Tampereen yliopisto

Mikael Petäjä

LEARNING-BASED 3D SCENE RECONSTRUCTION USING RGBD CAMERAS

Bachelor's Thesis
Faculty of Engineering and Natural Sciences
Examiner: Tarun Devalla
May 2021

ABSTRACT

Mikael Petäjä: Learning-based 3D scene reconstruction using RGBD cameras
Bachelor's Thesis
Tampere University
Bachelor of Science (Technology), Degree Programme in Engineering Sciences, Automation Engineering
May 2021

Machine learning methods and object recognition algorithms have improved much in the past decade, but computer perception and object recognition remain some of the biggest challenges of modern engineering. A close relative of these is scene reconstruction, in which the computer attempts to create a digital reconstruction of the environment it is perceiving.

This thesis considers the use of RGB-D cameras in room reconstruction, which is a particularly interesting field of scene reconstruction for mobile robots. RGB-D cameras and reconstruction algorithms are not as widespread as LiDAR-based applications in robots but can be cost-efficient replacements in certain situations. In this thesis, applications of room reconstruction methods are also discussed by giving an overlook of state-of-the-art algorithms.

This thesis is divided into two parts. First in the literary review section elucidates upon the basic theory of the subject and presents the current state of room reconstruction. The experimental part of the work examines in detail the operation of the algorithms used. Finally, achieved results are displayed and analyzed along with potential future research.

The reconstruction of the test environment was manufactured with a single moving RGB-D capable camera, and object recognition semantics was applied to this scene. This was achieved by applying InstanceFusion on a dataset collected with a Stereolabs ZED camera. The achieved reconstruction is not as good as examples in research material, and reasons for this are explored.

Keywords: reconstruction, 3D, RGBD, ZED, machine learning, computer vision, machine vision, semantic reconstruction, InstanceFusion, ElasticFusion, SemanticFusion, Mask R-CNN, SLAM, object recognition

TIIVISTELMÄ

Mikael Petäjä: Learning-based 3D scene reconstruction using RGBD cameras
Kandidaatintyö
Tampereen Yliopisto
Teknisten tieteiden kandidaattiohjelma, Automaatiotekniikka
Toukokuu 2021

Koneoppimisen menetelmät sekä kappaleentunnistusalgoritmit ovat kehittyneet paljon viime vuosikymmenen aikana, mutta konenäkö sekä esineiden tunnistus ovat edelleen nykytekniikan haastavimpia ongelmia. Näiden eräs sovellus on näkymärekonstruktio, jossa tietokone yrittää luoda digitaalisen jäljitelmän havaitsemastaan ympäristöstä.

Tässä työssä tarkastellaan RGB-D kameroiden käyttöä huonerekonstruktiossa, mikä on erityisen mielenkiintoinen näkymärekonstruktion sovellus mobiilirobotiikassa. RGB-D kameroihin pohjautuvat rekonstruktioalgoritmit eivät ole robotiikassa yhtä laajassa käytössä kuin LiDAR -pohjaiset sovellukset, mutta ne voivat olla joissain tilanteissa kustannustehokas vaihtoehto.

Työ itse on jaettu kahteen osaan. Ensiksi kirjallisuustutkimusosa keskittyy pohjateorian selvittämiseen sekä perehtyy huonerekonstruktion nykytilaan. Kokeellisessa osuudessa tarkastellaan tarkemmin käytettyjen algoritmien toimintaa sekä esitetään tiedonkeruuseen käytetyt menetelmät. Lopuksi tarkastellaan saatuja tuloksia sekä esitetään potentiaalisia tulevaisuuden tutkimusaiheita.

Rekonstruktio, joka työssä esitellään, aikaansaatiin yhdellä liikkuvalla RGB-D videokuvaan kykenevällä kameralla. Kappaleentunnistusta liitetään tähän ympäristöön semantiikka-analyysillä, jota ajettiin rekonstruktioalgoritmin ohessa. Tämä toteutettiin soveltamalla InstanceFusion -algoritmia Stereolabs ZED -kameralla kerättyyn datajoukkoon. Aikaansaatu rekonstruktio on tutkimusesimerkkejä huonompi ja syitä tälle tarkastellaan.

Avainsanat: rekonstruktio, 3D, RGBD, ZED, koneoppiminen, konenäkö, semanttinen rekonstruktio, InstanceFusion, ElasticFusion, SemanticFusion, Mask R-CNN, SLAM, esineiden tunnistus

FOREWORD

This thesis was chosen out of personal interest to a subject that was interesting for me. Installing InstanceFusion proved to be a surprisingly momentous hurdle, evident since I had to reinstall Linux thrice due to breaking it. I want to extend my thanks to my supervisor, Tarun Devalla, who was patient with my slow pace and provided extra time for me to finish this thesis.

I learned a lot of new things in the process of writing this thesis, such as how to use CMake and Ms. Visual Studio along with investigating a subject previously unknown to me. In the end, this thesis hasn't ended up as I first envisioned it. Results were disappointing to me, but in every failure, there is a lot of potential for learning. At first, I produced a lot of scans which gave almost completely unusable models, but the product seen in this thesis at least represents what I thought I would get at first. Had I more time, the ZED camera could be parametrized further and results thus bettered, but even bad results can be good when analyzed.

At Tampere, 4.5.2021

Updater: Mikael Petäjä

TABLE OF CONTENTS

ABBREVIATIONS AND SYMBOLS

AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CAD	Computer-Aided Design
CMYK	Cyan-Magenta-Yellow-Key -color model
CNN	Convolutional Neural Network
fps	Frames per second
Lidar	Light detection and ranging
MEMS	Microelectromechanical system
NI	Natural interface
png	portable network graphics
RGB	Red-Green-Blue
RGBD	RGB with depth
RGB-D	RGB with depth
SLAM	Simultaneous Localization and Mapping
VR	Virtual Reality
YCbCr	Luma, blue-difference, red-difference -chroma component model
YOLO	You Only Look Once
2D	Two-dimensional
3D	Three-dimensional
cm	Centimeter
mm	Millimeter
Hz	Hertz, 1/s
Px	Pixel
s	second

1. INTRODUCTION

While machine learning methods and object recognition algorithms have improved much in the past decade, computer perception and object recognition remain some of the biggest challenges of modern engineering. A close relative of these is scene reconstruction, in which the computer attempts to create a digital reconstruction of the environment it is perceiving. Object recognition algorithms are already making their way to our lives in the way of facial recognition algorithms or filters on Instagram, but new problems are found as engineers become more ambitious with their desired applications.

Mobile robots are robots capable of moving in an environment and possibly also interacting with it. Some industries are already implementing these robots, such as Amazon's robotic fulfillment centers. As of 2020, these shelving robots have however increased staff injuries by about 50% when compared to nonrobotic facilities (Evans 2020). One of the biggest current problems in mobile robots is how the robot navigates in an unknown environment. This problem is especially accentuated if the environment is changing as the robot is operating, such as with humans walking in it. According to Corke (2017), a robot only moves to a position or pose in which it expects the goal to be and as such, will fail if reality has changed from its assumptions.

Researchers have developed simultaneous localization and mapping (SLAM) techniques to solve this issue and recent algorithms by the influx of machine learning have become very fast. The idea behind a SLAM algorithm is to create a digital reconstruction of the environment and estimate a robot's state in it (Cadena et al. 2016). Additionally, an auxiliary Machine Learning algorithm can then be used to classify items in the reconstruction. To solve the issue of moving in a changing environment, these algorithms would need to run in real-time. Recent algorithms, such as InstanceFusion (Lu et al. 2020) and SemanticFusion (McCormac et al. 2017), can create these item estimations at a framerate of over 20 Hz on consumer-grade hardware. Increased estimation speeds result in a faster model, which in turn allows an actor utilizing this information to react quickly. Increased framerates could also improve the actor's positional awareness (localization) if it is moving in the environment or if the environment is moving.

Generally speaking, a reconstruction of a room can be made from static RGBD pictures at different angles, videos, or rotational scans from a single point. Single-angle reconstructions and scan reconstructions can be particularly useful if the goal is to create digital environments to create digital cities (Bláha et al. 2016). This type of reconstruction has created a market of its own with applications in aerospace & defense, bathymetry, mining, railways, and roadways. Companies, such as Euclideon (Euclideon 2021) provide fast point-cloud data visualizations, which could in the future be used in game environments and have current applications in the aforementioned fields. A reconstruction that a robot would use to navigate a given environment should also have the ability to be updated.

The purpose of this thesis is to create a semantic reconstruction of a room with InstanceFusion (Lu et al. 2020). The room will be made from a video dataset collected with a Stereolabs ZED stereo camera. InstanceFusion is designed to function in real-time and as such, using the algorithm with discrete data will also be a goal for this thesis. The second chapter gives a deeper but still general overview of the available reconstruction method. The third chapter focuses on the image acquisition methods and theory as well as giving a technical overview of the camera used. The fourth chapter depicts the used methodology and elaborates on the general function of the algorithms used. Finally, the fifth chapter gives an overview of attained results with visualizations and conclusions drawn from the results. The thesis closes with a small summary of attained results and conclusions.

2. LEARNING-BASED SCENE RECONSTRUCTION

In this chapter, we take a look at the general function of a learning-based scene reconstruction algorithm by presenting existing solutions. A general understanding of machine learning is expected of the reader, but a general overview of neural networks can be found in chapter 2.3 and its subchapter. An understanding of the RGBD model and point-cloud data is also recommended and their general overview can be found in chapter 3 and associated subchapters.

A point-cloud data, that is, the raw RGB data with depth (RGBD) provided by a capable sensor is typically very fragmented. Machine learning algorithms are applied to these datasets to give proper alignment to the depths of pixels. The goal of this is to identify surfaces and shapes. Note that accurate depth perception methods such as Light Detection and Ranging (Lidar) by mobile laser scanners can also be utilized, as the noise levels in Lidar-based depth data can be orders of magnitude lower than in stereo-generated point clouds (Babahajiani et al. 2017). The 3D data used in Google Maps Street View, for example, was collected with a car that had a roof-mounted Lidar scanner and auxiliary cameras. Fast-speed video-based real-time reconstruction from RGBD cameras is especially desirable in mobile robots because of the high cost associated with Lidar sensors. RGBD camera reconstruction however brings in more problems, such as how to properly align the frames of a video, which is connected to the problem of aligning static shots from multiple angles.

Generally, image segmentation (to properly align pictures) is made accurate by utilizing machine learning. For example, Lu et al. (2020) use similarities between pixels and based superpixels for distance, color, “point-to-plane” and the normals between these superpixels. The total similarity score is then maximized using neural regression. Superpixels themselves are made from groups of similar pixels to minimize the amount of data needed to describe a location in an image. The algorithm used in InstanceFusion divides an image into 1200 superpixels, reducing boundary errors.

Picture 1. *An RGB image and clustered superpixels (Lu et al. 2020)*

Picture 1 depicts the superpixel division done by InstanceFusion. Note how similar surfaces, such as the table, are combined into one large group. In contrast, the geometrically complex hoodie contains many smaller superpixels.

