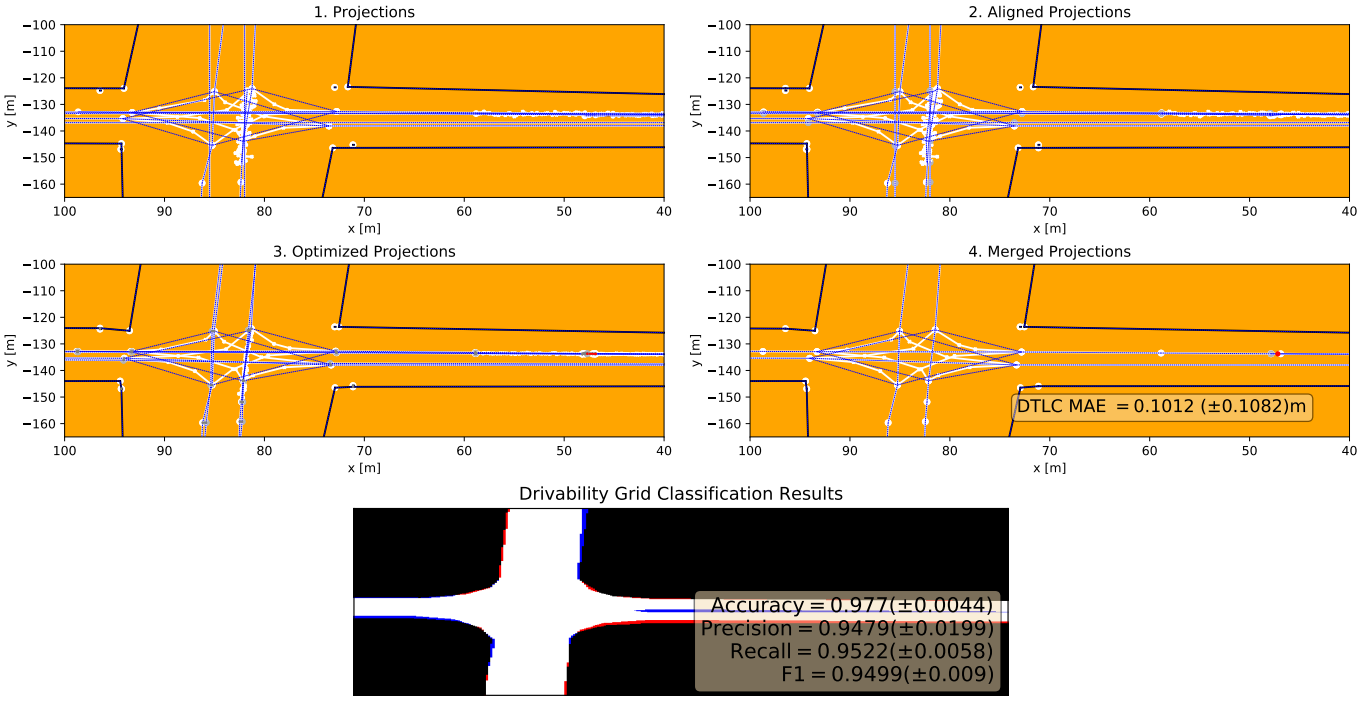


Scenario 6. Noise level 1.8. Missing input: Traffic lights

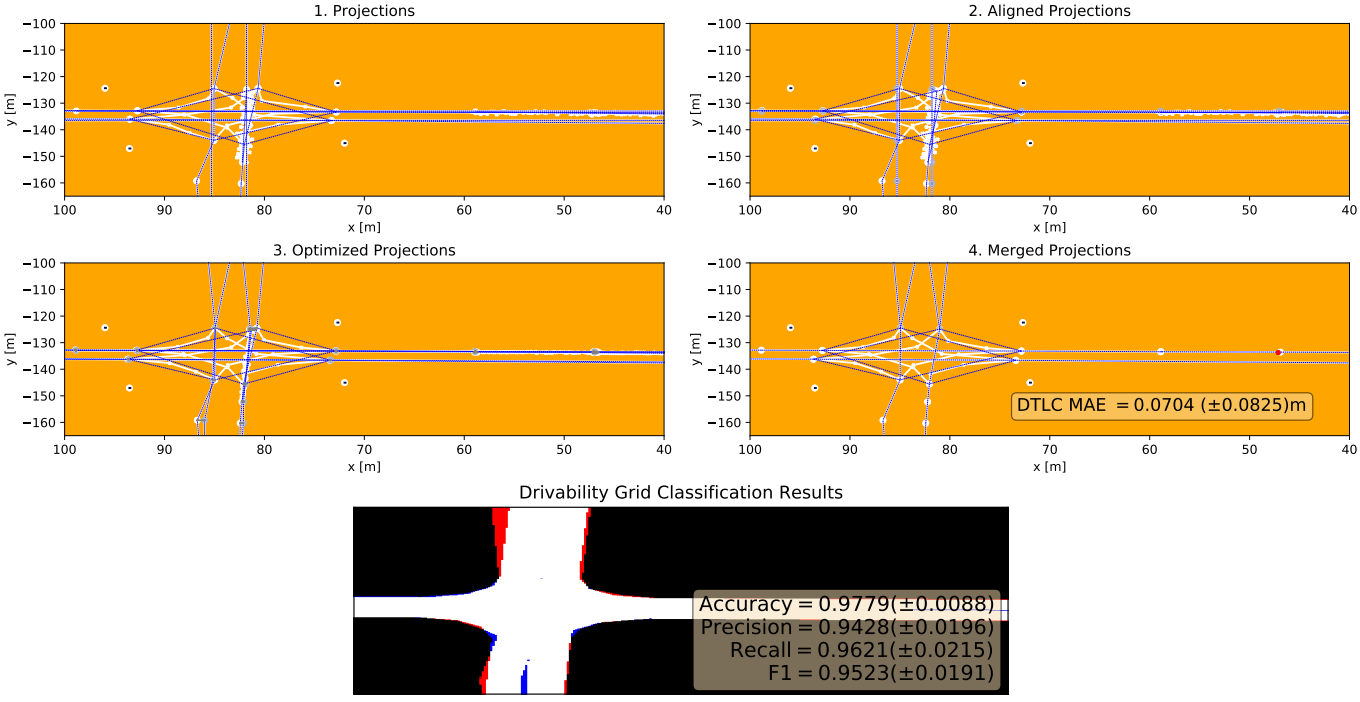




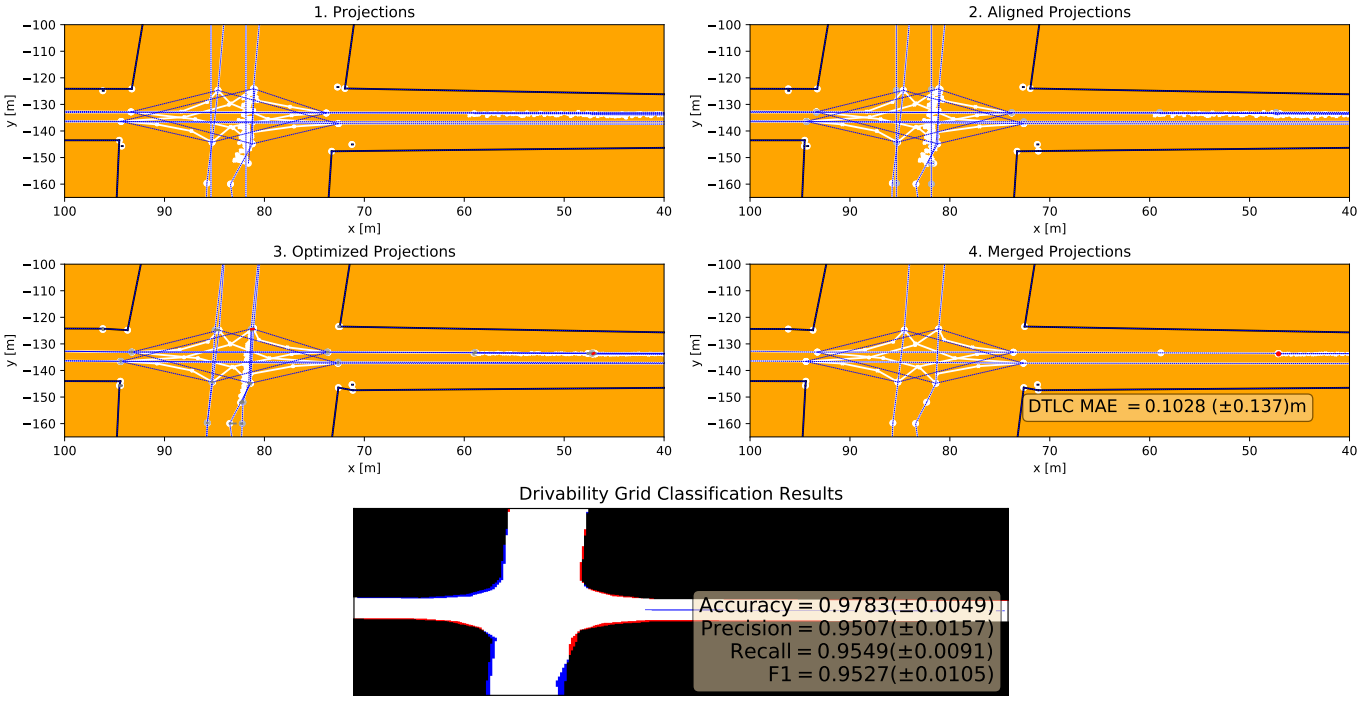
Scenario 6. Noise level 2.7. Missing input: None



