EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

3D mapping and change detection for patrolling mobile robots

Dr. Zoltán Istenes
Associate Professor at ELTE

Levente Göncz
Computer Science Student

Quan Zhou
Associate Professor at Aalto University

András Majdik
Senior Research Fellow at SZTAKI

ACKNOWLEDGMENTS

The research in this thesis, which was carried out by Levente Göncz in collaboration with SZTAKI, was supported by the Ministry of Innovation and Technology and the National Research, Development and Innovation Office in the framework of the National Laboratory for Artificial Intelligence.

This research would not have been possible without the help of Dr. Zoltán Istenes, Associate Professor in the Faculty of Informatics at Eötvös Loránd University (ELTE) and András Majdik, Senior Research Fellow in the Machine Perception Research Laboratory at Institute for Computer Science and Control (SZTAKI).

Appreciation is also due to the Digital Master School of the European Institute of Innovation & Technology (EIT) programme and to the Institute for Computer Science and Control (SZTAKI) for making this thesis possible.

Furthermore, the thesis could not have been accomplished without the help of Eötvös Loránd University (ELTE) and Aalto University.

TABLE OF CONTENTS

1. Introduction

As technology is evolving, mobile robots are becoming more and more popular. Thanks to the commercialization of camera enabled smart systems, a large variety of applications have become more accessible, e.g: traffic and construction monitoring, pedestrian tracking, security control. Furthermore, the expanding fields of deep learning and simultaneous localisation and mapping allow developers to combine computer vision and artificial intelligence to create more complex systems. The researcher Andrew Davison defined the name of this new level of evolution as Spatial AI [4].
The ZED 2 stereo camera was built for the purpose of promoting and advocating the concept of Spatial AI. It is the first to combine 3D sensing with AI to create next-generation spatial intelligence. It was specifically designed for autonomous navigation and mapping tasks. It is the first stereo camera that has built-in neural networks to help and expedite the development of computer vision perception systems. One of the main goals of the camera is to *"Make your robot see with a few lines of code"* [3].

The goal of this thesis is twofold: First, the capabilities of the ZED 2 stereo vision camera are to be assessed to determine whether the camera is suitable to use in real-time applications. Second, the thesis investigates the possibility of implementing a real-time change detection system using the camera.

The ZED 2 stereo camera combines many technologies in order to be able to contribute to complex solutions. In general, stereo cameras use two sensors to obtain two different views on a scene to simulate human vision and estimate the disparity map of the scene [2]. The accuracy of the depth estimation of an object or a scene depends highly on its distance from the viewpoint. The ZED 2 stereo camera has a reported depth range of 20 meters and a depth accuracy of $< 1\%$ up to 3 meters and $< 5\%$ up to 15 meters [1]. It is important to measure more accurately the depth precision and depth measurement consistency of the camera to determine the usefulness and usability of the device for patrolling.
The visual SLAM (simultaneous localisation and mapping) system allows the camera to provide an accurate 3D representation of its environment while being able to continuously locate itself in it. However, the manufacturer does not report any information on the camera's SLAM accuracy. Therefore, in the thesis, the accuracy of the ZED 2 visual SLAM algorithm is estimated and evaluated.
It is also important to measure the special relocalization ability of the ZED 2 stereo camera that enables developers to share a fixed reference frame between different recordings. The camera utilizes a so-called area memory for this task, which mimics human memory and incrementally stores the spatial information of the environment in a map [3]. This feature is key for fast change detection applications, because it enables the camera to locate itself in an already visited area and detect changes more easily. The accuracy of this area map is also measured in this thesis.
The ZED 2 stereo camera is capable of 2D and 3D object detection. The camera is able to use its proprietary object detector or can be interfaced with many well known frameworks to detect objects of interest, such as PyTorch, TensorFlow or YOLO. However, the object

detector of the ZED 2 stereo camera has no performance values associated with it. Therefore, in the thesis, it is going to be determined by a qualitative assessment whether the ZED object detector is able to provide fast and reliable object detection.

Following the assessment of the ZED 2 stereo camera, various change detection approaches will be considered for implementation. These change detection algorithms are presented in the section called *Research on 3D change detection*. The goal is to develop a change detection application that works on the level of objects and is able to detect changes real-time between multitemporal predefined patrol routes. For this task, 3D semantic object maps are reviewed and considered for using in the change detection application.

2. State of the art

In this section, the key concepts appearing in this thesis are described, with an emphasis on spatial AI, object detection, 3D change detection, camera model and stereo vision. Furthermore, the main attributes of the ZED 2 stereo camera are presented.

2.1 Spatial AI

In the introduction, the concept of **Spatial AI** was already mentioned. The definition was invented by the researcher Andrew Davison. He defined it as augmenting SLAM (Simultaneous Localisation and Mapping) technology with deep learning *"to enable the next generation of smart robots and devices to truly interact with their environments"* [5]. Spatial AI means the combination of computer vision and artificial intelligence: it empowers devices to continuously interact with their environment in real-time [4] [6].
The objective of current research is being shifted from plain environment mapping and understanding towards more complex topics, where the emphasis is on using the existing solutions (such as SLAM, deep learning, semantic segmentation) as building blocks to design bigger systems. Andrew Davison and his team in SLAMcore Ltd. continued the definition of Spatial AI by stating that it is "the process of turning sensor information into actionable, spatial understanding" [7]. More specifically, according to their definition, the 3 key questions Spatial AI has the answers to are: Where am I? Where are the objects around me? What are the objects around me?

In order to thoroughly understand what Spatial AI is really about, it is important to have a look at the evolution of machine and computer vision algorithms.
Visual SLAM has undergone 20 years of development and research to reach where it is today. From its evolution, three main levels could be highlighted. The first step in this advancement was **sparse localization**, where the camera was tracking a set of landmark features and built them incrementally into the 3D scene representation, thus achieving localization [8].

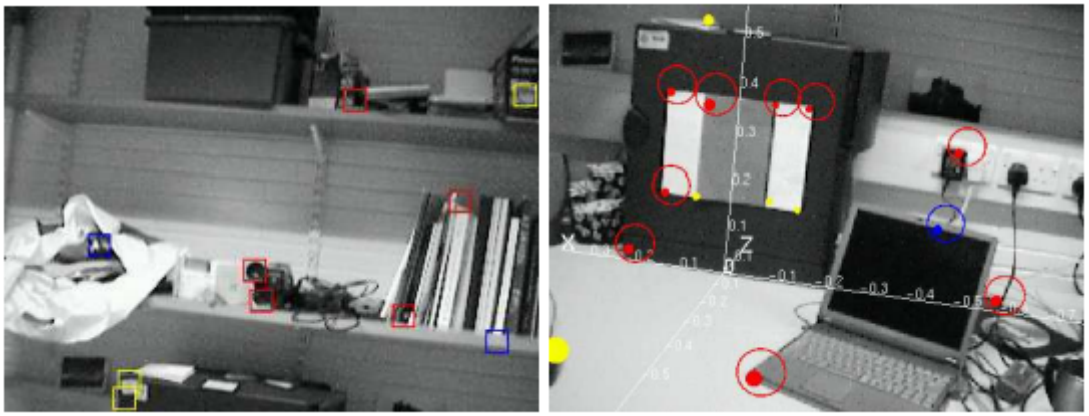


Figure 1. Sparse SLAM - Finding landmark features (Davison [8])

In sparse SLAM there wasn't any detailed knowledge about the environment yet, the emphasis was more on camera localization.

The second step was called **dense visual SLAM**. The intention was to start from the known localization capability of sparse SLAM and achieve a better, more detailed 3D mapping of the environment. This was made possible by the commercialization of GPUs and various depth cameras, enabling general-purpose computing on graphics processing units [9].

Figure 2. Dense visual SLAM 3D mesh model and found landmark features (R. A. Newcombe et al. [9])

The third level in SLAM evolution was **semantic visual slam**, which added an extra semantic layer to the achievements of sparse and dense visual slam. The semantic layer allows a wider range of functionalities compared to only having information about the geometry of a scene and also provides more information useful for human-centered applications. The evolution of deep learning algorithms and continuously developing GPUs enabled real-time scene reconstruction and semantic mapping, which led to a more detailed scene understanding [10].

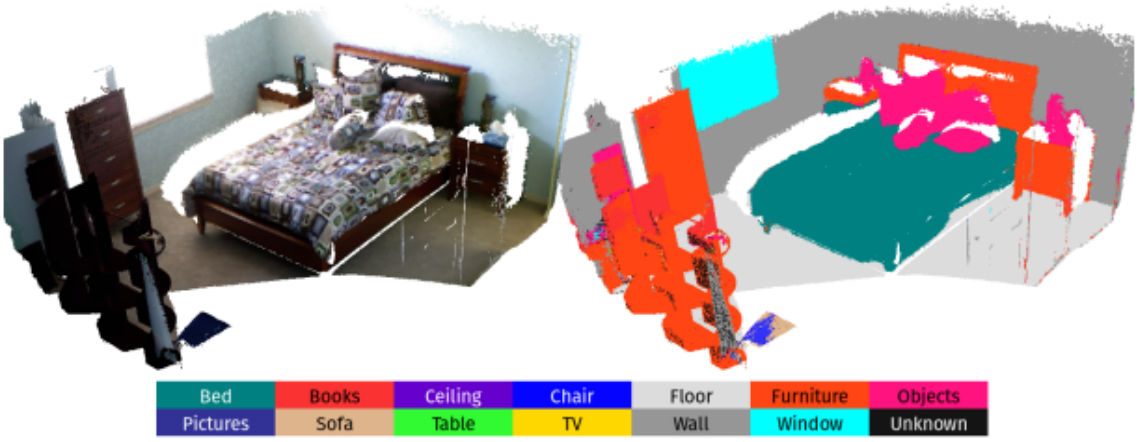


Figure 3. The output of semantic visual SLAM (J. McCormac et al. [10])

To complete the steps of SLAM development, it is worth mentioning that semantic visual SLAM has an object-oriented variant, called **SLAM++**. The SLAM++ does not use the bottom-up solution, where first the 3D scene is built and then semantic labels are added. On the contrary, it is using an already available object database and as soon as an object is found during the mapping of the environment (by comparing the scene with the database), it is added to the map. As a result, there aren't any partially fused objects in the map. An object is either present or not present at the end of the mapping [11].

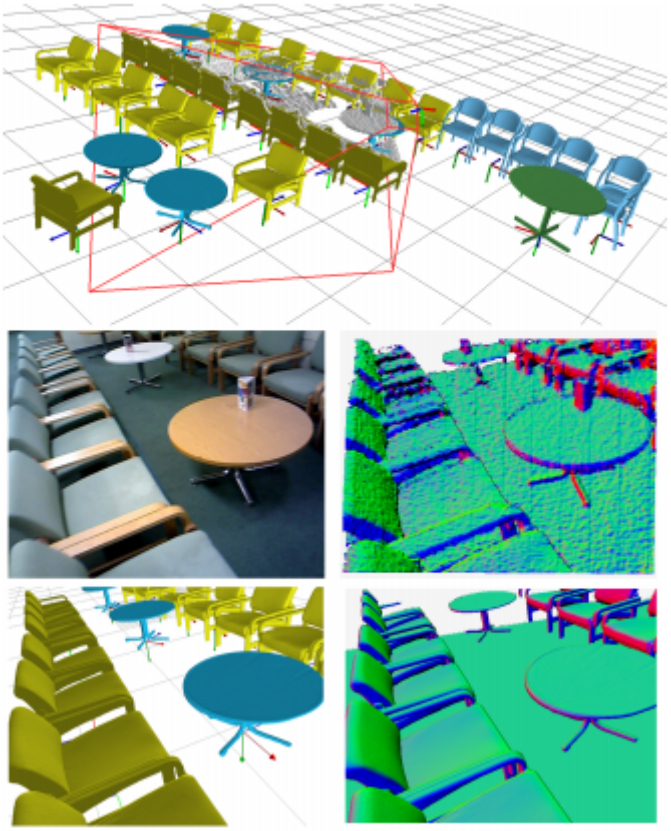


Figure 4. The output of SLAM++ (R. F. Salas-Moreno et al. [11])

As it can be seen, the joint use of SLAM and deep learning CNN (Convolutional Neural Network) algorithms allows researchers to design complex systems. However, one of the main obstacles of Spatial AI is the size, price and lack of computational power of the hardware devices. This is why the goal of Spatial AI is to co-design processors, sensors and algorithms to develop real-time systems that can carry out a wide variety of tasks while being able to continuously interact with their environment in real-time [4].

As mentioned before, the ZED 2 stereo camera was expressly designed to be one of the flagships of the Spatial AI era. It is the first stereo camera that includes neural networks to help compute the depth information of a scene and also facilitate complex computer vision solutions with built-in SLAM and object detection algorithms.

2.2 Research on 3D change detection

In this section, the state-of-the-art 3D change detection methods and algorithms are described to present various approaches and evaluate them for their possible usability with the ZED 2 stereo camera.

Change detection is *"the process of identifying differences in the state of an object or phenomenon by observing it at different times"* [12]. Change detection is used for numerous tasks, e.g: construction monitoring, traffic monitoring, pedestrian tracking or even as a security system. There are many ways to categorize change detection methods, e.g: based on input data, technology and sensors used to acquire the data, methods used to analyze the data etc. The input data for a change detector can be in various forms, e.g: satellite imagery, aerial photos, street-view photos, feature maps, three dimensional point clouds [12] [13].

3D change detection is a subset of traditional remote sensing (RS) [14], which analyzes and determines the (typically long-term) changes of the surface from multi-temporal images. However, in general, **3D change detection can be applied to both remote sensed data captured from top view or to close-range data that was captured near ground-level.** 3D change detection is a quite recent field of study that was motivated by the increasingly available 3D information and also the need for it in e.g: smart cities, more available autonomous vehicles and robots. The key difference between 3D change detection and traditional remote sensing is the **additional dimension of input data**. The data can be point clouds, 3D (city) models, digital surface models or stereo-view images that can express positions/shapes of objects along all three dimensions [15].
With available 3D geometric information, the input data is not affected by illumination or perspective distortion, thus 3D change detection can provide a more accurate description of the environment. The commercialization and advancement of LiDAR systems, UAVs and stereo cameras and the precise three dimensional co-registration algorithms allow cost-effective data acquisition and data processing. However, the quality and accuracy of 3D data depends highly on the precision of the acquiring sensors and on the algorithm that is used during data gathering. Additionally, the change from 2D to 3D space leads to an increase in problem modalities, e.g: occlusions, incomplete data, point cloud registrations, 3D feature extractions [15].

Change detection methods usually have 3 main components [15].

1. Data acquisition
2. Data co-registration / data association
3. Change analysis

There are many forms of change detection methods whose solutions differ in one or in all of these components [16].
Some approaches **compare the multi-temporal 3D point clouds directly (3D to 3D change detection)**. D. Girardeau-Montaut et al. [17] use a ground based laser to scan the 3D environment at different times and define a specific octree structure (an octree recursively

subdivides the 3D space into eight smaller spaces) which is used to compare cloud-to-cloud comparison techniques. The result of the change detection is displayed as a heatmap. However, this method is highly sensitive to point sampling variations between scans and the running time depends greatly on the octree level.

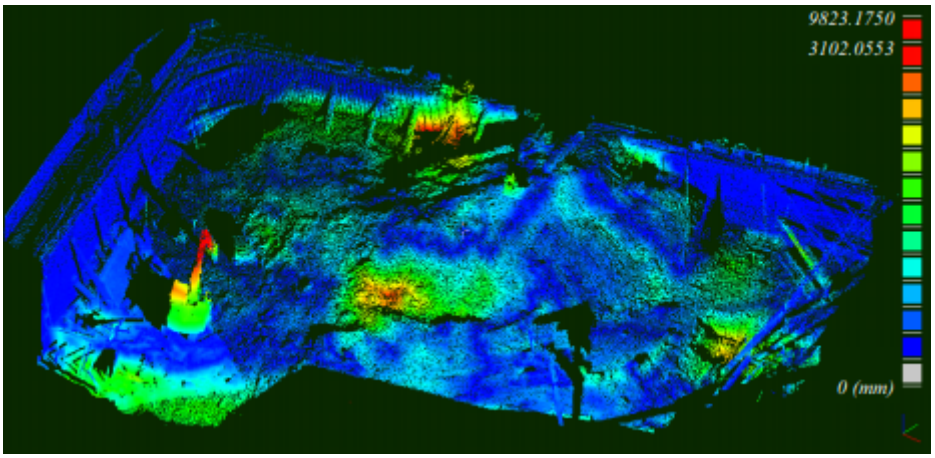


Figure 5. Using Hausdorff distance as a metric in cloud-to-cloud comparisons
(D. Girardeau-Montaut et al. [17])

Fanfani et al. [18] also use a cloud-to-cloud comparison technique. The point clouds are generated from multi-temporal image collections using the Structure from Motion (SfM) method. SfM is a method that uses the principles of stereo vision on multiple overlapping image pairs to create a 3D representation. It tracks features that are visible in a minimum of three images and estimates the camera positions, orientations and the coordinates of the features to produce a point cloud from the 2D image collection. After having the generated multi-temporal point clouds, the paper proposed to use a Random sample consensus (RANSAC) framework for the estimation of the rigid transformation between the point clouds using a previously generated 3D feature-match matrix. As a final step before the change detection, an Iterative Closest Point (ICP) algorithm is used to align the point clouds. The change detection algorithm works by shifting a 3D box over the aligned point clouds and computing a majority voting scheme that compares density, shape and distribution of 3D points. The result of the change detection is displayed as a heatmap. The change detection method proposed in the paper, and most methods that are based on **cloud-to-cloud comparison techniques**, are mainly used in post-processing and they are **inefficient as real-time solutions with regard to time and computation.**

There are several 3D change detection approaches, which assume that a 3D model of the scene is given and then 2D images are used for the change detection **(2D to 3D change detection)**. Golparvar-Fard et al. [19] assume that a Building Information Model (BIM, which has information about geometry and relationships of elements) is given and utilize a 2D image collection to reconstruct a construction scene using structure-from-motion, multi-view stereo (MVS, which is the extension of standard stereo vision to multiple images) and voxel coloring algorithms with a probabilistic framework. However, it relies greatly on

the quality of the obtainable MVS reconstruction and also on the novelty of the 3D Building Information Models.

Taneja et al. [20] and Taneja et al. [21] propose a method to detect changes in the geometry of a city using cadastral 3D models provided by the city administration and Google StreetView panoramic images captured all over the city. The key idea is to obtain a pixel-wise map of inconsistencies between the geometry of the environment and two multi-temporal images of said environment by projecting one of the images onto the other. Then based on this inconsistency map, the change detection algorithm estimates a binary labeling for each uniformly sized voxel in the entire grid of the 3D model. The results show an increase in accuracy, compared to the similar, state-of-the-art solutions. However, due to reflections and occlusions, the number of false positives is relatively high. Moreover, the computation time is around 1 minute per region, which makes these solutions not suitable for real-time change detection.

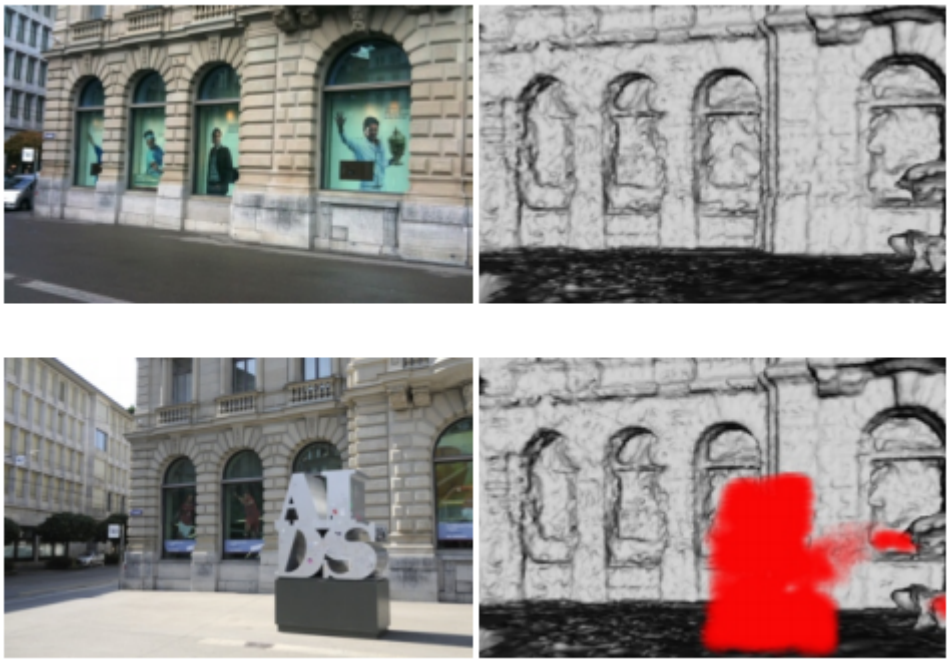
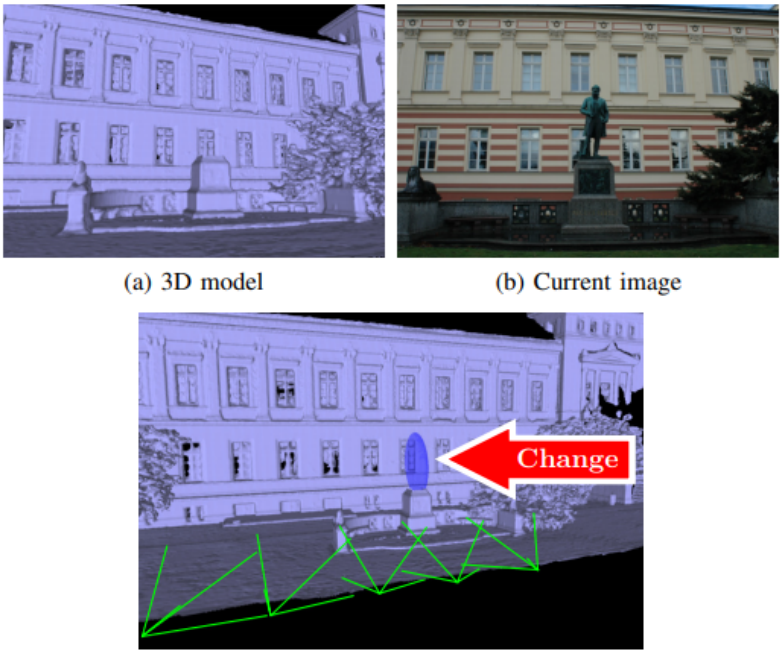


Figure 6. Change detection by projecting one of the multi-temporal images onto the other using the 3D geometry of the environment (Taneja et al. [20])

Palazzolo and Stachniss [22] use an approach similar to the one mentioned previously. However, instead of picking image pairs, the paper proposed combining multiple images such that the 3D location of the change can be estimated with fewer ambiguities. Apart from being more accurate, the above mentioned solution is suitable for real-time change detection as, according to the authors, the total computational time is less than two seconds. However, this solution builds upon an already built dense 3D map of the environment. This means that during change detection the 3D map is not being continuously updated, therefore it is not perfectly suitable for patrolling tasks that require consecutive comparisons and updated 3D models.



(a) 3D model (b) Current image

(c) Result

Figure 7. Comparison between the re-projected images and the one observed in reality using the 3D model of the scene (Palazzolo and Stachniss [22])

To reduce the computational time and to increase the accuracy of multi-view stereo based change detection methods, Sakurada et al. [23] propose a probabilistic framework to quickly detect damages of a city after earthquakes. The key idea in the paper is that the exact structure of a scene is not needed in order to detect changes. The estimation of the scene structure is done by using a probabilistic density of depths, and then this density is used in such a way that the depth ambiguity is well reflected in the final change estimation. Despite the reduced computational time, this solution still does not qualify to be able to be used in real-time change detection systems.

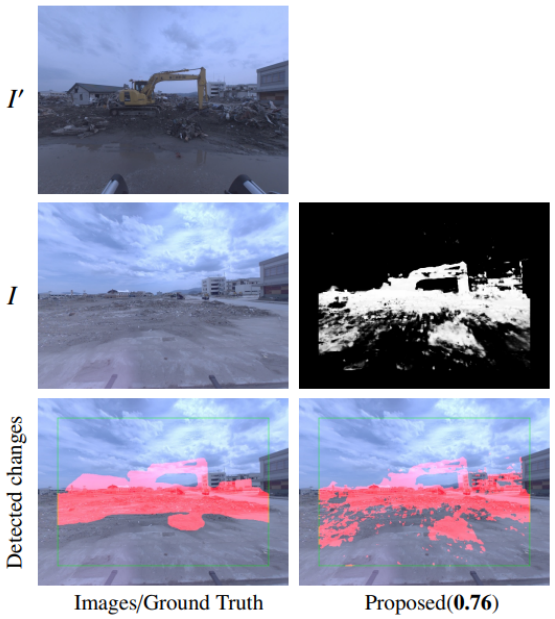


Figure 8. Result of the probabilistic framework
(Sakurada et al. [23])

Andreasson et al. [24] propose a change detection algorithm that uses a 3D range sensor and a camera to estimate the depth information only at the position of local visual features in the image. After registering the multi-temporal point clouds, the change is detected using a probabilistic framework based on colour and spatial difference.

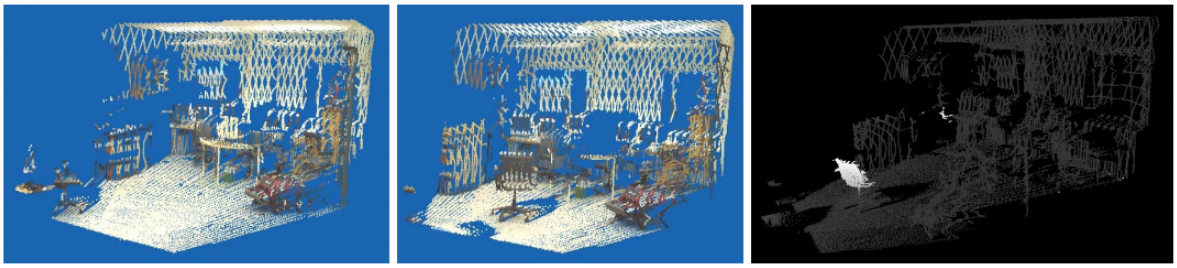


Figure 9. The multi-temporal point clouds and the detected changes
(Andreasson et al. [24])

With the evolution of deep learning algorithms, it became possible to implement AI architectures for detecting changes. Shi et al. [25] survey the state of the art change detection methods that are based on artificial intelligence. However, most of them use remote sensed, top-view data to detect environmental changes and changes on the Earth's surface.
Varghese et al. [26] introduce an architecture called ChangeNet that uses a parallel deep convolutional neural network for localizing and identifying the changes between image pairs. The results are promising, however, the changes are only in 2D. The architecture does not concern itself with three dimensional data.

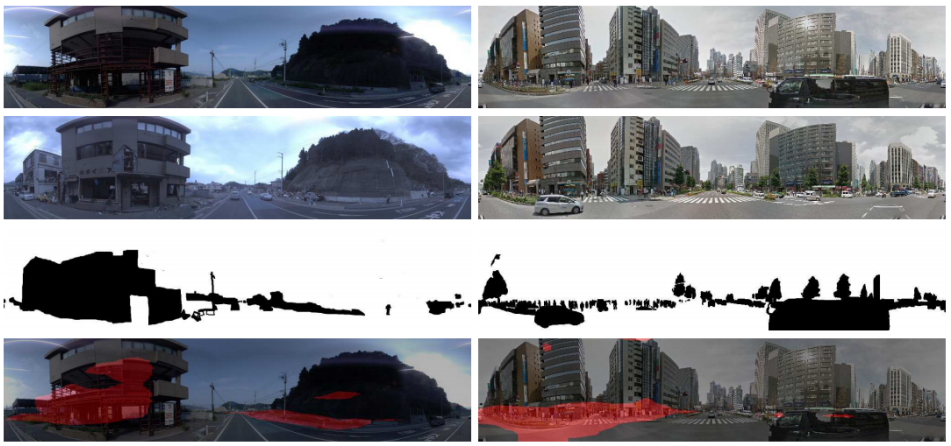


Figure 10. Output of the ChangeNet change detection architecture, compared to ground truth (Varghese et al. [26])

The previously mentioned 3D change detection methods work on a general level and they are susceptible to noise and are time-consuming for later classification. Mostly, they capture long-term structural changes. However, for patrolling tasks it is required to have 3D **change detection algorithms** that work on **object-level** and are fast enough to make decisions in real-time.

Lawson et al. [27] propose an algorithm that is between general and object-level change detection. The paper presents a method where the robot is fixed on a patrol route and tries to build a dictionary of the clustered features present in the scene. Later, based on this feature dictionary, the robot is able to detect scene-specific anomalous objects. The algorithm does not know any information about the objects, only detects if a feature cluster has changed or not. Additionally, the change detection algorithm, proposed in this paper, produces a very high number of false positives due to occlusions. The authors suggest a higher level reasoning to be added to their solution to identify the likely cause of the change and to rule out false positives.

The author of this thesis did not find any change detection algorithms that utilize object detection algorithms on near-ground captured data for 3D change detection. However, there are new publications (McCormac et al. [10], Nakajima et al. [29], Sünderhauf et al. [30], Zhang et al. [31], Truong et al. [33], Sung-Hyeon et al. [34], Kunze et al. [36]) that investigate the possibility of using object oriented semantic maps to give higher level data to robots. In the section called *Research on object detection based semantic maps*, these papers will be reviewed and their feasibility for change detection will be considered.

2.3 Research on object detection based semantic maps

Semantic maps are able to describe the environment from an object level perspective and give higher level reasonings to autonomous systems. Nuchter et al. [28] defined the **semantic map** as *"a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes. Further knowledge about these entities, independent of the map contents, is available for reasoning in some knowledge base with an associated reasoning engine."*.

McCormac et al. [10] and Nakajima et al. [29] propose semantic mapping systems combining CNN-based semantic labeling and SLAM. The architecture uses a real-time SLAM system for finding correspondences between frames and thus achieving localization. Separately, it also uses a CNN network to return 2D pixel class probabilities for RGB or RGBD images. Finally, the output of these two parts are taken into consideration for updating the 3D map. The idea to combine two separate algorithms for semantic mapping is useful and can be incorporated into a change detection system, however, the publications do not mention anything regarding how to use the semantic map for further manipulation or database creation.

The system proposed by Sünderhauf et al. [30] gives a solution to the previously mentioned problem. It simultaneously builds a 3D model of the environment and also creates an object-oriented semantic map with a database. In this case, object-orientedness means that object instances are the key entities in the semantic map.

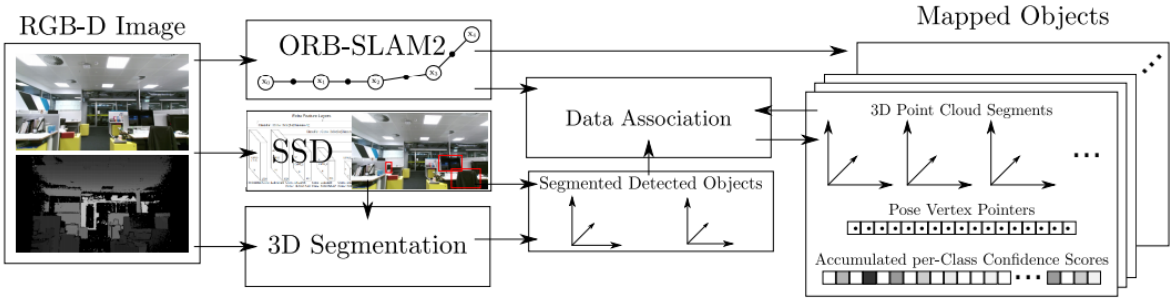


Figure 11. Overview of the semantic mapping system proposed by Sünderhauf et al. [30]

To provide more advanced scene understanding, bounding box-based object detection and unsupervised 3D segmentation were combined with simultaneous localization and mapping. The semantic map building happens in a step called ***Data Association***. It determines whether a detected object was already built into the map (hence only needing to update a previously saved map object) or it needs to be added as a new map object. The data association happens in two stages. For every detected object, the system queries the semantic object map to select previously saved objects whose point cloud centroids are below a predefined Euclidean distance threshold. Secondly, a nearest neighbor search is performed between the segmented 3D object point clouds (the detected object's and the queried, already saved object's point clouds) to calculate the Euclidean distance between the neighbouring point pairs. If at least 50% of the associated 3D point pairs of the two object point clouds have a distance smaller than a predefined threshold, the detection will be associated with an already saved object in the semantic object map. According to their measurements, this process takes around 30 ms for each comparison. Their work stops at the built 3D semantic map and only gives future

possibilities on how this map could be used. Still, this approach does not require previously built 3D models of the environment and the mapping and the building of the object database happens in real-time.
The time-efficiency of the semantic mapping and the productive combination of object detection and SLAM prove to be a useful solution for designing a more detailed scene understanding, which will be further investigated in the section called *The change detection algorithm*.