When a scene is reconstructed, object recognition algorithms can be run on it. This is typically done by convolutional neural networks (CNN), which are presented in chapter 2.3.1. This is called semantics analysis and with it, items in a scene can be labeled with object names. The InstanceFusion implementation used in this thesis can detect, for example, chairs computers, and people from an image by utilizing Mask R-CNN on the 2D RGB picture.

Typically, the depth data is provided by the RGBD capable camera, but depth prediction can also be done on 2D images from multiple angles or moving cameras if the depth data is not otherwise available. Scene reconstruction is typically done from RGBD data, as using no depth data yields somewhat more inaccurate scenes. Similarly, good results can be achieved from just RGB data in certain situations using predicted depths (Eigen et al. 2015).

Most moving cameras move rotationally in closed loops or a corridor-like motion with a forward-facing camera. In a loop, the end location is similar to the start location, and loop closure needs to be achieved. In the results chapter, we will see that loop closure can fail due to poorly collected data giving the model multiple instances of the same wall in different locations. Looping movement methods are simple to mathematically model and are present in current generation robots. Some recent research focuses algorithms also allow crossing loops such as in the paper by Whelan et al. (2016). This type of movement brings with it the problem of scanning the same surface multiple times, but it could be more akin to a robot exploring an unknown environment as opposed to a looping motion.

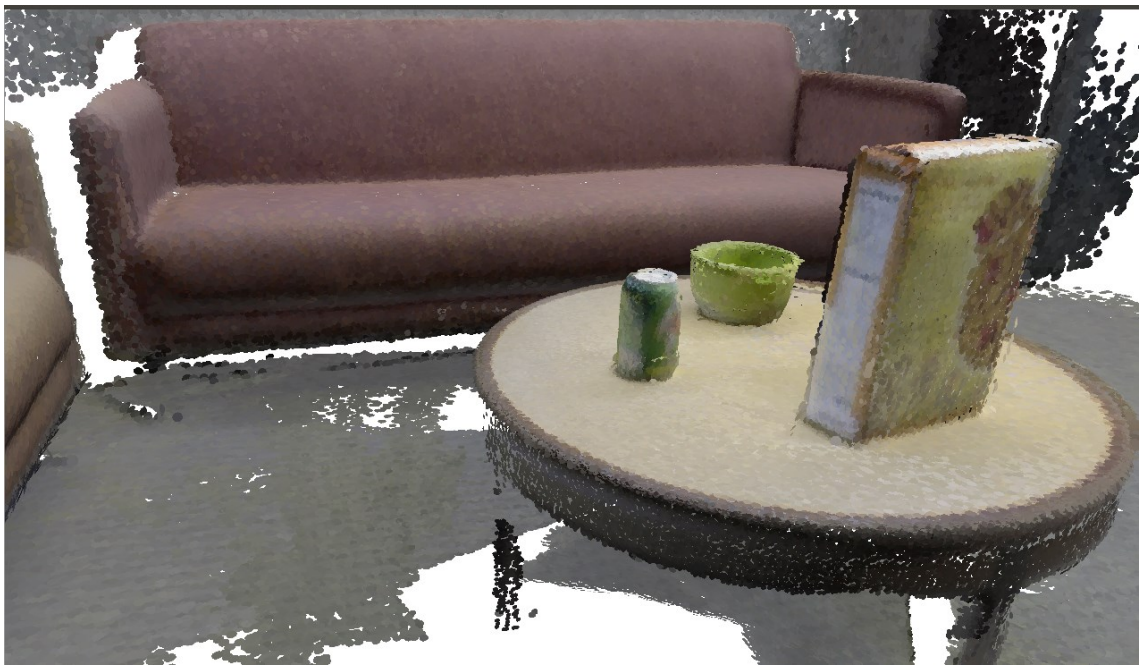
Current scene reconstruction research focuses on indoor environments or city architecture since nature is geometrically very complex and difficult to model. Some potential

applications of 3D scene reconstruction, such as self-driving cars, could potentially get into geometrically complex environments. According to Yoo et al. (2018), the autonomous driving industry uses mostly commercially available Lidar sensors and uses an optical phase array or microelectromechanical system (MEMS) mirrors as cheaper alternatives. The function of RGBD sensors in real environments is limited due to the inherent geometrical complexity. Potentially dangerous cases where reconstruction systems may fail include missing data from incomplete scans, thin and small parts such as a metallic chair handle, occlusions such as heavy shadowing, and object irregularity such as many shapes of pillows and ruffled fabric (Lu et al. 2020). Leaves from natural trees potentially fulfill all of these difficult cases.

Light reflection on surfaces can be problematic in scene reconstruction, especially in SLAM algorithms. Some reconstruction algorithms, such as ElasticFusion (Whelan et al. 2016) place special emphasis on light source estimation. The removal of estimated light source reflections improves tracking, reduces smearing, and reduces color distortion when applying future scans on reflective surfaces. Beyond robotics, light source estimation can create visually pleasing augmented reality (AR) environments and allow realistic object-scene interactions in virtual reality (VR), such as picking up a book (Whelan et al. 2016).

2.1 Surfels

Surfels are a way of storing image color and depth information slightly differently from separate RGB and depth pixels. In a surfel-based environment, an object is represented by a dense set of points discs oriented to the object geometry. According to Pfister et al. (2000), these discs have a shape, hold color and lighting information, and approximate the object only locally. An object represented by surfels can appear perforated when viewed at a close range. In a scene made from point-cloud data, the use of surfels can be practical. Another way to display point-cloud data is to use voxels instead of surfels. Voxels or volume elements are small 3D elements, or little cubes, used in polygonal modeling. Some reconstruction methods such as Bláha et al. (2016) propose an adaptive use of voxel resolution as a more efficient method of storing scenes. Unlike voxels, surfels themselves do not have depth and represent the object only from one direction by default. They represent a 3D shape by location and color but do not necessarily form a geometric surface themselves. Surfels can however be connected to form true surfaces.



Picture 2. *A surfel-based model of scene_09 from the dataset by Lai et al. (2014).*

Picture 2 presents a surfel-based view of the point-cloud data generated by InstanceFusion in chapter 5. Note that the results in chapter 5 are presented with raw point-cloud data without surfel normal estimations and as such, have less defined geometric features and more perforated surfaces.

Even when using surfels, initial RGBD reconstructions can be rough and unnatural due to the nature of the data. Items can however be filtered in the surfel-based models to get smooth and consistent objects, such as in the algorithm by Lu et al. (2020). With surfels, there is no need for neighbor information or other topology computation to get natural lighting in scenes. Therefore, they are well suited for dynamic geometry modeling, which is inherent to scene reconstruction.

2.2 Simultaneous localization and mapping (SLAM)

Simultaneous localization and mapping (SLAM) is an application of scene reconstruction. In SLAM, the environment is constructed simultaneously as the agent moves in it while also keeping track of the agent's position in this environment.

Artificial reality (AR) benefits from SLAM especially, as visual SLAM can be the base for such environments. With modern applications such as InstanceFusion, systems can detect and reconstruct instance-level objects. This is done using semantics analysis and may lead to new kinds of interactions in AR and applications. (Lu et al. 2020)

Robot perception is also at the heart of SLAM and instance-level object semantics. Robots can use generated reconstructions in such tasks as navigation and object grasping. (Lu et al. 2020). According to McCormac et al. (2017), in the future, such maps may also lead to robots understanding fetching tasks on a deeper level. Instructions such as “fetch the coffee mug from the nearest table on your right” become interpretable by the robot when it knows all the items in an environment and their relative positions. Semantic maps will also help the robot find the number of certain objects in a location and their recognition becomes a reality.

2.3 Machine Learning

Machine learning is a process in which a computer automatically approximates a mathematical function. Typical applications for machine learning come in problems that are difficult to mathematically model, such as object recognition in images. Machine learning itself bases on artificial neural networks, which generally function by comparing incoming signals and choosing an output signal. Individual neurons which do these comparisons can be arranged in a grid forming neural layers.

A typical neural network consists of multiple layers and is called a deep neural network once it has more than one layer. A neural network is typically initialized with weights which usually represent the default state of changing variables in the net and a cost function. Using the cost function, a neural network can learn. For linear regression, this cost function can be for example the mean squared error between the current state and the desired state. The cost changes when the neural network changes randomly, and typically it is minimized by weighing appropriately. This is done automatically by the neural network for each layer.

A maximally efficient neural network is considered to be such that it performs as well as possible with the minimum amount of required connections (Gordienko, 1993). Some recent research in the field also concentrates solely on shallowing older deep neural networks, such as Gorban et al. (2020).

2.3.1 Convolutional neural networks (CNN)

Convolutional Neural Networks are a special form of Neural Networks that process data with a known grid-like topology. Examples of this are time-dimensional data or 2D image data. Convolution, however, is rarely used as the only mechanism in machine learning. Rather, other functions are used in parallel, which in their entirety are not commutative,

which a sole CNN is. Commutativity means that suboperations can be done in any order feasible without changing the outcome of the whole operation. (Goodman et al. 2018)

Traditional layers in neural networks use matrix multiplication with a parameter matrix that has separate parameters describing the interaction between the individual input and output units (neurons). Each output would therefore theoretically interact with each input unit. CNNs usually have sparse interactions, in which all outputs don't interact with each input. This is beneficial in image processing, for example, because the input image can consist of thousands or millions of pixels, but the algorithm is only detecting small meaningful features such as edges. This detection can happen with groups of only a few dozen or hundred pixels since they are the only meaningful connections. Thus, a CNN is much faster than a traditional neural network in object recognition. (Goodman et al. 2018)

CNNs also make use of parameter sharing to make the learning more efficient. This means using a parameter for more than one function in a model. In a classic neural network, each element of the weighting matrix is used exactly once to calculate the output of a layer. It is then multiplied by an element of the input and then never needed again. In a CNN, every element of a given set is used at every position in the input. Parameter sharing during the convolution operation means, that only one set of parameters is learned instead of many sets. This does not affect the time it takes for the algorithm to finish, but it lessens the amount of memory allocated for the model parameters. In terms of memory requirements and terms of statistical efficiency, convolution is more efficient than multiplication with a fully coupled matrix. (Goodman et al. 2018)

Convolutional neural networks are also used in scene reconstruction. The problem of assessing depth from an RGB image is usually done with a CNN. Missing depth data can also be estimated with a specialized CNN, such as in Palla et al. (2017). Semantics analysis is also a problem that CNNs solve well. For example, SemanticFusion (McCormac et al. 2017) applies a 39-layer CNN to create a semantic analysis to the geometric information determined by ElasticFusion (Whelan et al. 2016)