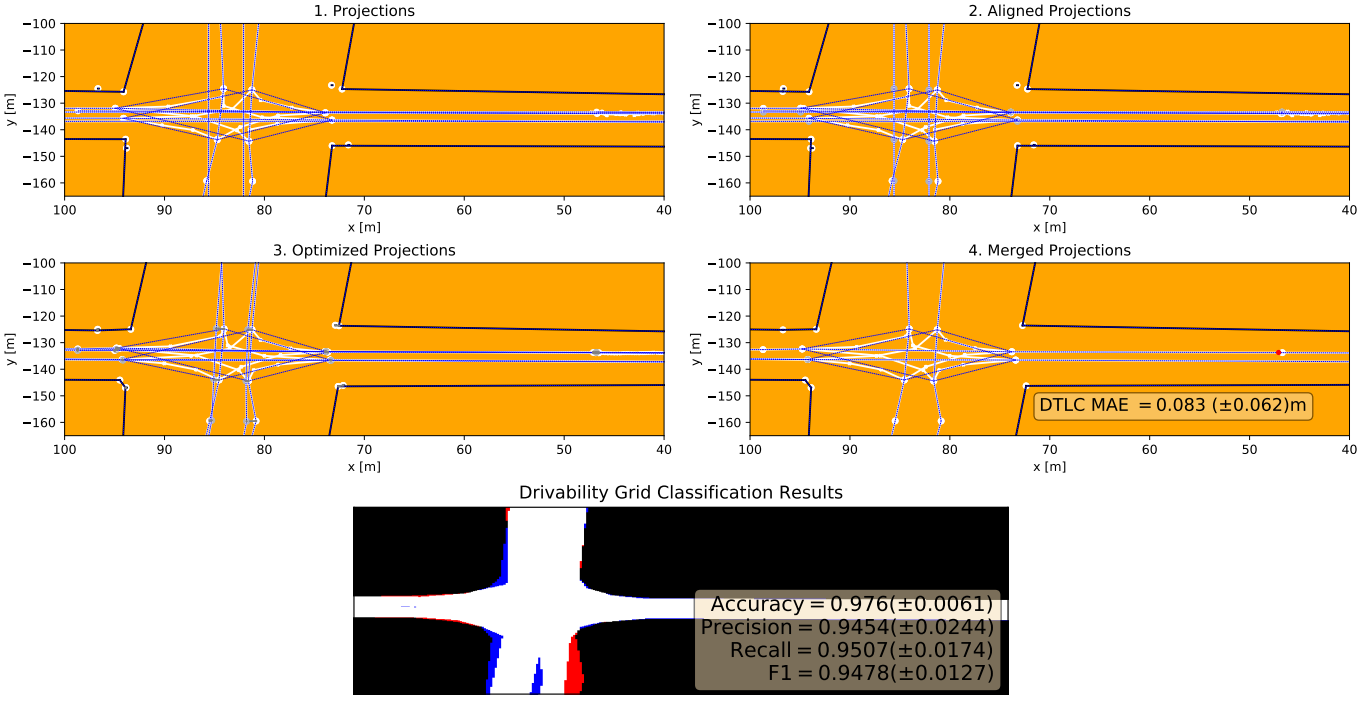
Scenario 6. Noise level 2.7. Missing input: Buildings



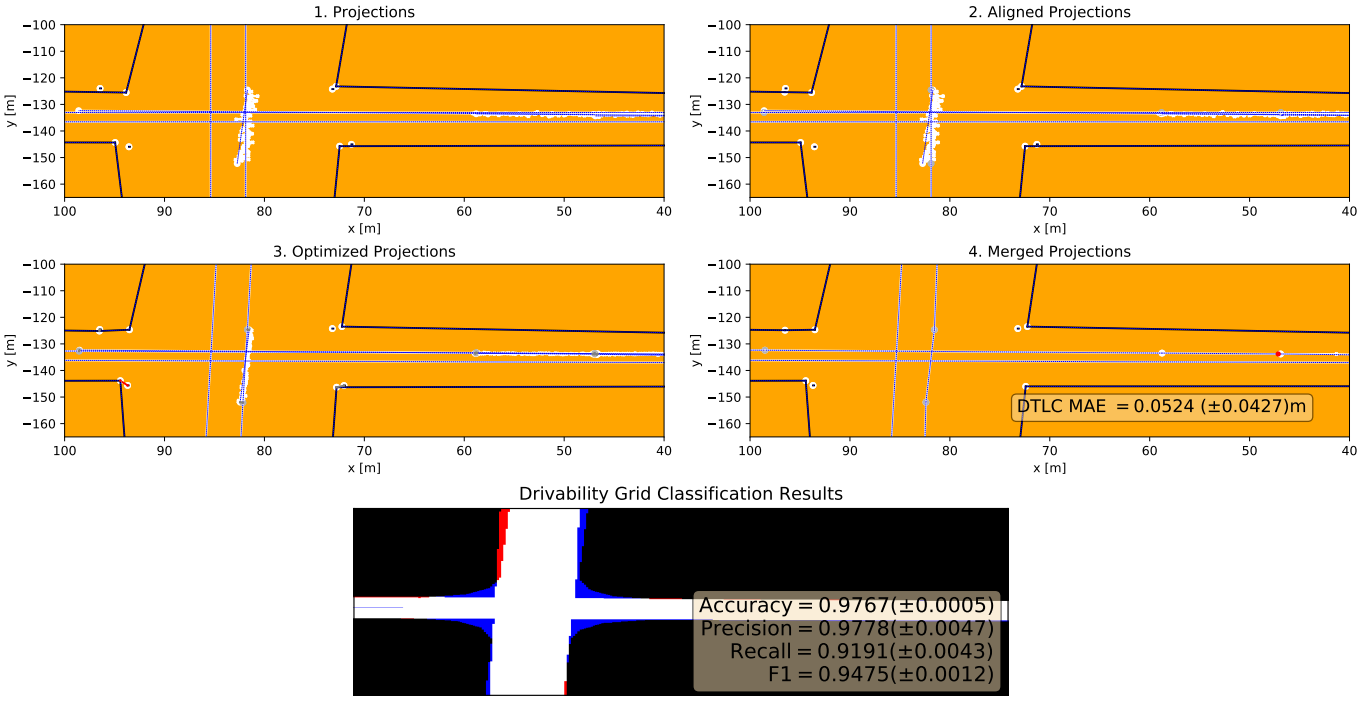
Scenario 6. Noise level 2.7. Missing input: Mobileye



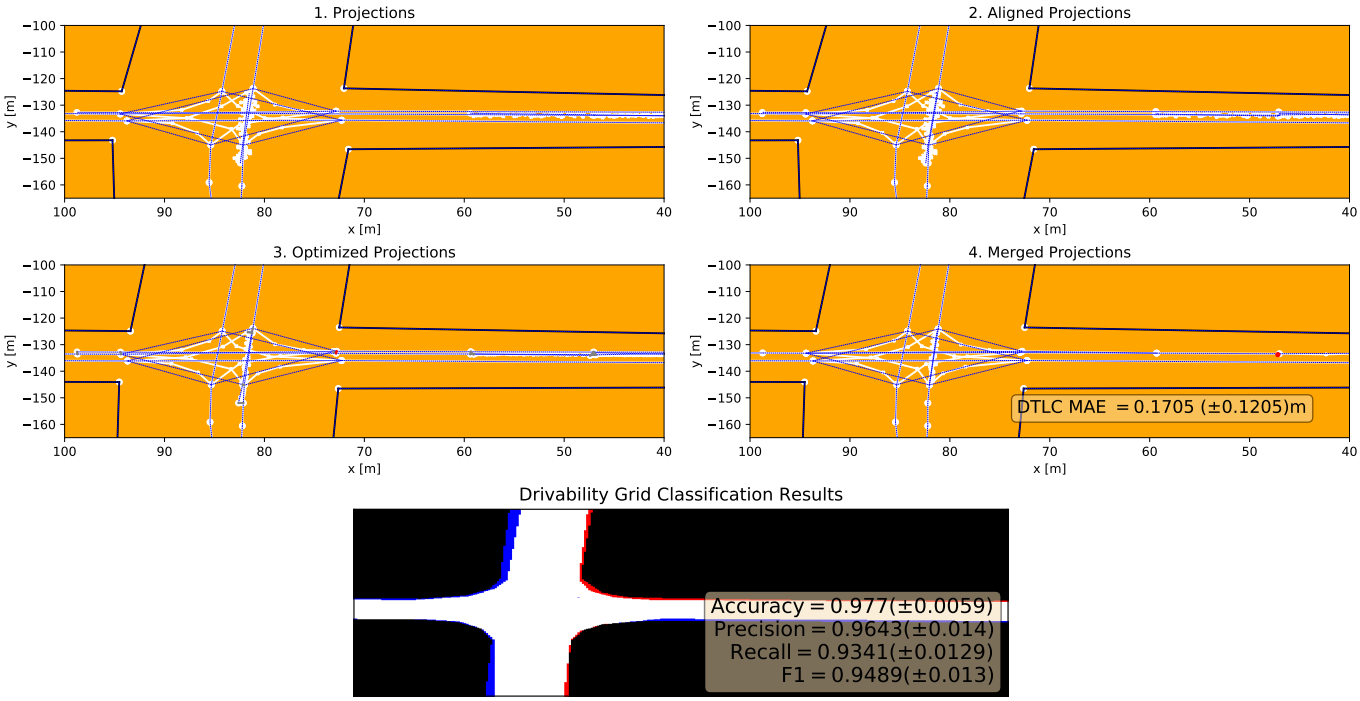
Scenario 6. Noise level 2.7. Missing input: Vehicles