Zhang et al. [31] developed an object detection based semantic SLAM system by using the YOLO object detector for real-time object detection and a SLAM system for simultaneous localization. The main difference between the proposed system and the previously mentioned semantic mapping solutions is that it uses an octomap based structure for storing the point clouds at every keyframe. The reason for using an octomap is that point clouds do not use any structures to store each point, which makes querying and searching the map inefficient. Another drawback of point clouds is that they cannot categorize e.g. the empty areas and cannot eliminate noise, which makes path planning and collision detection harder. Octomap is a probabilistic 3D mapping framework that solves all the previously mentioned problems by using an octree structure [32]. An octree recursively subdivides the 3D space into eight smaller spaces. Using an octomap accelerates the speed of mapping. However, one of the main advantages of the ZED 2 stereo camera is that it uses neural networks and GPU to deliver state of the art 3D spatial mapping by needing in average less than 10 ms for point cloud processing. Therefore, getting the 3D map of an environment is not a bottleneck in time-consumption for a ZED 2 camera based application.

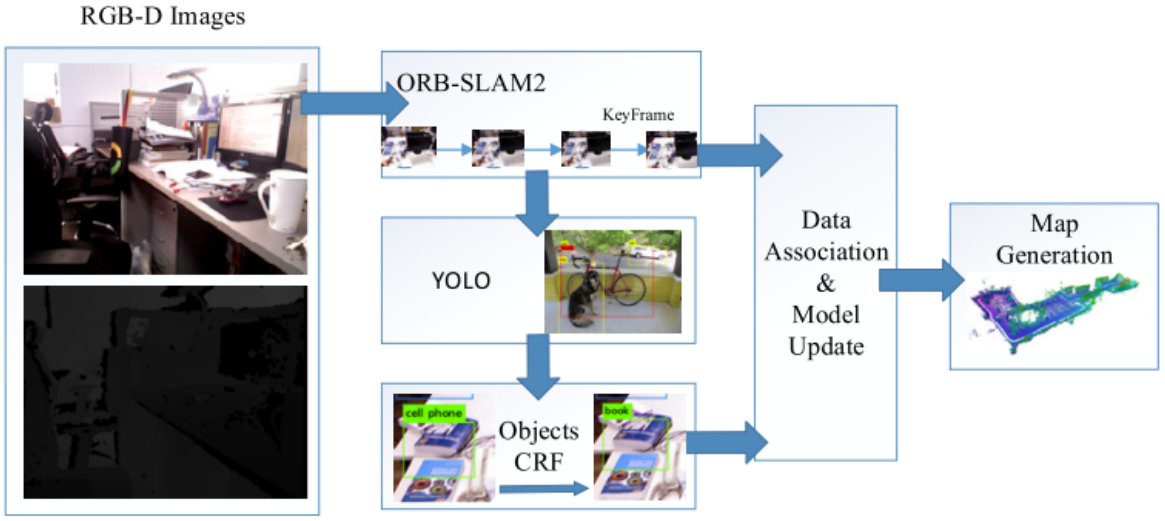


Figure 12. Architecture proposed in Zhang et al. [31]

The architecture proposed in this paper is similar to the one in [30], in that they both simultaneously traverse a 3D scene and build a semantic object database using the gathered data in real-time.
Truong et al. [33] also use an object detection algorithm on top of SLAM to obtain semantic information of landmarks in the map. The proposed solution in the paper is suitable for

dynamic environments and for building a semantic database it uses a Triplet Ontological Semantic Model (TOSM) database that was introduced by Sung-Hyeon et al. [34]. The goal in defining the TOSM database was to enable systems to carry out complex tasks using high-level and human-like knowledge. TOSM aims to mimic the brain GPS model efficiency by storing metric information, geometrical features and image information of objects, but also relations between environmental elements. For instance, when a robot explores its environment, it detects two objects using a CNN object detector and saves them into the semantic database along with their spatial relationship, as seen on the figure below.

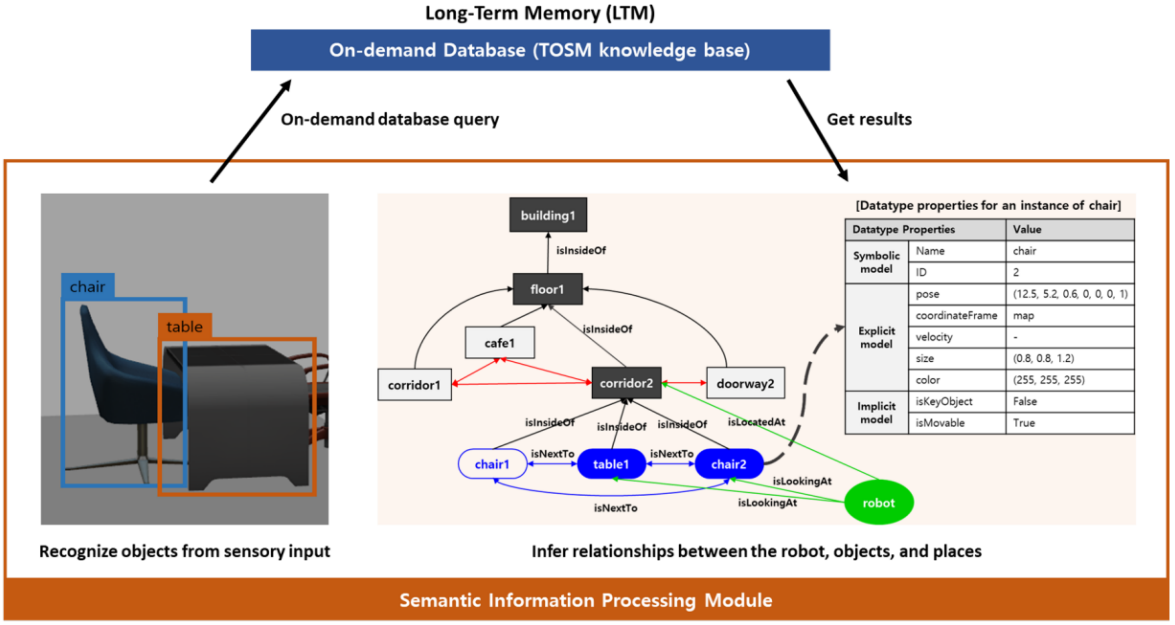


Figure 13. Semantic database with spatial relations between objects
(Sung-Hyeon et al. [34])

The solution proposed using the TOSM model is extensive and efficient for path planning and navigational tasks (Rocha et al. [35]), but requires retrained deep learning models and a complex database structure that are beyond the scope of this thesis.

Kunze et al. [36] propose a semantic object mapping framework called SOMa, designed to "*map locations of objects, regions of interest, and movements of people over time*". They specifically introduced this framework to help researchers understand change in dynamic environments. Their open-source system is adaptive for changes, can be queried and stores objects and regions of interests over time. However, SOMa stores discrete observations of objects and is more suitable for long-term case-studies, not real-time change detection.

2.4 Object detection

The object detection algorithm of the ZED 2 camera is proprietary, which means that its architecture is not accessible. Therefore, in order to assess the performance and capability of the ZED 2 object detection algorithm, it is meaningful to benchmark it against the state of the art object detection algorithms. In this module, a brief introduction into the evolution of object detection is given. Following the introduction, the current state-of-the-art deep learning object detection algorithms are presented. After presenting the state-of-the-art object detectors, the performance metrics used in object detection are described.

2.4.1 Object detection with deep learning

Object detection is the task of accurately localizing and classifying one or more predefined classes of objects. The human visual system is and has been doing this job very efficiently, which has motivated researchers to try to automate this process with the help of computer vision algorithms. Traditional object detection used **handcrafted features** based on basic shapes and convex polygons of contours. With the combination of these low-level features, it was possible to describe more complex, higher level features. However, these manually designed features are not invariant to illuminations or the diversity of appearances [37]. The bigger step in the evolution of object detection happened when automated feature extraction methods were introduced. The automated method of finding features aimed at finding descriptors, that are invariant to image translation, scaling, and rotation, illumination changes and robust to local geometric distortion. Such methods include for example the **Harris detector**, that finds object corners based on local intensity changes in an image, **Scale Invariant Feature Transform (SIFT)** or **Speed Up Robust Features (SURF)** that create robust feature vectors for points of interest in images. With the help of automated feature extraction algorithms, it is possible to represent objects with their keypoints and use these keypoints to perform object detection. However, after the initial success of such algorithms, the accuracy of object detections have stagnated for a couple of years between 2010 and 2012 [37].

With the beginning of the Deep Learning era in the year 2012, traditional feature engineering has been replaced with **end-to-end learning using convolutional neural networks** (CNNs) [38]. The spatial invariance of CNNs [39] made them perfect building blocks for Deep Neural Networks (DNNs), allowing efficient end-to-end learning. As a result, the deeper architectures were able to learn more complex features, leading to much higher object detection accuracy [40].

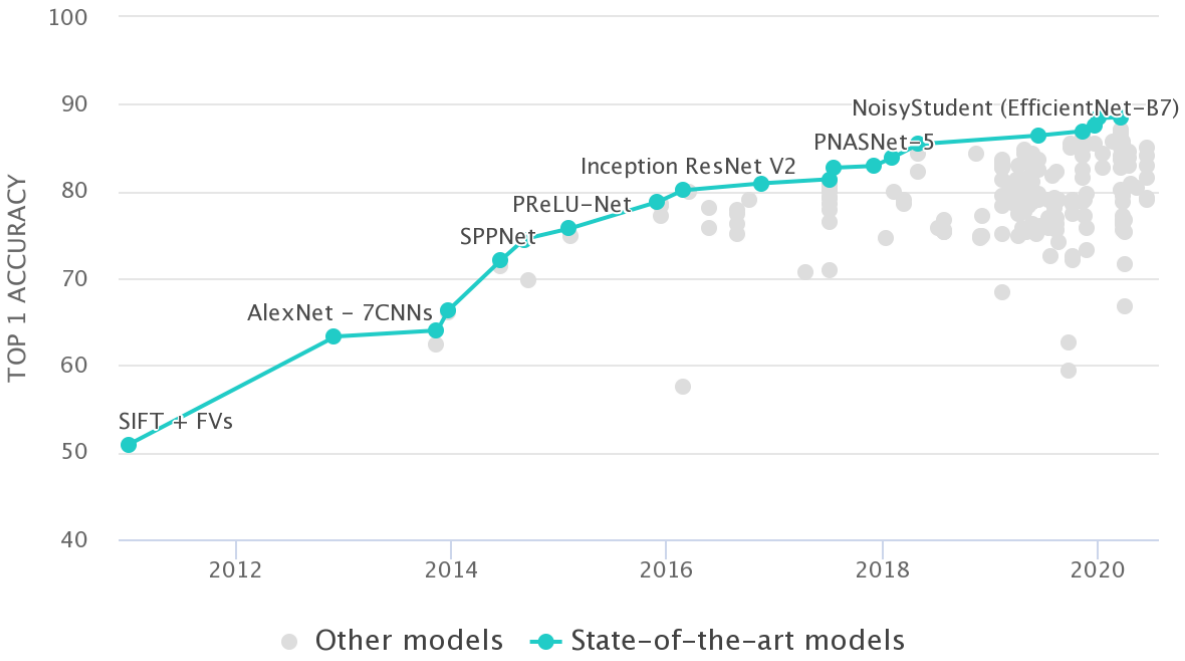


Figure 14. ImageNet results after switching from traditional feature engineering to Deep Learning [41]

There are two main types of CNN-based object detectors: **Region-based detectors** (also known as two-stage detectors) and **regression/classification based detectors** (also known as single-shot or one-stage detectors) [40].
Region-based detectors achieve object detection in two steps. First, they generate region proposals and then classify each proposal into one of the previously defined object categories. The region proposal based methods include e.g: region-based convolutional network (R-CNN), Fast R-CNN, Faster R-CNN, region-based fully convolutional networks (R-FCN), Feature Pyramid Network (FPN), Mask R-CNN.
Single-shot detectors achieve the final object detection result directly by splitting the images into a grid of cells and then for each cell they make different bounding-box guesses. Single-shot detectors methods include e.g: Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO) [40].

In this next section, three state of the art object detection methods are briefly described. Namely, Faster R-CNN, SSD, and YOLO. The performance of these three object detector architectures will be later benchmarked against the performance of the ZED 2 object detection algorithm. Therefore, it is necessary to know their architecture before doing a comparison between them. The goal is to provide a general overview of the algorithms, so that the reader can have a better understanding of the structure of these state of the art object detectors.

Faster R-CNN [42] is a region-based object detector and as its name indicates, a faster version of the regular Fast R-CNN detector. It has two networks: A region proposal network (RPN) for generating region proposals and a network for using these proposals for object

detection. RPN reduces the time for generating region proposals, instead of using selective search to do the same but slower. RPN is a fully convolutional network and it predicts object bounds and a metric called objectness score. Objectness measures membership to a set of object classes. The RPN uses a sliding window to generate region proposals and at each location it generates 9 anchors/region boxes. The RPN predicts if an anchor belongs to the foreground or background, thus telling the object detection network where to look for objects in the image.

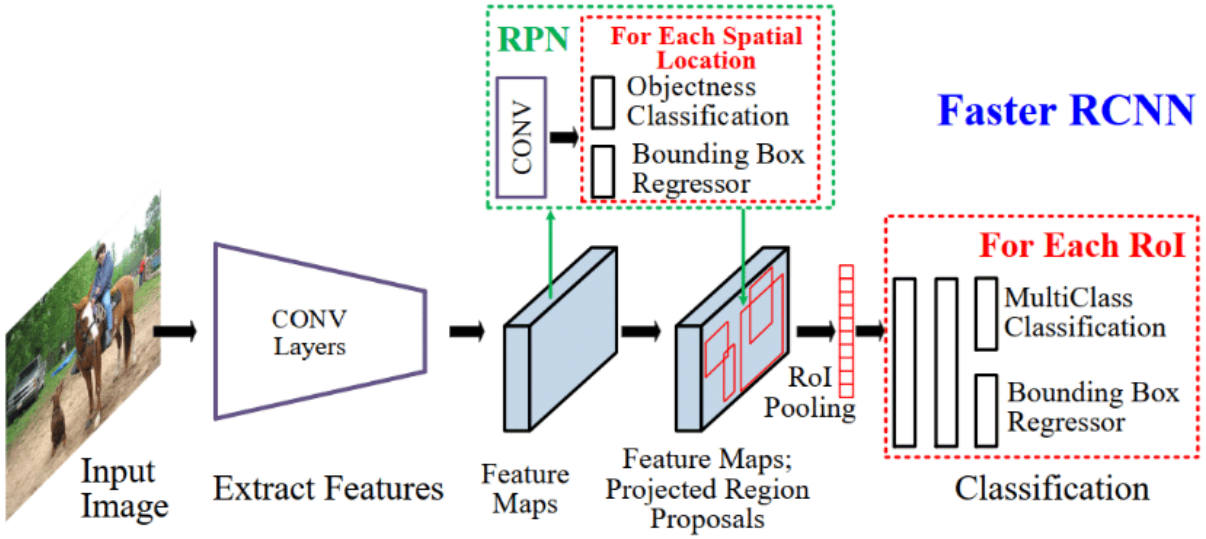


Figure 15. The high level architecture of the Faster R-CNN network (Liu et al. [43])

After generating the region proposals, the Region of Interest (ROI) Pooling layer makes the feature vectors fixed-sized. The extracted feature vector, using the ROI Pooling, is then passed through fully convolutional layers and then the output is split into two branches: Softmax layer to predict the class scores and a fully convolutional layer to predict the bounding boxes of the detected objects [42].

Faster R-CNN (and R-CNN networks in general) has three main drawbacks [44]:

- Training the network takes too long.
- Training happens in multiple phases, i.e: training region proposal and classifier.
- Too slow at inference time which would not allow real-time object detection.

SSD [45] is one of the single-shot object detectors. It outperformed the previous region-based object detectors in both accuracy and speed. As the name suggests, the object localization and classification happens in one single forward pass of the network, instead of two passes as in R-CNN networks, thus SSD speeds up the process by eliminating the need for the region proposal network The SSD framework has two main parts. One is based on the famous VGG-16 [46] architecture, which won the ImageNet Visual Recognition challenge in 2014. This part is responsible for the feature map extraction. It divides the input image into a grid and for each cell it evaluates/predicts a small set of default boxes of different aspect ratios. Each prediction entails the coordinates of the bounding box and c class confidence scores,

where *c-1* is the number of classes and the added extra class is for the background [45]. This method is the base for MultiBox [47].

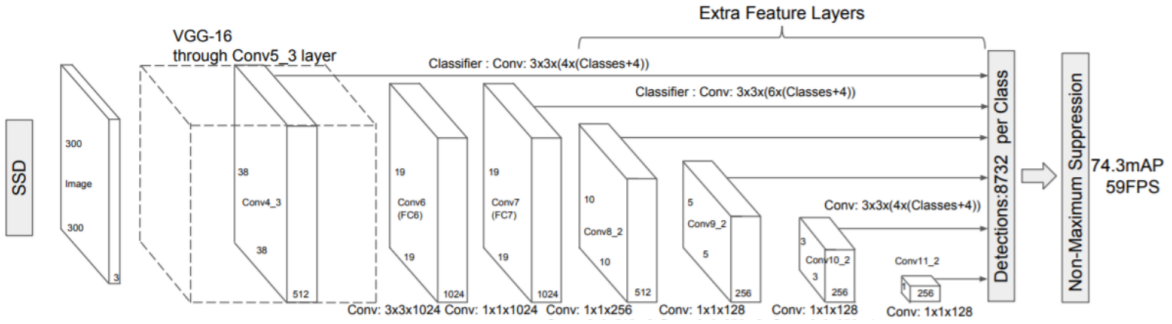


Figure 16. The architecture of SSD (Liu et al. [45])

The second main part of the SSD framework consists of a series of convolutional filters to detect objects. For each predicted bounding box from the MultiBox based feature generation, a set of *c* class predictions are computed for every possible class in the dataset, thus creating a class probability map [45].

SSD achieves 74.3% mAP (mean Average Precision) at 59 FPS (frame per second) on the VOC2007 dataset (the meaning of object detection metrics will be described later, the higher the mAP and FPS values are, the better the algorithm is) and for 512 × 512 sized input images it achieves 76.9% mAP. It outperforms Faster R-CNN in accuracy, because on the VOC2007 dataset the region based method only scored 73.2% mAP with only 7 FPS [45].

YOLO (You only look once) [48] is a state-of-the-art, real-time single-shot object detection system. It also applies only a single neural network to the full image and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. Because the object detector looks at the whole image at once, its predictions are based on the global context of the image. It is much faster than the region-based object detectors, enabling real-time object detection [49].

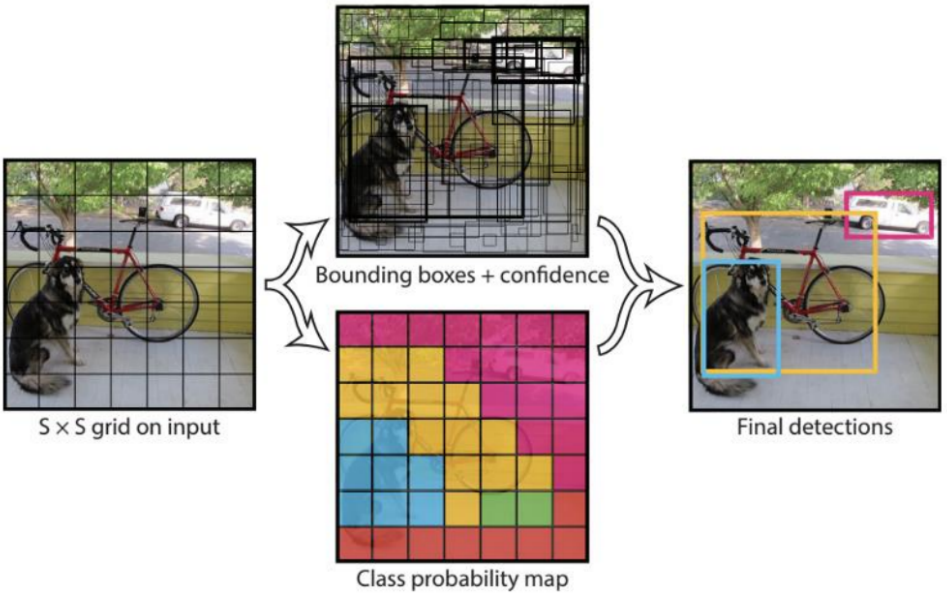


Figure 17. The high level model of YOLO (Redmon et al. [48])

The method of YOLO is similar to the SSD algorithm, however it only predicts one class per bounding box. If there are multiple objects of different classes in one grid cell, YOLO will fail to classify both of them correctly. Another limitation is that YOLO can only predict a limited number of bounding boxes per grid cell, which means that the algorithm cannot detect every object when multiple objects appear in a single grid cell [50].

YOLO comes with a trade-off between accuracy and speed. On the VOC2007 dataset, the original algorithm scored a 63.4% mAP with 46 FPS [49]. There are, of course, newer versions which achieve better results. However, it is insightful to know the original architecture in order to grasp how modern object detectors work. In the section, where the object detection test method is described with the ZED 2 stereo camera, these newer versions will be compared to the ZED 2 algorithm.

Table 1. Benchmark results on the PASCAL VOC 2007 test dataset

Model	**mAP (t=0.5)**	**FPS**
Faster R-CNN	73.2%	7
SSD	74.3%	59
YOLO	63.4%	46

2.4.2 Object detection performance metrics

The goal of object detectors is to find the location of objects (that belong to predefined classes) in an image with high confidence [37]. Once they find it, they place bounding boxes (rectangles) around the objects in the image. Consequently, a successful object detection is represented by three parameters: The class of the object or commonly called **object label**, the

bounding box of the detection, and the **confidence score** (usually a number between 0 and 1, or a percentage value between 0 and 100) showing how confident the object detector is about the prediction. When an object detector is being assessed, these three object detection attributes are being compared to the ground truth bounding boxes and object labels [37].
The number of object detectors is continuously increasing and in order to ease the training of various architectures, many datasets with a huge number of training and test samples have been created. Such datasets are e.g.: the Common Objects in Context (COCO) dataset, the PASCAL Visual Object Classes (VOC) dataset, Open Images Dataset. To help create datasets faster, many annotation tools have been designed to help researchers label their ground truth data easier. However, some datasets use specific annotation formats that differ in how they represent the ground truth and the detected object data. For instance, the COCO dataset uses a JSON file that contains all the bounding boxes of a given dataset and the PASCAL VOC dataset has one XML file with the corresponding bounding boxes for each image in the dataset. Furthermore, usually each dataset has specific evaluation tools for assessing the performance of object detectors with different performance metrics [37].
For this reason, there was a need for a unified toolbox that could provide cross-dataset assessment using the common object detection performance metrics: Padilla et al. [37] proposed an open-source toolkit [51] supporting different annotation formats and providing various performance metrics that are supported by most object detection datasets. The metrics that are usually used in object detection are presented in the following section.

The object detection performance metrics measure how close the detected bounding boxes and the ground truth bounding boxes are. This is indicated by the amount of overlap between these bounding boxes, measured by the **intersection over union (IOU)**, which is the area of their overlap between the detected and ground truth bounding boxes divided by the area of their union [52].

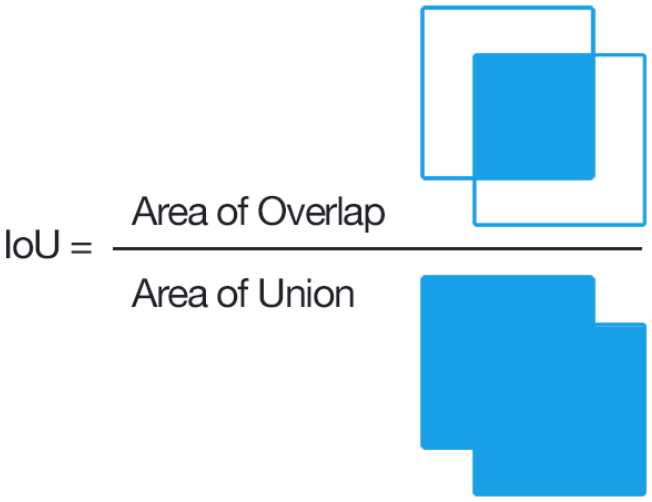


Figure 18. Intersection Over Union (IOU)
(Rosebrock [52])

The value of IOU can be between 0 and 1. The closer IOU is to 1, the better the object detection is. In order to decide whether a detection is correct, a threshold is usually set: IOU

values that are above said threshold are considered as correct detections, below the threshold are considered as incorrect. The most common threshold values are 0.5 and 0.75 [37].
The Intersection Over Union metric is used to define more metrics that describe the performance of object detectors: namely precision and recall. The confidence level of detections can be considered to work as a threshold for deciding whether a detection is correct when calculating the precision and recall metrics. **Precision** measures how accurate the prediction of an object detector is, i.e. the percentage of correct predictions. **Recall** measures how many of the ground truth objects are found, i.e. the percentage of correct predictions among all ground truths. Their equations are as follows:

$$Precision = \frac{TP}{TP + FP} \text{ and}$$
$$Recall = \frac{TP}{TP + FN}, \text{ where}$$

$TP = True\ positives$, the number of correct detections,
$FP = False\ positives$, the number of incorrect detections when the object detector detected an objects that were not there,
$FN = False\ negatives$, the number of undetected ground truth bounding boxes [53].

By setting the threshold for confidence score at different levels, different pairs of precision and recall values can be generated, which can be displayed using the **precision-recall curve**.

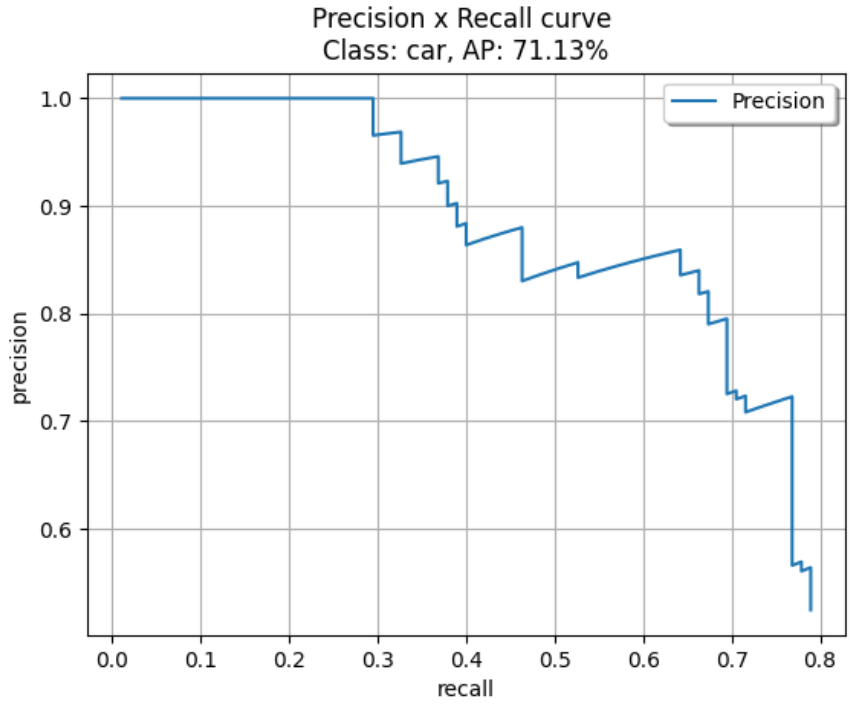


Figure 19. The precision-recall curve for the class *car* using the ZED 2 object detector (author's test result)

It is hard to compare the precision-recall curve of object detectors, therefore a numerical metric was introduced, namely the **Average Precision (AP)**, which is based on the area under the precision-recall curve. In order to measure the area under the curve (AUC), interpolation is usually used to reduce the impact of the wiggles in the curve. The interpolated precision for any recall level $r' \geq r$ is calculated as follows:

$$P_{interpolation}(r) = max_{r' \geq r} \; p(r')$$

There are two main methods for choosing recall points for the calculation of AP: ***N-point interpolation*** calculates the AP with N=11, the interpolation measures the recall in the points [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]. Using the ***All-point interpolation*** approach, all points are considered [53].

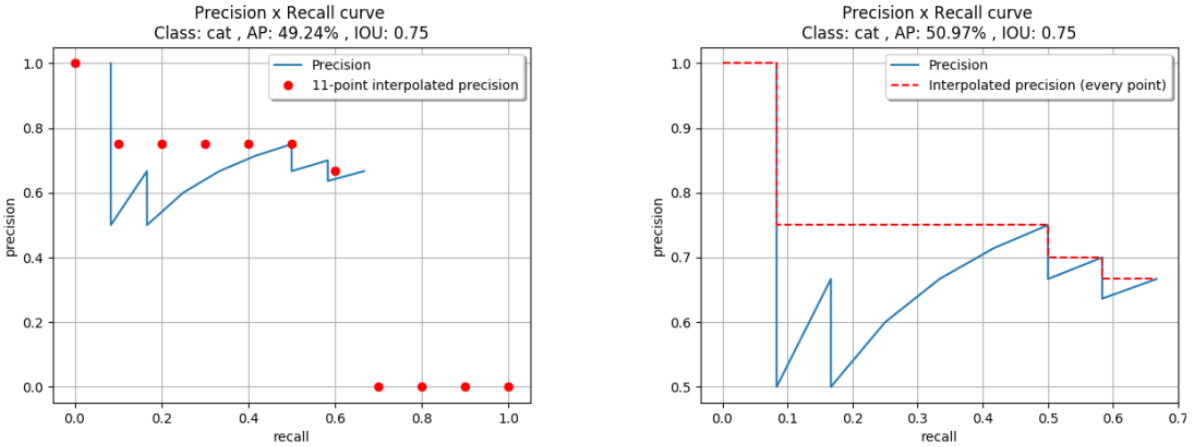


Figure 20. Example of the N-point and All-point interpolations (Padilla et al. [53])

Using the above described equation for interpolation, the equation of the Average Precision is as follows:

$$AP = \sum_{i=1}^{n-1} (r_{i+1} - r_i) \, p_{interpolation}(r_{i+1})$$

The Average Precision is calculated for each object class. The **mean Average Precision (mAP)** is defined as the mean of AP across all C classes:

$$mAP = \frac{\sum_{i=1}^{C} AP_i}{C}$$

Another relevant performance metric is the **Average Recall (AR)**. This metric does not take the detection confidences into account. AR is the recall averaged over all IOU $\in$ [0.5, 1.0] and can be computed as follows:

$$AR = 2 \int_{0.5}^{1} Recall_{IOU}(o)\, do,$$

where o is an IOU value in the range [0.5, 1.0] and $Recall_{IOU}(o)$ is the corresponding recall value.