3. IMAGE ACQUISITION

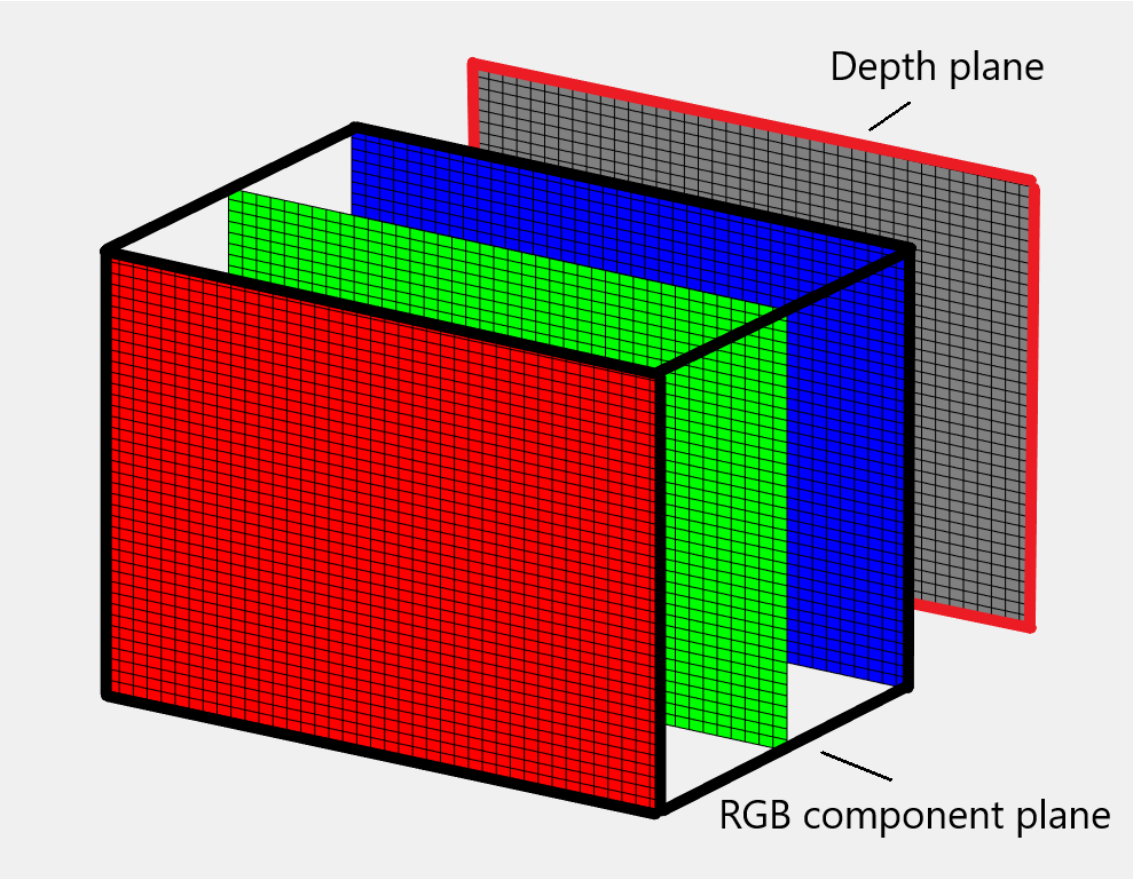
In this chapter, we give a brief overview of what RGBD data is, and the camera used in this thesis for data acquisition. Lidar-based solutions, which were mentioned in chapter 2 will not be explained further, as they are outside of the scope of this thesis.

A stereoscopic camera has two lenses separated by a distance, which allows it to estimate depth in a scene by comparing images taken from slightly differing angles. According to Shaik et al. (2020), in robotics, most current solutions are implemented using expensive laser measurements, but recent RGBD cameras can provide low-cost localization solutions. In certain situations, modern RGBD cameras reach similar uncertainties as laser measurements. Uncertainty however is closely linked to the complexity of the scene in which the robot is in (Shaik et al. 2020).

Most current solutions are implemented with laser range sensors due to their long-range, accuracy, and wide opening angle. However, the price of these sensors can be thousands of dollars, which makes them too expensive in autonomous mobile robots. Recent low-cost RGBD cameras can be low-cost alternatives to these sensors, as depth perception and localization improves (Shaik et al. 2020).

3.1 The RGB color model

The trichromatic theory states that human color perception is dependent on three types of photon receptors, which are sensitive to the red, green, and blue regions in the light spectrum. The eye then combines these into an achromatic response to decorrelate the signal and reduce noise. This signal is then transported via neurons to the visual cortex and interpreted as perception (Fairchild 2013). The Red-Green-Blue (RGB) -color model follows the sensing part of the trichromatic theory closely. In RGB, a color image is produced by superimposing the red, green, and blue color components of an image on top of each other.



Picture 3. *Breakdown of the structure of an RGBD image*

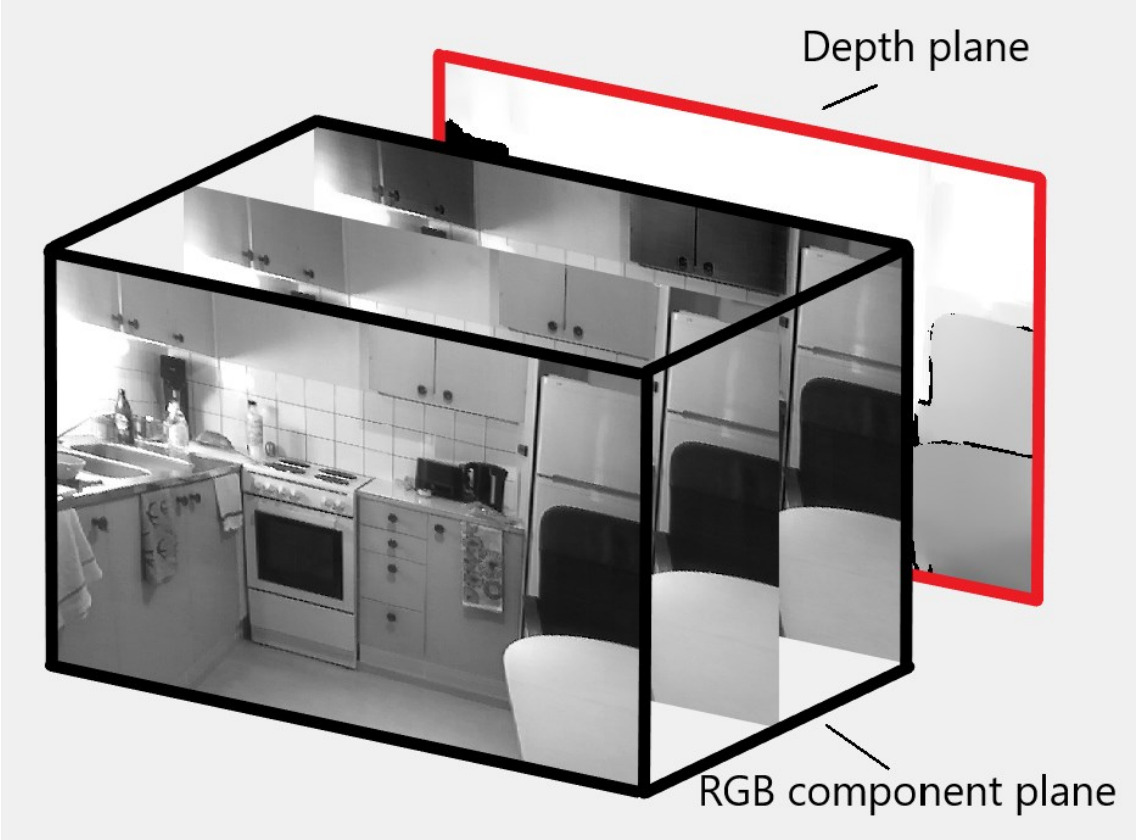
Picture 3 presents the breakdown of an RGB-D image. The portion marked as 'RGB component plane' represents a regular 2D RGB image. The addition of the depth data is further explained in chapter 3.2.

Mathematically, the maximal presence of all RGB component colors would yield a pure white image, but in the real world mixing red, green, and blue paint yields the color black instead. This is because in the real world these colors are not additive. The RGB model is useful in digital devices, where this representation works, but the Cyan-Magenta-Yellow-Key (CMYK) combination is typically used in printers. There are also other systems, such as the YCrCb -model, which could also be coupled with depth perception. However, the Stereolabs ZED cameras (chapter 3.3) used in this thesis capture depth coupled with RGB color data, so other models will not be analyzed further.

3.2 RGB model with depth

An RGB image with depth can be represented in three-dimensional space instead of 2-dimensional planes. An example of these representations is the **point cloud**, in which every captured pixel and its color data is placed to the spot of its real-space equivalent. These models can be rough, and for visual engagement, these are sometimes converted

to 3D surfaces (Berger et al. 2017). These surfaces can be further utilized in CAD modeling and computer graphics. Examples of point clouds can be found in the results chapter, where the result models are colored point-cloud data maps.



Picture 4. *Example of an RGB-D sequence*

Picture 4 presents a sample of the data used in this thesis with the associated depth displayed behind. The RGB component plane represents a colored image, and the three-color data is attached to corresponding depth pixels.

A stereoscopic camera calculates the depth plane by comparing the two-feed images to each other and inferring distance to objects based on angular discrepancies. The Stereolabs ZED camera API provides an option to calculate the depth distance either to the left eye or the right eye. The camera defaults to the left eye, and this setting was used in chapter 4. In the reconstruction of large models, such as rooms, visibility information is typically considered. According to Berger et al (2017), scanners capable of interactive acquisition of geometry, such as Microsoft Kinect or the Stereolabs ZED, can infer the visibility of areas from an image using a truncated signed distance function (TSDF). This function is the predominant representation for dynamic reconstruction in these systems. The TSDF of a scan can be found by comparing the distance between the scanner head and where a straight ray from the scanner head intersects a triangulated range scan (Berger et al. 2017).

3.3 The Stereolabs ZED stereo camera

The Stereolabs ZED is a stereoscopic camera that is capable of taking RGBD-images, and it is the camera used to capture the data in this thesis. Stereolabs provides multi-sensor cameras that include stereo vision, motion, position, and environmental sensing. For ease of use, they also distribute an API that provides low-level access to the camera and sensors. The API also facilitates high-quality video recording and streaming (Stereolabs 2021a).

The camera itself is capable of recording stereo video at configurations, such as: 4416x1242 pixels (px), 3840x1080 px, 2560x720 px and 1344x376 px. The Stereolabs stereo cameras provide depth to cameras by reproducing the way human binocular sight works. Human eyes are separated by 6.5 cm on average, and to emulate this, Stereolabs cameras have two eyes separated by 6 to 12 cm depending on the model. The ZED camera has eyes separated by 12 cm. The depth data itself is captured in depth maps, which store a distance value for each pixel. The distance to the scene object is calculated in metric units from the back of the left eye of the camera. (Stereolabs 2021b). The front of the ZED camera is presented in Picture 5. The ZED camera used in this thesis was encased in an additional plastic shell to protect it from environmental damage.

Picture 5. *The front of the ZED camera (Stereolabs 2021c)*

The Stereolabs cameras are also capable of streaming the captured data directly to a computer, as the depth map is also created in real-time. In this thesis depth data. InstanceFusion was created to work with sensors that are inherently compatible with OpenNI2 such as Microsoft Kinect. The Stereolabs documentation also contains directions on how to use the ZED camera with OpenNI2 but using the algorithm in real-time falls outside the scope of this thesis.