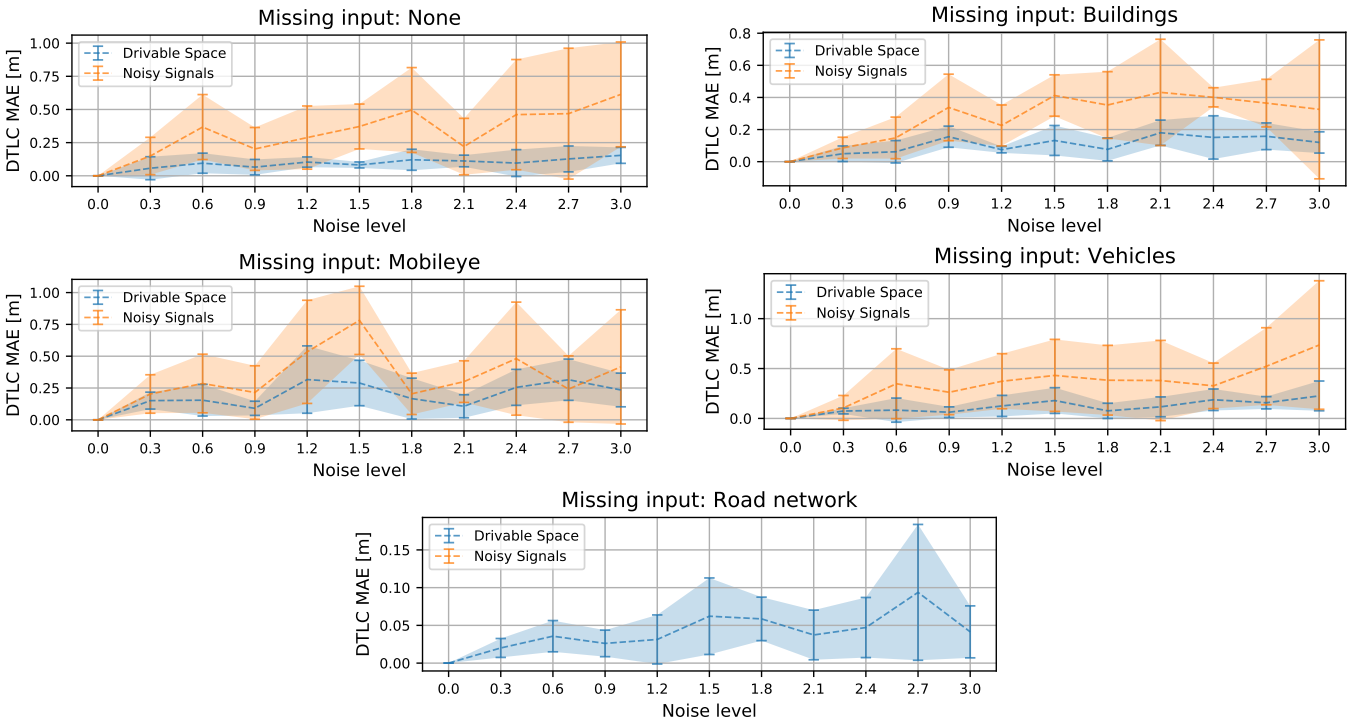
Scenario 6. Noise level 2.7. Missing input: Road network



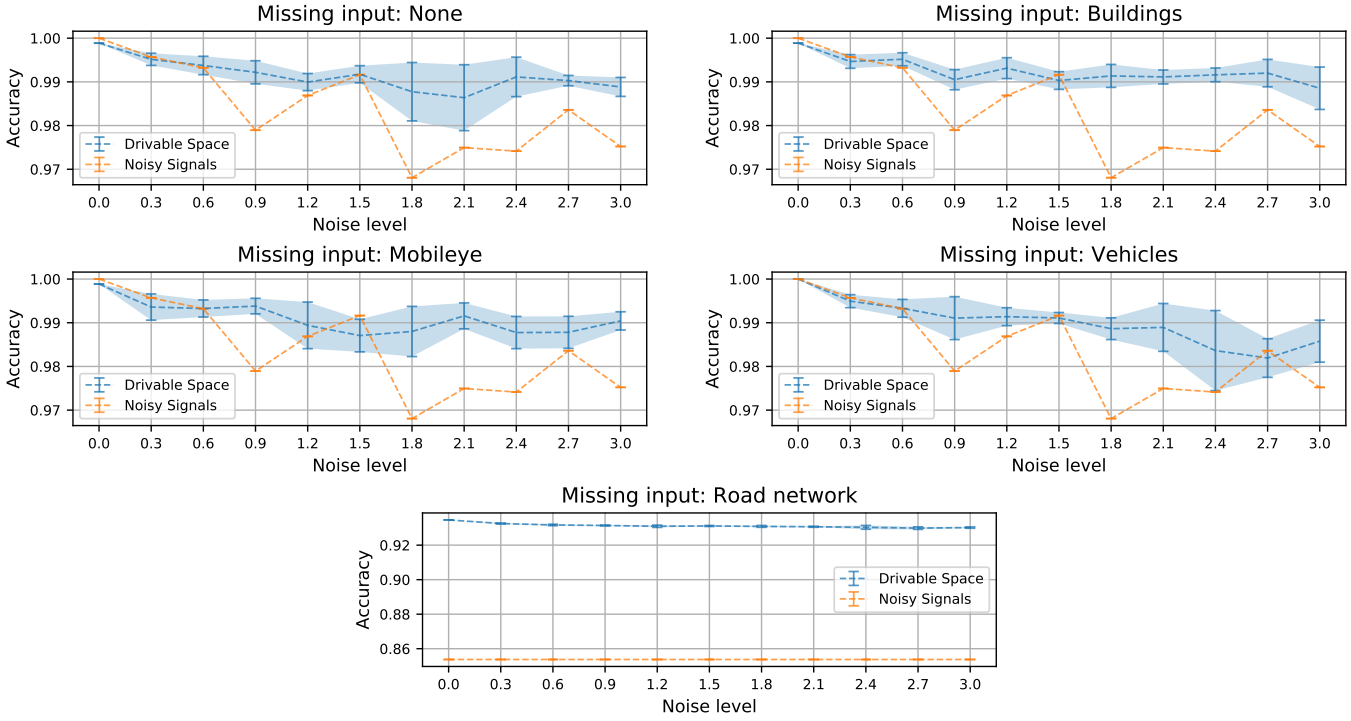
Scenario 6. Noise level 2.7. Missing input: Traffic lights