Mean Average Recall (mAR) is defined as the mean of AR across all C classes:

$$mAR = \frac{\sum_{i=1}^{C} AR_i}{C}$$

There are many ways to evaluate object detectors based on the metrics presented above [37]. In the list below, the few mainly used variations are mentioned:

- **AP with IOU threshold t=0.5** - a widely used per-class metric in the PASCAL VOC dataset
- **mAP with IOU threshold t=0.5** - a widely used metric in the PASCAL VOC dataset, averaging per-class AP values with IOU = 0.5
- **mAP with IOU threshold t=0.75** - a strict metric used in the COCO dataset, averaging per-class AP values with IOU = 0.75
- **AP@.5 and AP@.75** - metrics commonly used in the COCO dataset, interpolation is performed in N = 101 recall points, then the per-class results are summed up and divided by the number of classes. AP@.5 uses IOU = 0.5 and AP@.75 uses IOU = 0.75.
- **AP@[.5:.05:.95]** - averaging the previously described AP@ results obtained with 10 different IOU values in the range of [0.5, 0.55, ..., 0.95]
- **mAP** ^{small} = applying AP@[.5:.05:.95] only on small objects that cover area less than32^2pixels, used in the COCO evaluation dataset
- **mAP** ^{medium} = applying AP@[.5:.05:.95] only on medium sized objects that cover area greater than32^2pixels but less than 96^2pixels, used in the COCO evaluation dataset
- **mAP** ^{large} = applying AP@[.5:.05:.95] only on large objects that cover area greater than96^2pixels, used in the COCO evaluation dataset [53]

2.5 Introduction to stereo vision

In this section, two key concepts of computer vision are introduced. Namely, the camera model and the description of standard stereo vision. These two concepts are detailed below to provide useful background information for understanding the tests conducted on the ZED 2 stereo camera.

2.5.1 Camera model

It is important to introduce the camera model used in computer vision to better understand how the camera and the world coordinate frames relate to each other and how triangulation in stereo vision works. Moreover, many crucial parameters of cameras are described in this section, which will be frequently mentioned in later parts of this thesis. The camera model section is based on the computer stereo vision theory described by Sturm P. et al. [54].

The camera model used in computer vision is called the **pinhole camera model**. The name originates from *camera obscura* (means dark chamber in latin) which is the predecessor of modern cameras. *Camera obscuras* are usually dark boxes, allowing the light to penetrate only through a small hole on their surface. This results in an inverted and reversed reprojection on the opposite side of the box, but with color and perspective preserved [55].

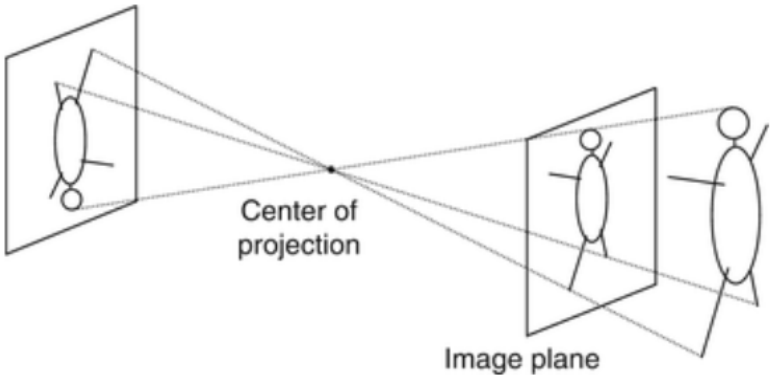


Figure 21. The two main components of the pinhole camera model (Sturm et al. [54])

The pinhole camera model defines the relationship between three dimensional coordinates and their projection onto the image plane. It is a simplified, ideal model of true cameras, where the aperture point (also called optical or center of projection), through which light travels, is described as a point. The ideal model does not include geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures.

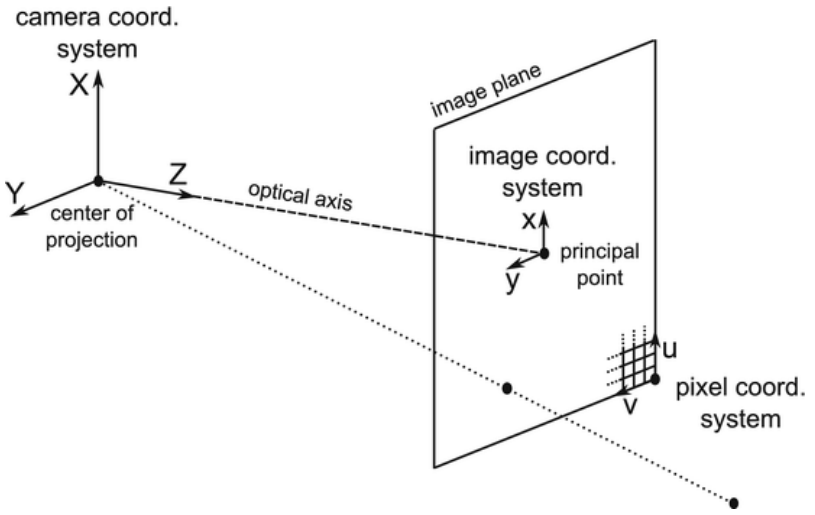


Figure 22. The components of the pinhole camera model (Sturm et al. [54])

The pinhole camera model introduces the following important properties:

- **Image plane**: The true image plane would show the inverted reprojection of objects, thus, for convenience, an alternative image plane is commonly used. This virtual image plane shows the objects with their real orientations, and it is at the same distance (focal length) from the center of projection as the true image plane but parallel to it.
- **Focal length, f**: The distance between the center of projection and the image plane.
- **Optical axis**: The orthogonal line between the optical center and the image plane.
- **Principal point**: The point where the optical axis intersects with the image plane.

The origin of the 3D orthogonal coordinate system of the camera is in the center of projection and its Z axis is coincident with the optical axis. The X and Y axes of the **camera coordinate system** are, by convention, parallel to the image coordinate system's axes.
The origin of the **image coordinate system** is in the principal point and its x and y axes are parallel to the X and Y axes of the camera coordinate system.
The **pixel coordinate system** is used to represent the final digital image and its origin is usually in one of the corners of the image. The u and v axes of the pixel coordinate system are parallel to the x and y coordinates of the image coordinate system [54].

By the properties of the pinhole camera model, the equation of the projection from a three dimensional point (expressed in the camera coordinate system with coordinates X, Y and Z) to a two dimensional point (expressed in the image coordinate system with coordinate x and y) is as follows:

$$x = f\frac{X}{Z}$$
$$y = f\frac{Y}{Z}$$

The conversion from the image coordinate system to the pixel coordinate system is as follows:

$$u = k_u(x + x_0) = k_u f \frac{X}{Z} + k_u x_0$$
$$v = k_v(y + y_0) = k_v f \frac{Y}{Z} + k_v y_0$$

Where k_u and k_v are the horizontal and vertical pixel sizes, respectively, and x_0 and y_0 are the coordinates of the principal point in the camera image. Commonly, x_0 and y_0 are replaced as $u_0 = k_u x_0$ and $v_0 = k_v y_0$, where u_0 and v_0 are the coordinates of the principal point given in the pixel coordinate system:

The camera matrix, which is usually denoted by K, can be constructed from the above mentioned intrinsic camera parameters.

$$K = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_u & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Using the camera matrix, the projection of a three dimensional point from the camera coordinate frame to the pixel coordinate frame can be defined as follows:

$$K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} fk_u X + Z u_0 \\ fk_v Y + Z v_0 \\ Z \end{bmatrix}$$

$$\begin{bmatrix} fk_u X + Z u_0 \\ fk_v Y + Z v_0 \\ Z \end{bmatrix} \sim \begin{bmatrix} fk_u \frac{X}{Z} + u_0 \\ fk_v \frac{Y}{Z} + v_0 \\ 1 \end{bmatrix}$$

In order to be able to model the camera in motion, it is necessary to describe its position and orientation with respect to some fixed frame. The origin of the **world coordinate system** is arbitrary, but has to be fixed throughout an application. The coordinates of the center of projection in the world coordinate system are usually denoted by ***t***, and the camera's orientation is represented by the rotation matrix ***R***.

A three dimensional point expressed in the world coordinate system can be mapped to the camera coordinate system with the following equation:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix}$$

The **general camera model** contains the rotation **R**, the translation **t** and the projection to the image coordinate system, where **R** and **t** are the extrinsic camera parameters.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} k_u f & 0 & k_u x_0 & 0 \\ 0 & k_v f & k_v y_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & -Rt \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix}$$

Which can be rewritten to the equation below:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \underbrace{\mathtt{KR}\,(\,\mathtt{Id}_3 \quad -\mathbf{t}\,)}_{\mathtt{P}} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix}$$

Where Id_3 is a 3 × 3 identity matrix and the 3 × 4 matrix **P** is called a projection matrix. The spatial coordinates [X^w, Y^w, Z^w] are given in the world coordinate system [54].

2.5.2 Standard stereo vision

In this section, the standard stereo vision is described to understand how stereo cameras are able to measure the depth of a scene.
There are several solutions that provide three dimensional reconstruction of the environment. Such solutions include stereo cameras, laser scanning, structured-light scanning or depth cameras that utilize the concept of Time-of-Flight (ToF).
Stereo cameras use two horizontally displaced lenses to obtain two differing views on a scene to simulate human binocular vision and estimate the disparity map of the scene. By having seen an object from two different positions, its three dimensional position can be calculated [2].
The challenge in stereo vision is finding the corresponding points in the two images captured by the stereo camera. [56] In standard stereo vision, the two cameras have to be calibrated and the stereo images have to be undistorted in order to better approximate the projection of an ideal pinhole camera. Furthermore, the stereo images have to be rectified, which means that they are projected onto a common image plane in order to decrease the search space for correspondences. Reconstructing the 3D scene from 2D point correspondences given in standard stereo images is called **triangulation** [57].

There are three main stereo reconstruction types:

1. **Fully calibrated reconstruction**
   Both the intrinsic and extrinsic camera parameters are known and the reconstruction is done by **triangulation**.
2. **Metric (Euclidean) reconstruction**
   Only the intrinsic camera parameters are known and at least n >= 8 point correspondences are given. The extrinsic camera parameters can be calculated by the so-called essential matrix and the reconstruction is up to a similarity transformation.
3. **Projective reconstruction**
   The intrinsic and extrinsic camera parameters are unknown and at least n >= 8 point correspondences are given. The projective matrices can be composed from the so-called fundamental matrix and the reconstruction can be computed up to a projective transformation [58].

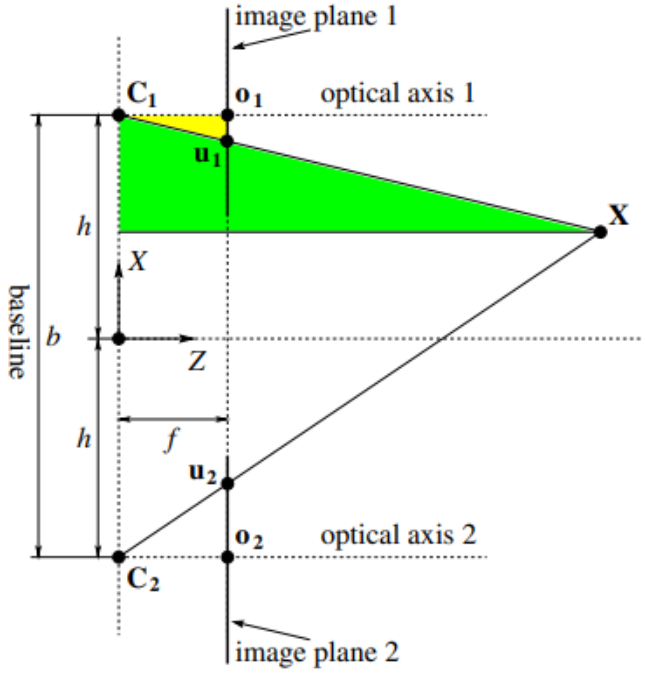


Figure 23. Geometry of standard stereo vision (Hajder and Chetverikov [58])

For triangulation, it is necessary to know [58] [59]:

- the **baseline, b**: which is the distance between the two camera's center of projection,
- the **focal length,** the distance between the center of projection and the image plane,
- (u_1, v_1) and (u_2, v_2) coordinates of **two correspondances** in the image coordinate system.
- and the **disparity**, which is the point location difference between images, i.e. in Figure 23. $d = u_1 - u_2$

By using the method of similar triangles on the parameters that can be seen on Figure 23, the relationships are as follows:

$$\frac{u_1}{f} = \frac{h - X}{Z}$$

$$-\frac{u_2}{f} = \frac{h + X}{Z}$$

$$v_1 = v_2$$

From these correspondences, the equations of the 3D coordinates are as follows:

$$X = \frac{b\,(u_1 + u_2)}{2\,d}$$

$$Y = \frac{b\,v_1}{d} = \frac{b\,v_2}{d}$$

$$Z = \frac{2\,h\,f}{u_1 - u_2} = \frac{b\,f}{d}$$

where $d = u_1 - u_2$ is the disparity.

2.6 ZED 2 stereo camera

In this section the main attributes of the ZED 2 stereo camera will be presented, with a strong emphasis on those characteristics that are useful for the change detection application. The examples and figures presented in this section are the results of the author's tests on the camera, with a few exceptions where the figures are from the manufacturer's website (in this case, the captions of the images will indicate the source).

Figure 24. ZED 2 stereo camera
(ZED 2 Camera Datasheet [1])

2.6.1 Requirements

The ZED 2 is Universal Video Class (UVC) compliant, which means that the device is able to stream its images and videos on any Windows or Linux platform. In order to use every feature of the camera, the developers recommend the specifications [60] found in the table below. It is meaningful to mention that some of core features would still be accessible even if the GPU specifications are not met. However, they would function with decreased abilities.

Table 2. Requirement specifications of the ZED 2 stereo camera

	MINIMUM	**RECOMMENDED**	**EMBEDDED SYSTEMS**
Processor	Dual-core 2,3GHz	Quad-core 2,7GHz or faster	Jetson Nano, TX2, Xavier
RAM	4GB	8GB	8GB
Graphics Card	NVIDIA GPU with **Compute Capabilities > 3**	GTX1060 or higher	Nano, TX2, Xavier
USB port	USB 3.0		
Operating System	Windows 10, Ubuntu 16.04, 18.04		L4T (Jetpack)

To meet the requirements and to be able to use all the features of the camera, a computer with **Intel Core i7 processor @2.7 GHz, 32 GB RAM and NVIDIA GeForce GTX 980 GPU with CUDA driver v11.1** will be used for the thesis.

The ZED cameras come with a software development kit called **ZED SDK**. Large videos can be recorded with H.264, H.265 or lossless compression. The videos are saved as SVO files,

which is a ZED-specific proprietary file format that stores videos along with additional metadata such as timestamp, depth and sensor data (temperature, magnetometer, barometer gyroscope, accelerometer). With SVO files, ZED API will behave as if a ZED was connected and live feed was available [3]. Recording videos are possible in four video modes, seen in the figure below.

VIDEO MODE	OUTPUT RESOLUTION (SIDE BY SIDE)	FRAME RATE (FPS)	FIELD OF VIEW
2.2K	4416x1242	15	Wide
1080p	3840x1080	30, 15	Wide
720p	2560x720	60, 30, 15	Extra Wide
WVGA	1344x376	100, 60, 30, 15	Extra Wide

Figure 25. Available resolutions of the ZED 2 stereo camera [61]

The ZED SDK can be installed without having CUDA (Compute Unified Device Architecture, a parallel computing platform and application programming interface by NVIDIA) installed on the computer. However, image acquisition and depth extraction cannot be used without it. The reason for this is that the underlying processes are entirely based on CUDA. Without it, only left and right video streams of the camera can be captured, but without the GPU-reliant compression modes, which would result in recordings of huge size.

2.6.2 Depth sensing

The ZED 2 stereo camera uses stereo vision to estimate the depth of a scene. The physical parameters of the camera can be found in the datasheet [1], however it is worth to mention the most important ones:

- field of view (FOV): up to 120°
- baseline: 120 mm
- focal length: 2.12mm
- weight: 166 g
- dimensions 175 x 30 x 33 mm

The ZED 2 stereo camera can estimate depth between 0.3 and 20 meters, with a reported accuracy of <1% up to 3 meters and <5% up to 15 meters. These reported accuracies will be measured and tested in later sections.
The ZED SDK provides two modes for depth sensing: *standard* and *fill*. The *standard* mode has accurate distance measurements and is faster. However, it also has holes due to visual occlusions. This mode is recommended for autonomous applications, because of its speed and accuracy. The *fill* mode provides a dense depth map by filling up the holes and occlusions

and also applies filtering to improve edges. It is more recommended for mixed reality and VR applications [61].

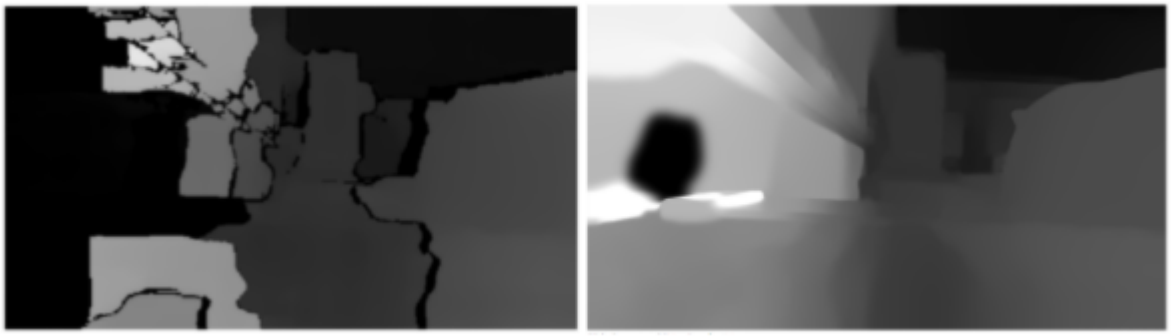


Figure 26. Example for the standard and fill depth sensing modes of the same scene (author's test result)

There are three depth modes that are available for developers. The *ultra* mode offers the largest depth range and is the most accurate but requires more GPU memory and computation power. The *quality* mode is designed to output smoother surfaces because it uses a strong filtering stage. The *performance* mode is optimized for speed and is recommended when an application requires a lot of resources [61].

In the camera tests and in the change detection application the ***ultra* depth mode** will be used along with the ***standard* depth sensing** mode.

2.6.3 Positional tracking

The ZED 2 camera uses a proprietary ***visual SLAM*** to estimate its position relative to a reference frame. This reference frame can either be the ***world frame*** or the previous ***camera frame***. The origin of the camera frame is the left "eye" of the stereo camera and can be used to get the relative pose between two consecutive camera views. The origin of the world frame is where the camera started the recording from. This point is stationary and does not change throughout the motion tracking of the camera. The type of the coordinate system can be selected from the following list [61]:

- Right handed, y-down (default)
- Left handed, y-up
- Right handed, y-up
- Left handed, z-up
- Right handed, z-up

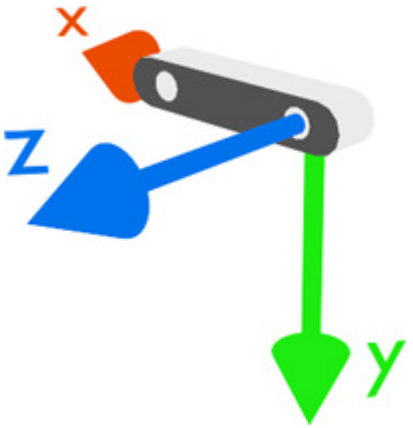


Figure 27. Example for the ZED coordinate system - Right handed, y-down (default) [61]

The accuracy of the positional tracking ability of the camera will be tested and reported in later sections.

2.6.4 Area memory

During recording, the camera continuously stores information about its surroundings in the form of key images and contextual information [61]. The term *area memory* refers to the capability of the camera that allows the device to relocate itself when it revisits an already visited area, thus achieving improved positional tracking by correcting drift. This solution is called **loop-closure** and is one of the key parts of visual SLAM algorithms.
Area memory can also be used to enable the camera to start searching for spatial similarities between the current and a previously recorded area. Once the search is successful, the positional tracking of the current recording will start using the previously recorded world frame as reference frame, thus achieving a **shared common reference frame** between sessions. This is a very useful feature, because it eliminates the need for point cloud registration. The generated point clouds, after a successful relocalization, share a common reference frame, thus they are already registered with respect to each other. Without area memory, different devices or devices in different recordings would not understand their absolute position in the world, and would track their position with respect to their own starting point.
As an example, see Figure 28. On the left, the 3D mesh of a room can be seen. While exploring the room and obtaining the 3D mesh, the area memory with key images corresponding to that room was saved into a file with **.area** extension. When revisiting the room, the camera used the previously saved area memory file to relocate itself and change its world coordinate frame to the previously recorded world frame, thus having a common reference frame. In the second recording, more of the environment was discovered compared to the first recording. See the result of the relocalization on the right: The 3D meshes of the first and second recording are loaded and shown together.

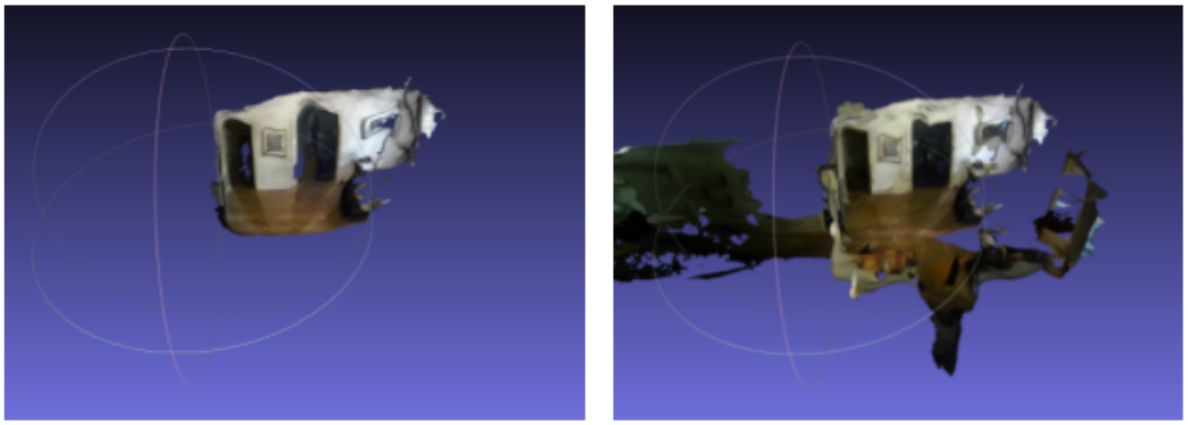


Figure 28. Common reference frame using area memory
On the left: 3D mesh of the first recording,
on the right: the 3D meshes of the first and second recording displayed together
(author's test result)

The accuracy of relocalization using area memory will be tested in later sections.

2.6.5 Spatial mapping

Spatial mapping, also called 3D reconstruction, is the ability to create a three dimensional map of the environment. A spatial map, using the ZED SDK, can either be a ***mesh*** or a ***fused point cloud***. A three dimensional mesh represents the scene with a 3D model consisting of polygons: usually quadrangles or triangles. The ZED SDK can use 2D images recorded during mapping to give texture to the 3D model surface [61].

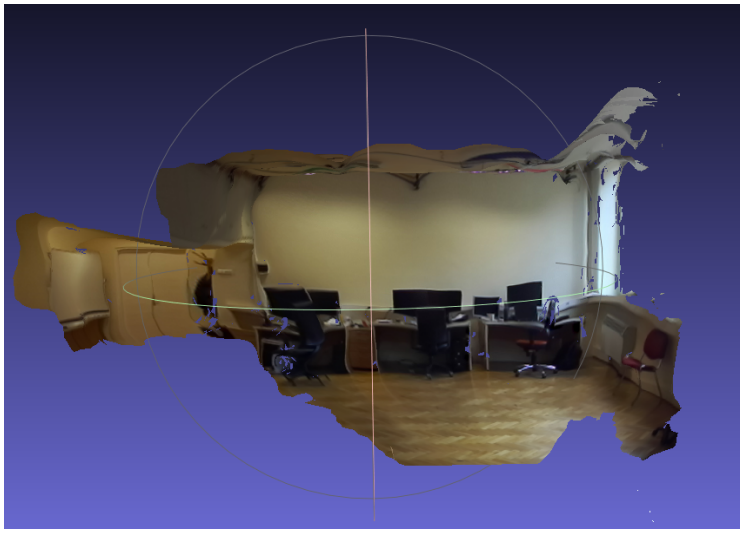


Figure 29. 3D textured mesh of a room from a stationary position
(author's test result)

The fused point cloud represents the geometry of the scene by a set of colored 3D points.

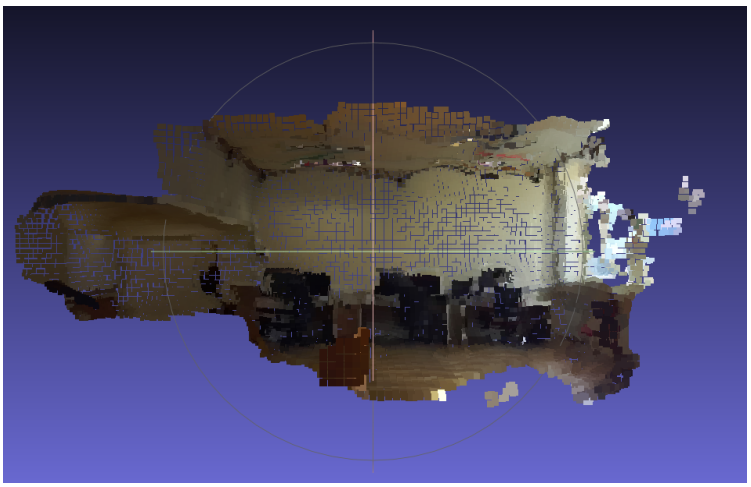


Figure 30. 3D point cloud of a room from a stationary position
(author's test result)

The 3D mesh can be filtered after a recording to improve the performance of an application. Figure 31. shows the three filtering modes. The *high* and *medium* modes decimate the mesh polygon count to optimize the 3D model. *Low* mode also filters the mesh from outliers.

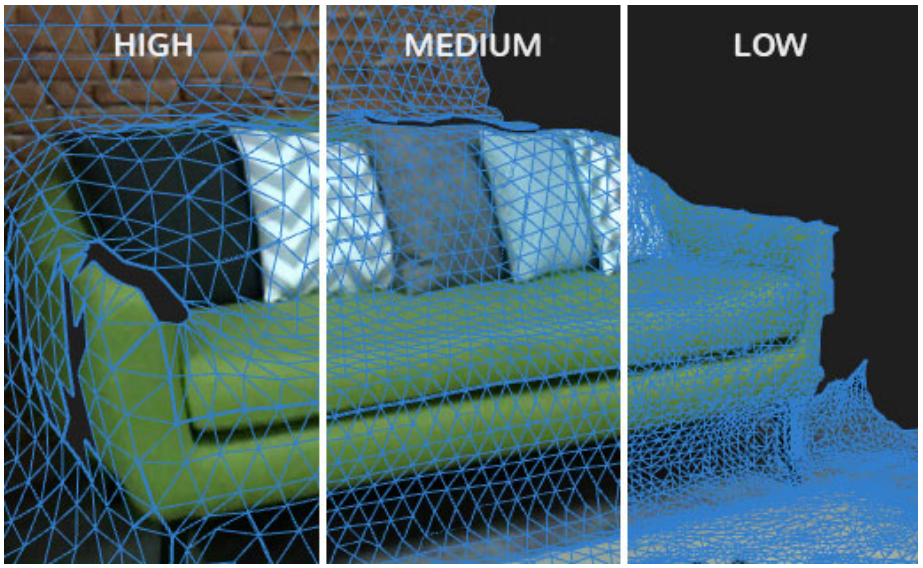


Figure 31. The mesh filtering modes of the ZED SDK [61]

Both the resolution and the range of spatial mapping can be set before a recording. Higher the resolution, the more detailed the spatial mapping is. Resolution can be set between 1 cm and 12 cm. The mapping range can be set manually or can be set using predefined presets, such as:

- NEAR: estimates depth up to 3.5 meters
- MEDIUM: estimates depth up to 5 meters
- FAR: estimates depth up to 10 meters - this limitation is introduced, because the depth ***accuracy decreases quadratically*** with distance, thus it is less reliable

- $d_z = \frac{z^2 d_e}{f b}$, where d_z is the depth error in meters and d_e is the disparity error in pixels
- Depth accuracy can also be affected by homogenous and textureless surfaces such as e.g. white walls or specular areas. These surfaces usually cause instability in depth measurements [61].

Apart from being able to generate a 3D spatial map, the camera is also able to do **plane detection** using 3D information of the environment. The result of the detection is a plane model with information such as: plane pose relative to the global reference frame, plane equation and 3D mesh of the plane.

Figure 32. 3D plane floor plane detection
(author's test result)

2.6.6 Object detection and tracking

The ZED 2 stereo camera is able to detect objects in an image. Using depth sensing and 3D information of the environment, the camera is also able to return not only 2D, but also 3D location of detected objects. The detected objects can also be tracked in the environment over time. This means that an object can keep the same object ID throughout a recording [61].

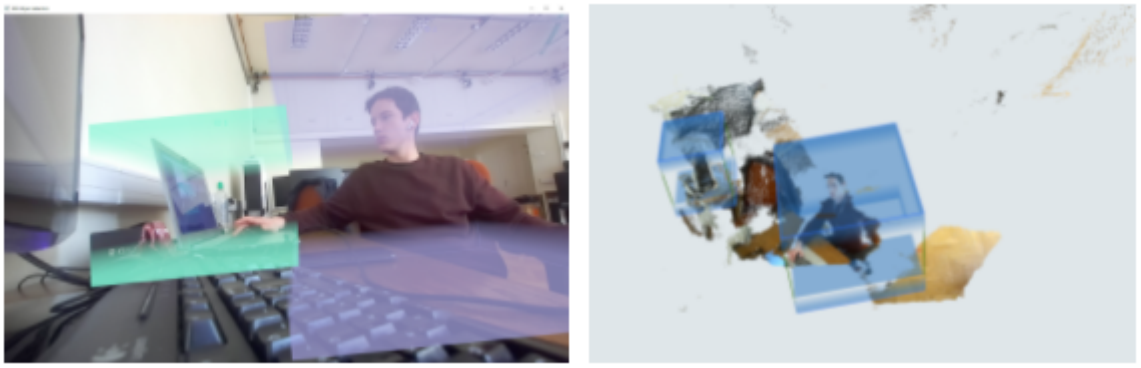


Figure 33. ZED 2 object detection module with 2D and 3D bounding boxes
(author's test result)

The ZED SDK determines which pixels of the image belong to an object and creates the 2D bounding boxes around the detected object based on this information. With the help of the depth map that was generated using the stereo images, the SDK can also estimate the 3D bounding box of the detected objects.
Each detected object has the following structure associated with them:

OBJECT DATA	DESCRIPTION	OUTPUT
ID	Fixed ID for identifying an object over time.	`Integer`
Label	Identifies the object type.	`Person`, `Vehicle`
Tracking state	Defines if an object is currently tracked or lost.	`Ok`, `Off`, `Searching`, `Terminate`
Action state	Defines if an object is currently idle or moving.	`Idle`, `Moving`
Position	Provides the 3D position of the object according to the camera as a 3D vector (x,y,z).	`[x, y, z]`
Velocity	Provides the velocity of the object in space as a 3D vector (x,y,z).	$[v_x, v_y, v_z]$
Dimensions	Provides the width, height and length of the object.	`[width, height, length]`
Detection confidence	A lower confidence means the object might not be localized perfectly or that its label is uncertain.	`0 - 100`
2D bounding box	Defines the box surrounding the object in the image represented as four 2D points.	`Four pixel coordinates`
3D bounding box	Defines the box surrounding the object in space represented as eight 3D points.	`Eight 3D coordinates`
Mask	Provides the pixels which really belong to the object and those of the background.	`Binary mask`

Figure 34. The stored structure of a detected object [61]

Currently, the ZED 2 object detection model can detect the following 22 object classes:

1. person
2. bicycle
3. car
4. motorbike
5. bus
6. truck
7. boat
8. backpack
9. handbag
10. suitcase
11. bird
12. cat
13. dog
14. horse
15. sheep
16. cow
17. cellphone
18. laptop
19. banana
20. apple
21. orange
22. carrot

The architecture of the ZED 2 object detection algorithm is proprietary and only the output of the detection is accessible to the developers. Therefore, in order to assess the performance and capability of the ZED2 object detection algorithm, a qualitative assessment will be presented, in which the ZED algorithm will be benchmarked against the state of the art object detection algorithms with approximately 50 pictures taken by the ZED 2 stereo camera. The test will be described in the *object detection performance measurement* section.