4. METHODOLOGY AND IMPLEMENTATION

In this chapter, we will explain the way data collection was carried out and introducing the algorithms used to process the data. The results of these implementations as well as selected visualizations will be visible in chapter 5.

4.1 Methodology

As expressed in chapter 3.3, the data was acquired with a Stereolabs ZED camera. The environment used for the data capture was a study room at Tampere University. The university data was collected with faulty parameters and the results were too bad to use further. The ZED camera was recalibrated, and further data collection continued in my apartment. The image data was captured on a Windows 10 machine and processed on an Ubuntu 16.04 for maximum compatibility with InstanceFusion. The Stereolabs ZED camera used was provided by the Faculty of Engineering and Natural Sciences at Tampere University.

The data was captured using the MATLAB integration of the Stereolabs ZED API. This allowed for easy manipulation of the data but had problems, especially in data capture frequency. The data was captured and saved using a modified version of Stereolabs' example codes. The algorithm used for this purpose was a slightly modified version of an algorithm provided by Joni Tepsa, who made a bachelor's thesis on a similar subject by Joni Tepsa (Tepsa 2020). In the script, depth was defined to only have values between 0.15-6.5 m due to the size of the scanning environment and the predicted distance of the camera to any given surface.

The Stereolabs ZED camera is capable of capturing full-HD imagery and depth data, but InstanceFusion expected a capture size of 640 x 480 px. To minimize possible occlusion in the ZED camera, data was captured in 1080 x 720 px and then cropped from the middle to the required size. ElasticFusion, which is the framework on which InstanceFusion runs on assumes that the data was captured in exactly 30 frames per second (fps), but the MATLAB code could not reach this frequency. Most data was captured at a frequency under 25 fps. According to ElasticFusion documentation, this can cause fundamental errors in image segmentation. The ZED camera also had a lot of picture smearing due to the operator's fast movements and sometimes contained significant horizontal tilt, which typically caused segmentation to fail.

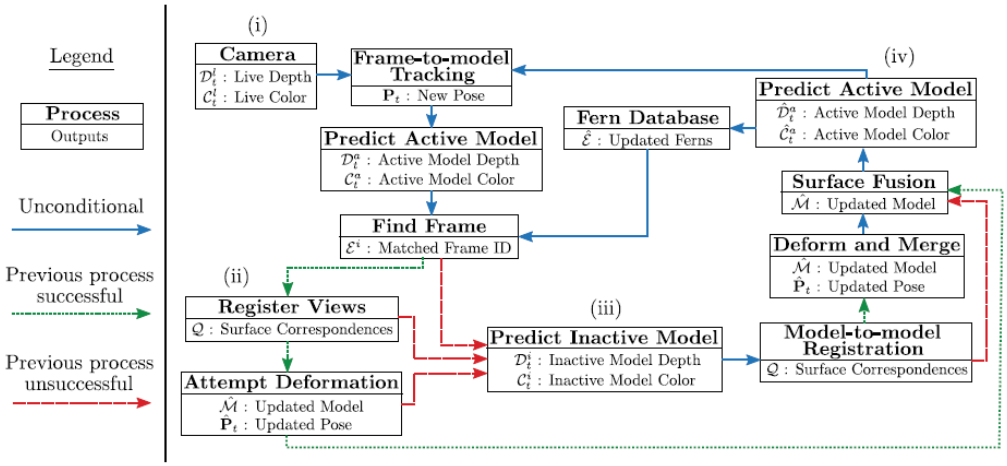
Also, a possible source of erroneous depth perception comes from data capture methods. The ZED camera stores depth-per-pixel as raw distances in float numbers depicting the depth in meters. The available dataset however contained data encoded in what was assumed to be unsigned 16-bit integers in an unknown spatial coordinate system. These distances were also assumed to represent meters in the power of 10^4 and the raw data captured by the ZED camera was converted to imitate this format. With this conversion, InstanceFusion produced models that seemed to have the correct depth dimensionality.

The computer on which InstanceFusion was run contained an NVIDIA RTX 2070 MAX-Q, an Intel Core i7-9750H @ 2.60GHz. This was deemed to be sufficient for the demanded minimum specifications found in the GitHub page of ElasticFusion (ElasticFusion 2021). NVIDIA recommends CUDA 10 or newer for RTX GPUs and therefore CUDA 10.1 was installed instead of CUDA 8, for which ElasticFusion and InstanceFusion were written to function. Initial tests with CUDA 8 were unsuccessful as the interface of InstanceFusion did not work.

4.2 ElasticFusion

ElasticFusion (Whelan et al. 2016) is a state-of-the-art room reconstruction algorithm developed in 2016 by DysonLabs. InstanceFusion uses a slightly modified version of this algorithm, but both need to be initialized in the same directory. Therefore, comparisons are easy to make and the data used by InstanceFusion is readily simulable. Elastic fusion is a SLAM algorithm that can also estimate all the light sources in a scene without any prior information. The algorithm is also capable of making these reconstructions real-time as a user-held RGB-D camera explores a scene. The model itself is represented with surfels that contain light information.

ElasticFusion works according to the following system architecture diagram (adapted from Whelan et al. (2016):



Picture 6. *ElasticFusion architecture diagram.*

Picture 6 gives an overview of the pipeline ElasticFusion takes when running. Step (i) represents the injection of raw sensor RGB-D data into the framework. This imagery is then aligned with the previous view, for which reason the system is required to run constantly at 30 fps. From the new pose, a predicted view render is considered for the active model. This render is compared with the fern database, which stores all previous views. If a matching view is discovered, the model is updated (deformed) correspondingly (ii). Otherwise, the system creates a new inactive area to the database (iii), and the model is deformed to include this new view. In step (iv), the live camera data and the latest updated model are combined, and a new prediction of the active model is rendered for tracking the next frame.

The specifics of the function of these steps is not the focus of this thesis further examination is left out. Elaboration on the function of pose estimation, tracking, the deformation graph, and loop closure algorithms can be found in the research paper (Whelan et al. 2016).

Light source estimation was also a research goal for ElasticFusion. It is a separate function of the algorithm that is not necessarily utilized in InstanceFusion. Light sources and surfels are generally explained in chapter 2.1.

According to Whelan et al. (2016), the light estimation algorithm detects reflections off of individual surfels and decides by Hough-voting in which direction a given light source is. The information of the geometry is integrated into the scene and portions of the model where these predictions intersect are removed as predicted light sources. All hypothesized light sources are combined and retrieved as a set of, which can be used to create interactive AR models, for example. This system runs parallel to the estimation of a fused surfel-based environment, which functions as described in Picture 6.

Hough-voting, mentioned in step 3, is a popular computer vision technique for detecting lines in 2D images. Hough voting can be utilized to detect shapes such as circles and ellipses from these images. The key idea is to perform voting of the image features, which are collected into an array. The dimensionality of this array equals the number of unknown parameters of the considered shape. This system can be extended for 3D point-cloud data to detect spheres and planes (Tombari & Stefano, 2012).

The light source detection pipeline can be run in real-time at camera framerate while a user is exploring a scene. The system is also not limited to estimating a single light source but can detect multiple sources in a single environment. Some InstanceFusion models manufactured in this thesis have apparent removals of light-reflection surfaces, as will be seen for example in the hole in the wall of Picture 25.

4.3 Mask R-CNN

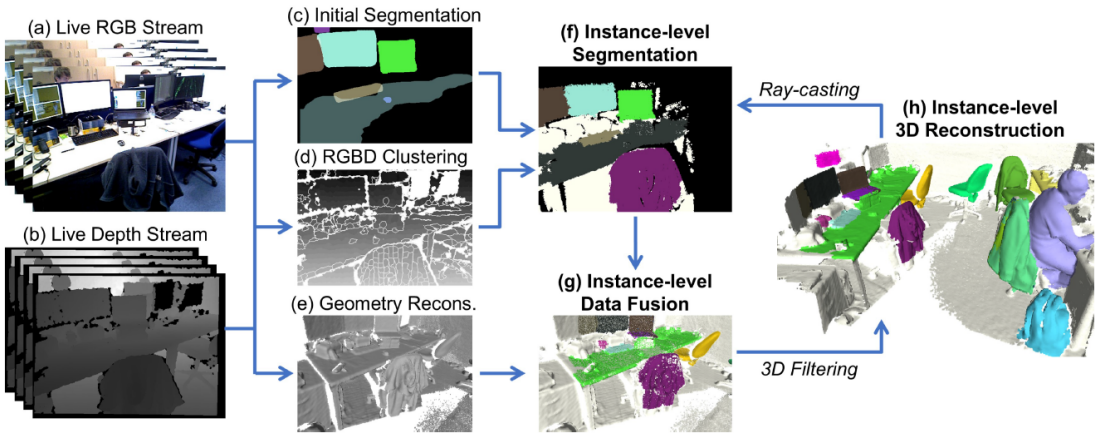
Mask R-CNN (He et al. 2017) is a convolutional network for object instance segmentation. It creates a high-quality segmentation mask for each instance or object. The implementation that is included with the InstanceFusion GitHub project contains the pre-trained MS COCO weights for specific object recognition in scenes. InstanceFusion ties this with the scene point-cloud data from ElasticFusion and provides a colored semantic map of the scene. The version used by InstanceFusion is the Python3 – Keras – TensorFlow implementation, also available on GitHub (Mask R-CNN 2021).

The You Only Look Once (YOLO) (Rademon et al. 2016) and its subsequent versions are a different and comparable masking algorithm. As of YOLOv3, which is the third version of the YOLO algorithm, it has been tested to outperform Mask R-CNN in some tests, such as in the paper by Prasetvo et al. (2020). In some tests, however, the predecessor of Mask R-CNN, Faster R-CNN has outperformed the older YOLOv2, such as in the paper by Scheider et al. (2018). Stereolabs' documentation contains instructions for the direct use of the YOLO v3 and v4 algorithms on the ZED cameras.

Mask R-CNN can produce bounded ground-truth boxes around recognized objects, but the point-cloud models produced by InstanceFusion did not have them. The GUI provided by the InstanceFusion project did also not draw these, so the classes of the recognized objects in the results chapter are pure postulation. Further parametrizing of the InstanceFusion project could provide scenes with these boxes. And subsequent class names.

4.4 InstanceFusion

InstanceFusion is a real-time algorithm used for instance-level 3D reconstruction with a single RGBD camera. It is a robust system for reconstructing 3D objects without any preceding knowledge of the scene or previously defined template models. InstanceFusion uses ElasticFusion and Mask-RCNN as dependency packages, but they have both been slightly modified in their implementation. InstanceFusion (and ElasticFusion) is designed to function in real-time and can do this at 20.5 Hz on consumer-grade hardware (Lu et al. 2020). A considerable source of difficulty in using InstanceFusion in real-time is that the camera used is required to be recognized by OpenNI2, which the ZED camera by default is not. As expressed in the introduction chapter, using InstanceFusion in real-time falls outside the scope of this thesis. It is notable, that Stereolabs provides an integration for OpenNI2 as well in their internet guides. OpenNI2 is an open-source software that aims to improve the interoperability of natural interface (NI) devices, such as Microsoft Kinect.