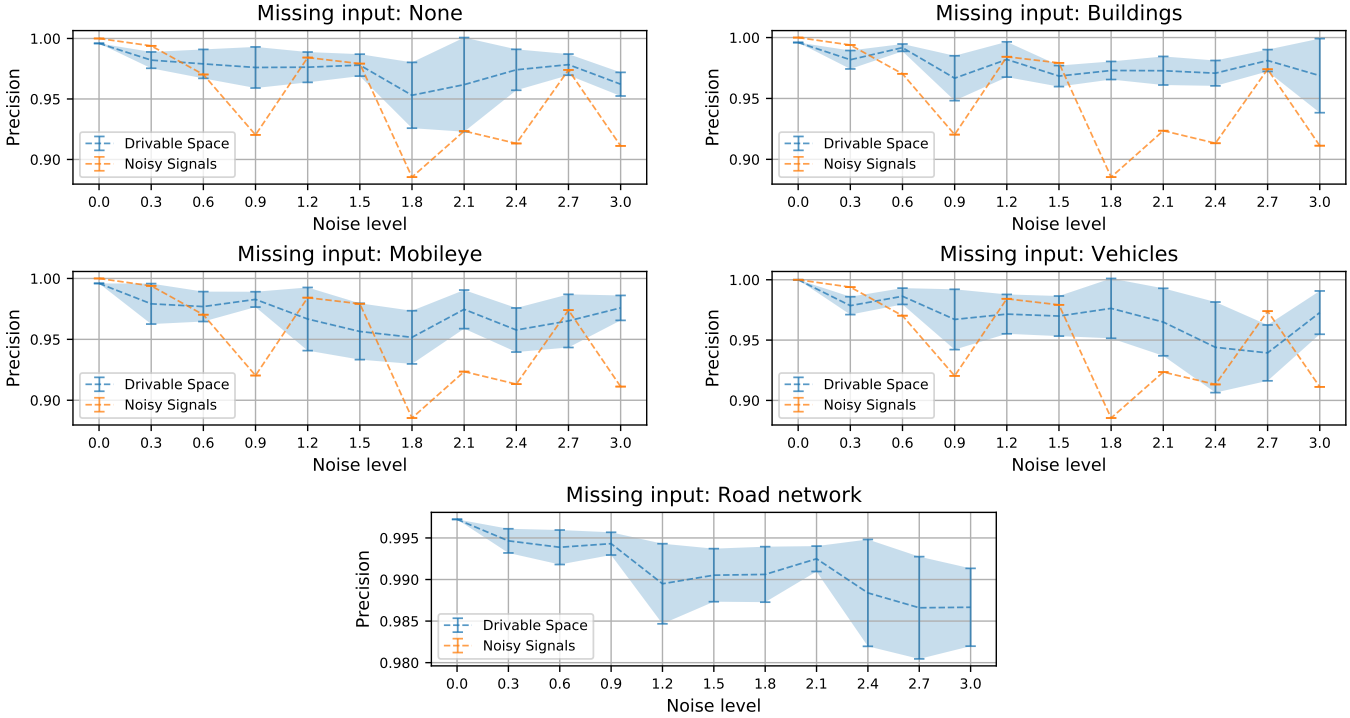
Scenario 3 - DTLC Error



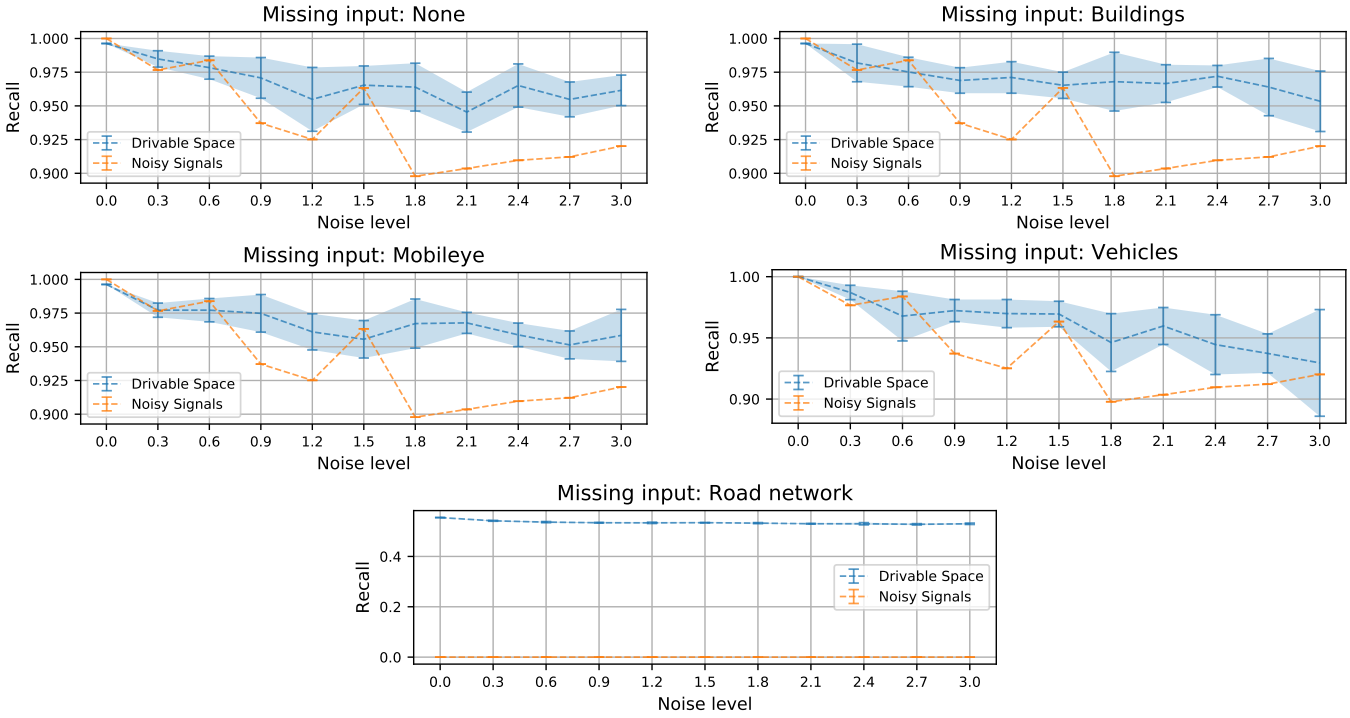
Scenario 3 - Accuracy



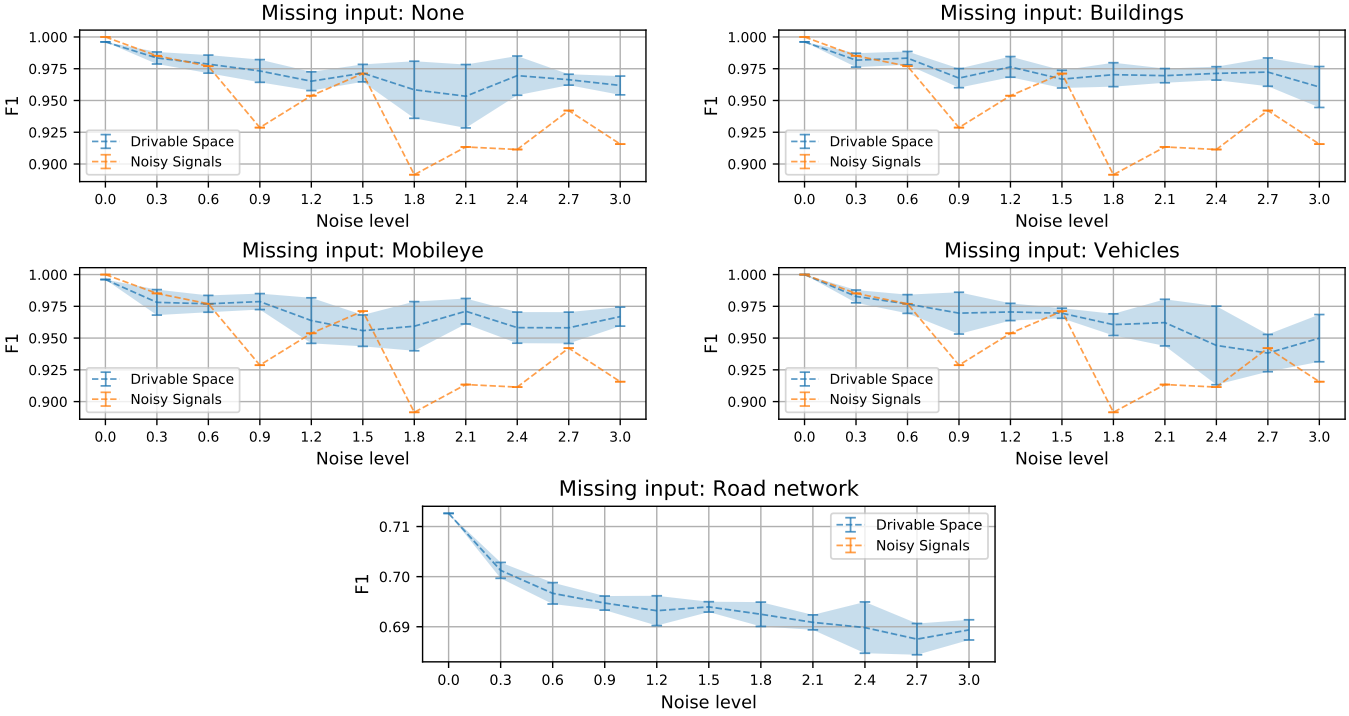
Scenario 3 - Precision



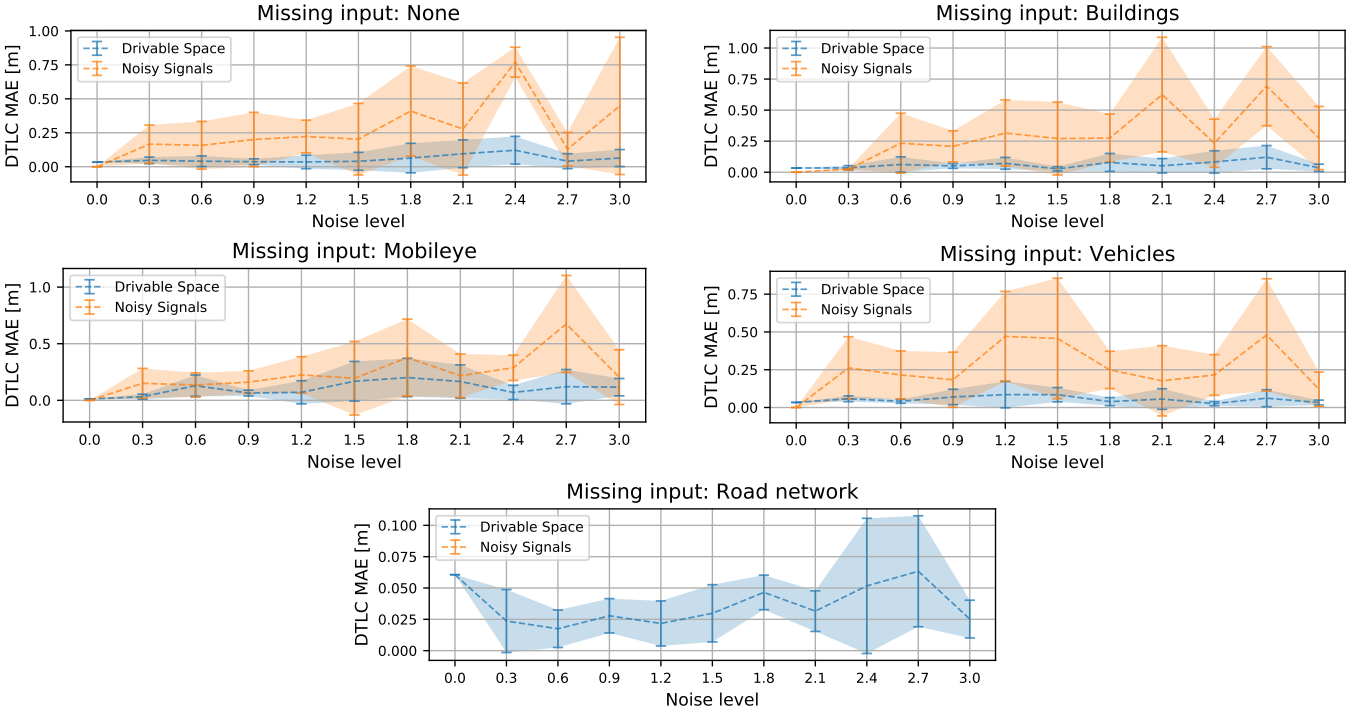
Scenario 3 - Recall



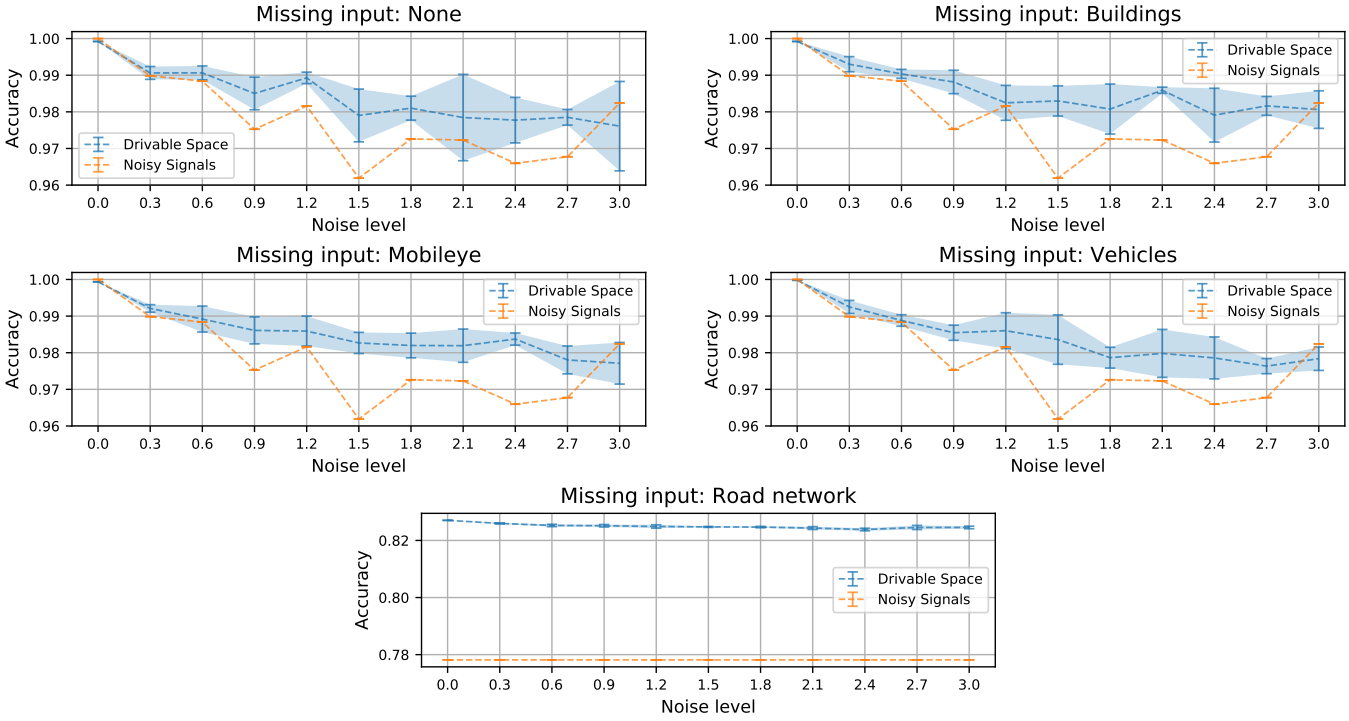
Scenario 3 - F1



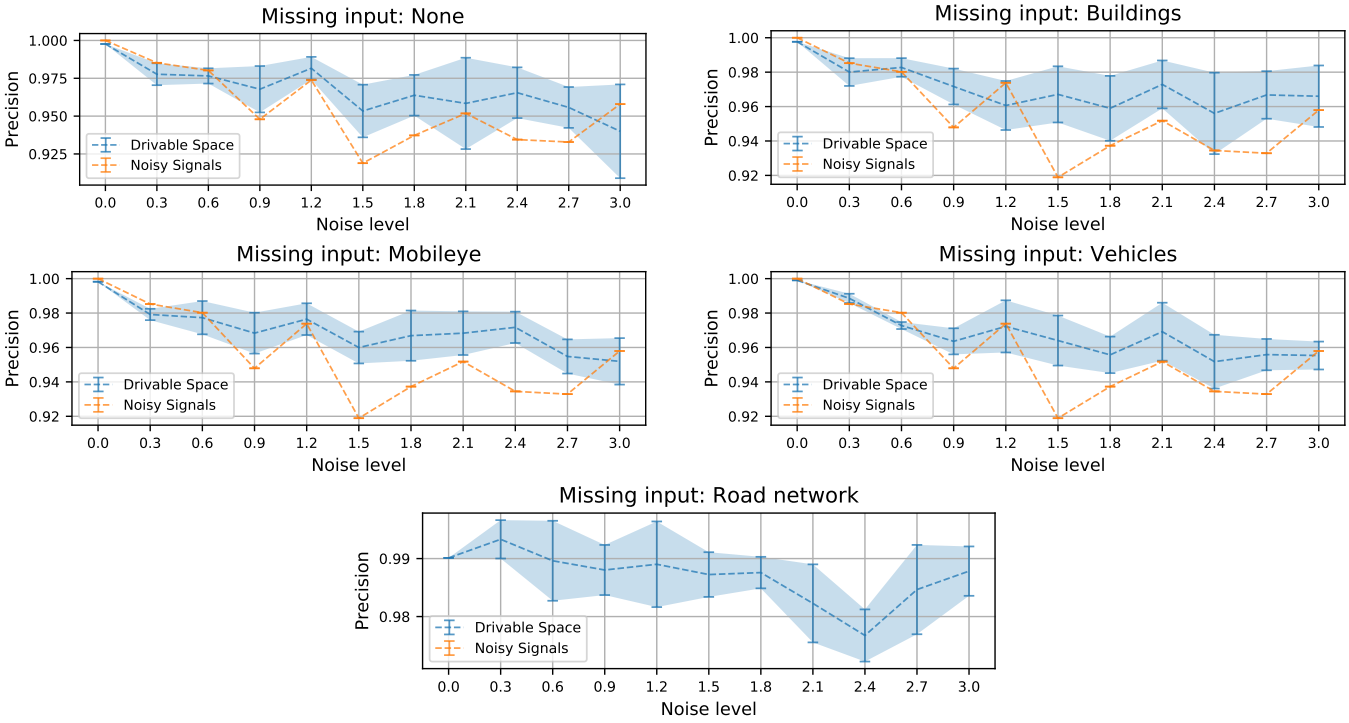
Scenario 4 - DTLC Error



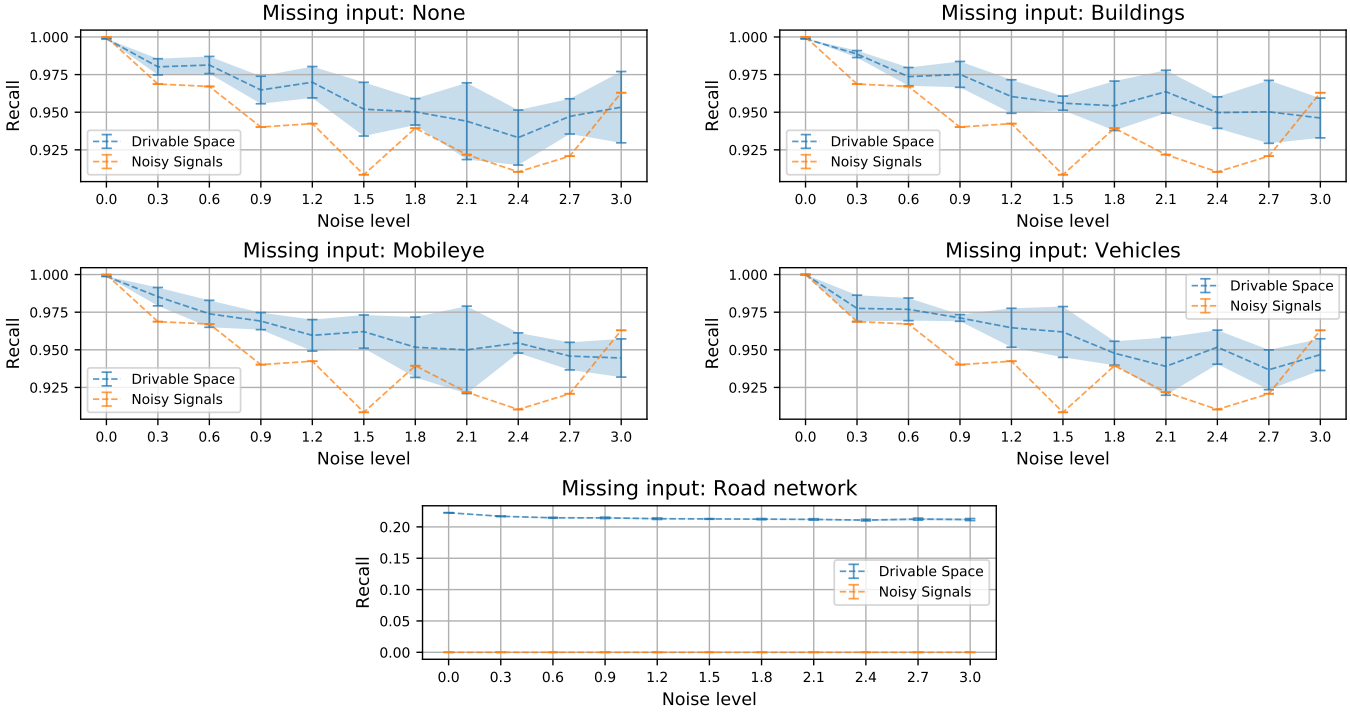
Scenario 4 - Accuracy