It is also possible to deploy custom object detection algorithms using the ZED 2 compatible PyTorch, TensorFlow or YOLO frameworks. This allows developers to benefit from the fast and accurate spatial mapping capabilities of the ZED 2 camera and also use their own object detection algorithms for various projects.

Apart from object detection, the ZED 2 stereo camera is also able to detect numerous keypoints of people (in total, 18 keypoints). Furthermore, the camera can track the detected skeletons of people, as seen on Figure 35. below, enabling the developers to use this information for 3D motion estimation.

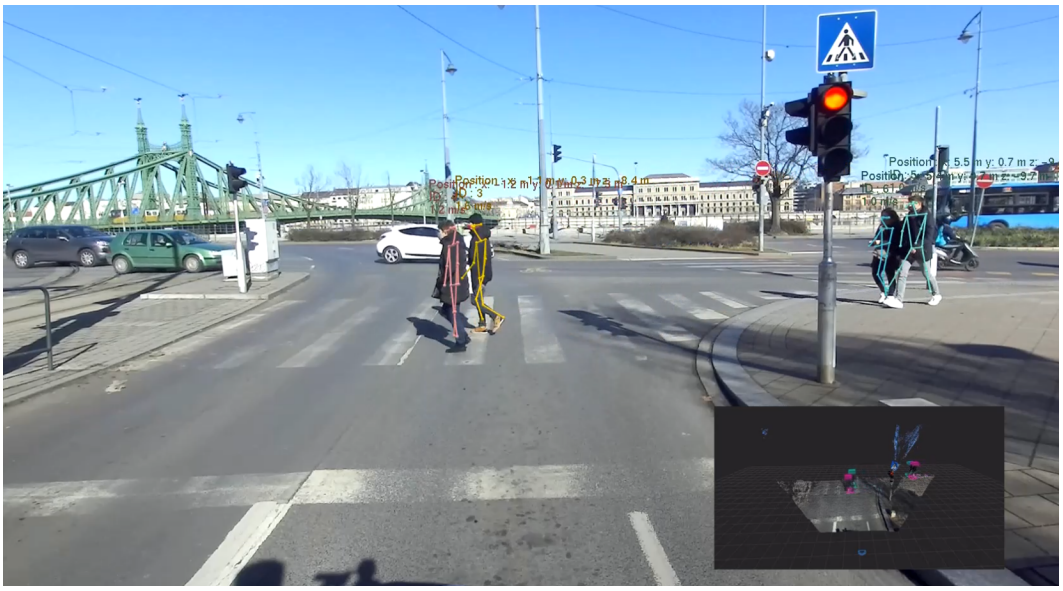


Figure 35. Body tracking with the ZED 2 stereo camera
(author's test result)

2.6.7 Compatibility

The ZED 2 stereo camera can be interfaced [62] with many third-party libraries and environments. For example:

- OpenCV
- PyTorch
- YOLO
- Unity
- GStreamer
- ROS
- TensorFlow
- Docker
- Unreal
- Matlab
- OpenGL
- PCL
- CUDA
- OpenPose
- Aruco

3. Tests on the ZED 2 stereo camera

In this section, the tests conducted on the camera are described. The tests measure the camera's depth accuracy, depth measurement consistency, visual SLAM accuracy, relocalization accuracy with area memory and object detection performance.

3.1 Depth accuracy measurement using a 3D checkerboard

Stereo cameras estimate the depth of a scene using triangulation. In the standard stereo vision section, the computation of depth has already been described:

$$Z = \frac{b\,f}{d}$$

where $d = u_1 - u_2$ is the disparity.

With the test system proposed in this section, the accuracy of depth calculation will be measured.

There are several techniques to measure the depth measurement accuracy of stereo cameras. Hassan et al. [63] used a checkerboard perpendicular to the flat surface of the floor. The board is similar to the ones that are usually used in the calibration process of cameras. In the test, the camera is used to detect the four corners of the board and then these points are used to calculate the center of the board. Using triangulation, it is possible to estimate the depth of this center point. Finally, the test compared the values obtained by triangulation with the ground truth values acquired using a laser-type measuring device.

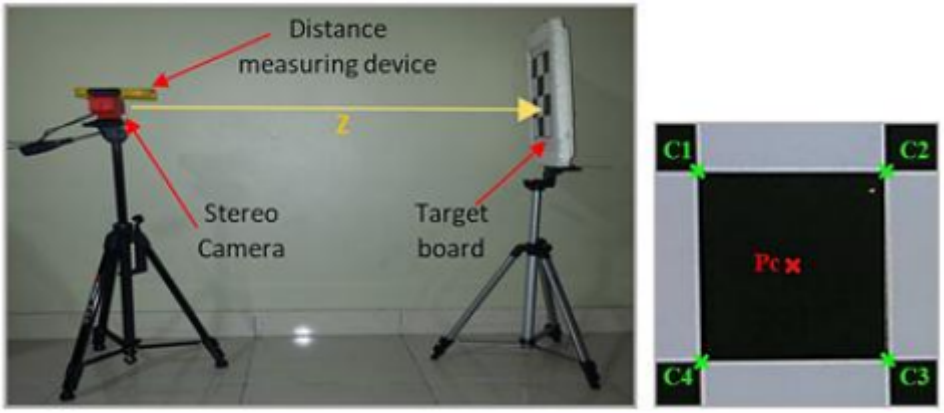


Figure 36. The test system proposed in Hassan et al. [63]

The technique proposed by Hassan et al. [63] is useful for measuring depth accuracy of stereo cameras, however, it requires an additional hardware (laser-type measuring device) to be available for the testers.

Nonetheless, the corner detection algorithm used in the paper, that is proposed by Geiger et al. [64], is very helpful for the detection of checkerboard corners. Many of the publications presented later in this section make use of this corner detection architecture. The authors of [64] introduced an algorithm that is more robust than Harris or Shi-Tomasi corner detectors.

In short, they use two different n × n corner prototypes to compute a corner likelihood score at each pixel in the image. This score is acquired by doing convolution between the kernels of these prototypes and the image.

(a) Corner Prototype 1 (b) Corner Prototype 2

Figure 37. Corner prototypes proposed by Geiger et al. [64]

Following the corner likelihood map generation, they apply non-maxima suppression and verify the corners by evaluating their gradient in a local neighborhood. At the end, using a weighted orientation histogram they find the two main edge orientations. With the help of edge orientations and the corner likelihood score, the corners of a checkerboard can be found.

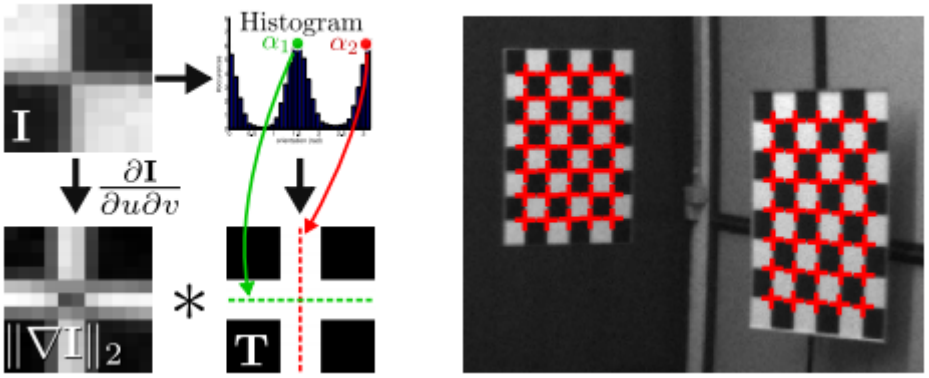


Figure 38. Image filtering and corner verification,
and the result of corner detection proposed by Geiger et al. [64]

The test method proposed by Kytö et al. [65] uses a checkerboard-system consisting of 9 plain checkerboards on levels at different distances with respect to each other.

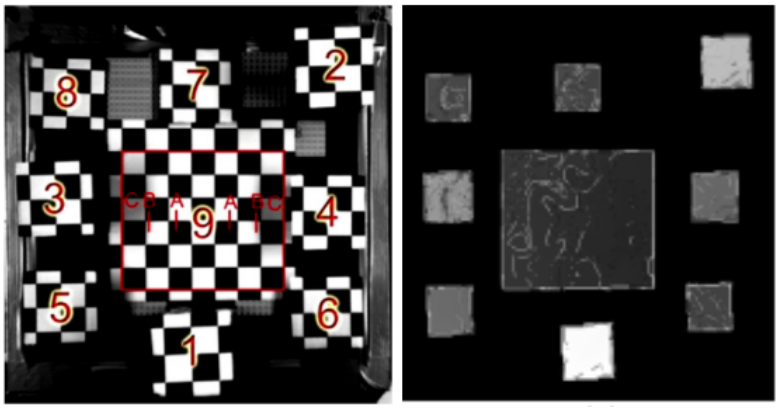


Figure 39. Test system proposed by Kytö et al. [65]

For ground-truth generation, the Matlab Camera Calibration Toolbox [66] was used, which detects corners, using the Harris corner detector, with 0.1 pixel accuracy. The ground truth disparity map was created based on these detected corners. The depth accuracy measurement

is done by comparing the ground truth and stereo reconstructed disparity maps pixel-wise and returning the result as a sum of absolute differences.
This test method is more tailored towards the comparison between the human eye and stereo cameras and requires a complex system which is often not easy to create.

In this thesis, the test method described by Fernandez et al. [67] and Ortiz et al. [68] will be used to assess the depth accuracy of the ZED 2 stereo camera. Both of these papers used the ZED stereo camera, which is the previous ZED model before the ZED 2 stereo camera. This allows comparison between the reported results in the papers and the measured data in this thesis.
The proposed method is based on a 3D checkerboard similar to the one shown in Figure 40.

Figure 40. 3D checkerboard, used for depth accuracy measurement (Fernandez et al. [67])

The test process is the following:

1. Before the test, the **corners of the 3D checkerboard are measured manually** and the **reference/ground truth 3D point cloud** is constructed from the measurements. In the thesis, a similar checkerboard is used to the one displayed above. There are 24 squares on each side of the checkerboard and the sides of each square on the checkerboard are 4,5 cm long.
2. **The 3D checkerboard is placed at different distances** from the stereo camera. Since the reported depth range of the camera is between 0.3 and 20 meters, the checkerboard is placed at every meter between 1 and 20 meters.
3. Using the **corner detection algorithm** (Geiger et al. [64]) described above and the depth data acquired by triangulation of the stereo camera, a **3D point cloud can be obtained**. The corner detection algorithm returns the (x, y) 2D pixel locations for each detected corner in the left stereo camera image, from which the (X, Y, Z) 3D coordinates can be triangulated.
4. **The reference point cloud and the point cloud obtained by the stereo camera are registered** using optimal point registration with Singular Value Decomposition (SVD) [69].

5. Finally, by determining the Euclidean-distances between corresponding point-pairs in the point clouds, the **RMS error is calculated**. The squared difference between two corresponding points is calculated as follows:

$$SquaredDifference = || q_i - (R\,p_i + T) ||^2 \quad \text{, where}$$

q_i is a 3D point in the 3D reference point cloud,

p_i is a 3D point in the 3D stereo reconstructed point cloud

and R and T are the corresponding rotation and translation matrices applied to p_i during point cloud registration.

For the RMS error calculation, first the above described equation is applied to every corresponding point pair in the point clouds:

$$SummedSquaredDifference = \sum_{i=1}^{N} || q_i - (R\,p_i + T) ||^2 \quad \text{, where}$$

N is the number of points in the point clouds.

Then the square root of the arithmetic mean is calculated using the above described summation:

$$RMS_{error} = \sqrt{\frac{1}{N} SummedSquaredDifference}$$

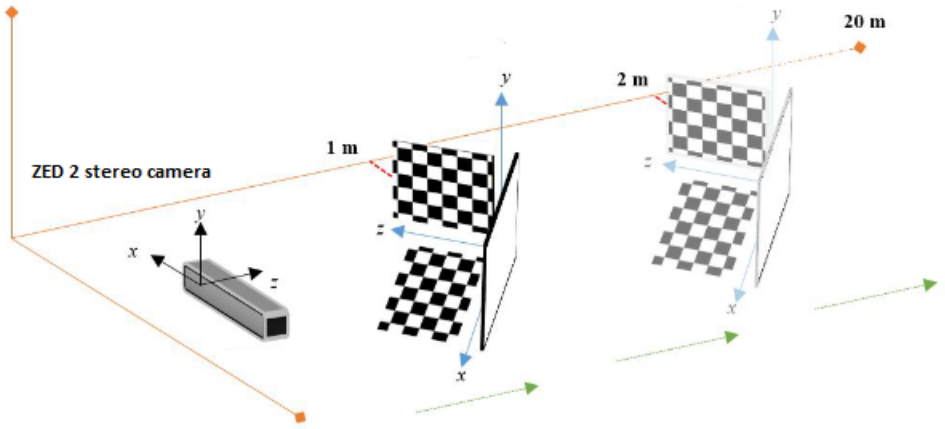


Figure 41. Placing the 3D checkerboard at different distances from the stereo camera (Ortiz et al. [68])

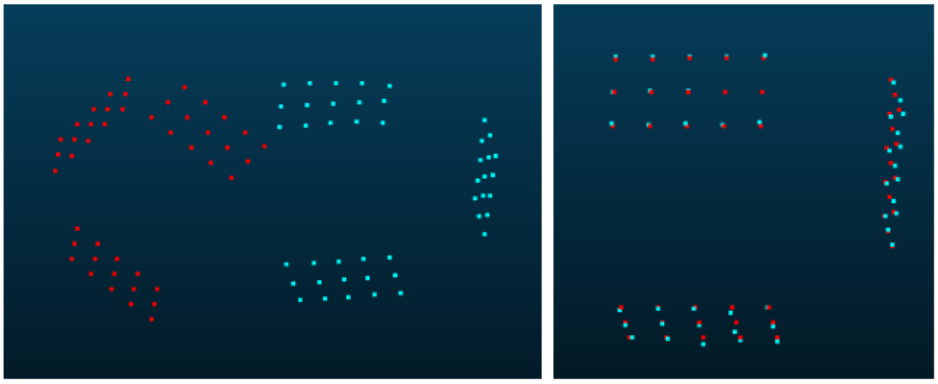


Figure 42. Example of two point clouds (the reference and the stereo reconstructed) before and after registration

The measurements are done in all four resolutions of the camera, with the corresponding maximum possible frame rate of each resolution:

1. **WVGA** -Wide Video Graphics Array - 672×376 pixel and 100 FPS
2. **HD 720p** - Standard HD - 1280×720 pixel and 60 FPS
3. **HD 1080p** - Full HD - 1920×1080 pixel - 30 FPS
4. **HD 2.2K** - 2208×1242 pixel - 15 FPS

The calculated RMS error values for each resolution and distance are visualized in the graph below. In the interest of better representing the RMS error for each resolution, curve fitting is applied on the measured data. It was empirically determined that optimally the exponential function fits the measured data the best.
The exponential model of the RMS depth error is as follows:

$$RMSE_{depth} = a * exp^{b * depth}, \text{ where}$$

a and *b* are the coefficients of the exponential model and
depth is the distance between the camera and the 3D checkerboard

In the numerical analysis of curve fitting, there are two mainly used statistical values for assessing the quality/goodness of the fitting [68]:

1. SSE (Sum of Squared Errors)
2. R*squared*

SSE measures the difference between the measured data and the model with the equation as follows:

$$SSE = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \text{ where}$$

N is the number of data points,
y_i is a measured data point and

$\hat{y}_i$ is the corresponding data point of y_i, obtained using the curve fitting model.

R*squared* measures how well the observed data is replicated by the model with respect of the variation of the data by using the equation below:

$$R_{squared} = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}, \text{ where}$$

N is the number of data points,
y_i is a measured data point,

$\overline{y}$ is the mean of the measured data points and

$\hat{y}_i$ is the corresponding data point of y_i, obtained using the curve fitting model

In general, the closer the value of SSE is to 0 and the closer the value of R*squared* is to 1, the better the curve fitting is [68].

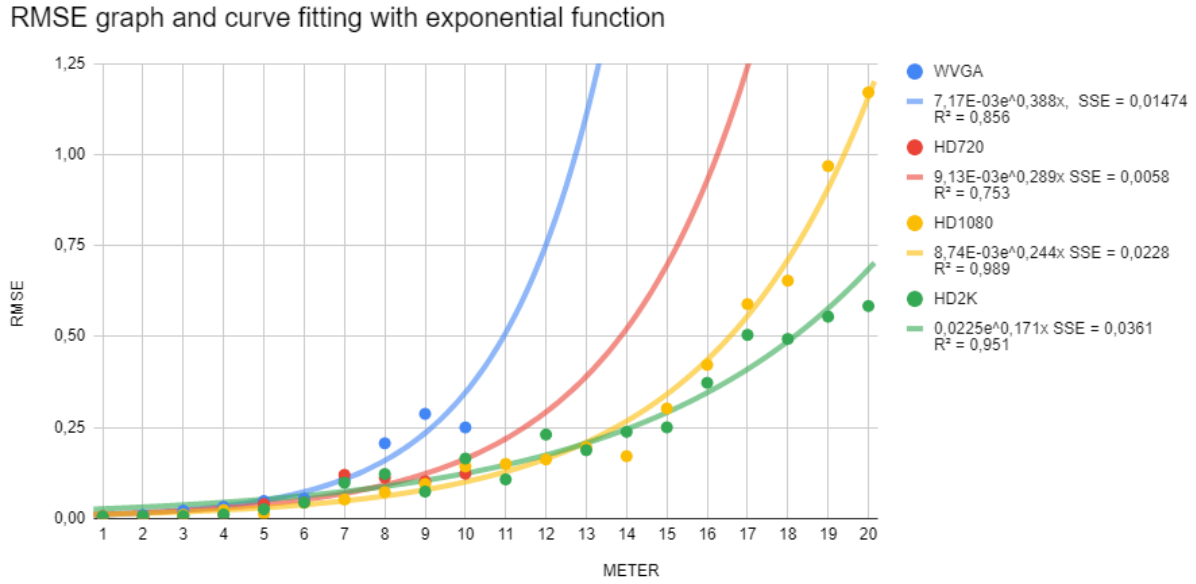


Figure 43. The calculated RMS error in each resolution and distance

The calculated SSE and R*squared* values are visible on the graph above, however, for better transparency they are included in the table below as well.

Table 3. SSE and R*squared* values between 1 and 20 meters

Resolution	**Exponential model coefficients**	**SSE** $[m^2]$	**R*squared***
WVGA	a = 0.00717, b = 0.388	0.0147	0.856
HD720	a = 0.00913, b = 0.289	0.0058	0.753
HD1080	**a = 0.00874, b = 0.244**	**0.0228**	**0.989**
HD2K	a = 0.0225, b = 0.171	0.0361	0.951

It can be noticed on the graph that the two lower resolutions, WVGA and HD720, do not have measurements associated with distances between 10 and 20 meters. The reason for this is that at distances over 10 meters the quality of the resolution was not good enough to distinguish the 3D checkerboard corner points. Consequently, the RMS error could not be calculated.
There was another difficulty in the detection of checkerboard corner points: With distances above 15 meters, the corner detector proposed by Geiger et al. [64] had difficulties in detecting all the corner points in the two higher resolutions. To solve the problem, the corners that were missed by the corner detector were manually complemented.

According to the table above, the HD720 resolution has the lowest SSE which would mean that potentially this exponential function fits the RMS error data the best, but a relatively low R*squared* value. The reason for this is that on HD720 resolution the RMS error was only calculated up to 10 meters where the depth accuracy is, in general, higher. For better comparison between the resolutions, the table below contains the SSE and R*squared* values for the RMS error up to only 10 meters in distance.

Table 4. SSE and R*squared* values between 1 and 10 meters

Resolution	**Exponential model coefficients**	**SSE** $[m^2]$	**R*squared***
WVGA	a = 0.00717, b = 0.388	0.0147	0.856
HD720	a = 0.00913, b = 0.289	0.0058	0.753
HD1080	**a = 0.00426, b = 0.352**	**0.0003**	**0.984**
HD2K	a = 0.0564, b = 0.346	0.0055	0.805

By observing these values, it can be seen that the HD1080 resolution achieves the highest R*squared* value and the lowest SSE value. This means that the exponential depth error model of the data recorded in HD1080 resolution fits the measured RMS error the best.

In the table below, the mean and variance of the RMS error estimated by the exponential model can be seen for the entire range of 20 meters.

Table 5. Mean and variance RMSE values between 1 and 20 meters meters

Resolution	**Mean of RMSE** [m]	**Variance of RMSE** [m]
WVGA	2.6129	20.3768
HD720	0.5870	0.6848
HD1080	**0.2637**	**0.1072**
HD2K	0.2116	0.0387

The reason for such high numbers in all resolutions is that the camera does not estimate accurate depth information at distances greater than approximately 10 meters. For instance, if the same table is constructed (see the table below), but with distances only up to 10 meters, the difference (compared to the previous table) between the mean and variance values of RMS error estimated by the exponential model can be seen.

Table 6. Mean and variance RMSE values at distances between 1 and 10 meters

Resolution	**Mean of RMSE** [m]	**Variance of RMSE** [m]
WVGA	0.1023	0.0112
HD720	0.0571	0.0026
HD1080	**0.0471**	**0.0021**
HD2K	0.0594	0.0032

The HD1080 resolution has the lowest mean and variance value of RMS error at distances between 1 and 10 meters. Therefore, this is the camera resolution that is going to be used in the change detection application, because this resolution seems to be the most reliable in terms of depth accuracy.

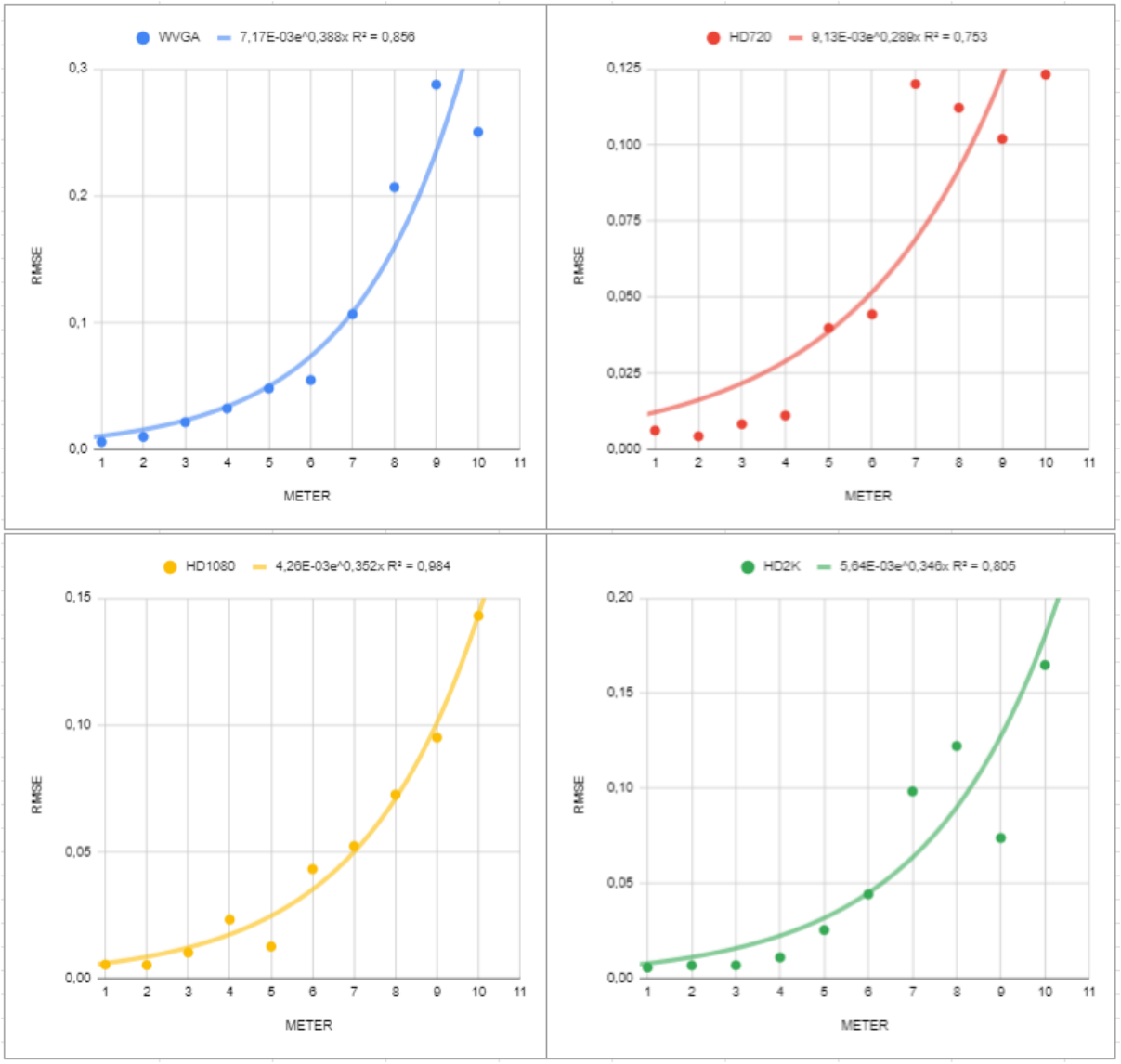


Figure 44. The calculated RMS error obtained at distances up to 10 meters with all resolutions

It is meaningful to compare the results of the test in this thesis and the results reported by Ortiz et al. [68]. The test environment in Ortiz et al. [68] is indoors with artificial light in order to achieve the maximum performance of the camera. However, due to the lack of a similar option, the depth accuracy test of this thesis was carried out in an outdoor environment with natural light. This could be an explanation for the better results reported by Ortiz et al. [68], visible in the table below.

Nonetheless, carrying out the experiment outdoors and reporting the outdoor measured results can be meaningful for outdoor applications that cannot benefit from the lab-like environment.

Table 7. SSE and R*squared* values between 1 and 20 meters, reported by Ortiz et al. [68]

Resolution	**Exponential model coefficients**	**SSE** $[m^2]$	**R*squared***
WVGA	a = 0.0115, b = 0.2986	0.0004	0.970
HD720	a = 0.0184, b = 0.2106	0.0044	0.964
HD1080	a = 0.0106, b = 0.2215	0.0031	0.960
HD2K	**a = 0.01805, b = 0.1746**	**0.0026**	**0.968**

Table 8. Mean and variance RMSE values between 1 and 20 meters, reported by Ortiz et al. [68]

Resolution	**Mean of RMSE** [m]	**Variance of RMSE** [m]
WVGA	0.83	1.54
HD720	0.30	0.12
HD1080	0.21	0.06
HD2K	**0.17**	**0.02**

By looking at the tables above, it is discernible that the HD2K resolution has the lowest mean and variance in the RMS error measurements, and the HD1080 resolution is the second according to these values. However, there are two advantages that speak for using the HD1080 resolution over the HD2K resolution:

1. In HD2K the ZED cameras can only record at 15 FPS, while in HD1080 the maximum is 30 FPS, allowing better recordings.
2. According to Ortiz et al. [68], the depth map acquisition/computation time is higher in HD2K (12.867 ms) compared to HD1080 (9.640 ms), which is more beneficial for real-time applications.

The reported depth accuracy by the manufacturer [1] of the ZED 2 stereo camera is:

- < 1% up to 3 meters
- < 5% up to 15 meters

If the RMS error values, estimated by the exponential model between 1 and 15 meters, are calculated into percentage values by dividing them with their corresponding distances in meters and multiplying with 100, the graph below can be constructed.

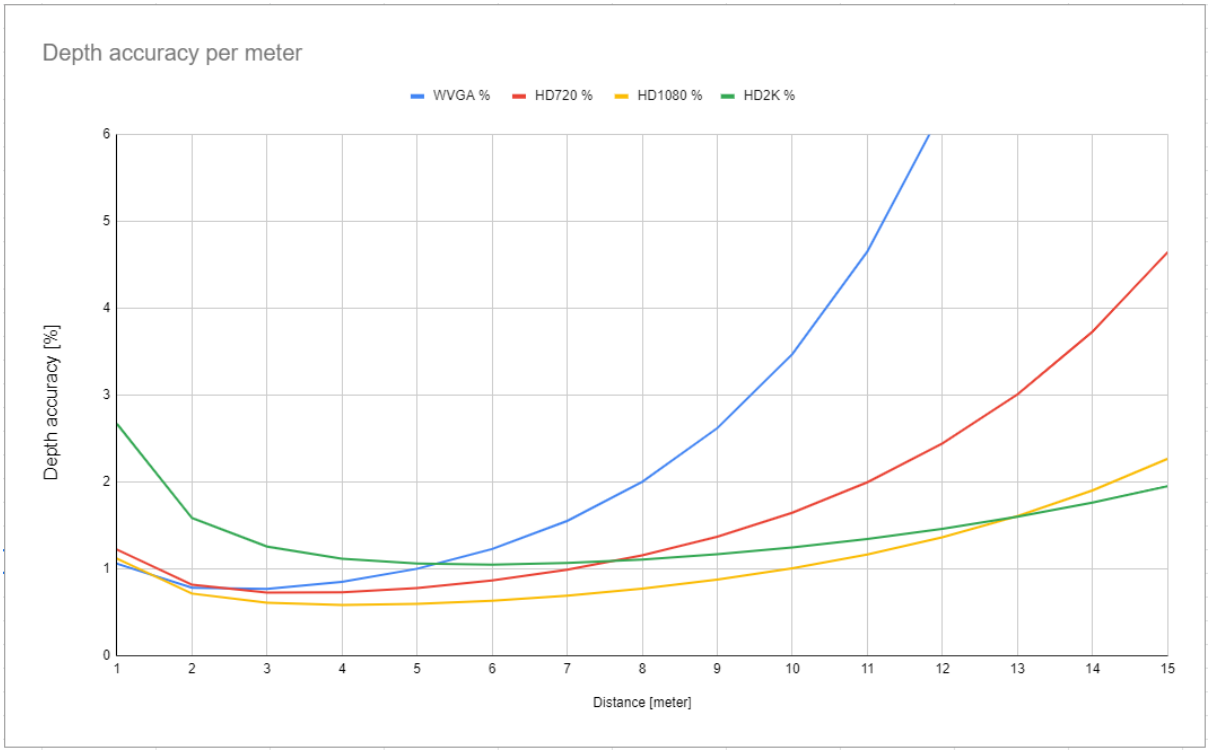


Figure 45. Depth accuracy in percentages estimated by the exponential model between 1 and 15 meters

Apart from the initial bias of the exponential error model at the distance of 1 meter, the accuracy is below 1% for 5 meters for every resolution except for the HD2K. The HD2K resolution never goes below 1%, however, the accuracy calculated in this resolution is below 2% at the distance of 15 meters too. Moreover, in the HD720 resolution, the accuracy is below 1% for 7 meters and in the HD1080 resolution it is below 1% for 10 meters.
The WVGA resolution passes the 5% accuracy threshold already around 11 meters. At a distance of 15 meters, the HD720 resolution has an accuracy of 4.64%, while the HD1080 has 2.26% and the HD2K has 1.95%.

3.2 Depth measurement consistency

The accuracy of the ZED 2 stereo camera was already verified with a 3D checkerboard in the previous section. However, the consistency of the depth measurement was not yet estimated.

Similarly to the test proposed by Ortiz et al. [68], 100 consecutive depth maps were acquired of the same scene in all four resolutions of the camera. In total, there were two scenes captured: one recorded indoors with artificial light and one outdoors with natural light. Both of them have a maximum depth of around 6 meters.