Picture 7. *The pipeline of InstanceFusion (Lu et al. 2020).*

Picture 7 presents the pipeline of InstanceFusion. The input streams (a, b) are the raw data from the RGB-D camera. Parts (c, d, f) display a two-stage segmentation algorithm, which is used for segmentation analysis or object detection. Parts (e, g) depict the surfel fusion with the detected classes. Part (h) depicts the position of the GPU-accelerated 3D filtering method. Finally, these models are rendered to guide the 3D reconstruction in the next frame, much as in the system architecture description of ElasticFusion displayed in Picture 6.

InstanceFusion has a distribution publicly available on GitHub, which is used in this thesis for the reconstruction as-is only with required changes to pathnames. The main goal for implementing the algorithm in this thesis comes from data collection and getting good results out of the algorithms.

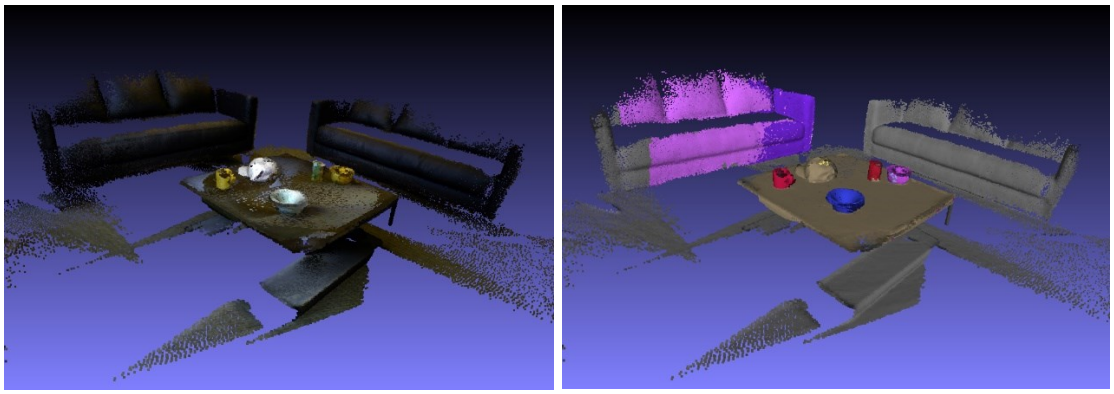
5. RESULTS AND ANALYSIS

In this chapter, we will be presenting the attained results according to the methodology in chapter 4.1. We will be presenting faulty models, the best model attained and comparing them to an exemplary dataset. Most results will be presented as screen captures from a point-cloud data viewing software called Meshlabs. All images presented in this chapter and its subchapters were manufactured alongside the thesis unless explicitly stated otherwise.

5.1 Example scenes

Example scenes are good for comparison here, as they have been used to quantify the success of the algorithm in the design phase. With example scenes, we can also verify the correct function of the algorithm. The dataset used here is the RGB-D Scenes Dataset v2 (Lai et al. 2014). This dataset was suggested on the GitHub page of InstanceFusion, and they provide an algorithm for organizing the data. This dataset was especially applicable in this thesis due to the simple format of the data: the RGB and depth pictures were stored as '.png' files. The algorithm to read these scenes with InstanceFusion sorted these and divided the values of the depth data by 10.

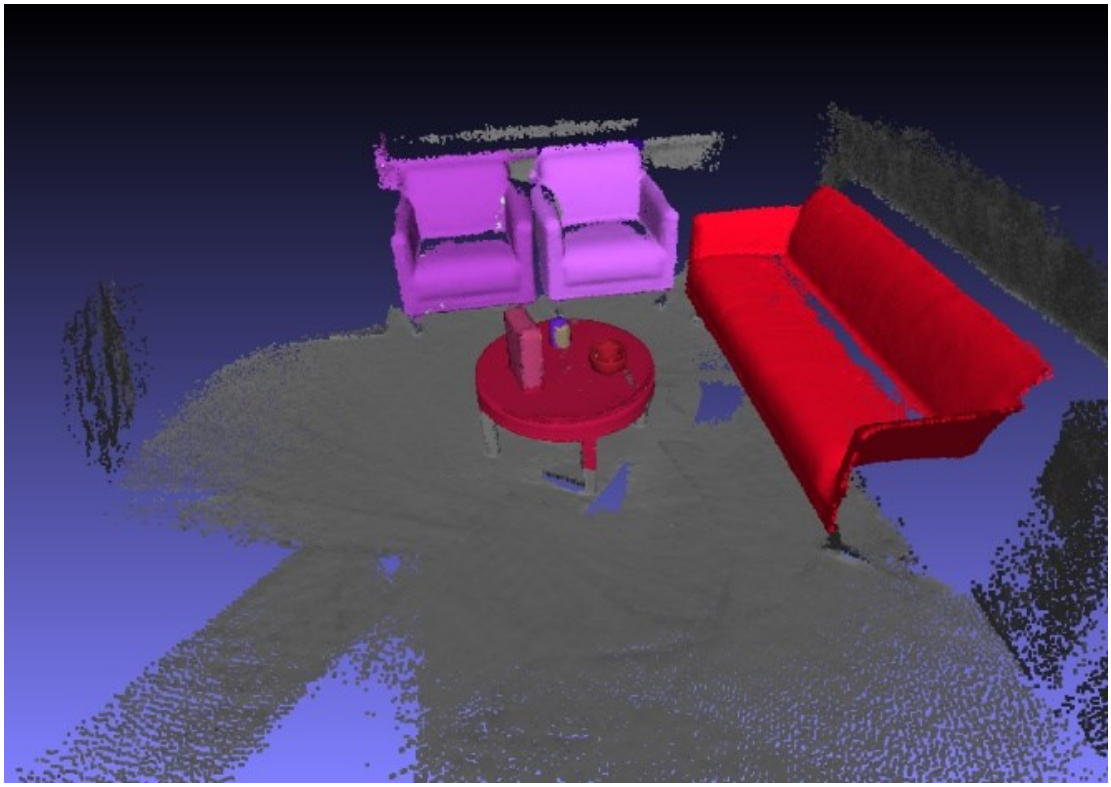
From this dataset, we will be using scene_01 and scene_09. Both of these sets contain a table with objects on top and move around them in a rotational loop. The camera in these sets also doesn't tilt horizontally which was a source of major segmentation faults in conducted data collection.



Picture 8. *Scene 01 colored point cloud (ElasticFusion) and a corresponding semantic model (InstanceFusion)*

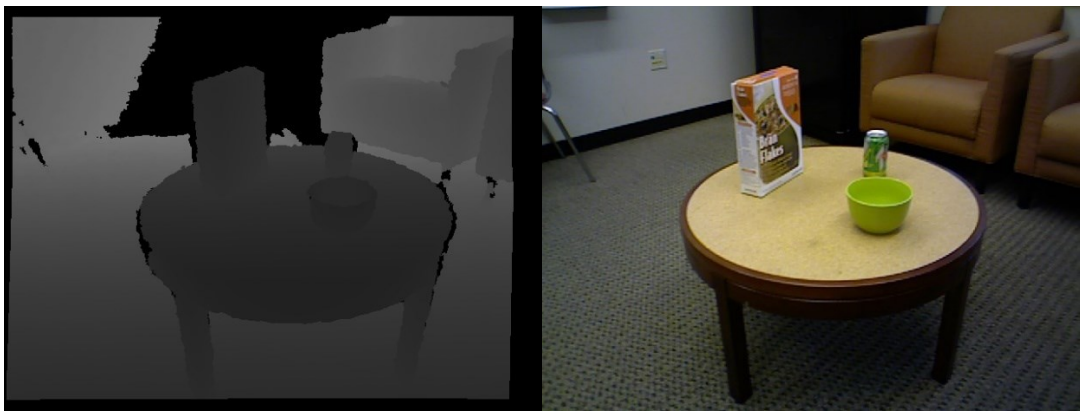
Picture 8 presents the result models from the first dataset available in the RGB-D Scenes Dataset v2. In this model, the semantic analysis confuses the couch on the right side as

two seats but recognizes the second couch correctly. The table is also detected and colored with a light brown scheme. On the table, there are two mugs (red and pink), a beverage can (red), a bowl (blue), and a cap (yellow).



Picture 9. *Scene_09 semantic model (the result of InstanceFusion)*

Picture 9 presents the resulting model from Scene_09 of the RGB-D Scenes Dataset v2. This model is also presented as an angled top-down png example in the InstanceFusion project files and can be used to confirm the correct function of the algorithm. In this dataset, all seats contain sharp angles and are discernible from the point-cloud data. The beverage can is gray on the table from the presented angle but is recognized during the scan process from a different direction. The colormap that is applied on it does not contain data from the undetected side presented here.



Picture 10. *Example of the depth data (left) fed to InstanceFusion and a corresponding RGB picture (right) (scene_09, frame 36)*

Picture 10 presents an example of the depth-data used by InstanceFusion. In testing, it was discovered that the ZED scans do not always replicate the clear edges visible on the table legs and seat corners. Additionally, the gradual whitening of the floor was not typically visible in the ZED scans, possibly due to difficult textures apparent in the scanning environment.

5.2 RGBD data

In this chapter, we take a look at different views of the depth data used in creating the InstanceFusion semantic models. In this chapter, some observations will also be presented on detected or potential inaccuracies. The data used for the final model was captured with MATLAB and all views from the MATLAB depth data are presented with the inverse of InstanceFusions scaling to provide visually pleasing images.

The scan was conducted by trying to present the algorithm views of the environment from different angles while moving about in the scanning environment. The final view of the scan was approximately the same as the starting view to make loop closure possible.

5.2.1 MATLAB views

Here we look at the RGB-D data used in manufacturing the final model presented in chapter 5.3. It is noteworthy that this particular scan is the result of a relatively quick look at all angles of the scan area. The scan consisted of only 1869 frames captured at a frequency of 26 Hz which could be the reason for much of the apparent segmentation faults as expressed in chapter 4.1. The low frame count was therefore selected to make segmentation errors on long scans from the same angle as scarce as possible. Better results may have been achieved with a longer scan and a higher framerate.

Picture 11. *Frame 38 Depth (left) and RGB (right) from the final dataset presented in this thesis*

Picture 11 presents the approximate start position of the final map which is referred to as ‘table’ in this thesis. To test loop closure methods, this also corresponds closely to the end frame of the scan. In the final model, we will see that loop closure fails due to the bright reflection on the wall even though the similarity of the end is high though mis-aligned. Note from the depth image that the thin legs on the lower-left corner are not detected by the camera and that most edges are not clearly defined.

This view does not completely correspond to the raw depth sensed by the ZED camera, but we will see in ensuing subchapters that the unparameterized depth sensing of the ZED camera doesn’t correspond to the sample datasets closely.

Picture 12. *Frame 647 Depth (left) and RGB (right) data*

Picture 12 presents the view of the ‘kitchen’ area of the used dataset. Note that flakiness is decreased compared to ensuing subchapters’ raw depth views due to the applied cropping proposed in chapter 4.1. This view is one of the most geometrically complex in the set but also resulted in one of the better semantic views on the final model.