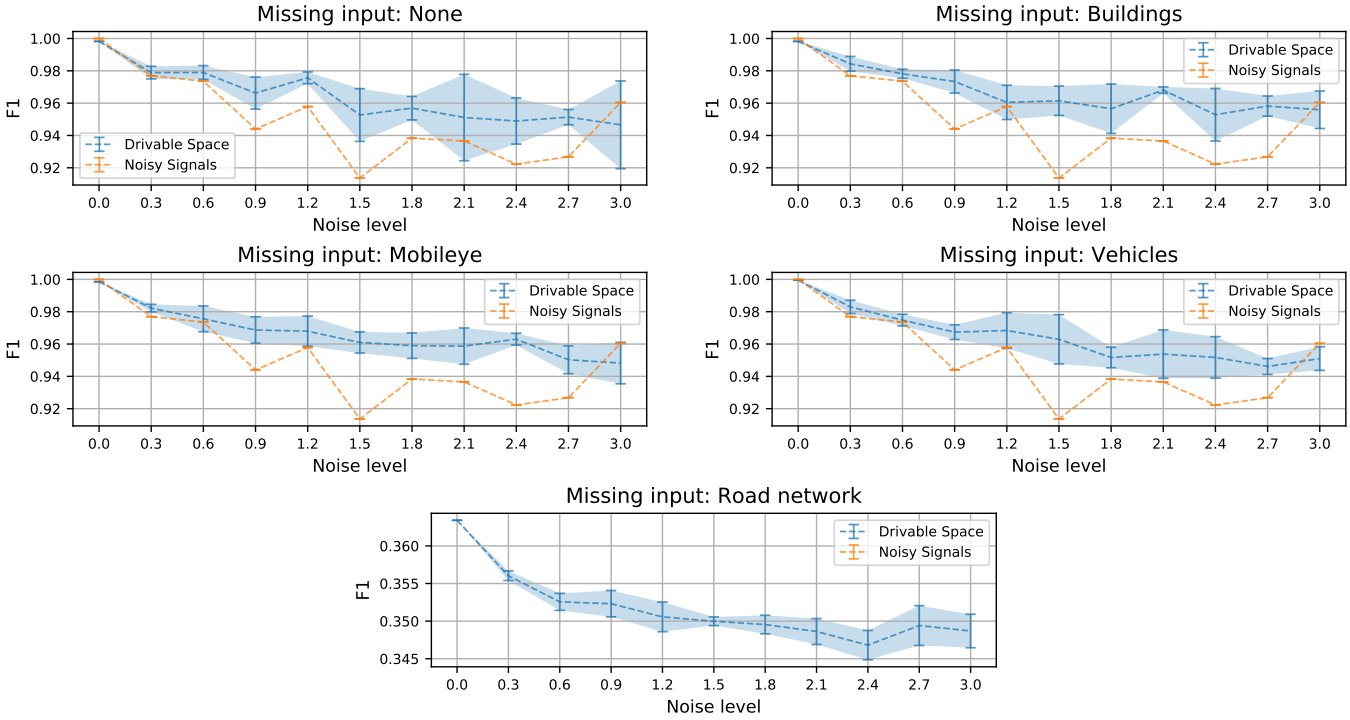
Scenario 4 - Precision



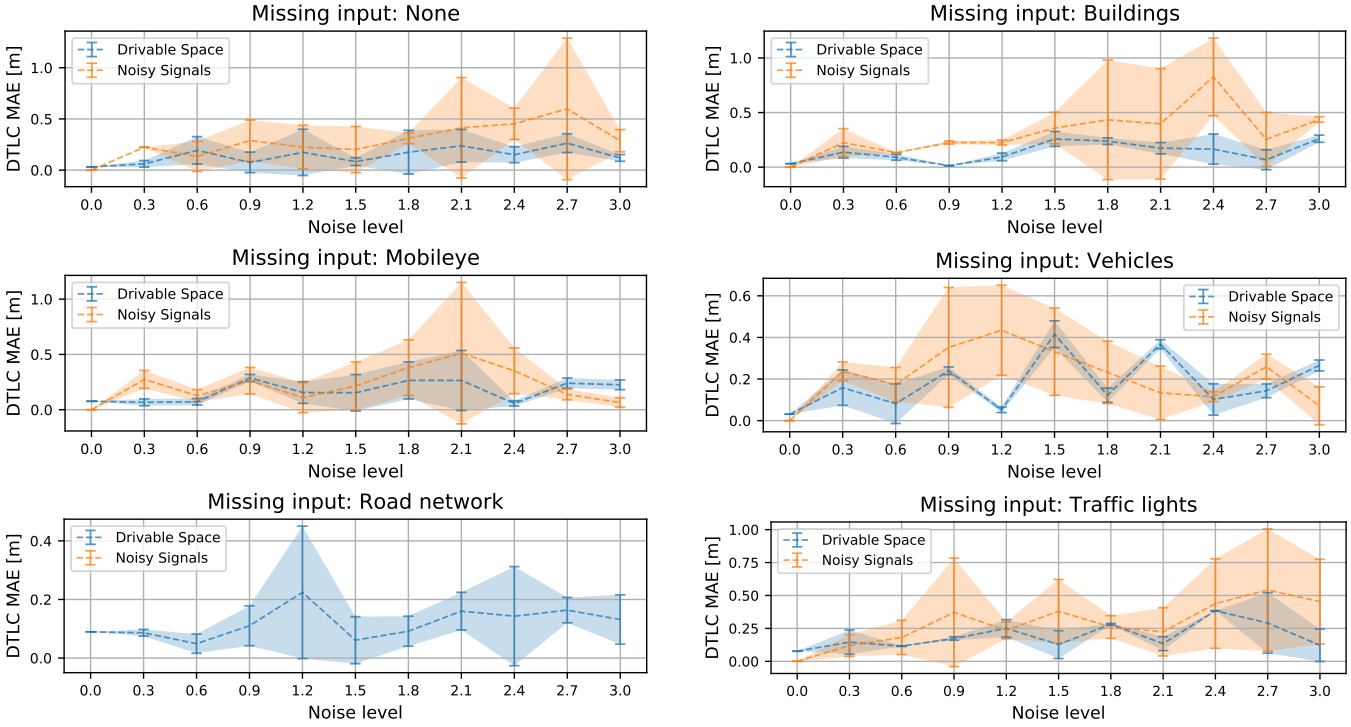
Scenario 4 - Recall



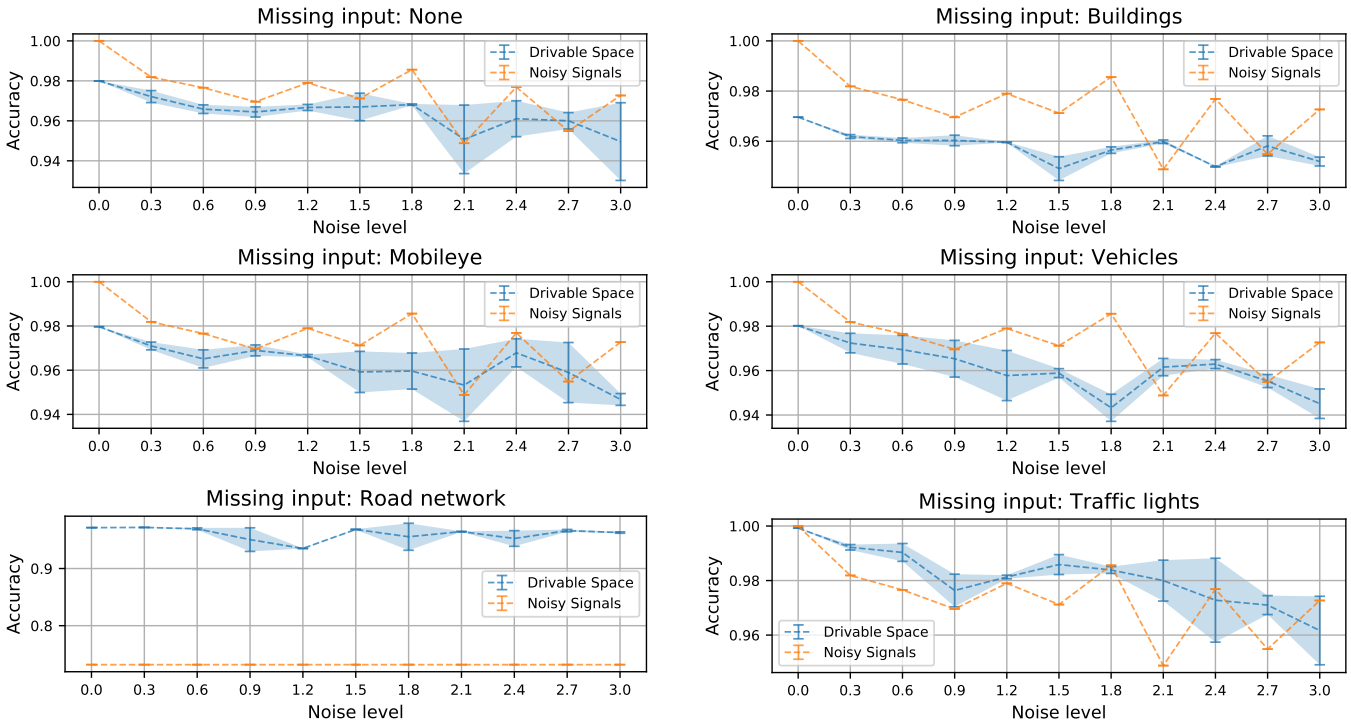
Scenario 4 - F1



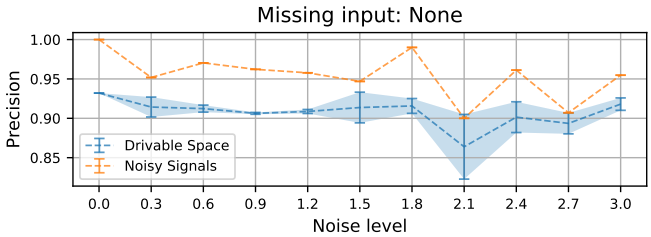
Scenario 5 - DTLC Error

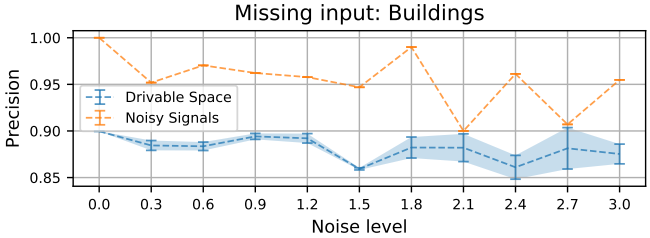


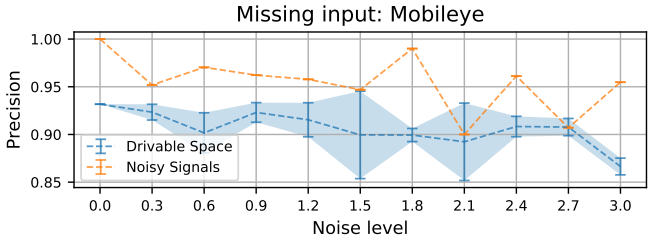
Scenario 5 - Accuracy

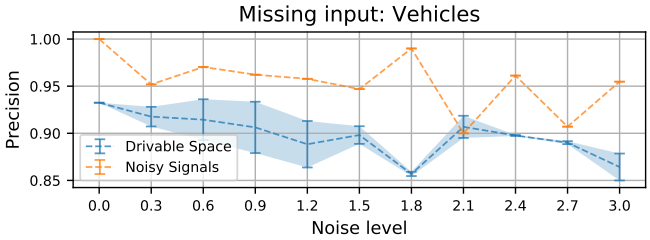


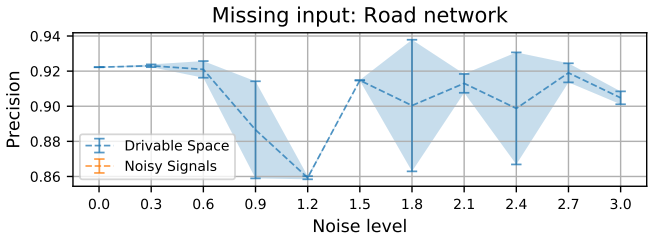