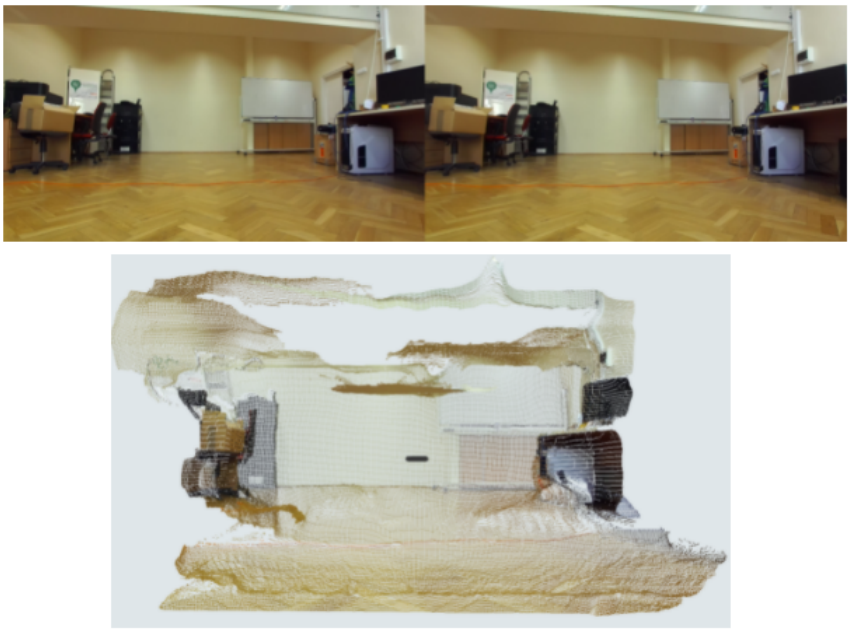


Figure 46. Left and right images and point cloud of the indoor scene

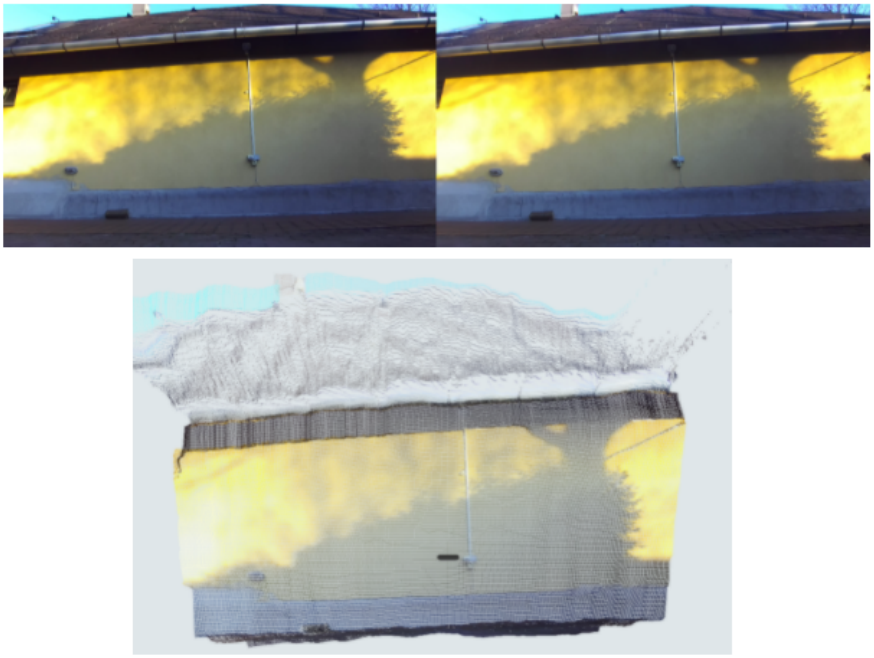


Figure 47. Left and right images and point cloud of the outdoor scene

The depth sensing of the camera assigns values of *NULL* and *INF* to those pixels where it does not estimate depth values. These values have been cleaned from the result. The table below contains how much percentage of the measurement was removed because it did not contain valid data. The last column contains the results reported by Ortiz et al. [68] measuring an outdoor scene with the ZED stereo camera.

Table 9. The percentage of the removed values where the camera did not estimate depth

Resolution	**Indoors** Removed NULL and INF values [%]	**Outdoors** Removed NULL and INF values [%]	**Outdoors** Removed NULL and INF values [%] by Ortiz et al. [68]
WVGA	3.25	3.09	2.09
HD720	3.68	2.48	2.06
HD1080	4.59	2.13	2.43
HD2K	4.21	2.25	2.3

Comparing the results of the tests in this thesis obtained outdoors with the results reported by Ortiz et al. [68], the removed percentage values are in a similar range. There is not much improvement in this sense between the ZED 2 stereo and the ZED stereo camera. The reason for the higher percentage values for the indoor scene could be that the scene contains objects closer to the ZED 2 stereo camera, where the camera does not estimate correct depth values.

To estimate the consistency of the camera, the standard deviations of every pixel in the scene for the 100 consecutive depth maps in all four resolutions of the camera were computed. The results can be seen in the table below. The minimum, maximum, median and mean values of the standard deviations were computed. The heatmaps of the standard deviations are also displayed in the images below for both scenes (Figure 48 and Figure 49).

Table 10. Standard deviations of every pixel in the scene for the 100 consecutive depth maps in all four resolutions for the **indoor scene**

Resolution	Mean [m]	Median [m]	Minimum [m]	Maximum [m]
WVGA	0.0436	0.0207	0	3.1053
HD720	0.0409	0.0296	0	3.3151
HD1080	**0.0361**	**0.0107**	**0**	**3.0960**
HD2K	0.0370	0.0111	0	3.1041

Table 11. Standard deviations of every pixel in the scene for the 100 consecutive depth maps in all four resolutions for the **outdoor** scene

Resolution	Mean [m]	Median [m]	Minimum [m]	Maximum [m]
WVGA	0.3696	0.0267	0	9.8983
HD720	0.0805	0.0139	0	9.1521
HD1080	**0.0336**	**0.0082**	**0**	**6.1823**
HD2K	0.0641	0.0083	0	6.6811

Table 12. Standard deviations of every pixel in the scene for the 100 consecutive depth maps in all four resolutions for the **outdoor scene** reported by Ortiz et al. [68]

Resolution	Mean [m]	Median [m]	Minimum [m]	Maximum [m]
WVGA	not reported	0.0082	0	0.04
HD720	not reported	0.0080	0	0.04
HD1080	not reported	0.0060	0	0.02
HD2K	not reported	0.010*	0	0.90*

* computed on the 75% of the data

Both for the indoor and outdoor scene, the HD1080 resolution proved to be the most consistent one. It has the lowest mean, median and maximum values out of all four resolutions. In the outdoor scene, the WVGA and HD720 resolutions have a much higher maximum value compared to the other two resolutions and the WVGA also has a mean value of around 10 times higher compared to the HD1080 resolution, making it less reliable for depth sensing.

The reported maximum values by Ortiz et al. [68] are much lower compared to the maximum values of the test carried out in this thesis in the outdoor setting, which could be explained with a more noise-free setup of the environment and a less-crowded scene layout. Although the tests carried out in this thesis report high maximum values, they take up only 1-5% of the measured data.

Nonetheless, the median values of each resolution are in a similar range. The test carried out in this thesis measured a lower median for the HD2K resolution. However, it is still the highest among all the four resolutions.

The test results reported by Ortiz et al. [68] confirm the results of the tests in this thesis. Namely, that the **HD1080** resolution has the lowest minimum-maximum difference, mean and the lowest median value as well, making it the **most reliable and consistent depth sensing camera resolution** of the ZED and ZED 2 stereo cameras.

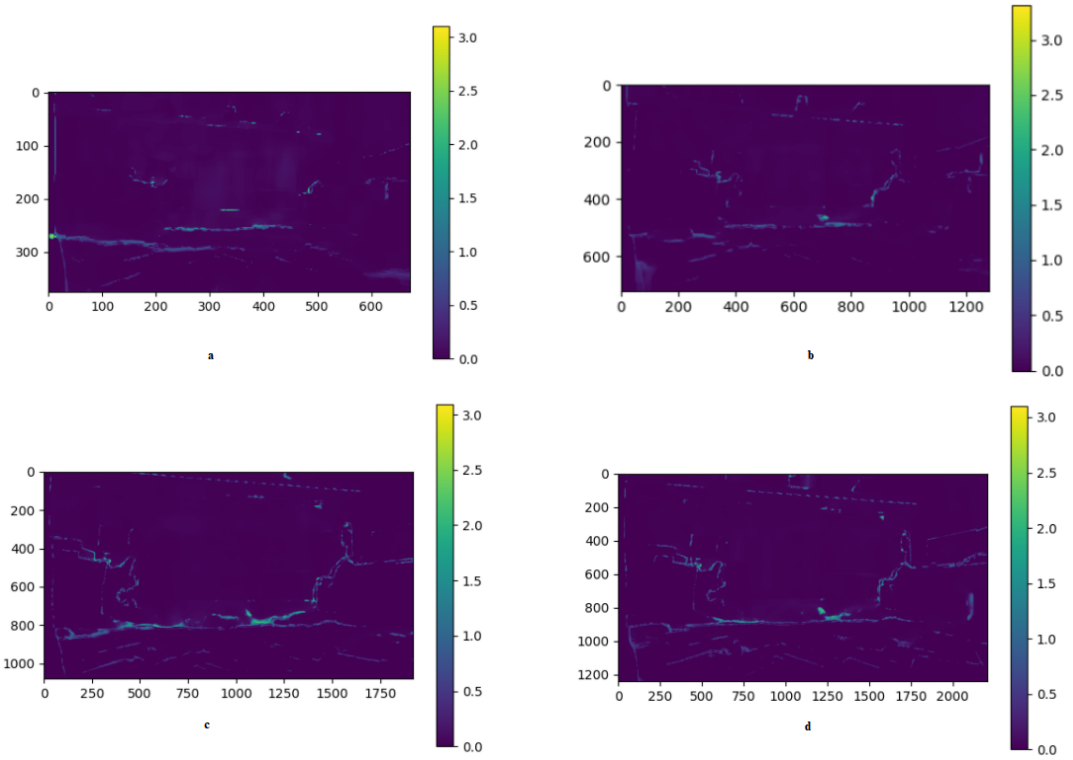


Figure 48. Standard deviation of the **indoor** scene computed for each corresponding pixel for 100 consecutive depth maps, displayed as a heat map
(a) WVGA resolution, (b) HD720 resolution, (c) HD1080 resolution, (d) HD2K resolution

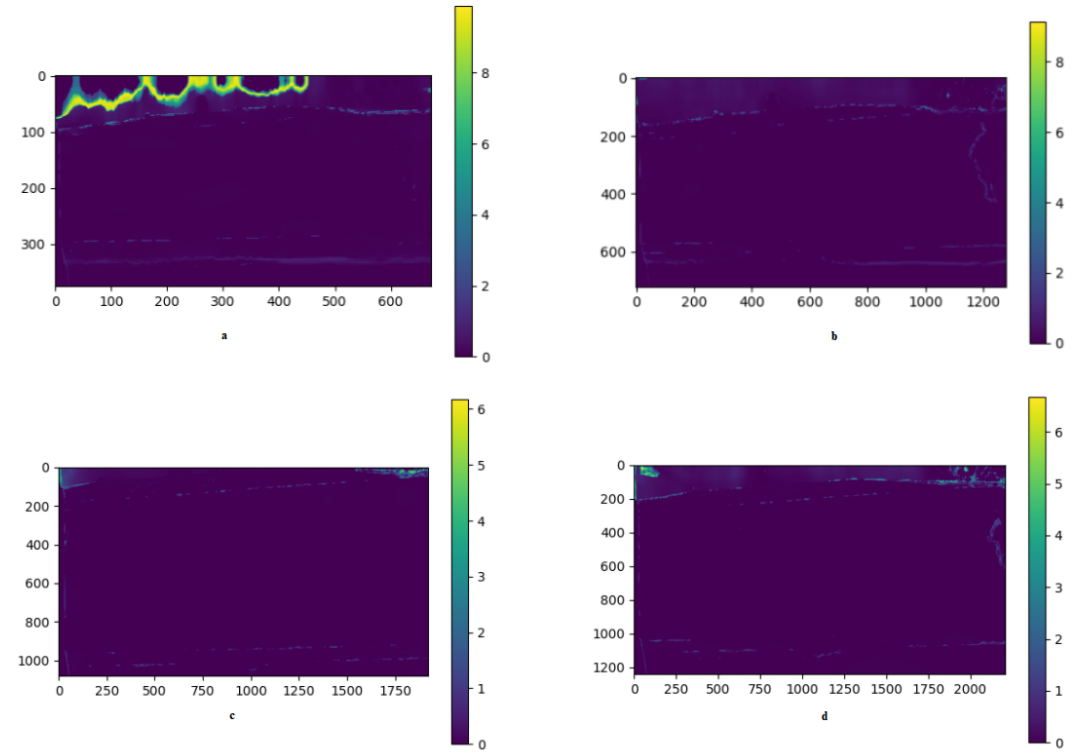


Figure 49. Standard deviation of the **outdoor** scene computed for each corresponding pixel for 100 consecutive depth maps, displayed as a heat map
(a) WVGA resolution, (b) HD720 resolution, (c) HD1080 resolution, (d) HD2K resolution

Table 13. The time needed to obtain the depth map and point cloud of a scene. Calculated by averaging 50 consecutive depth and point cloud retrieval for 4 resolutions with their maximum possible frame rate.

Resolution - FPS	Depth Map Time [ms]	Point Cloud Time [ms]
WVGA- 100 FPS	0.356	0.829
HD720 - 60 FPS	0.960	2.934
HD1080- 30 FPS	2.113	6.396
HD2K - 15 FPS	2.860	8.841

3.3 Positional tracking accuracy measurement

In order to evaluate the positional tracking accuracy and the relocalization accuracy of the ZED 2 stereo camera, the SZTAKI MIMO (MIcro aerial vehicle and MOtion capture) Arena proposed by Rozenberszki and Majdik [70] will be used. The dimensions of the arena are 7 meter x 8 meters x 3.5 meters and has an OptiTrack [71] system set up consisting of 10 cameras. The OptiTrack system uses Prime 13 cameras [72], which have their own FPGA modules for real-time onboard image processing and are connected with Ethernet cables. Every position in the area is seen by at least 4 cameras. According to the documentation [71] of the manufacturer, in optimal conditions the OptiTrack system is capable of sub-20 µm accuracy and is capable of consistently producing positional error less than 0.3mm and rotational error less than 0.05°.

Figure 50. Motion tracking through markers placed on a drone
(Rozenberszki and Majdik [70])

The cameras can track retroreflective markers seen on Figure 50. They can be tracked individually or handled as a rigid body that has orientation vectors too (at least three markers are needed).
During the positional tracking accuracy and the relocalization accuracy tests, three markers were placed on the ZED 2 stereo camera: two markers on top of the two lenses of the camera and the third one in-between. The real-time data of the OptiTrack system can be accessed using the optical motion capture software called Motive [73], seen on Figure 51. below.

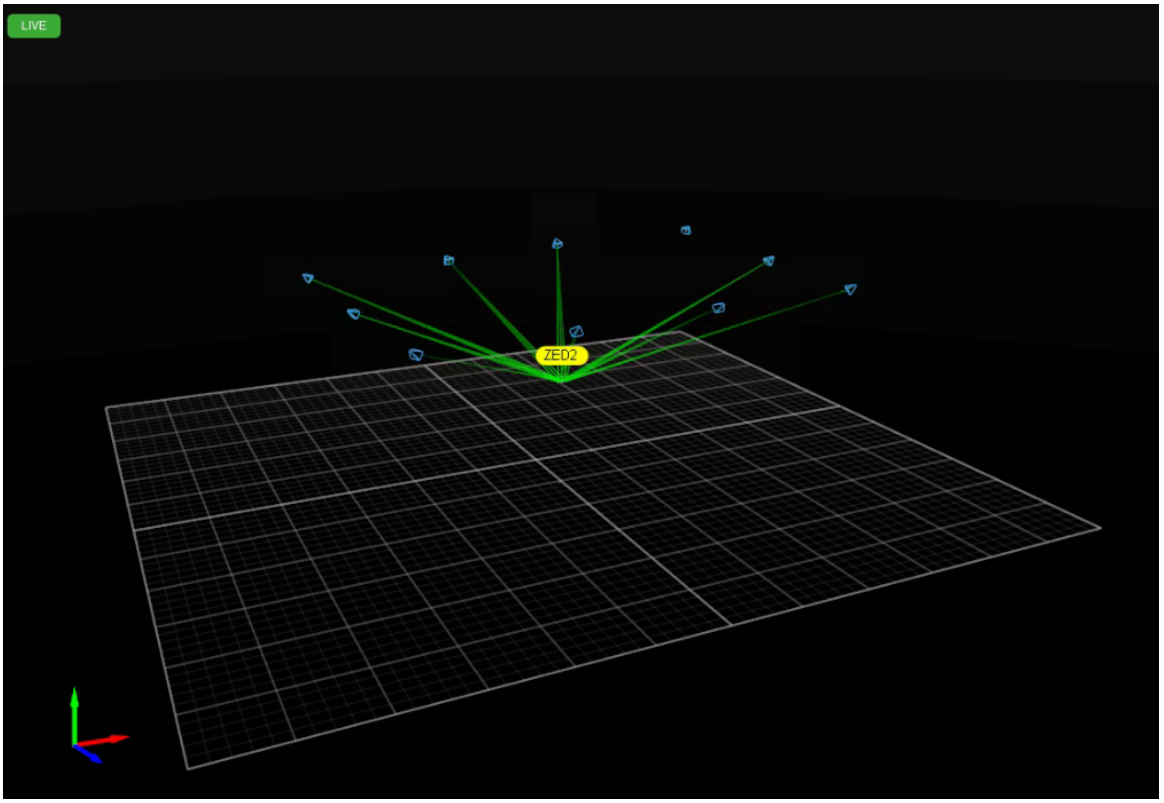


Figure 51. Tracking the movements of the ZED 2 stereo camera using the Motive software

The ZED 2 stereo camera uses a visual SLAM algorithm to track its surroundings thus understanding its position and orientation relative to the environment. There isn't any official reported information in the documentation about the accuracy of the positional tracking ability of the camera. Therefore, it is meaningful to test it to have an estimate about the efficiency of the camera's visual SLAM system. The measurement of the positional tracking accuracy of the ZED 2 stereo camera was done by using the previously mentioned SZTAKI MIMO Arena.

3.4 Visual SLAM accuracy tests in the SZTAKI MIMO Arena

In total, 3 tests were carried out inside the SZTAKI MIMO Arena. The OptiTrack system was continuously tracking the three markers on top of the camera and recorded the ground truth/reference values: the 3D coordinates of the trajectory and the orientation of the camera. These reference values were then compared to the 3D position and orientation reported by the ZED 2 stereo camera using the HD1080 resolution. The mean length of the test sequences were around 12 meters. The coordinate systems of the two trajectories have been registered to each other. The same coordinate system type has been chosen (y-down) and the initial position measured by the ZED 2 stereo camera has been shifted to the initial position recorded by the SZTAKI MIMO Arena. The frequency of pose information of the Arena has been downsampled to match the data output frequency of the camera.

To estimate the translation error, the RMS error was calculated between the two recorded trajectories. Furthermore, the evaluation metrics of the KITTI Vision Benchmark Suite (KITTI) [74] were used. The KITTI dataset uses two key metrics for comparison of different

visual SLAM algorithms. They take specific path length distances and measure the translation and rotation error at distances specified by the path lengths in the following way:

$$pathLenghts = \{100, 200, 300, 400, 500, 600, 700, 800\}$$

$$translationError_i = \frac{\sqrt{(x_2 - x_1)^2 * (y_2 - y_1)^2 * (z_2 - z_1)^2}}{pathLenghts_i}, \text{ where}$$

x_1, y_1, z_1 are the 3D coordinates of the position estimated by the camera at a given path length, and

x_2, y_2, z_2 are the 3D coordinates of the ground truth position at a given path length

$$rotationError_i = \frac{acos\{0.5 * [trace\,(R_{measured}T * R_{groundtruth}) - 1]\}}{pathLenghts_i}, \text{ where}$$

$R_{measured}$ is the rotation matrix estimated by the camera at a given path length, and

$R_{groundtruth}$ is the ground truth rotation matrix at a given path length

Their evaluation table ranks the visual SLAM methods according to the average of the translation error and rotation error values described above. The dataset of KITTI has test sequences of length of around 800 meters and measure the aforementioned errors at every 100 meters. In the tests of the thesis, the path length of the test runs were also measured at 8 different evenly distributed distances. The translation error is measured in percent and the rotation error as degrees per meter. See the results in the table below. The coordinate axes are expressed in a *right handed y-up coordinate system*.

Table 14. Translation and rotation error of three test sequences

Test run	**Total RMS translation error** [m]	**RMS error along the *x* axis** [m]	**RMS error along the *y* axis** [m]	**RMS error along the *z* axis** [m]	**Mean rotational error [deg/m]**	**Mean translational error [%]**
1	0.2223	0.1109	0.0452	0.0661	0.1729	2.605
2	0.1666	0.0794	0.0241	0.0631	0.3252	4.871
3	0.3010	0.1319	0.0453	0.1236	0.0539	1.742
Average	**0.2299**	0.1074	0.0382	0.0843	**0.184**	**3.073**

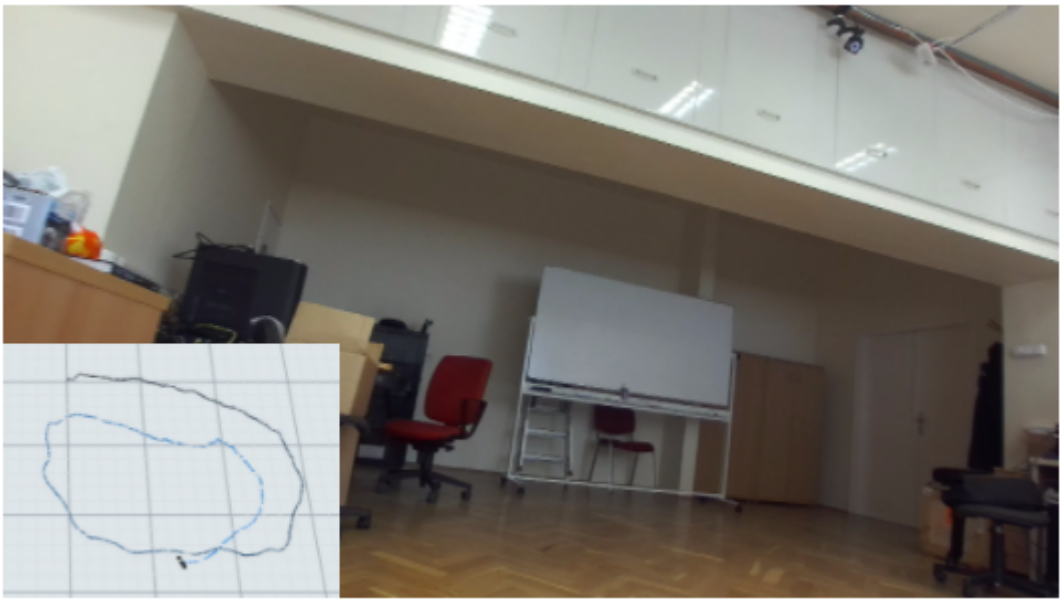


Figure 52. Example image of one frame of a positional tracking test run,
with the current real-time trajectory reported by the camera in the bottom left corner

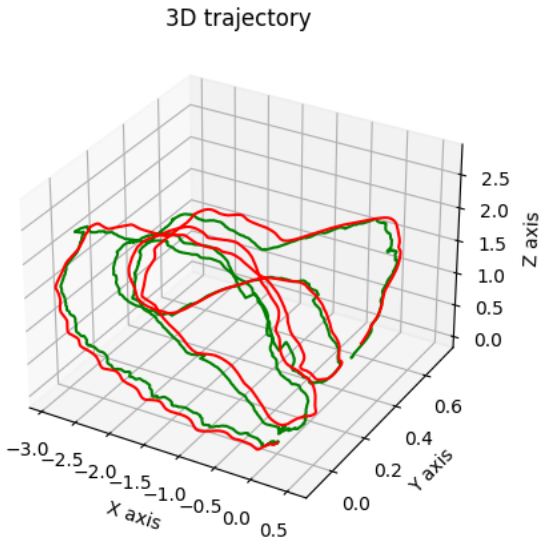


Figure 53. 3D trajectory of a test run [in meter]
Trajectory reported by the OptiTrack system shown in red.
Trajectory reported by the camera shown in green.

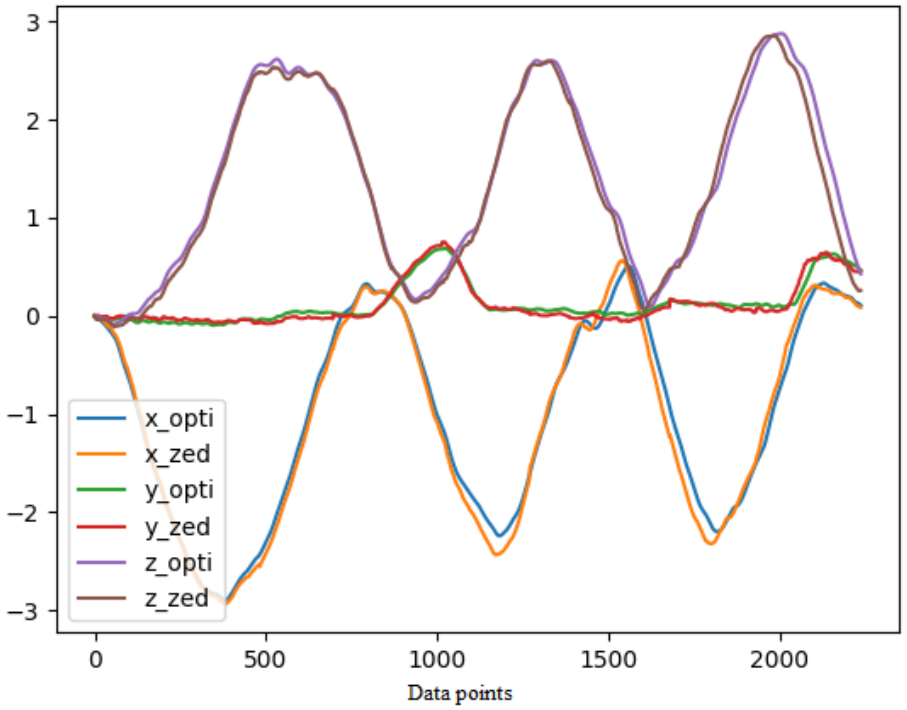


Figure 54. The reported 3D coordinates (in meter) of the reference and ZED-measured trajectories of a test run

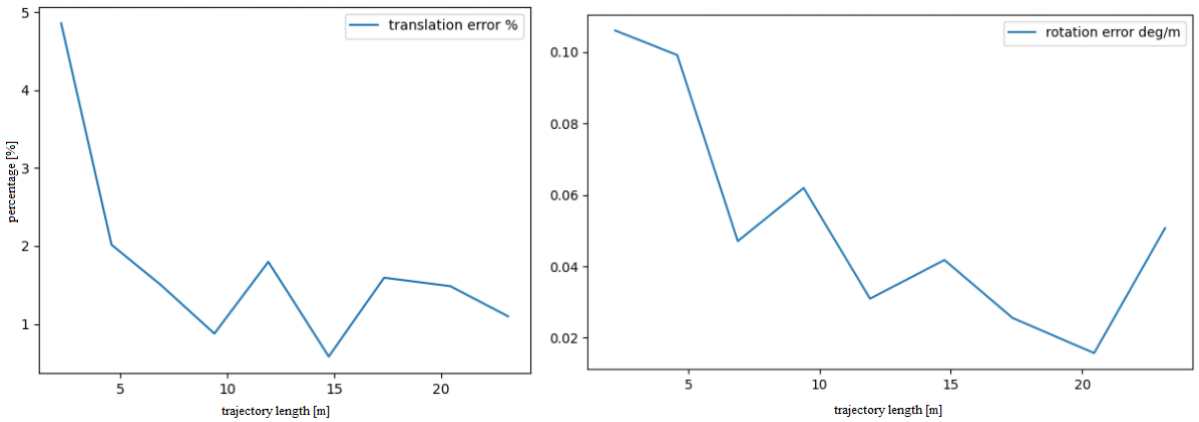


Figure 55. Translation error [%] and rotation error [deg/m] of a test run

The visual SLAM system of the ZED 2 stereo camera has an average score of 0.184 deg/m for rotation error and 3.073% for translation error, calculated from the three test runs. The measured Total RMS translation error expressed as percentage is around 2%.
Compared to the best performing visual SLAM system of the KITTI Vision Benchmark Suite dataset, which has a rotation error of 0.0009 [deg/m] and a translation error of 0.53 %, the SLAM system of the camera seems to be underperforming. The reasons for that can be the following:

- The metrics divide the calculated errors by the length of the path. Since the length of the trajectory in the KITTI dataset is around 800 meters, this gives better results on longer trajectories.

- The SZTAKI MIMO Arena, where the test runs were carried out, has a lot of flat surfaces, which makes tracking for visual SLAM systems harder. Therefore, the accuracy can decrease.

3.4.1 Measurement of the visual SLAM's cumulative drift

The positional tracking accuracy of the visual SLAM system of the ZED 2 stereo camera was also measured by comparing the difference between the 3D position reported by the camera at the beginning and at the end of a long test sequence. In total, 3 test sequences were conducted using the HD1080 resolution of the camera. Their results are seen in the table below. The cumulative drift is measured as the distance between the 3D coordinates of the start and stop location, which in ideal cases should be the equal. The tests were carried out inside one of the buildings of SZTAKI going from one level to another and coming back on a different path.

Table 15. Length of the test runs and the acquired cumulative drift

Runs	**Length** [m]	**Cumulative drift** [m]	**Error** [%]
1	100.3	0.85	0.847
2	146	3.45	2.363
3	119.4	2.10	1.759
Average	**121.9**	**2.13**	

The picture below shows an instant of the real-time camera tracking and 3D mapping of a long test sequence.

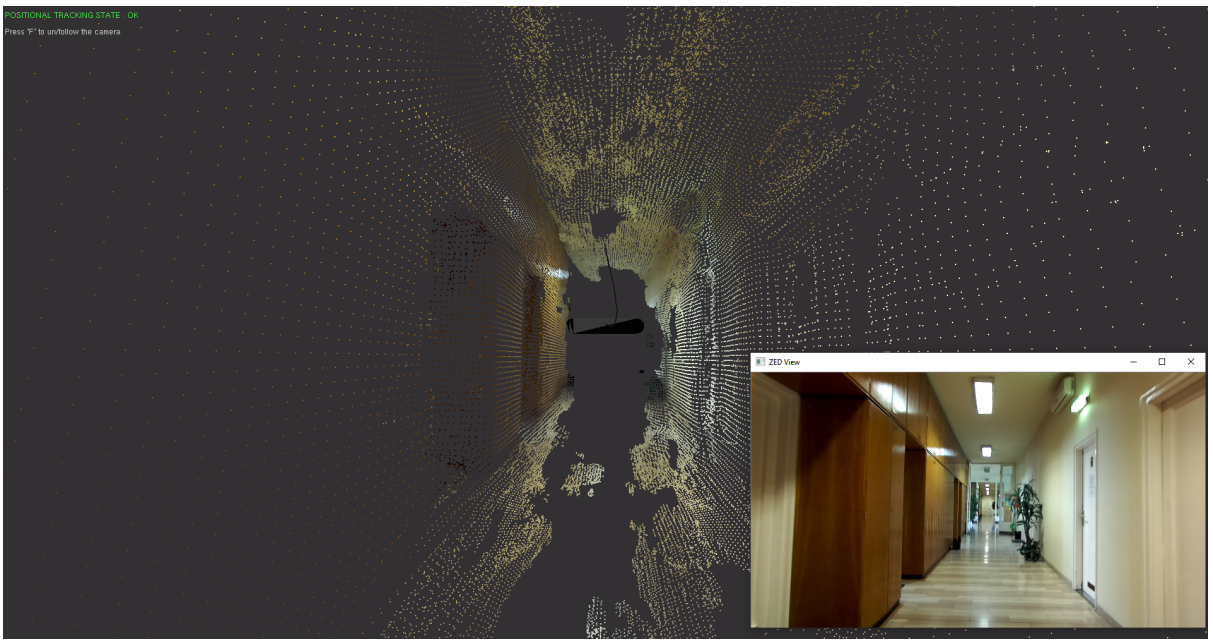


Figure 56. Real-time 3D point cloud generation with the trajectory and 2D left image of the camera

The three test sequences had an average trajectory length of 121.9 meters. The average drift measured was 2.13 meter. Averaging the three test sequences, the ZED 2 stereo camera had an average drift of 1.747%. To be able to benchmark this result to other solutions, the result of the currently first visual SLAM method of the KITTI Vision Benchmark Suite was looked at. Zhang et al. [75] reported a 0.75% relative position drift using the KITTI dataset.