Picture 13. *Frame 1039 Depth (left) and RGB (right) data*

Picture 13 shows a view of the 'middle' portion of the scanning environment. Note how the arm holder of the couch is almost indiscernible from the depth model and that its edge is detected in the wrong location. This is seen as null data on the bottom middle. This causes a segmentation error in the final model, but the couch is still detected as only one object.

5.2.2 OpenNI2

Due to the dissimilarity of the dataset depth data and the ZED depth data, we will examine other implementations to see if depth sensing was parametrized in a faulty way. For OpenNI2 NiViewer.exe was used with the necessary Stereolabs integration. Note that these views closely correspond to the MATLAB dataset but are flakier due to the higher resolution present in both the RGB and Depth data.

Picture 14. *OpenNI2 view of 'table' with the depth on left and RGB on right*

Picture 14 shows a comparable view of the 'table' area on the dataset. Note that in this particular image there is far less light available and depth detection is worsened as a result.

Picture 15. *OpenNI2 overview of the scan area*

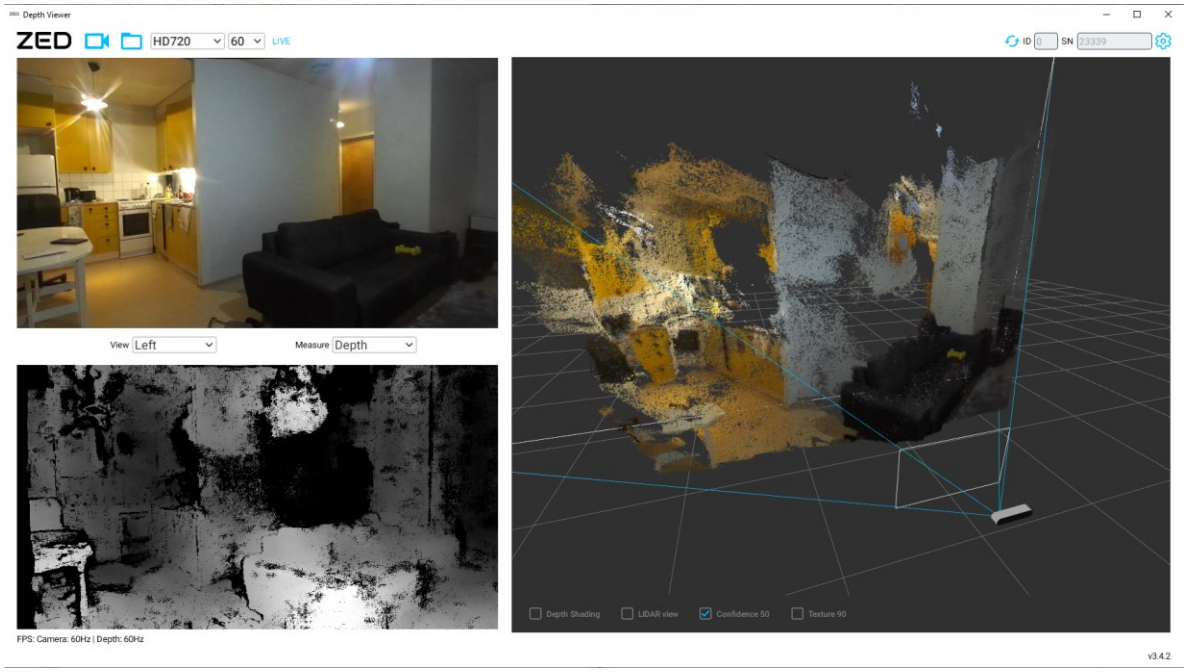
Picture 15 shows a comparable view of the ‘middle’ area of the scanning environment. Note that this view is not necessarily any better than the MATLAB version as the geometry of the couch is still largely indiscernible. Note also how the wall textures of the environment are difficult to sense for the ZED camera, particularly in the areas behind the couch.

Picture 16. *OpenNI2 view of ‘kitchen’*

Picture 16 shows a comparable view of the ‘kitchen’ area of the scanning environment. Note that in the used scans, the camera did not take images from a static position but wandered around the scanning environment, so views were closer. Comparing the RGB portions of Picture 15 and Picture 16, we can see the heavy distortion of the couch arm holder, which was the prime reason behind cropping the image. Note, that in theory distortion should not matter to RGB-D point-cloud models, as long as all pixels are correctly calculated to correct positions.

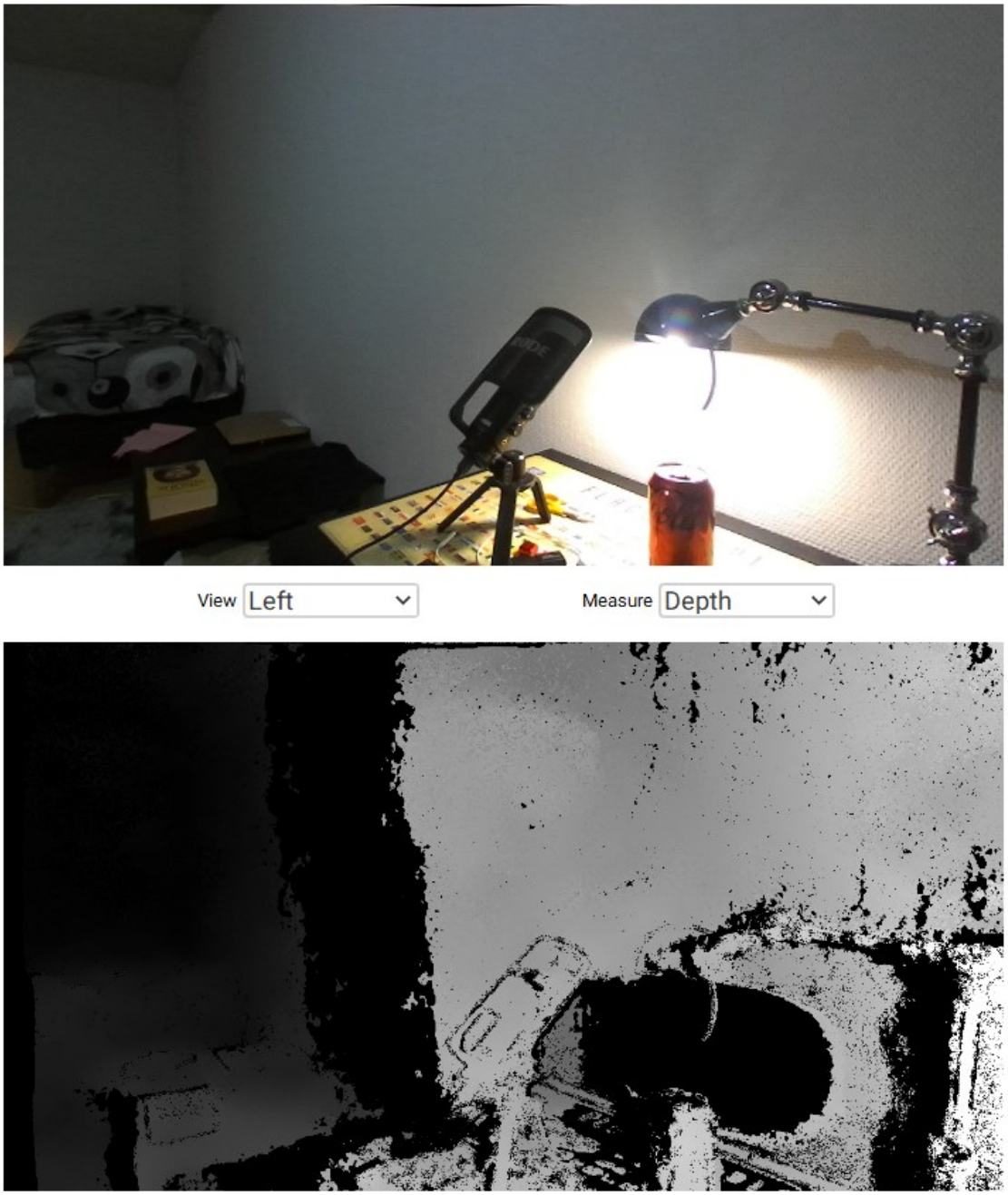
5.2.3 ZED Depth Viewer

The ZED Depth Viewer was used as-is from the ZED SDK. These views represent the best results achieved in this thesis, as we can assume all parameters of the camera have been selected optimally by its designers. The ZED depth viewer is also capable of storing RGB-D data, but only as ‘.svo’ files which were unusable by InstanceFusion, and extracting this file type or modifying InstanceFusion falls outside the scope of this thesis.



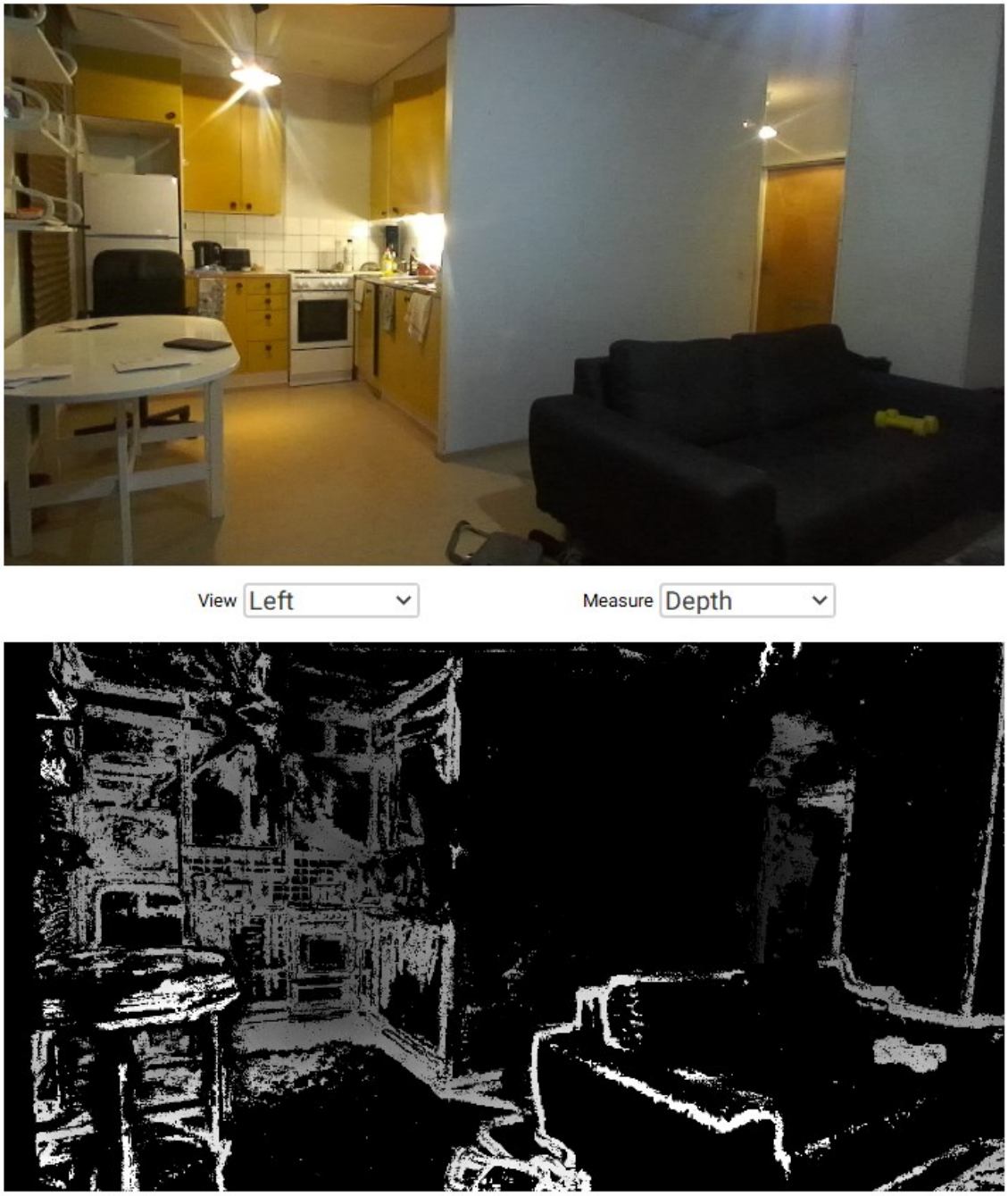
Picture 17. *ZED Depth Viewer view of 'kitchen'*