Scenario 5 - Precision

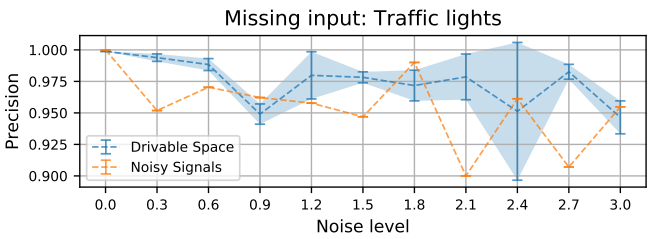




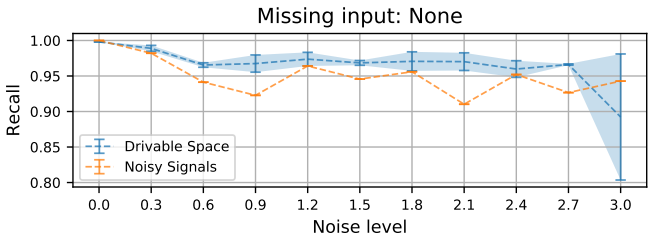


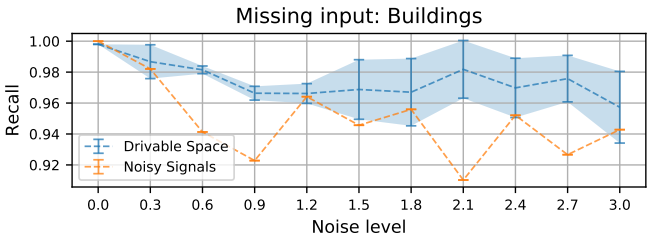


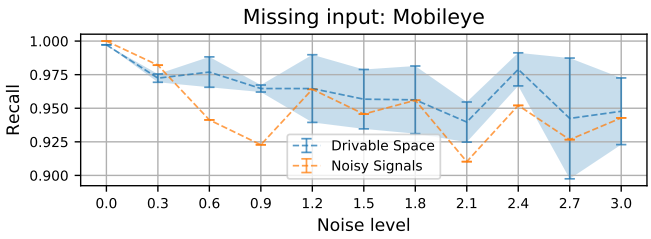


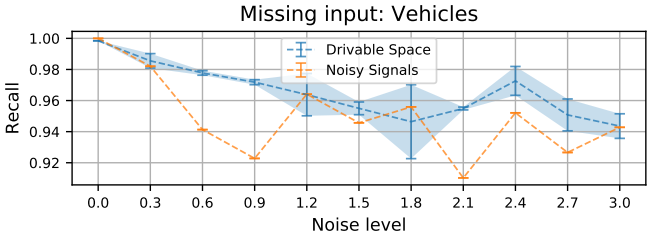


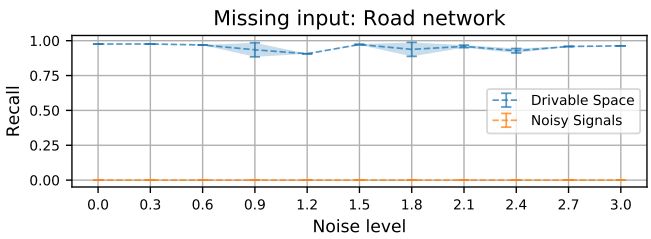
Scenario 5 - Recall

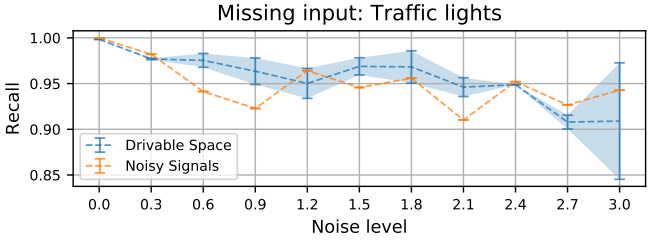




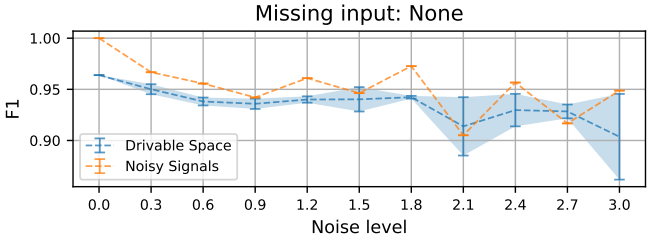


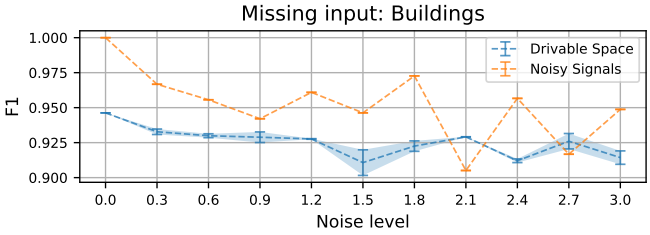


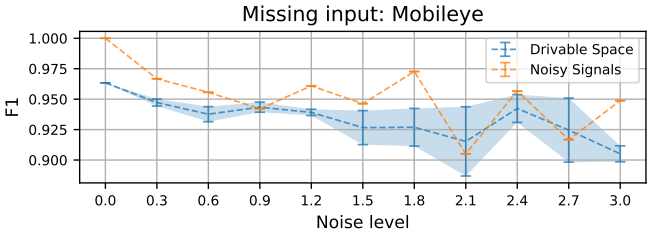