3.5 Relocalization accuracy measurement

To estimate the relocalization accuracy of the camera, the previously described SZTAKI MIMO Arena is used. The test method is the following:
During the first recording in the MIMO Arena, the ZED 2 stereo camera is placed to predetermined locations measured by the OptiTrack system. All along, the area map containing the key images of the scene is continuously being stored. At these specific locations, the 3D coordinates estimated by the camera are saved.
These 3D coordinates at the specific 3D locations will then be compared to the 3D coordinates obtained during the second recording, after the area map has been successfully loaded and the coordinate frames of the two recordings have been automatically aligned. The MIMO Arena is only used to locate the specific benchmark positions. In total, three tests were conducted with 6 different benchmark locations. The relocalization accuracy is measured as the discrepancy between the two recordings' reported 3D positions by the ZED 2 stereo camera at these 6 benchmark locations. See the results in the table below.

Table 16. Relocalizational error of three test sequences

Test sequence	**Mean discrepancy of the 6 locations** [m]
1	0.05371
2	0.08368
3	0.06273

Average	**0.06671**

The three test sequences, after relocalization, had an average discrepancy of 66.71 millimeters.

3.6 Object detection performance measurement

Previously it was mentioned that the ZED 2 object detection algorithm is proprietary, which means that its architecture is not accessible to developers. Furthermore, the object detector of the ZED 2 stereo camera can only process input images that are coming from either a live video stream of the camera or from its specific *.svo* recording format. This makes using large online test image databases impossible for performance measurement.
In order to assess the performance of the ZED object detector, the qualitative test process described below was proposed.

Using the ZED 2 stereo camera, 51 images (HD1080 resolution) were captured and the output of the ZED 2 object detector was saved. The output of each detected image consists of the following properties for each detected object in the image:

a. class name
b. confidence value
c. absolute bounding box coordinates - (x,y) pixel values
d. inference time, which is the time needed for the object detector to make a prediction

The image acquisition happened:

- on the road while the camera was mounted on a test vehicle,
- in a garden with various objects present
- and inside a house.

The goal of the image acquisition was to gather images in a broad spectrum, representing as many object classes as possible. The distribution of classes in the 51 images can be seen in the chart below.

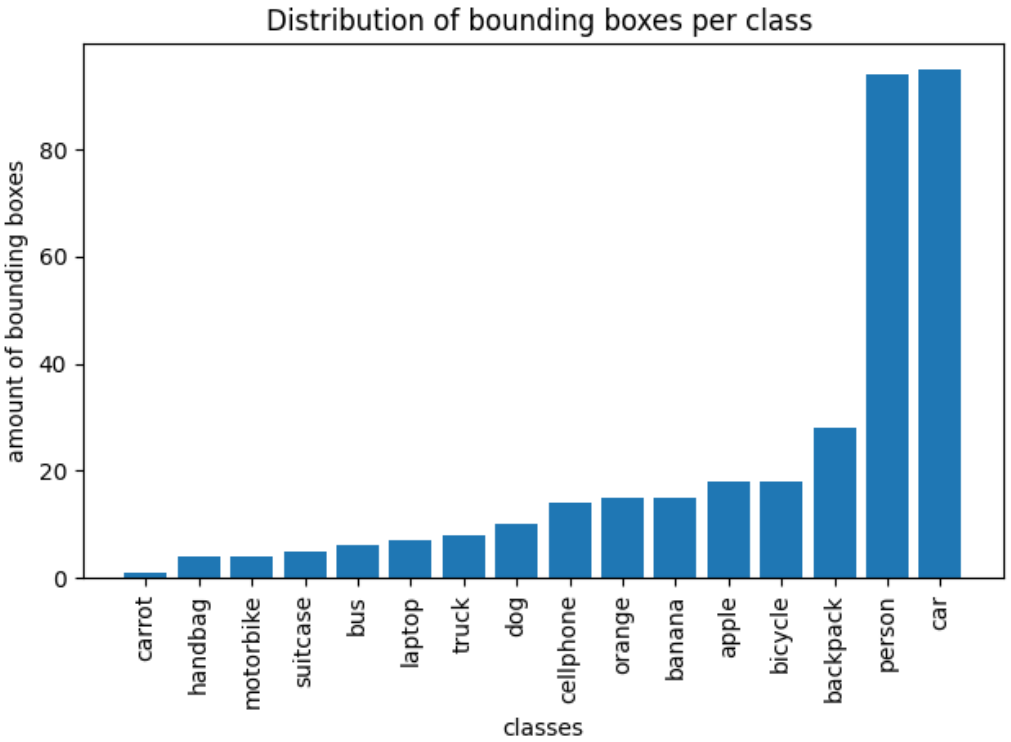


Figure 57. Distribution of ground truth bounding boxes per class in the 51 captured images

After the recording of the dataset of 51 images, an **online annotation tool** [76] **was used to label the classes on the captured images**, thus creating the ***ground truth data*** for the object detection performance measurement. With the help of annotation tools, people can label their images by drawing bounding boxes around objects of interest. This step is needed to be able to calculate the metrics described in the *Object detection performance metrics* section, such as: intersection over union (IOU), recall, precision and average precision.

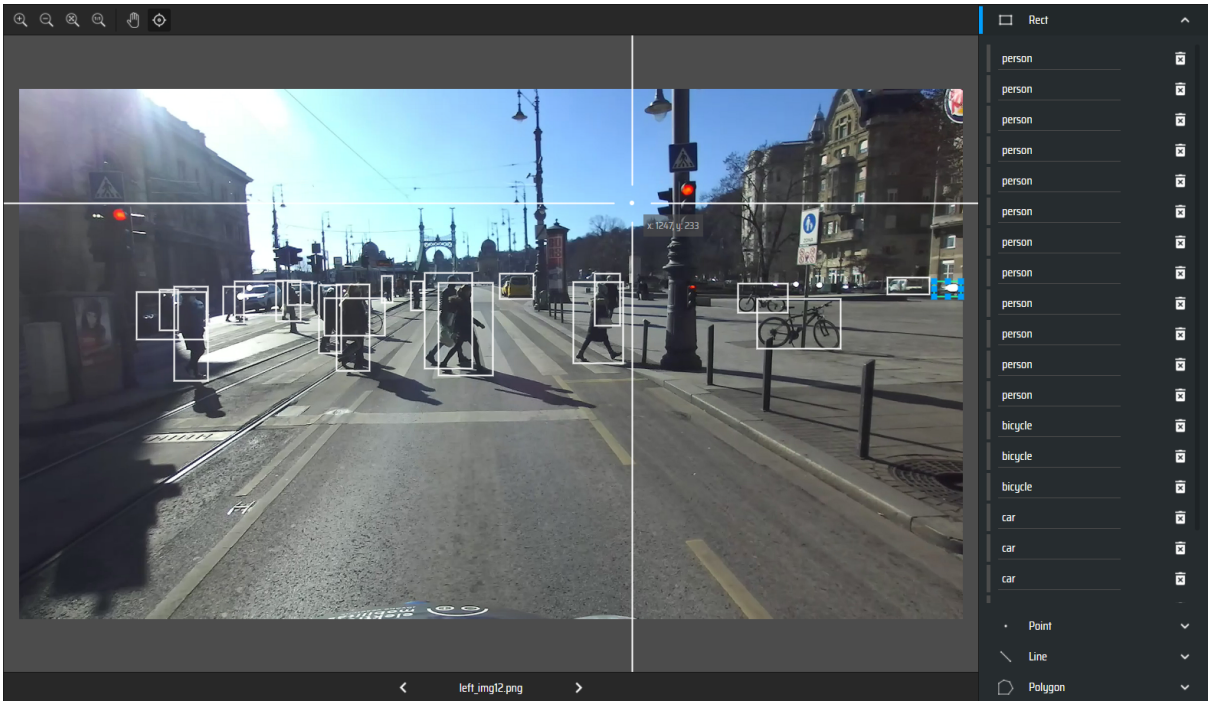


Figure 58. The online annotation tool

In total, there are 342 bounding boxes labeled in the 51 images. The average area of the bounding boxes is 55293.19 pixels. The 342 bounding boxes belong to 16 different classes and the distribution can be seen in the chart above.

The assessment of the object detection performance was done by **using** the previously mentioned **toolkit** proposed by Padilla et al. [37]. The GUI (Graphical User Interface) of the toolkit can be seen in the picture below. There are three main parts of the layout: In the *Ground truth* part, the folders of the ground truth annotations and the captured images can be given as input. Following the loading of the folders, the format of the ground truth data can be selected. There are many types of formats, some of them were mentioned in the *object detection performance metrics* section. In the thesis, the PASCAL VOC format was used, which stores a file with *.xml* extension for each image in the dataset. This format was generated by the online annotation tool and it contains the name, width, height and depth of the image file, along with the names of the detected object classes and the absolute coordinates of the corresponding bounding boxes.
In the *Detections* part, the folder containing the results of the object detection has to be included. Each image has a corresponding file with *.txt* extension. After the folder is included, the format of the object detection results has to be chosen. In this thesis, the file with *.txt* extension that contains detections corresponding to an image has the format for each detected object in the image as follows:

≪class_name≫≪confidence≫≪left≫≪top≫≪right≫≪bottom≫

where the four directions represent the corners of the bounding box, in their absolute coordinates, corresponding to a detected object.

In the *Metrics* part, various metric formats can be selected, in which we would like to receive the result of the performance assessment. These metrics were already described in the *Object detection performance metrics* section.

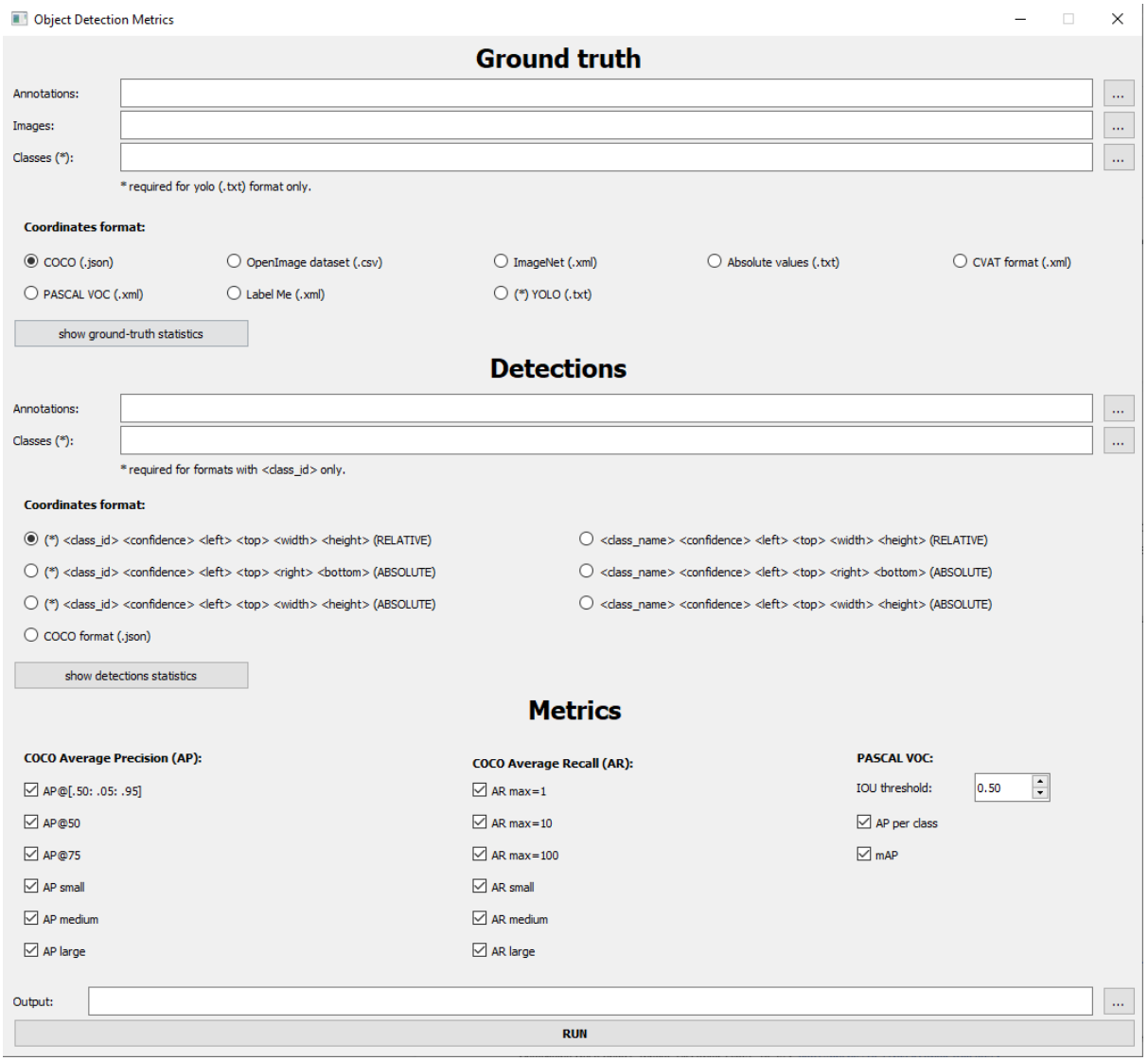


Figure 59. GUI of the toolkit proposed by Padilla et al. [37]

The statistics of the detections of the ZED 2 object detector with an example image showing the ground truth and detected bounding boxes can be seen below.
A total of 325 bounding boxes were found in 51 images. The average area of the bounding boxes is 54856.34 pixels. The 325 bounding boxes belong to 16 different classes. The amount of bounding boxes per class is as seen below.

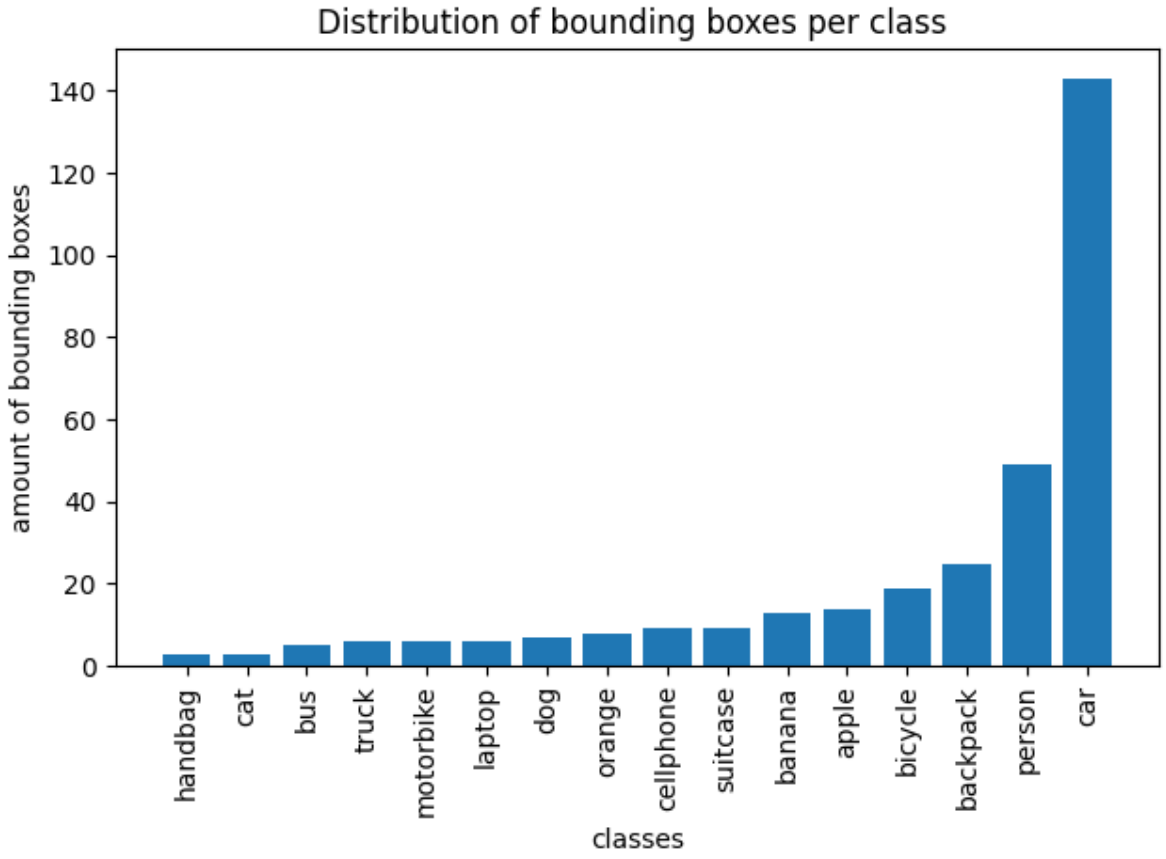


Figure 60. The distribution of bounding boxes per detected class of the ZED 2 object detector

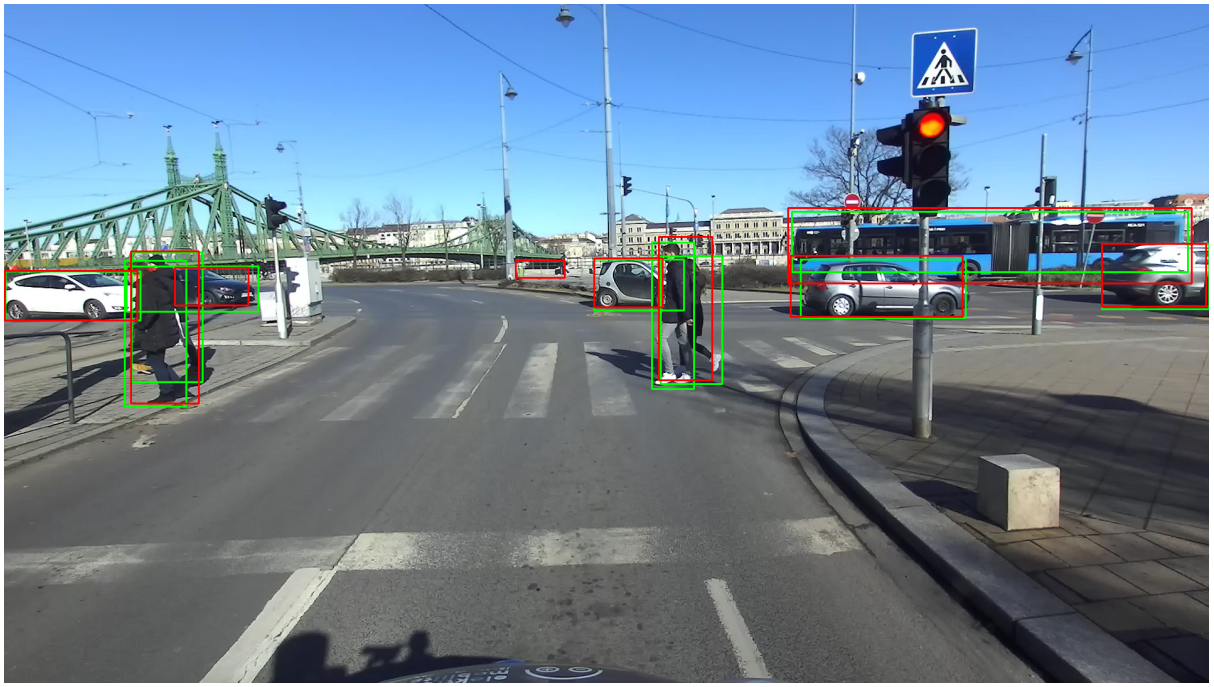


Figure 61. Example image of the output of the ZED 2 object detector
generated using the toolkit proposed by Padilla et al. [37].
Bounding boxes of the detections are shown in **red**.
Bounding boxes of the ground truth annotations are shown in **green**.

Using the above described toolkit, the **evaluation of the ZED object detector** was carried out. The toolkit outputs the precision-recall curve for every object class, along with the AP (Average Precision) percentage calculated according to metric used the Pascal VOC dataset, i.e. IOU threshold is set to t=0.5. An example of the per class precision-recall curve can be seen below.

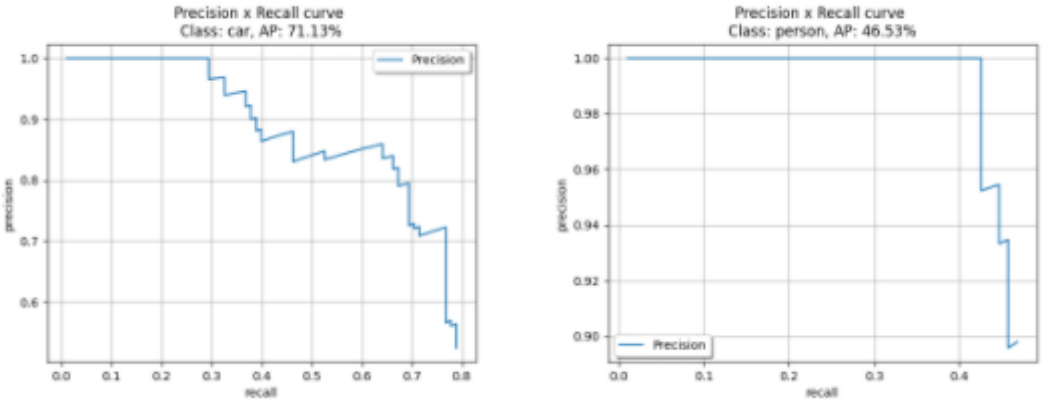


Figure 62. Example of the precision-recall curves for the object classes car and person

For the sake of completeness, the metrics output by the toolkit in the various object detection performance metrics used by the COCO and the PASCAL VOC dataset can be found in the list below.

COCO METRICS

- AP: 0.3785
- AP50: 0.5333
- AP75: 0.3974
- APsmall: 0.0801
- APmedium: 0.1074
- APlarge: 0.5689
- AR1: 0.3617
- AR10: 0.4344
- AR100: 0.4349
- ARsmall: 0.0888
- ARmedium: 0.1447
- ARlarge: 0.6277

PASCAL METRIC (AP per class)

- banana: 0.4583
- backpack: 0.6454
- dog: 0.7
- bicycle: 0.5995
- person: 0.4653
- car: 0.7113
- truck: 0.625
- motorbike: 0.4499
- bus: 0.5
- orange: 0.3166
- apple: 0.4622
- handbag: 0.75
- laptop: 0.8571
- carrot: 0
- cellphone: 0.6428
- suitcase: 0.35

When comparing the result of the ZED object detector with other object detectors, the mAP (mean Average Precision) value of the PASCAL VOC dataset is used (where the IOU threshold is set to t=0.5), along with the mean inference time of the detections.

The ZED object detector has a mAP value of 0.5333 and a mean inference time of 51,74 ms.

To be able to benchmark this result with other object detectors, four algorithms were chosen that are based on the three architectures described in the *object detection with deep learning* section. The four object detection algorithms are all pre-trained on COCO. It is important to note that the COCO dataset has 80 different classes, however, the 22 classes that the ZED object detector can distinguish are included. Retraining these object detectors are out of the scope of this thesis. Therefore, this benchmark should be treated only as a relative timing and accuracy comparison. Consequently, the detected classes that are not those the ZED object detector can detect are suppressed.

The four object detectors are as follows:

- SSD ResNet [77], that uses SSD as the basic network structure and replaces the VGG16 inside with a ResNet101 network.
- SSD Inception [78], that uses SSD as the basic network structure and replaces the VGG16 inside with an inception block.
- Faster R-CNN NasNet [79], that optimizes the Faster R-CNN network with Neural Architecture Search optimization and should have a higher mean inference time and a higher accuracy
- YOLO v4 [80], which is an improved version of the classic YOLO architecture.

The 51 images that were previously captured were fed to all four object detectors above and the results were saved similarly as with the ZED object detector. The table below contains the mAP and inference time values achieved with these object detectors.

Table 17. Performance values of object detectors

Object detector	mAP	Inference time (rounded) [ms]
SSD ResNet	0.3039	440
SSD Inception	0.29309	414
Faster R-CNN NasNet	0.3897	3960
YOLO v4	0.5054	210
ZED object detector	**0.5333**	**52**

Both of the two SSD object detectors achieve a similar mAP of ~0.3 and an inference time of ~400 ms. The Faster R-CNN object detector has a higher mAP (0.3897) compared to the SSD networks, however the inference time is increased about 10 times. On the other hand, the YOLO-based object detector has a mAP of 0.5050, which is very close to the mAP of the ZED object detector (0.5333). Despite that, the inference time is about 4 times higher than the inference time of the ZED object detector.

Overall, the ZED object detector is the most accurate and the fastest compared to these four algorithms. Moreover, the object detector of ZED also outputs 3D information about the detected objects, along with tracking and velocity data. Nonetheless, these results should be treated only as relative benchmark values, since the ZED object detector detects only 22 classes, while the other four can detect 80 different objects. Moreover, the object detection performance was measured using a qualitative test process and the test dataset only consisted of 51 images, which is short of a complete coverage of performance.

Nevertheless, it has been proved that the ZED object detection algorithm is capable of reliable real-time object detection.

4. The change detection algorithm

Previously it was mentioned that to the best of the author's knowledge, there are no change detection methods that utilize object detection algorithms on near-ground captured data for 3D change detection. Therefore, the focus of the research has been shifted towards object detection based semantic mapping. There were many solutions that seemed promising, especially the architecture proposed by Sünderhauf et al. [30], which has been already described in the *research on object detection based semantic maps* section. In this section, the 3D change detection algorithm is presented, utilizing some key parts of the architecture proposed by Sünderhauf et al. [30]. The algorithm is written in the C++ programming language and is using the:

- Point Cloud Library (**PCL**) for point cloud data manipulation and display
- ZED camera specific library (**SL**, stands for StereoLabs, which is the manufacturer of the camera) for controlling the camera
- **OpenCV** library for displaying and manipulating 2D images
- **boost** library for XML document generation
- **CUDA** and Open Graphics Library (**OpenGL**) libraries for displaying 3D point clouds

The entire code of the algorithm was written by the author of this thesis. The parts proposed by Sünderhauf et al. [30] have been used as a written inspiration, the code itself was entirely programmed by the author of this thesis.

Change detection identifies changes using multi-temporal observations. This means that, in effect, change detection compares two states inspected at different times. Along this idea, the change detection algorithm has two main parts: At time t_i, the environment has to be discovered and somehow stored in order to use the data gained through the exploration for change detection by comparing it with the discovery at time t_{i+1}.

The flowchart below (Figure 63) presents the process of the change detection algorithm's initial environment discovery and object-map building using the ZED 2 stereo camera.

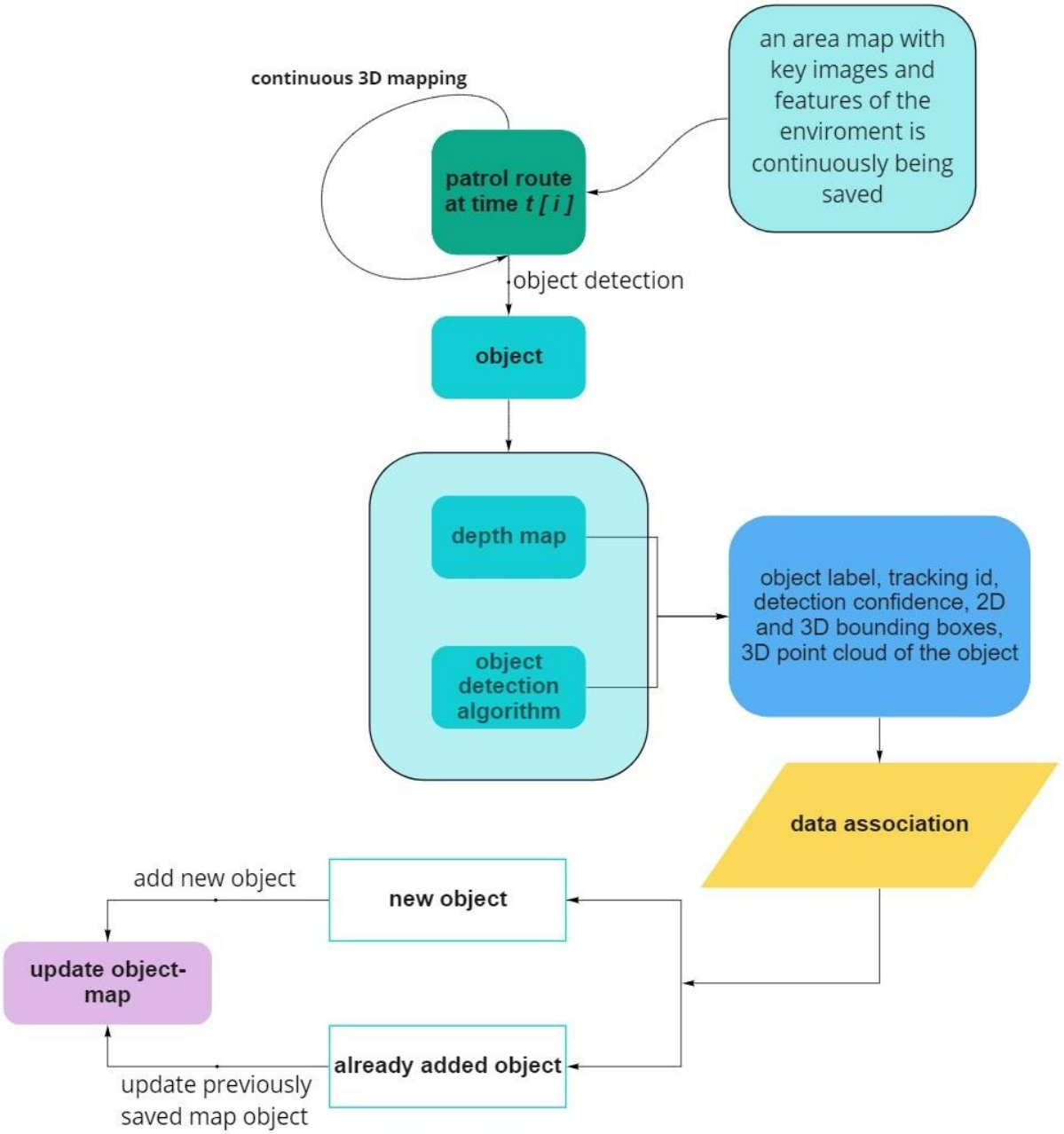


Figure 63. The flowchart of the initial environment discovery and object-map building of the change detection algorithm

It is meaningful to mention some key parameters of the ZED 2 stereo camera that play an important role in change detection. As seen in the *depth accuracy measurement* section, the HD1080 resolution is the most reliable in terms of depth sensing accuracy. However, even this resolution has limitations: It is most reliable up to around 10 meters. After that, the depth error starts increasing. Therefore, the maximum distance of depth sensing was limited to 7 meters in the change detection application. This helps maintain a coherent semantic object-map, since the data association step, which is described later, relies heavily on the 3D information obtained through the depth sensing of the camera. Note, that this 'hard-coded' limitation can be improved and the possibility will be investigated later to introduce a

probabilistic framework to counter the aforementioned issue. However, it is out of the scope of this thesis.