In the bottom left corner of Picture 17, we can see the assumed optimal depth that a ZED camera can present. The large window on the right presents a freely movable point-cloud representation of the data, and this data was never distorted in testing. In this particular view, we can however see that the ZED camera is attempting to connect the wall texture behind the couch to the back of the couch, which can explain the difficulty all systems faced when detecting this surface. It is also notable that the depth-sensing capability of the ZED camera was deemed to be dependent on the capture framerate in testing. This view is an excerpt of the camera functioning at 60 fps, but the final model was captured at around 26 fps.



Picture 18. *Wall bending depth-smear caused by the light source and a microphone*

Picture 18 presents the two leftmost windows visible in Picture 17. Here we see an inherent flaw of the ZED camera in geometrically and texturally misleading environments, as the wall is detected to continue immediately behind the microphone with a sudden drop to the back. This wall bend was intentionally left in the final model and is visible from all directions in the final scan.



Picture 19. *ZED Depth Viewer RGB and depth view of 'kitchen' with texture confidence set to 90.*

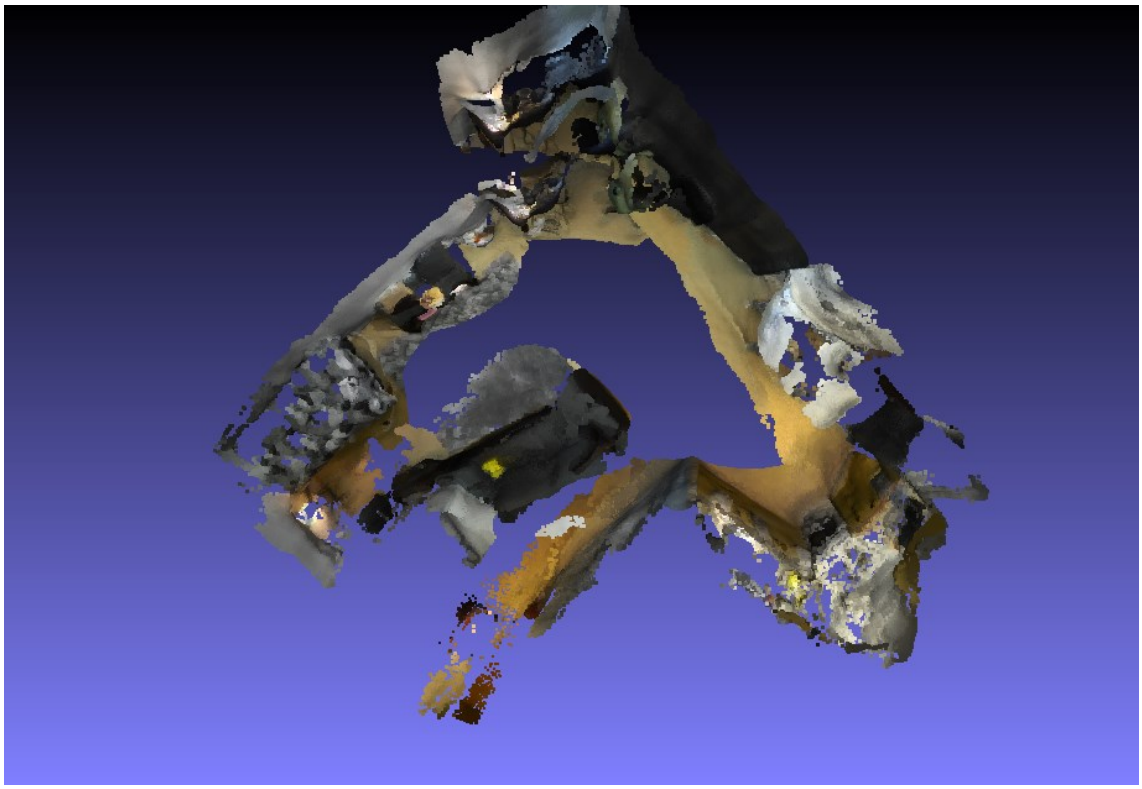
Picture 19 presents the ZED Depth Viewer in a different depth perception storage mode. The setting changed alters texture confidence and partially fixes the edge detection problems. This view however is not compatible with InstanceFusion, which expects per-pixel depth data. Note from this picture how the wall is no longer mistakenly detected as a part of the sofa. Different views with this mode detected corners of areas particularly well.

Picture 20. *ZED Depth Viewer view of the kitchen area with texture fill enabled*

Picture 20 presents the best results achieved with the ZED camera in this thesis. It was created utilizing the ZED camera's inherent texture prediction, but models created by capturing data with this setting proved even worse than models captured without it. This view is however presented here, as from this image it is very easy to make out certain geometry, such as the couch and the chair behind. It is also notable, that the environment has fewer roof-mounted light sources in the instance in which this image was captured.

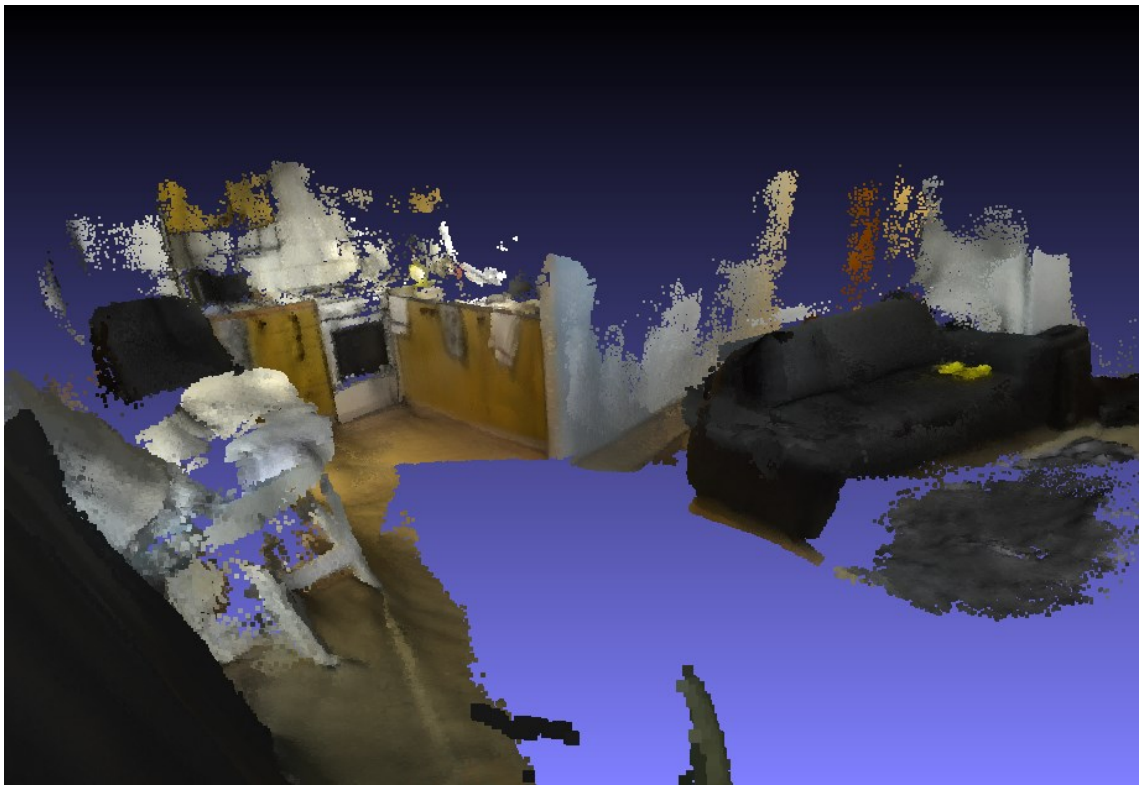
5.3 InstanceFusion results with ZED data

In this subchapter, we will be presenting both the ElasticFusion and the InstanceFusion models of the final dataset. Note that these datasets do not necessarily represent the function of the algorithm at its best and most errors might be tied to the incorrect parametrizing of the ZED camera on the MATLAB integration.



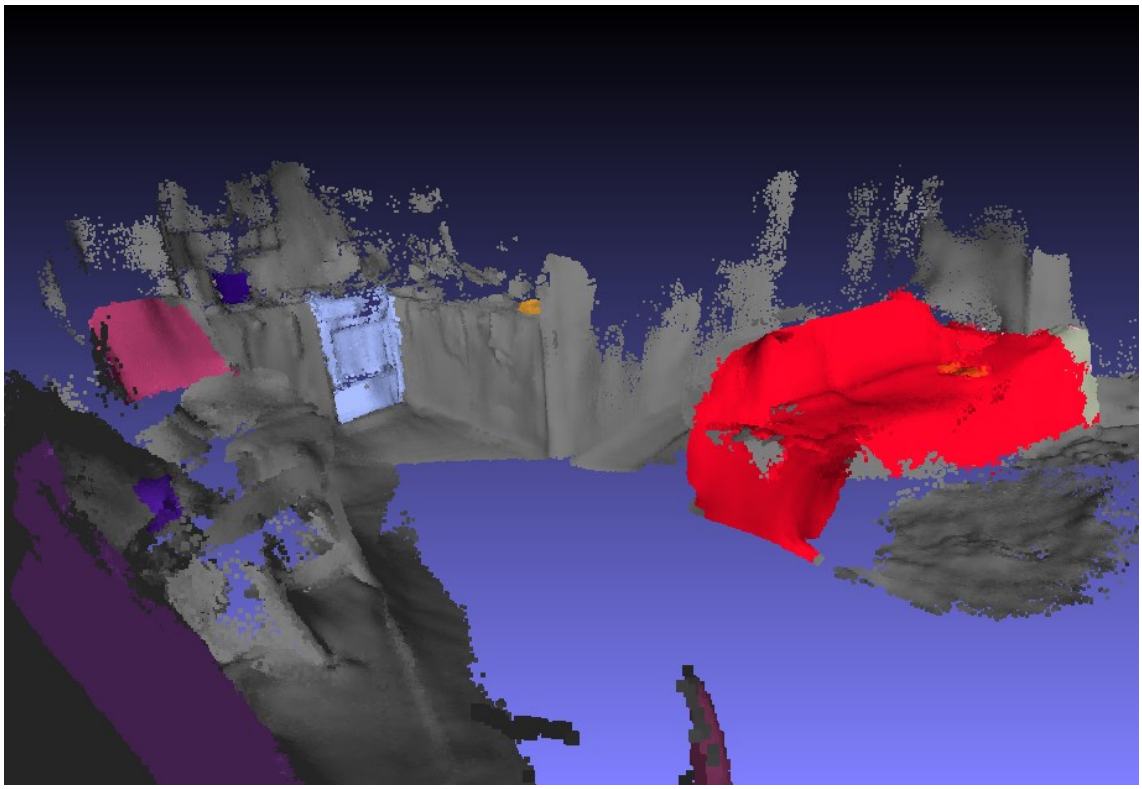
Picture 21. *Overview and shape of the final dataset model (ElasticFusion)*

Picture 21 presents the final shape of the dataset. Note that the general shape of the room is distorted and lacks definite corners. This is especially noticeable with the couch that is present in the middle of the model, as it is in reality parallel to the wall behind it. This model represents the best result of 15 test runs with slightly altering parameters, scan locations, and dataset sizes. Note also that the beginning and end of the dataset are in the top corner of the room, and that the loop closure fails or is not implemented.