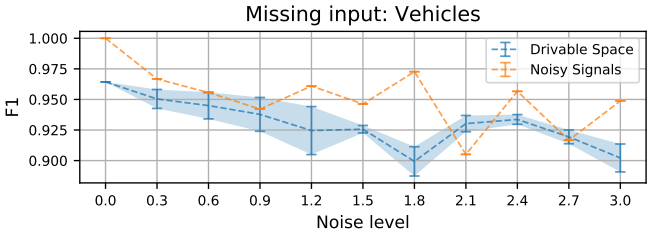


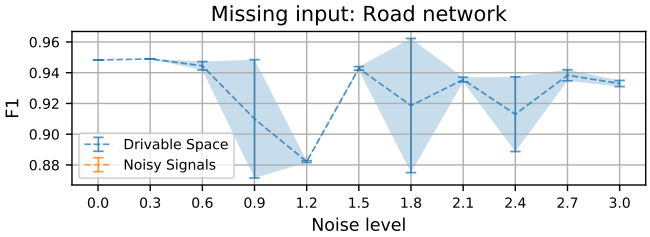
Scenario 5 - F1

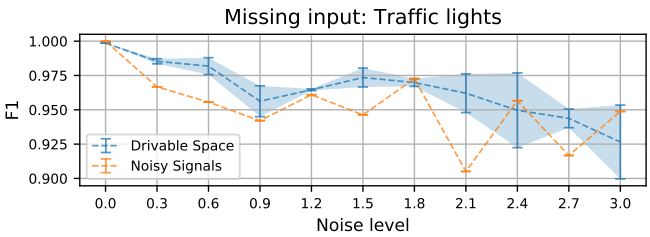




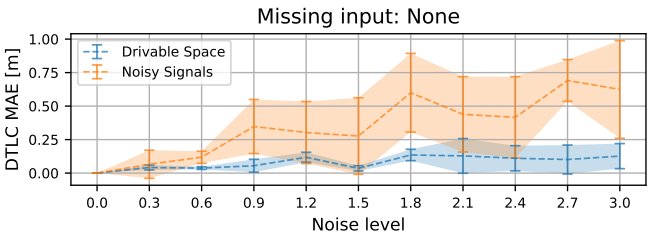


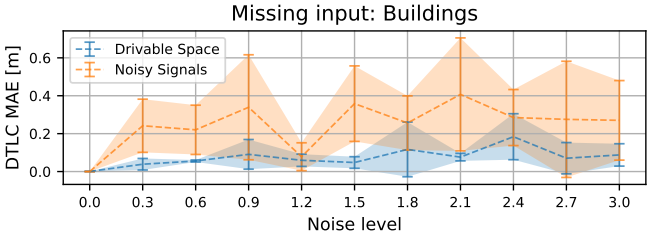


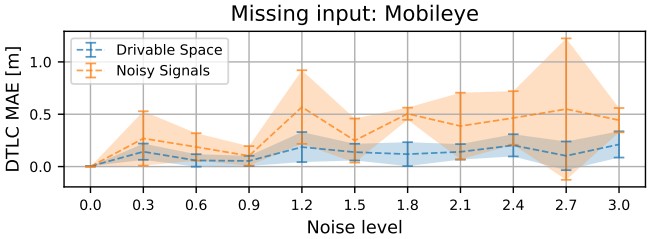


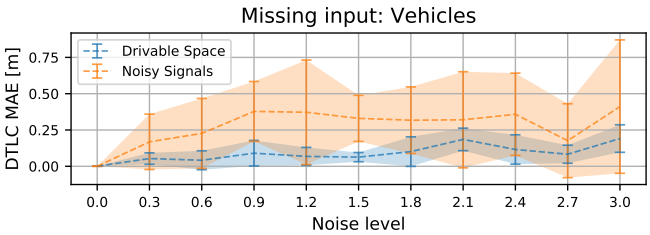


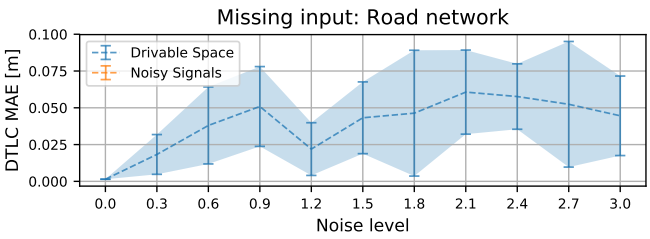
Scenario 6 - DTLC Error

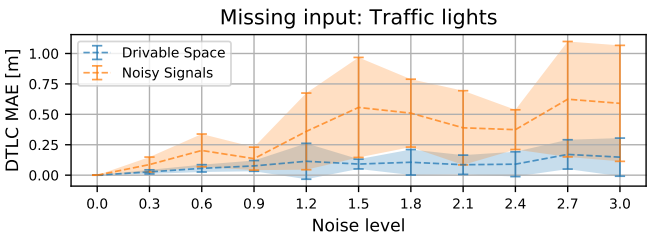