4.1 Initial environment exploration

During a patrol route at time *t[i]*, the ZED 2 stereo camera continuously builds the 3D spatial map of the environment. While discovering the environment, the camera incrementally builds an area map, containing the key features of the environment. Using this area map at the beginning of a patrol route at time *t[i+1]*, it is possible to maintain a shared reference world coordinate frame across patrol routes.
As the camera moves and explores its environment, the object detection algorithm is continuously scanning for available objects in the scene. Once an object is found, with the help of the depth sensing of the camera, the following informations are extracted:

- **object label**, which is amongst the 22 object classes the ZED can recognize
- **object id** - the camera can track objects as long as they are in the field of view (FOV) of the camera and can assign them the same object ID
- **detection confidence**, that shows how confident the object detector is about the label of the detected object
- **2D bounding box** - 2D pixel coordinates of the four corners of the two-dimensional rectangle that encapsulates the object in the 2D left image of the stereo camera
- **3D bounding box** - combining the four corners of the 2D bounding box with the depth map of the scene results in the eight 3D coordinates of the three dimensional cube that encapsulates the object
- **3D position** - the 3D coordinates of the object's centroid
- **segmented 3D point cloud** - using the 3D bounding box of a detected object, it is possible to **segment the 3D point cloud** that belongs to the object from the 3D point cloud of the scene

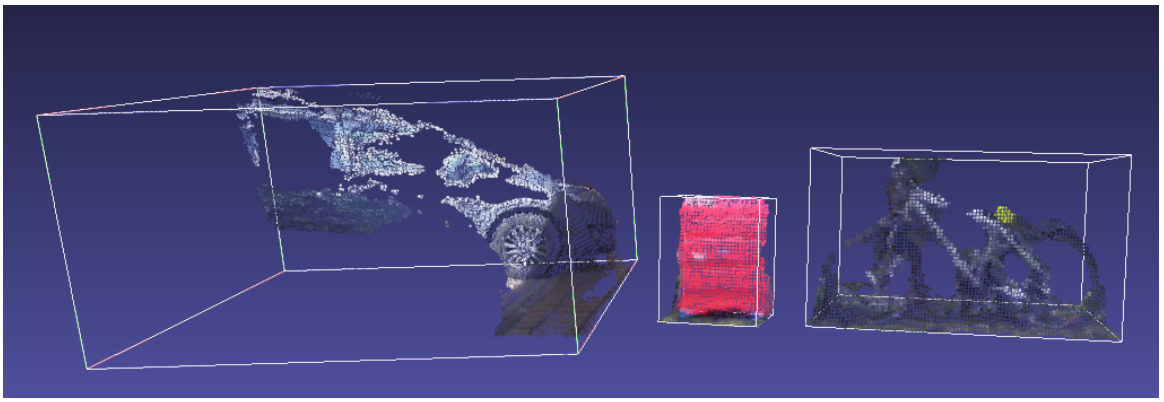


Figure 64. Example for 3D segmented point clouds of three objects: car, suitcase and bicycle

Following the object detection, the **data association step** (built upon the similar method proposed by Sünderhauf et al. [30]) decides if the detected object is a new object that has not been stored in the object-map yet or it is a new, yet undetected and unsaved object. The flowchart of the data association step is seen below (Figure 65). The object detection

algorithm returns a list of the detected objects in the scene and the data association method iterates over this list. The change detection algorithm of this thesis excludes moving objects, therefore in the beginning of the data association this attribute is checked. If the object-map that the algorithm is building is empty, the detected object is instantly registered to the object-map as a new object. If that is not the case, the object-map is queried to return the closest objects to the detected object. This step is done by taking the 3D position of the object's centroid and calculating the Euclidean distance between the map-objects and the detected object. At the end of the step called '*find closest objects'*, those already stored map-objects whose centroids are closer to the detected object than a predefined threshold, are obtained. If there are no close objects already stored in the object-map, the detected object is added as a new object to the object-map.

Otherwise, a nearest neighbor search based on a k-d tree [81] (k-dimensional tree) structure (*'knn_search'* step) is performed between the detected object's segmented 3D point cloud and the queried, already saved object's point cloud to calculate the Euclidean distance between the neighbouring point pairs. If at least 50% of the associated 3D point pairs of the two point clouds have a distance smaller than a predefined threshold, the detected object will be associated with the already saved object in the semantic object-map. Otherwise, the detected object is added as a new object to the object-map.

A k-d tree is a space-partitioning data structure for organizing points in a k-dimensional space. In essence, it is a binary tree [82] where every leaf node is a k-dimensional point and every non-leaf node divides the space into two half-spaces. Using a k-d tree for nearest neighbour search is very effective, because the properties of the tree allow it to swiftly disqualify large parts of the search space. In the worst case, the runtime is O(n), but in practise it is closer to $O(2^d + \log n)$.

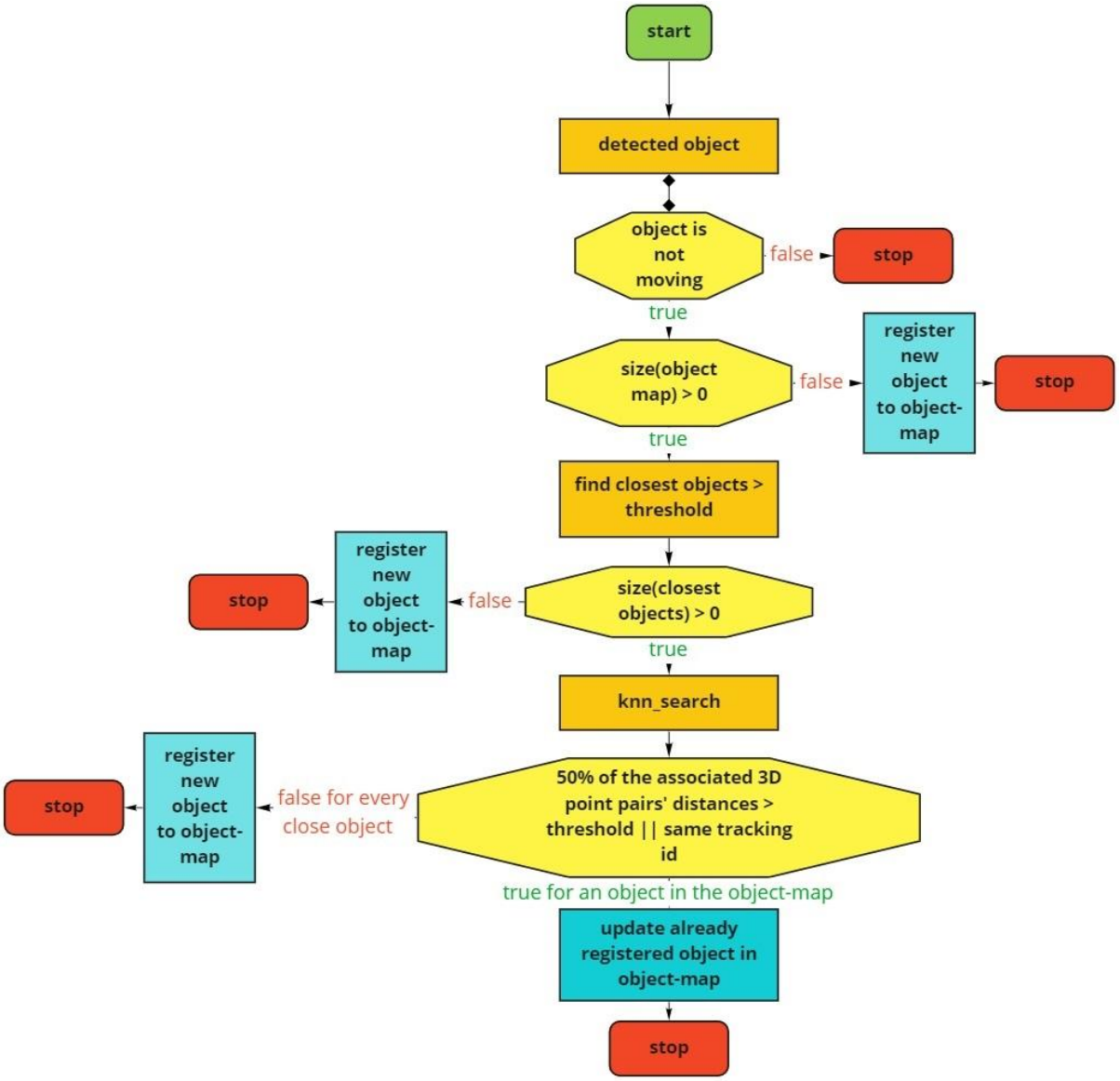


Figure 65. Flowchart of the data association algorithm

So far, the **update of an already registered object** was mentioned only as a 'black box'. However, it is an important step in maintaining a structured semantic object-map. With every stored object in the semantic database, a *label* - *confidence* and a *label* - *number_of_detections* pair is associated. This helps determine the final label and confidence score of every object in the semantic database. During an update of an object in the database, its 3D bounding box coordinates are also updated by taking the average of the cumulative sum of previously detected 3D bounding box corners.

There was an initial attempt to incrementally build and maintain the 3D point cloud of a stored object by adding the 3D point cloud of a detected object to it and then sorting to only keep the unique 3D coordinates and eliminate the duplicates. However, this step increased the runtime of the data association step so much that it was no longer suitable for real-time change detection. Instead of this solution, at every object update, the 3D point clouds are compared and the one with the larger size is kept. This step is meant to store the more

detailed 3D point cloud and also keep the operation fast enough for real-time usage. The data association step takes on average 15ms per detected object.

At the end of a patrol route at time *t[i]*, a post-processing step determines the final state of the semantic object-map. A stored object's label is decided by the previously mentioned *label - confidence* pair. The label with the highest accumulated confidence score is kept and based on the *label - number_of_detections* pair, the average confidence score for the object is calculated. Furthermore, to eliminate objects added due to noise, objects with a number of detections less than 5 are excluded from the database.

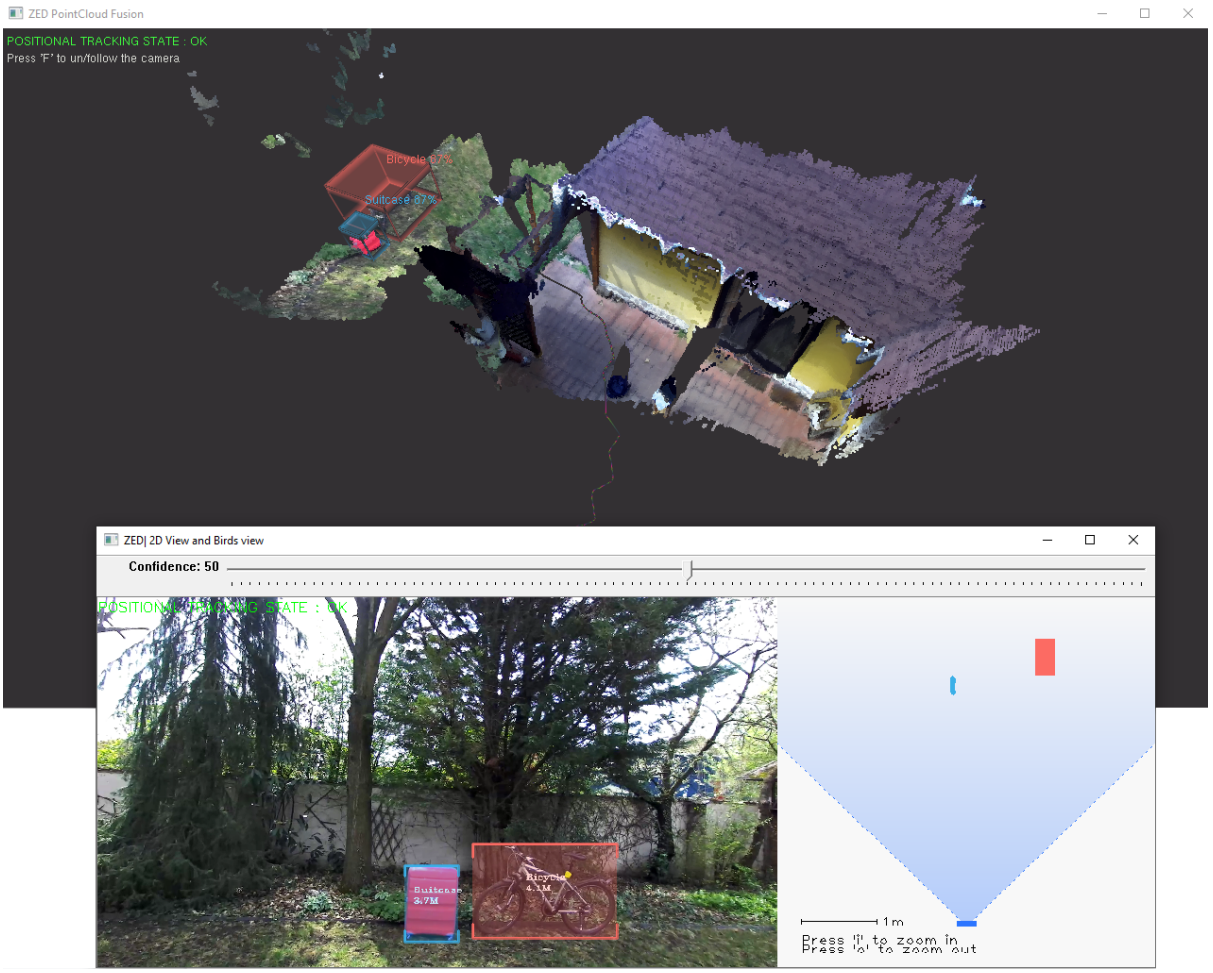


Figure 66. Simultaneous 3D mapping and semantic object database building
Incremental 3D point cloud reconstruction with overlaid camera trajectory and 3D object detection (top)
Result of the object detection displayed on the left image of the camera with a 2D object tracking view (bottom)

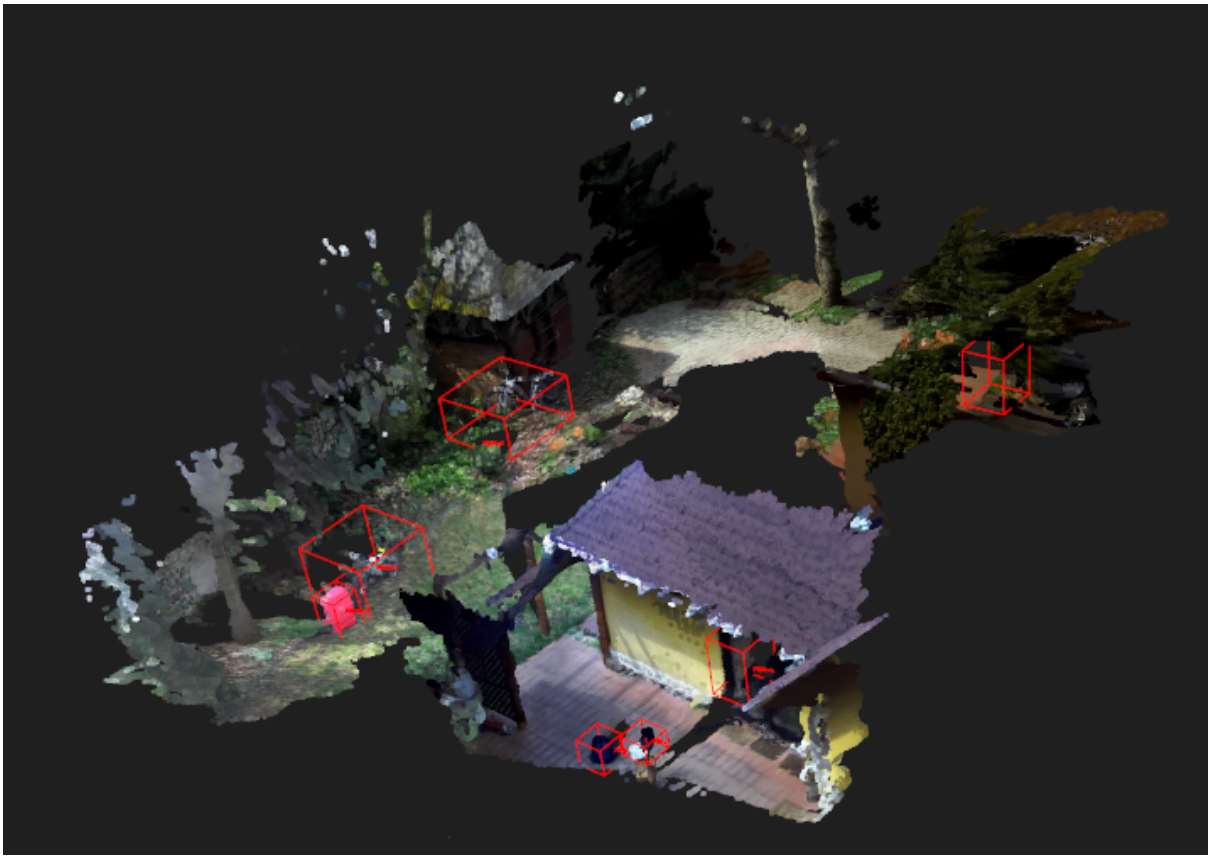


Figure 67. Result of the semantic object-map after post-processing
with displayed 3D bounding boxes of objects

After post-processing, the final semantic object-map has to be somehow stored in order to use the information gathered during the exploration of the environment for change detection. For this purpose, an XML Tree Structure is presented. The XML file stores the path to the final 3D point cloud of the entire patrol route and to the area map constructed during mapping. Furthermore, it also stores the following attributes for every object in the database:

- object id
- object label
- object detection confidence
- number of detections of the object
- 3D coordinates of the object's centroid in the world coordinate frame
- boolean value for determining if the stored object has a segmented 3D point cloud associated with it
- path to the 3D point cloud of the object
- 2D bounding box coordinates of the object
- 3D bounding box coordinates of the object in the world coordinate frame

```
<?xml version="1.0" encoding="utf-8"?>
<library>
    <areaMap>
        <path> </path>
    </areaMap>
    <inputPointcloud>
        <path> </path>
    </inputPointcloud>
    <objects>
        <object id="">
            <label> </label>
            <confidence> </confidence>
            <numberOfDetections> </numberOfDetections>
            <position> </position>
            <has3DPointCloud> </has3DPointCloud>
            <pathTo3DPointCloud> </pathTo3DPointCloud>
            <2DBoundingBox> </2DBoundingBox>
            <3DBoundingBox> </3DBoundingBox>
        </object>
        <object id="">
            <label> </label>
            <confidence> </confidence>
            <numberOfDetections> </numberOfDetections>
            <position> </position>
            <has3DPointCloud> </has3DPointCloud>
            <pathTo3DPointCloud> </pathTo3DPointCloud>
            <2DBoundingBox> </2DBoundingBox>
            <3DBoundingBox> </3DBoundingBox>
        </object>
    </objects>
</library>
```

Figure 68. Structure of the proposed XML Tree

4.2 Detecting changes by using the semantic object database

At the beginning of the patrol route at time *t[i+1]*, the previously saved XML file is loaded. This allows the algorithm to use the area map file to create the shared reference world coordinate frame between the patrol routes. The objects in the semantic object database are also loaded and are used to look for changes in the environment. The exact method of change detection is described in this section. The flowchart of the change detection process is seen below (Figure 69).

The spatial mapping of the ZED 2 stereo camera during patrol route at time *t[i+1]* does not start until the camera was able to relocate itself using the area map. Once the world coordinate frames of the two consecutive patrol routes are aligned, the steps described in the *initial environment exploration* section are continuously repeated, including the data association step. This is needed in order to maintain an up-to-date semantic map of the environment and recognize changes later compared to the current patrol route. While building the new semantic object-map, the semantic object-map recorded during the previous patrol route is being queried repeatedly based on the 3D FOV (field of view) of the stereo camera. At each new camera frame, the algorithm iterates over the semantic object-map loaded from the XML file and checks if any of the objects' 3D centroids fall into the 3D FOV of the stereo camera. If the query is true for an object in the previously recorded semantic object-map, a similar scenario takes place as during the initial environment exploration.

Namely, the currently incrementally built semantic object-map is compared with the previously recorded semantic object-map. For visualization, the point clouds of those objects in the previous semantic object-map that are in the 3D FOV of the stereo camera, are 3D projected onto the 2D image of the camera.

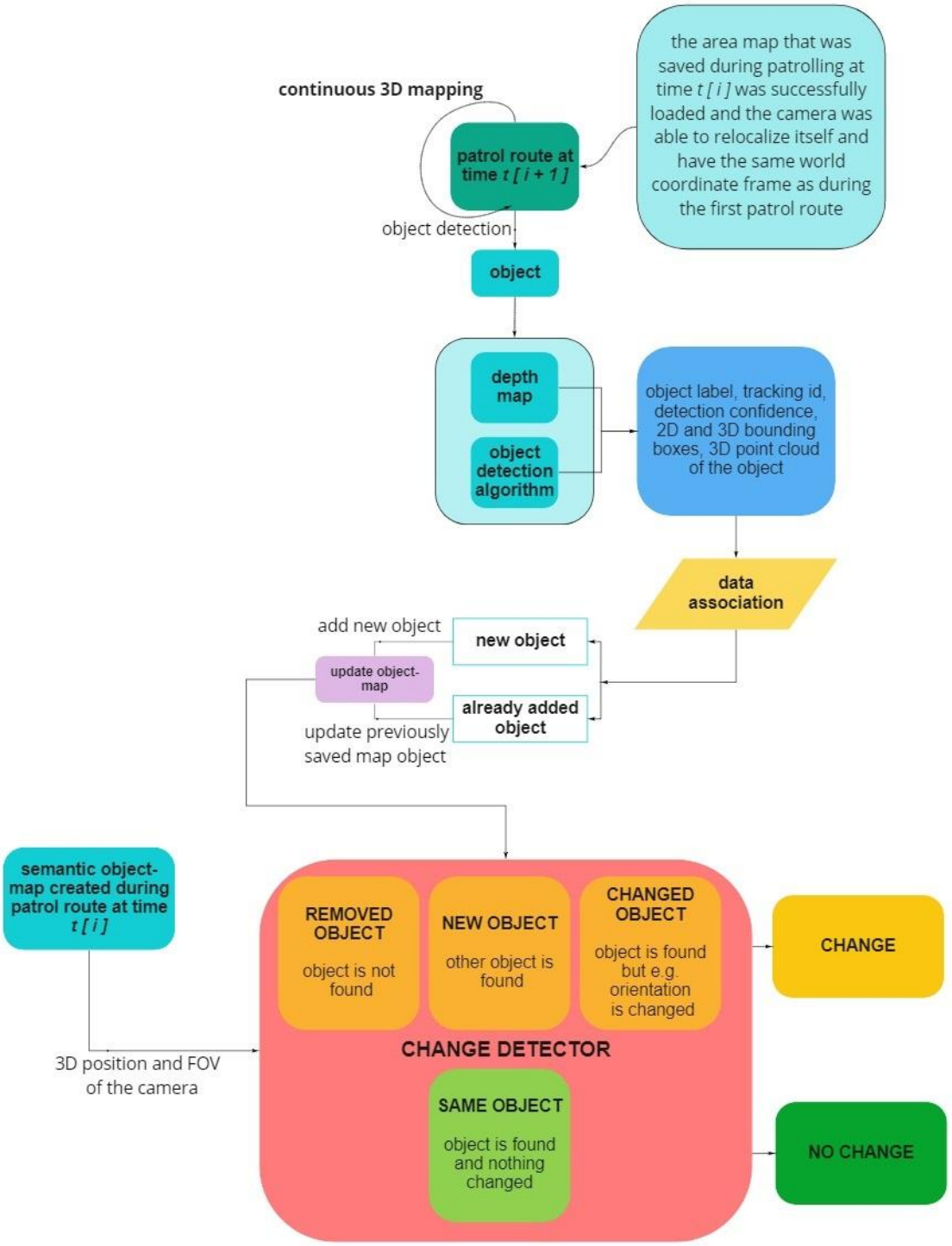


Figure 69. The flowchart of the semantic object database based change detection

For every previously detected object that is in the 3D FOV of the camera, the algorithm returns the closest objects in the current semantic object-map, using the Euclidean distance of

3D object centroid coordinates. If there are no objects in the current semantic object-map that are closer than the predefined threshold, that means the **object has been removed between the two patrol routes.** Therefore, a change has occurred. Otherwise, the same *'knn_search'* step is conducted to compare the point clouds of candidate objects as during the initial environment exploration. If at least 50% of the associated 3D point pairs of the two point clouds have a distance smaller than a predefined threshold, the objects are considered as a match, which means no change has occurred. If the 50% mark is not reached, it means that the point cloud of the two compared objects differ too much, which means that a change has occurred, because potentially **another object has been placed to the same location or the object has changed between two patrol routes**.
Similarly, to detect **new objects that appear during the current patrol route, but not the previous**, the objects of the current semantic object-map are queried if they have close objects in the previous semantic object-map. If not, a change has occurred.

To visualize the changes or matches between two consecutive patrol routes, a 2D mask is overlaid on the 2D left image of the ZED 2 stereo camera. The color green indicates a match, and orange indicates a change. Three examples for change detection are seen below (Figure 70-72).

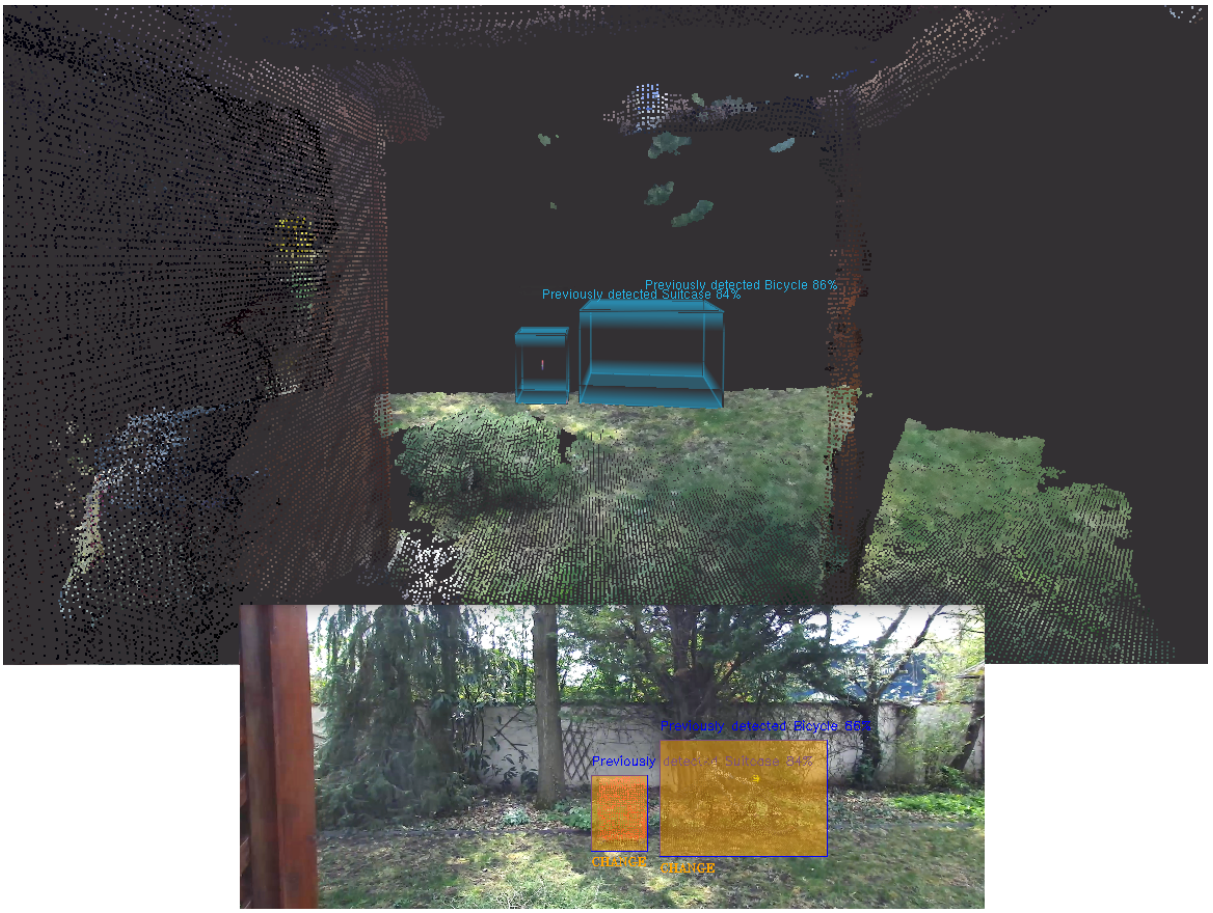


Figure 70. Example for two missing objects that were detected during the first patrol route
but they are not detected in the second patrol route (overlaid with orange on the 2D image).
The 3D bounding boxes in the point cloud show where the objects were during the first patrol route (top)
The point clouds of the two objects are projected back onto the 2D image (bottom).

Figure 71. Example for one object that did not move (bicycle, overlaid with green on the 2D image = match) and another object that is new compared to the first patrol route (person, overlaid in orange on the 2D image = change)

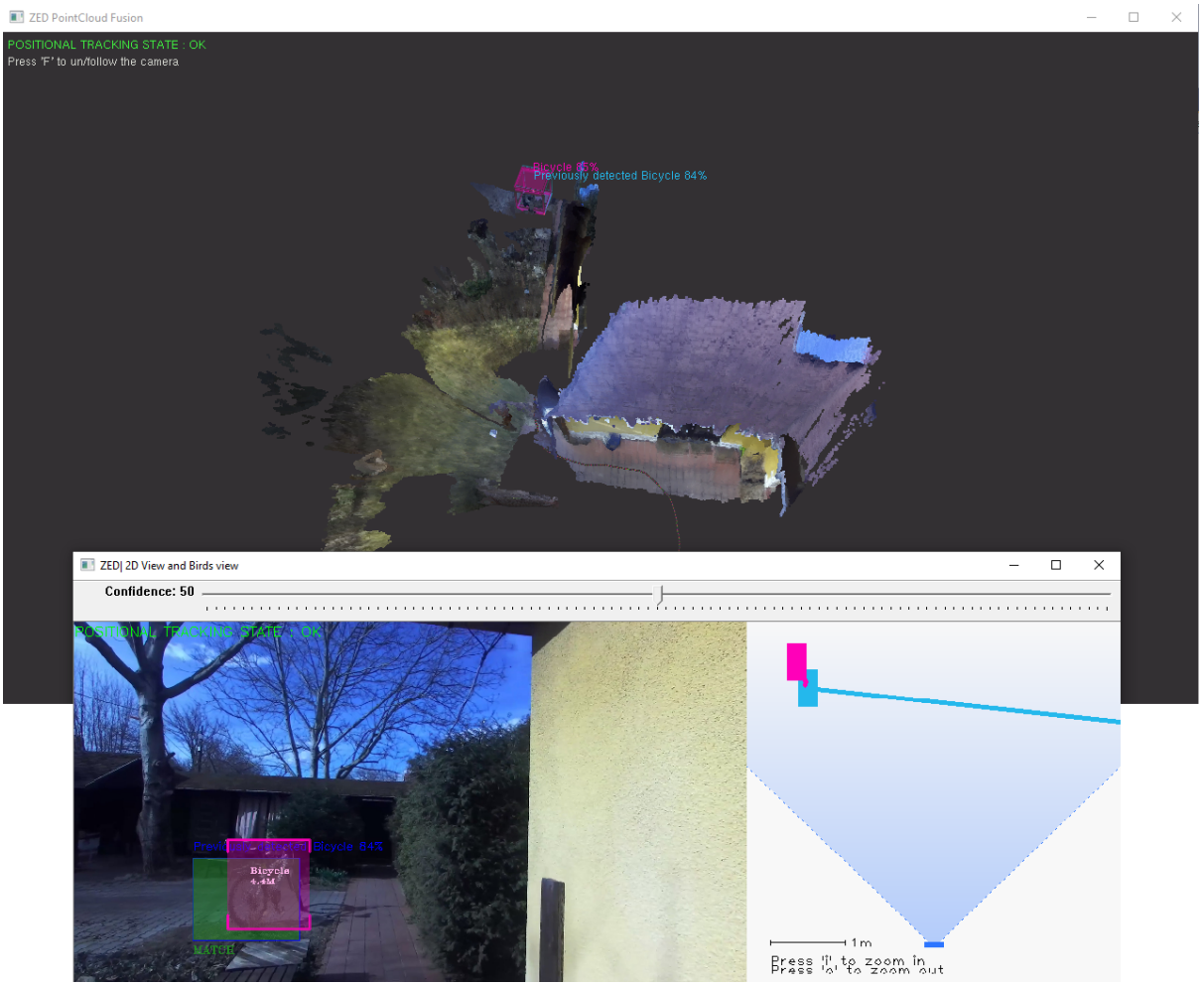


Figure 72. Example of one object that did not move compared to the first patrol route (bicycle, overlaid with green on the 2D image = match)

4.3 Qualitative performance assessment of the change detection algorithm

According to the knowledge of the author, there are no existing databases with *svo* video files. Nonetheless, to be able to assess the performance of the change detection algorithm, a qualitative test is performed.