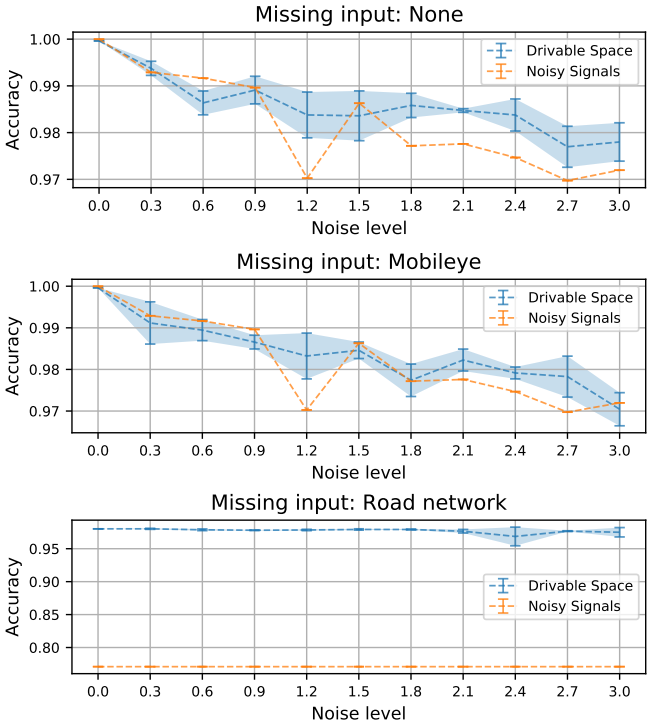


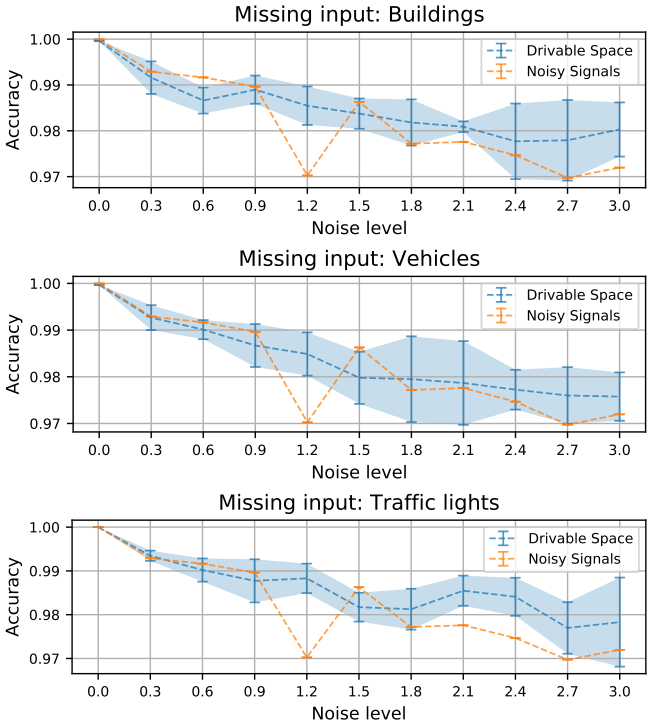




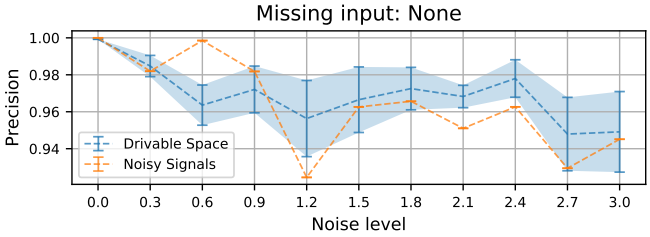


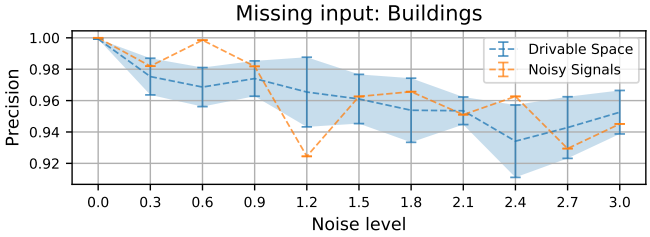
Scenario 6 - Accuracy

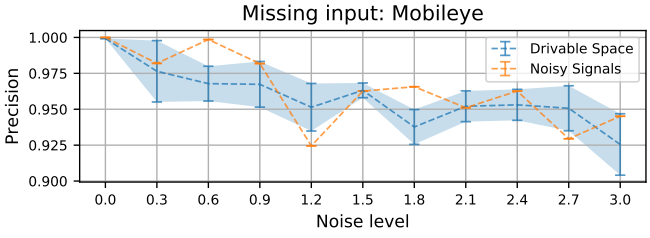


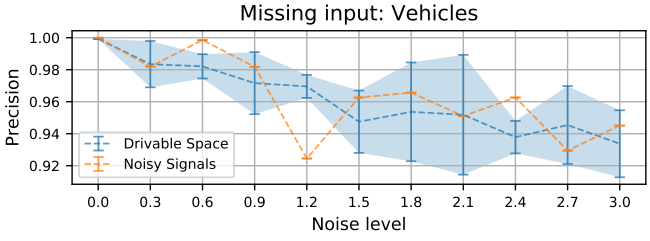


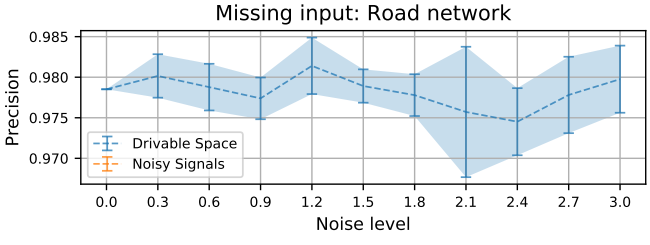
Scenario 6 - Precision

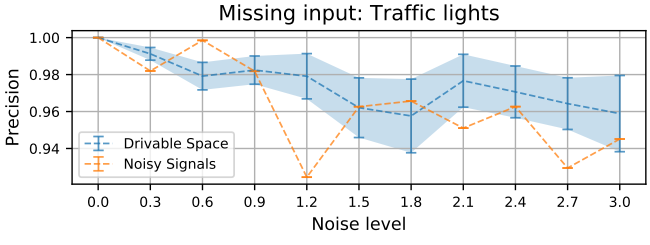




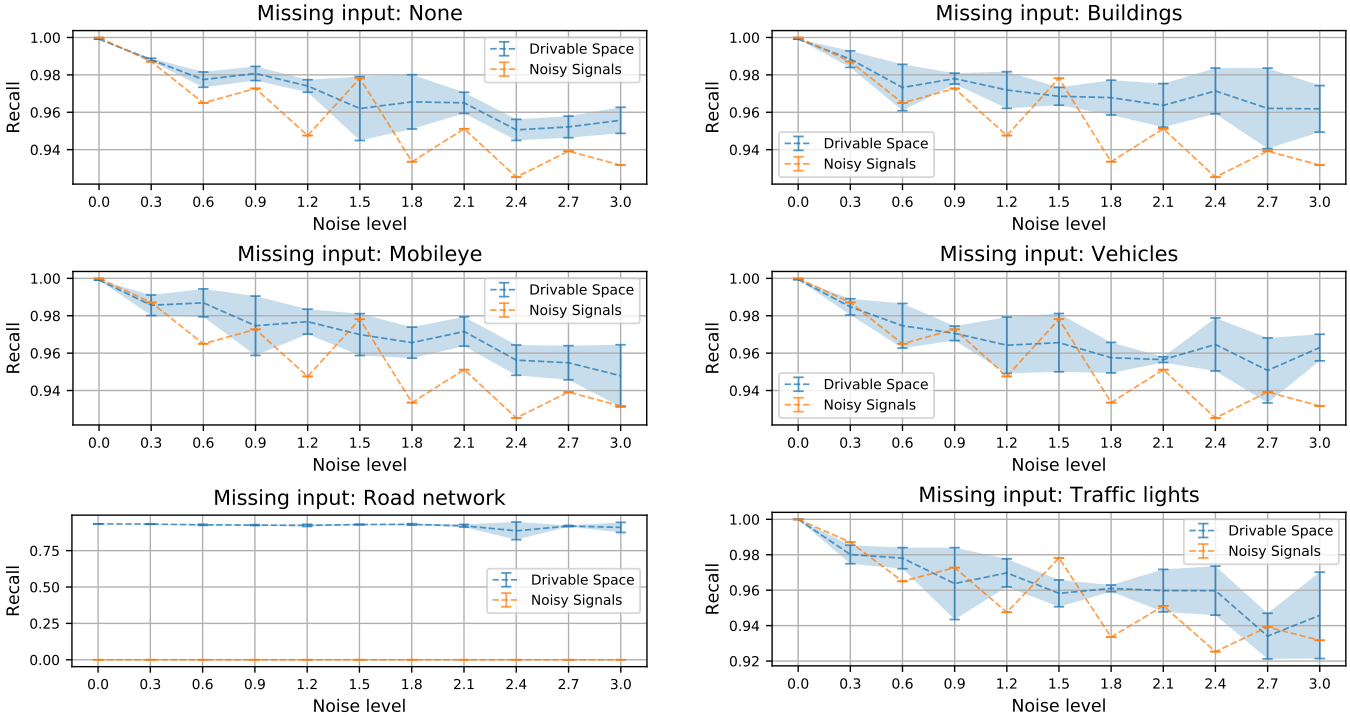








Scenario 6 - Recall



Scenario 6 - F1