The two threshold values of the data association algorithm were empirically adjusted and during testing the Euclidean threshold was set to 1 meter, while the threshold of the nearest neighbor search was set to 10 centimeter. Note that the Euclidean threshold value was not reported and the threshold of the nearest neighbor search was set to 2 centimeters in Sünderhauf et al. [30]. However, the threshold of 2 centimeters was found to be too small in an outdoor setting (the work of Sünderhauf et al. [30] is based on a controlled, indoor office environment), thus it has been increased in the change detection algorithm of this thesis. The tests were carried out both in outdoor and indoor environments with various objects and with all possible scenarios, such as: same object, removed object, changed object, added object.

The change detection algorithm can successfully and accurately detect changes that happened because a previously detected object has disappeared or a new object has appeared in a place where there was no object before. In these cases, the algorithm does not carry out a comparison, because there are no close objects nearby. See Figure 73. as an example.

Figure 73. Examples of changes due to removed or added objects (shown in orange overlay).
For changes caused by removed objects, the point clouds of previous detections are overlaid on the 2D image.
(top right, bottom left and bottom right example)

The algorithm can also handle objects closer to each other than the 1 meter Euclidean distance threshold. The algorithm was able to find all the *changes* and *matches* in the 3 tests that were carried out with an average of 9 objects each. However, it had difficulties when the detected objects were partially occluded or were too close to the camera (below ~1 meter). In these cases, the accuracy of the change detection algorithm has dropped and because of the partially available point clouds of objects, it detected changes even though the objects did not move compared to the previous patrol route. In the future, one of the main goals of the change detection algorithm development will investigate the issue of occluded and close objects.

In order to show and assess the capabilities of the semantic object map based change detection algorithm, a suitcase (which the ZED 2 stereo camera can detect, because it is included in the 22 object detectable by the object detection algorithm of the camera) was placed in the middle of the SZTAKI office (Figure 74). The suitcase was recorded by walking around and capturing it from all possible angles. Following the initial environment discovery and semantic object map building, the same patrol route was carried out.
The results can be seen on Figure 75. The changes in the background of the suitcase are due to the alterations made by colleagues that work in the office. These changes can be discarded,

since the focus of this test is the suitcase in the middle of the office. When the suitcase was at the same location with the same orientation, the algorithm successfully detected no change in the object between the two patrol routes.

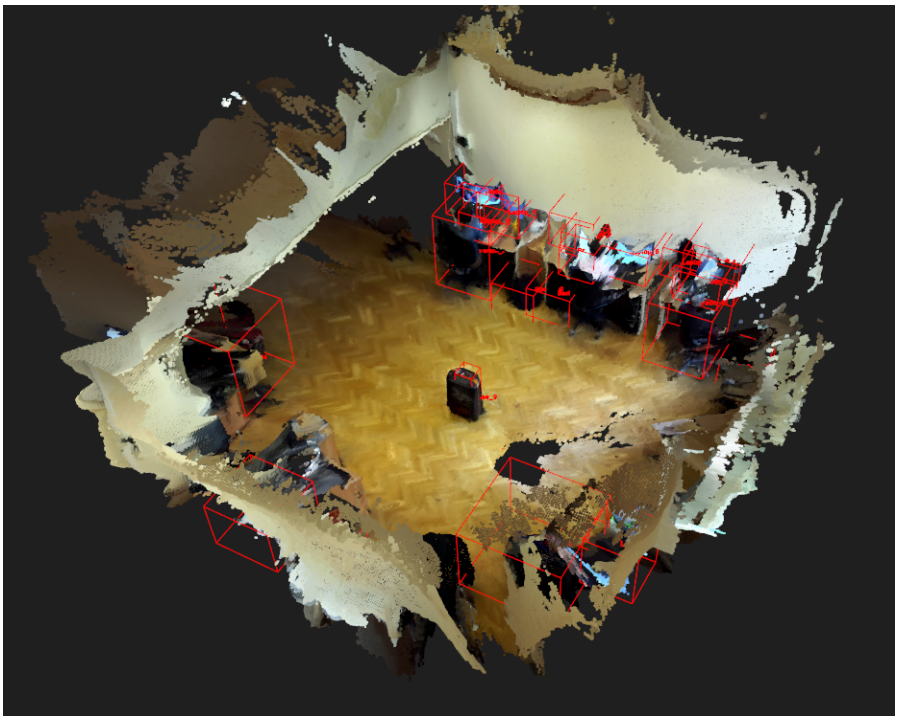


Figure 74. The 3D semantic object map of the SZTAKI office and the suitcase in the middle of it

Figure 75. Two examples of the change detection algorithm recognizing the unchanged suitcase from two different angles. The *match* is shown in green overlay

During the third patrol route in the same office area, the suitcase was moved to various locations. First, it was moved by around 5 centimeters, then 10 and 20. In the first two cases, the algorithm did not detect changes. This happened because the threshold of the nearest neighbour search was set to 10 centimeters beforehand. Therefore, translation changes that are below 10 centimeters, are not registered as changes by the algorithm. However, during the last trial, when the translation was around 20 centimeters, the algorithm was able to detect the change in the location of the suitcase. Figure 76. shows the aforementioned process.

Figure 76. Moving the suitcase ~5 cm (left), ~10 cm (middle) and ~20 cm (right)

The change detection algorithm does not perform any Iterative Closest Point (ICP) based algorithm to check whether a point cloud of a rotated object belongs to a previously detected object. Therefore, it is sensitive to changes of orientation of objects. To measure this sensitivity in this test case, three retroreflective markers were placed on the top of the suitcase. Using the SZTAKI MIMO Arena, the suitcase was being rotated until the algorithm no longer recognized it as a *matching* object between patrol routes. The sensitivity of the change detection algorithm depends greatly on the object in question. Since the stored features in the object-map are 3D point cloud points, the point cloud density and the object's texture influence how susceptible the algorithm is to change in orientations of objects. For the above presented suitcase, the change detector did not detect changes even when it was rotated by 90°. The reason for this is that the texture of the suitcase is very simple and similar on all sides. Moreover, the 3D shape of the suitcase is also simple and when rotated, it does not change that much. On the other hand, with a more complex object, such as the bicycle in Figure 64, the change detector is more sensitive to changes and detects smaller rotations too.

5. Conclusion

In the thesis, the assessment of the ZED 2 stereo camera has been successfully performed. For the measurement of **depth accuracy**, various solutions have been considered and in the end a 3D checkerboard based test method was chosen, described by Fernandez et al. [67] and Ortiz et al. [68]. The depth accuracy has been determined for all four available resolutions of the camera (see Figure 45) and the results have been compared with the reported values by the manufacturer. It has been concluded that the **HD1080 resolution meets the accuracy requirements stated in the documentation** of the ZED 2 stereo camera [1] and has a measured accuracy below 1% up to 10 meters and below 2.26% up to 15 meters. Apart from the initial bias of the exponential error model at the distance of 1 meter, the accuracy is below 1% for 5 meters for every resolution except for the HD2K. The HD2K resolution never goes below 1%, however, the accuracy calculated in this resolution is consistently below 2% at the distance of 15 meters too. The manufacturer reported an accuracy below 5% for up to 15 meters of distance for every resolution. The WVGA resolution passes the 5% accuracy threshold already around 11 meters. At a distance of 15 meters, the HD720 resolution has an accuracy of 4.64%, while the HD1080 has 2.26% and the HD2K has 1.95%.

After the assessment of depth accuracy of the stereo camera, the **consistency of the depth measurement** was also estimated. Similarly to the test proposed by Ortiz et al. [68], 100 consecutive depth maps were acquired of the same scene in all four resolutions of the camera. In total, there were two scenes captured (see Figure 46 and 47): one recorded indoors with artificial light and one outdoors with natural light. Both of them have a maximum depth of around 6 meters. To estimate the depth measurement consistency of the camera, the standard deviations of every pixel in the scene for the 100 consecutive depth maps in all four resolutions of the camera were computed. For comparison, the mean, median, minimum and maximum values of the pixel-wise standard deviations were calculated. Both for the indoor and outdoor scene, the **HD1080 resolution proved to be the most consistent one**. It has the lowest mean (0.0361 m for the indoor and 0.0336 m for the outdoor scene), lowest median (0.0107 m for the indoor and 0.0082 m for the outdoor scene) and lowest maximum values out of all four resolutions. In the outdoor scene, the WVGA and HD720 resolutions have a much higher maximum value compared to the other two resolutions and the WVGA also has a mean value of around 10 times higher compared to the HD1080 resolution, making it less reliable for depth sensing.
The test results reported by Ortiz et al. [68], which used a ZED stereo camera during the tests, confirm the results of the tests in this thesis. Namely, that the HD1080 resolution has the lowest minimum-maximum difference, lowest mean and the lowest median value as well, making it the most reliable and consistent depth sensing camera resolution of the ZED and ZED 2 stereo cameras.

For the measurement of the positional tracking accuracy of the camera, the SZTAKI MIMO Arena, proposed by Rozenberszki and Majdik [70], was used to create accurate ground truth trajectories. The Arena has ten cameras set up in such a way that every position in the area is seen by at least 4 cameras. According to the documentation of the manufacturer [71], in

optimal conditions the camera system is capable of sub-20 μm accuracy and is capable of consistently producing positional error less than 0.3mm and rotational error less than 0.05°. There were three retroreflective markers attached to the ZED 2 stereo camera (see Figure 50) and thus the SZTAKI MIMO Arena was able to record ground truth position and orientation data during the tests (see Figure 51).
To estimate the translation error of the visual SLAM algorithm of the camera, the RMS error was calculated between the trajectory reported by the camera and the ground truth trajectory measured with the SZAKI MIMO Arena. Furthermore, the evaluation metrics of the KITTI Vision Benchmark Suite [74] were used: Namely, rotational error measured as degrees/meter and translational error measured as percentage at specific path length distances. In total, there were three tests concluded in the SZTAKI MIMO Arena (see Table 14). They had a mean RMS translation error of 0.2299 m, a mean KITTI translation error of 3.073% and a mean KITTI rotational error of 0.184 deg/m.
Compared to the best performing visual SLAM system of the KITTI Vision Benchmark Suite dataset, which has a rotation error of 0.0009 [deg/m] and a translation error of 0.53%, **the visual SLAM system of the camera seems to be underperforming**. The reasons for that can be the following:

- The metrics divide the calculated errors by the length of the path. Since the length of the trajectory in the KITTI dataset is around 800 meters, this gives better results on longer trajectories.
- The SZTAKI MIMO Arena, where the test runs were carried out, has a lot of flat surfaces, which makes tracking for visual SLAM systems harder. Therefore, the accuracy can decrease.

Apart from measuring the translational and rotational error of the camera, the **cumulative drift of the visual SLAM algorithm** was also estimated. In total, three test sequences were conducted using the HD1080 resolution of the camera. The cumulative drift was measured by comparing the difference between the 3D position reported by the camera at the beginning and at the end of a long test circle, which in ideal cases should be the equal. The three test sequences had an average length of 121.9 meters and on average they had a cumulative drift of 2.13 meters (see Table 15 and Figure 56). Averaging the three test sequences, the ZED 2 stereo camera had an average drift of 1.747%. To be able to benchmark this result to other solutions, the result of the current first visual SLAM method of the KITTI Vision Benchmark Suite was looked at. Zhang et al. [75] reported a 0.75% relative position drift using the KITTI dataset. The slightly worse result can be explained with the bare, flat surfaces of the office, which makes tracking for visual SLAM systems harder. Therefore, the accuracy can decrease.

To estimate the **relocalization accuracy** of the camera, its position was measured at specific locations in the SZTAKI MIMO Arena. Once while recording the Arena and generating the *area map* and once after successfully loading the previously recorded *area map* and the coordinate frames of the two recordings have been automatically aligned. The three test sequences, after relocalization, had an average discrepancy of 66.71 millimeters. (Table 16)

As a final test, the **object detection** of the camera was also assessed. A qualitative test process was carried out and 51 images (HD1080 resolution) were captured using the ZED 2 stereo camera. After annotating the images using an online annotation tool (Figure 58), the assessment of the object detection performance was done by using the previously mentioned toolkit proposed by Padilla et al. [37] (Figure 59). The result of the qualitative test is that the **ZED object detector has a mean Average Precision value of 0.5333 and a mean inference time of 51,74 ms.** These values were compared to four other state-of-the-art object detectors (Table 17). The ZED object detector outperformed these architectures. However, the ZED object detector detects only 22 classes, while the other four can detect 80 different objects. Therefore, these results should be treated only as relative benchmark values. Moreover, the object detection performance was measured using a qualitative test process and the test dataset only consisted of 51 images, which is short of a complete coverage of performance. **Nevertheless, it has been proved that the ZED object detection algorithm is capable of reliable real-time object detection.**

Following the assessment of the ZED 2 stereo camera, a **3D change detection architecture** was proposed. The algorithm is able to detect 3D object-level changes between two consecutive patrol routes. The system was initially inspired by the work of Sünderhauf et al. [30], which described an object-oriented semantic mapping framework. The change detection algorithm of this thesis builds an initial object-oriented semantic map (Figure 63) with the help of the ZED object detector, depth sensing of the camera and a **data association step** (Figure 65), which takes care of creating a coherent object database. In the data association step, the objects are queried based on their Euclidean distance and on the 3D point cloud of the closest objects, a nearest neighbour search is performed. If at least 50% of the associated 3D point pairs of the two point clouds have a distance smaller than a predefined threshold, the detected object will be associated with the already saved object in the semantic object-map. Otherwise, the detected object is added as a new object to the object-map. The object database is then saved (Figure 67 and 68) along with the ZED-generated *area map* of the environment (which consists of key images and features of the vitised scenes), and is loaded during the next patrol route. The changes are detected by comparing the previously saved object database with the currently being built object database (Figure 69). Those objects are being considered for change detection assessment that lie in the 3D Field Of View (FOV) of the camera.
The **change detection algorithm**, due to the unavailability of *.svo* format databases and lack of time, **was tested qualitatively**. The algorithm **accurately detected unchanged, removed and new objects in all three test sequences** (Figure 73). However, it had **difficulties** when the detected objects were **partially occluded** or were **too close to the camera** (below ~1 meter). In these cases, the accuracy of the change detection algorithm has dropped and because of the partially available point clouds of objects, it detected changes even though the objects did not move compared to the previous patrol route. To test objects with changes in orientation or translation, the SZTAKI MIMO Arena was used. It has been determined that due to the fact that during testing the nearest neighbour threshold was set to 10 centimeters, the algorithm did not detect changes in translation less than the threshold (Figure 76). This **threshold is a trade-off between accurate change detection and coherent semantic**

object-map. It has also been determined that the sensitivity of the change detection algorithm depends greatly on the object in question. More in detail, it is very sensitive to the texture and 3D shape and the obtained 3D point cloud representation of the object.

Previously in the thesis, it was mentioned that the HD1080 resolution is only reliable up to around 10 meters. After that, the depth error starts increasing. Therefore, the maximum distance of depth sensing was limited to 7 meters in the change detection application to achieve a coherent semantic object-map. In the future development of the 3D change detection algorithm, the possibility to remove this 'hard-coded' limitation will be investigated by introducing a probabilistic framework that associates a probabilistic value with each measurement. This would help create a more coherent semantic object-map which in theory could work up to 20 meters in distance.
In the future, next to the probabilistic framework for the 3D depth measurement of the ZED 2 stereo camera, the performance of the change detection algorithm will also be more thoroughly measured and an investigation will be conducted into the possibility of improving the current state of the algorithm, such as the handling of partially occluded and close objects. Furthermore, the data association step will also be updated to not only rely on the segmented 3D point cloud of objects, but their additional features, such as color, point cloud density.

6. Bibliography

[1] ZED 2 Camera Datasheet and SDK Overview. [Online]. Available: https://www.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf [Accessed: 13-May-2021]

[2] “Computer stereo vision” Wikipedia (2020). https://en.wikipedia.org/wiki/Computer_stereo_vision. [Accessed: 13-May-2021]

[3] “Built for the spatial AI era”. StereoLabs. [Online]. Available: https://www.stereolabs.com/zed-2/ [Accessed: 13-May-2021]

[4] Davison, A. (2018). FutureMapping: The Computational Structure of Spatial AI Systems. ArXiv, abs/1803.11288.

[5] *“From SLAM to Spatial AI” - Andrew Davison*. 2020 [Online]. Available: https://www.youtube.com/watch?v=lGIM2WVp5t0. [Accessed: 13-May-2021]

[6] “Spatial AI: Augmenting SLAM technology with deep learning”. SlamCore [Online] Available: https://blog.slamcore.com/spatial-ai-slam-deep-learning [Accessed: 13-May-2021]

[7] Spatial intelligence for robots & machines. 2021 [Online]. Available: https://www.slamcore.com/. [Accessed: 13-May-2021]

[8] Davison, "Real-time simultaneous localisation and mapping with a single camera," *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 1403-1410 vol.2, doi: 10.1109/ICCV.2003.1238654.

[9] R. A. Newcombe, S. J. Lovegrove and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," 2011 International Conference on Computer Vision, 2011, pp. 2320-2327, doi: 10.1109/ICCV.2011.6126513.

[10] J. McCormac, A. Handa, A. Davison and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 4628-4635, doi: 10.1109/ICRA.2017.7989538.

[11] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1352-1359, doi: 10.1109/CVPR.2013.178.

[12] Ashbindu Singh (1989) Review Article Digital change detection techniques using remotely-sensed data, International Journal of Remote Sensing, 10:6, 989-1003, DOI: 10.1080/01431168908903939

[13] Théau J. (2008) Change Detection. In: Shekhar S., Xiong H. (eds) Encyclopedia of GIS. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-35973-1_129

[14] You, Y.; Cao, J.; Zhou, W. A Survey of Change Detection Methods Based on Remote Sensing Images for Multi-Source and Multi-Objective Scenarios. Remote Sens. 2020, 12, 2460. https://doi.org/10.3390/rs12152460

[15] Rongjun Qin, Jiaojiao Tian, Peter Reinartz, "3D change detection – Approaches and applications", ISPRS Journal of Photogrammetry and Remote Sensing, Volume 122, 2016, Pages 41-56, ISSN 0924-2716, https://doi.org/10.1016/j.isprsjprs.2016.09.013

[16] D. Lu Corresponding author, P. Mausel, E. Brondízio & E. Moran (2004) Change detection techniques, International Journal of Remote Sensing, 25:12, 2365-2401, DOI: 10.1080/0143116031000139863

[17] Girardeau-Montaut, Daniel & Roux, Michel & Marc, Raphaël & Thibault, Guillaume. (2005). Change detection on point cloud data acquired with a ground laser

scanner. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 36.
[18] Fanfani, M. and Colombo, C. (2019). Structural Change Detection by Direct 3D Model Comparison. In Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP, ISBN 978-989-758-354-4 ISSN 2184-4321, pages 760-767. DOI: 10.5220/0007260607600767
[19] M. Golparvar-Fard, F. Peña-Mora and S. Savarese, "Monitoring changes of 3D building elements from unordered photo collections," 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), 2011, pp. 249-256, doi: 10.1109/ICCVW.2011.6130250.
[20] A. Taneja, L. Ballan and M. Pollefeys, "Geometric Change Detection in Urban Environments Using Images," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 11, pp. 2193-2206, 1 Nov. 2015, doi: 10.1109/TPAMI.2015.2404834.
[21] A. Taneja, L. Ballan and M. Pollefeys, "City-Scale Change Detection in Cadastral 3D Models Using Images," 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 113-120, doi: 10.1109/CVPR.2013.22.
[22] E. Palazzolo and C. Stachniss, "Fast Image-Based Geometric Change Detection Given a 3D Model," 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018, pp. 6308-6315, doi: 10.1109/ICRA.2018.8461019.
[23] K. Sakurada, T. Okatani and K. Deguchi, "Detecting Changes in 3D Structure of a Scene from Multi-view Images Captured by a Vehicle-Mounted Camera," 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 137-144, doi: 10.1109/CVPR.2013.25.
[24] H. Andreasson, M. Magnusson and A. Lilienthal, "Has something changed here? Autonomous difference detection for security patrol robots," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 3429-3435, doi: 10.1109/IROS.2007.4399381.
[25] Shi W, Zhang M, Zhang R, Chen S, Zhan Z. Change Detection Based on Artificial Intelligence: State-of-the-Art and Challenges. Remote Sensing. 2020; 12(10):1688. https://doi.org/10.3390/rs12101688
[26] Varghese A., Gubbi J., Ramaswamy A., Balamuralidhar P. (2019) ChangeNet: A Deep Learning Architecture for Visual Change Detection. In: Leal-Taixé L., Roth S. (eds) Computer Vision – ECCV 2018 Workshops. ECCV 2018. Lecture Notes in Computer Science, vol 11130. Springer, Cham. https://doi.org/10.1007/978-3-030-11012-3_10
[27] W. Lawson, L. Hiatt and K. Sullivan, "Detecting Anomalous Objects on Mobile Platforms," 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2016, pp. 1426-1433, doi: 10.1109/CVPRW.2016.179.
[28] Andreas Nüchter, Joachim Hertzberg, "Towards semantic maps for mobile robots", Robotics and Autonomous Systems, Volume 56, Issue 11, 2008, Pages 915-926, ISSN 0921-8890, https://doi.org/10.1016/j.robot.2008.08.001.
[29] Y. Nakajima and H. Saito, "Efficient Object-Oriented Semantic Mapping With Object Detector," in IEEE Access, vol. 7, pp. 3206-3213, 2019, doi: 10.1109/ACCESS.2018.2887022.
[30] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford and I. Reid, "Meaningful maps with object-oriented semantic mapping," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 5079-5085, doi: 10.1109/IROS.2017.8206392.

[31] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu and J. Song, "Semantic SLAM Based on Object Detection and Improved Octomap," in IEEE Access, vol. 6, pp. 75545-75559, 2018, doi: 10.1109/ACCESS.2018.2873617.
[32] Freeman H., Meagher D.J. (1985) Octrees: A Data Structure for Solid-Object Modeling. In: Freeman H., Pieroni G.G. (eds) Computer Architectures for Spatially Distributed Data. NATO ASI Series (Series F: Computer and Systems Sciences), vol 18. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-82150-9_14
[33] P. H. Truong, S. You and S. Ji, "Object Detection-based Semantic Map Building for A Semantic Visual SLAM System," 2020 20th International Conference on Control, Automation and Systems (ICCAS), 2020, pp. 1198-1201, doi: 10.23919/ICCAS50221.2020.9268441.
[34] Joo S-H, Manzoor S, Rocha YG, Bae S-H, Lee K-H, Kuc T-Y, Kim M. Autonomous Navigation Framework for Intelligent Robots Based on a Semantic Environment Modeling. Applied Sciences. 2020; 10(9):3219. https://doi.org/10.3390/app10093219
[35] Rocha, Y.G.; Kuc, T.Y. Mental simulation for autonomous learning and planning based on triplet ontological semantic model. CEUR Workshop Proc. 2019, 2487, 65–73.
[36] Kunze, L., Karaoguz, H., Young, J., Jovan, F., Folkesson, J., Jensfelt, P., & Hawes, N. SOMA: a framework for understanding change in everyday environments using Semantic Object Maps. 47–54.
[37] Padilla, Rafael; Passos, Wesley L.; Dias, Thadeu L.B.; Netto, Sergio L.; da Silva, Eduardo A.B. 2021. "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit" Electronics 10, no. 3: 279. https://doi.org/10.3390/electronics10030279
[38] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.
[39] Dumoulin, Vincent and Francesco Visin. “A guide to convolution arithmetic for deep learning.” ArXiv abs/1603.07285 (2016): n. pag.
[40] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
[41] Image Classification on ImageNet [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet [Accessed: 13-May-2021]
[42] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.
[43] Liu, L., Ouyang, W., Wang, X. et al. Deep Learning for Generic Object Detection: A Survey. Int J Comput Vis 128, 261–318 (2020). https://doi.org/10.1007/s11263-019-01247-4
[44] “Faster R-CNN Explained for Object Detection Tasks”. Ahmed Fawzy Gad. 2021. [Online]. Available: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/ [Accessed: 13-May-2021]
[45] Liu W. et al. (2016) SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham.

https://doi.org/10.1007/978-3-319-46448-0_2
[46] Karen Simonyan, and Andrew Zisserman. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.
[47] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe. (2015). Scalable, High-Quality Object Detection. arXiv 1412.1441
[48] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
[49] "YOLO: Real-Time Object Detection". J. Redmon. [Online]. Available: https://pjreddie.com/darknet/yolov2/. [Accessed: 13-May-2021]
[50] "YOLO Explained". Ani Aggarwal. [Online]. Available: https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31. [Accessed: 13-May-2021]
[51] "Open-Source Toolbox for Object Detection Metrics". Padilla, Rafael and Passos, Wesley L. and Dias, Thadeu L. B. and Netto, Sergio L. and da Silva, Eduardo A. B.. 2021. [Online]. Available: https://github.com/rafaelpadilla/review_object_detection_metrics. [Accessed: 13-May-2021]
[52] "Intersection over Union (IoU) for object detection". Adrian Rosebrock. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. [Accessed: 13-May-2021]
[53] R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), 2020, pp. 237-242, doi: 10.1109/IWSSIP48289.2020.9145130.
[54] Sturm P. (2014) Pinhole Camera Model. In: Ikeuchi K. (eds) Computer Vision. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-31439-6_472
[55] "Camera obscura". Wikipedia (2011). [Online] Available: http://en.wikipedia.org/wiki/Camera_obscura. [Accessed: 13-May-2021]
[56] Shirai Y. (1987) Stereo Vision. In: Three-Dimensional Computer Vision. Symbolic Computation (Computer Graphics — Systems and Applications). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-82429-6_7
[57] "Image rectification" Wikipedia (2021). https://en.wikipedia.org/wiki/Image_rectification [Accessed: 13-May-2021]
[58] L. Hajder, D. Chetverikov "Basics of Stereo Vision" [Online lecture] Available: http://cg.elte.hu/~hajder/vision/slides/lec02_stereo.pdf [Accessed: 13-May-2021]
[59] Hahne, C., Aggoun, A., Velisavljevic, V. et al. Baseline and Triangulation Geometry in a Standard Plenoptic Camera. Int J Comput Vis 126, 21–35 (2018). https://doi.org/10.1007/s11263-017-1036-4
[60] Recommended Specifications for ZED SDK. [Online]. Available: https://www.stereolabs.com/docs/installation/specifications/ [Accessed: 13-May-2021]
[61] Introduction of ZED 2 stereo camera and ZED SDK. [Online]. Available: https://www.stereolabs.com/docs/ [Accessed: 13-May-2021]
[62] GitHub repository of the ZED SDK [Online]. Available: https://github.com/stereolabs. [Accessed: 13-May-2021]
[63] Abu Hassan, Mohd Fadzil & Hussain, Aini & Md Saad, Mohamad Hanif & Win, Kong. (2017). 3D DISTANCE MEASUREMENT ACCURACY ON LOW-COST STEREO CAMERA. Scientific International. 29. 599-605.

[64] A. Geiger, F. Moosmann, Ö. Car and B. Schuster, "Automatic camera and range sensor calibration using a single shot," 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 3936-3943, doi: 10.1109/ICRA.2012.6224570.
[65] Mikko Kytö, Mikko Nuutinen, Pirkko Oittinen, "Method for measuring stereo camera depth accuracy based on stereoscopic vision," Proc. SPIE 7864, Three-Dimensional Imaging, Interaction, and Measurement, 78640I (27 January 2011); https://doi.org/10.1117/12.872015
[66] Camera Calibration Toolbox for Matlab [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/. [Accessed: 13-May-2021]
[67] Fernandez, L.; Avila, V.; Goncalves, L. A Generic Approach for Error Estimation of Depth Data from (Stereo and RGB-D) 3D Sensors. Preprints 2017, 2017050170 (doi: 10.20944/preprints201705.0170.v1).
[68] Ortiz, Luis & Cabrera, Elizabeth & Gonçalves, Luiz. (2018). Depth Data Error Modeling of the ZED 3D Vision Sensor from Stereolabs. Electronic Letters on Computer Vision and Image Analysis. 17. 10.5565/rev/elcvia.1084.
[69] Least-Squares Rigid Motion Using SVD. Olga Sorkine-Hornung and Michael Rabinovich. [Online]. Available: https://igl.ethz.ch/projects/ARAP/svd_rot.pdf. [Accessed: 13-May-2021]
[70] Rozenberszki, Dávid, Majdik, András. (2019) The MTA SZTAKI micro aerial vehicle and motion capture arena. In: KÉPAF 2019: Képfeldolgozók és Alakfelismerők Társaságának 12. országos konferenciája, 2019.01.28-2019.01.31, Debrecen. https://eprints.sztaki.hu/9701/1/Rozenberszki_1_30762070_ny.pdf
[71] OptiTrack Motion Capture System [Online]. Available: https://optitrack.com/. [Accessed: 13-May-2021]
[72] Prime 13 camera. [Online]. Available: https://www.optitrack.com/cameras/prime-13/. [Accessed: 13-May-2021]
[73] Motive, Optical motion capture software. [Online]. Available: https://optitrack.com/software/motive/. [Accessed: 13-May-2021]
[74] The KITTI Vision Benchmark Suite. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_odometry.php. [Accessed: 13-May-2021]
[75] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2174-2181, doi: 10.1109/ICRA.2015.7139486.
[76] MakeSense Annotation Tool. [Online]. Available: https://www.makesense.ai/. [Accessed: 13-May-2021]
[77] X. Lu, X. Kang, S. Nishide and F. Ren, "Object detection based on SSD-ResNet," 2019 IEEE 6th International Conference on Cloud Computing and Intelligence Systems (CCIS), 2019, pp. 89-92, doi: 10.1109/CCIS48116.2019.9073753.
[78] Chengcheng Ning, Huajun Zhou, Yan Song and Jinhui Tang, "Inception Single Shot MultiBox Detector for object detection," 2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW), 2017, pp. 549-554, doi: 10.1109/ICMEW.2017.8026312.
[79] Ji-qing Luo, Hu-sheng Fang, Fa-ming Shao, Yue Zhong, Xia Hua, Multi-scale traffic vehicle detection based on faster R–CNN with NAS optimization and feature enrichment, Defence Technology, 2020, ISSN 2214-9147, https://doi.org/10.1016/j.dt.2020.10.006.
[80] Bochkovskiy, Alexey & Wang, Chien-Yao & Liao, Hong-yuan. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.
[81] Jon Louis Bentley. 1990. K-d trees for semi dynamic point sets. In Proceedings of the sixth annual symposium on Computational geometry (SCG '90). Association for

Computing Machinery, New York, NY, USA, 187–197. DOI:https://doi.org/10.1145/98524.98564

[82] kd-trees. Carnegie Mellon School of Computer Science. [Online]. Available: https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/kdtrees.pdf. [Accessed: 13-May-2021]