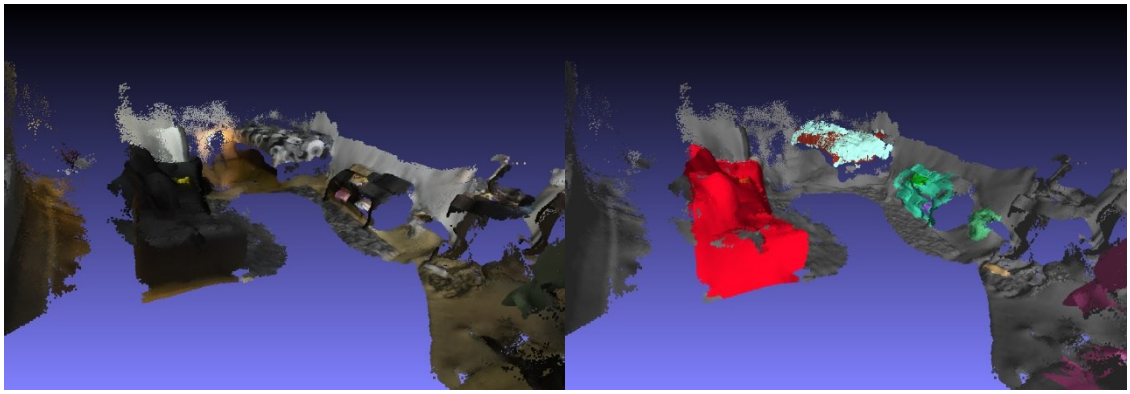
Picture 22. *ElasticFusion model view of the 'middle' and 'kitchen' areas*

Picture 22 presents the ElasticFusion view of the 'middle' and 'kitchen' areas of the model. Note how certain areas around the 'kitchen' are very well defined and how the arm holder of the couch in the 'middle' area is distorted. One proponent in improving this model could be the inclusion of the floor areas in as many shots as possible since floors almost universally were mapped correctly if the segmentation did not fail. In the scanning environment, there is a grey carpet around the 'middle' area of this dataset, which was however difficult to correctly map for the algorithm in other datasets.



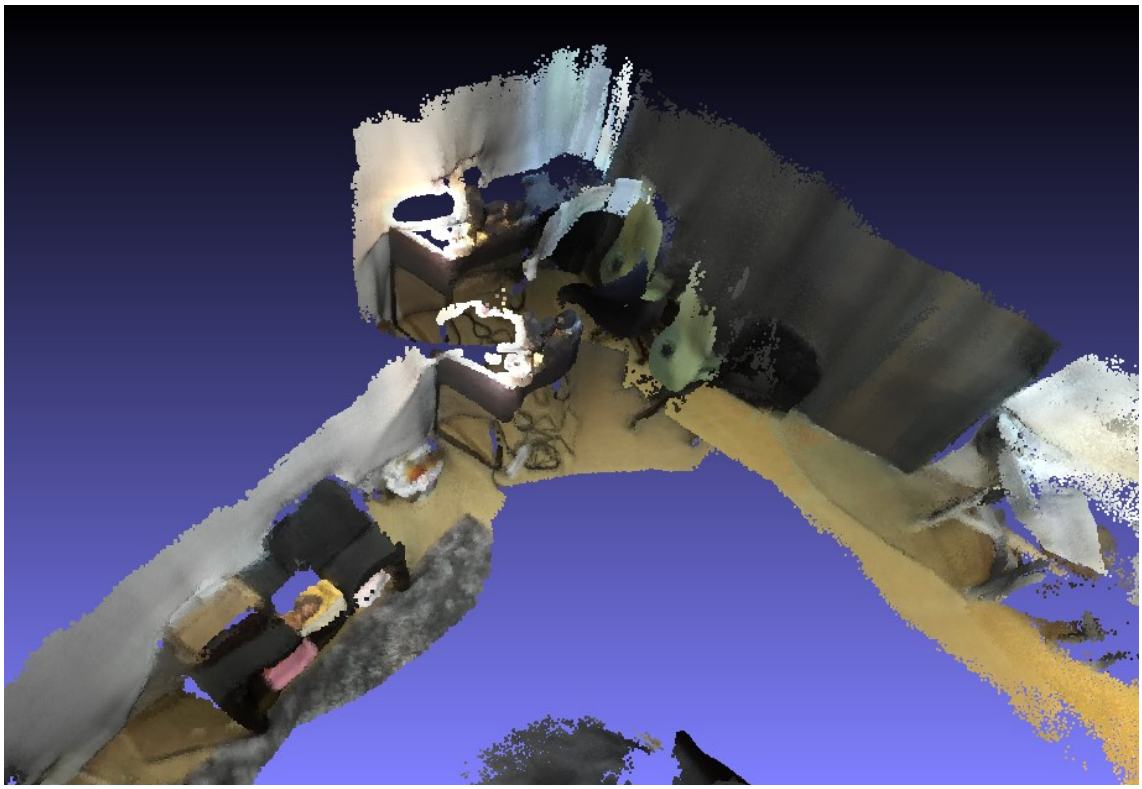
Picture 23. *InstanceFusion model view of the 'middle' and 'kitchen' areas*

Picture 23 presents the comparable InstanceFusion model seen in Picture 22. The views are presented from the same angle. Note how the chair (pink) is detected even though only a small portion of it is visible in the dataset. Also, the couch is mapped as only one object even though its segmentation is faulty around the arm holder.



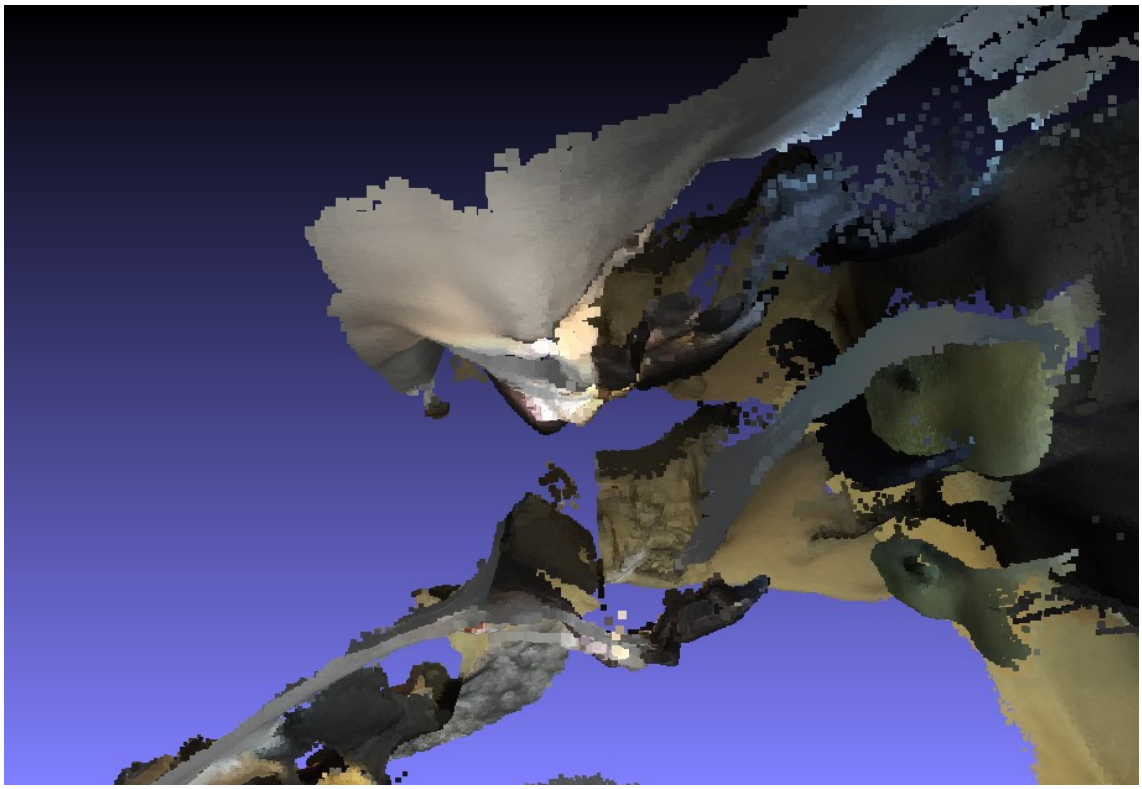
Picture 24. *ElasticFusion (left) and InstanceFusion (right) model views of the 'middle' area with the far corner of the scan area*

Picture 24 presents a previously unseen corner of the final model, where there is a bed in the furthest corner and a table in the middle right. This bed has two colors which means it is detected twice as a different object, much like with the couch in the example model depicted in Picture 8. In the closing point of the model, we can also see the green chair detected as an object, but it is distorted.



Picture 25. *ElasticFusion model view of the loop closure location*

Picture 25 presents the end and start point of the dataset. Note that the rightmost chair belongs to the endpoint and the leftmost point to the start point. In addition to being vertically misaligned, these two points are also horizontally in slightly different positions. This could also be rectified with the inclusion of additional floor data. Note also that the brightest reflection point of the light source on the wall is left out of the depth data due to the light source removal function of ElasticFusion.



Picture 26. *ElasticFusion model view of the wall bend distortion*

Picture 26 presents a view of the 3d wall bending effect of the microphone and light source as discussed in chapter 5.2.3 and displayed in Picture 18. Note however that this bend is universal from all scan directions and is present in both the endpoint and the start point. Better loop closure results could be achieved by using geometrically less complex closing points.

5.4 Result analysis

Based on the results displayed in chapter 5.2, it is clear that the implementation used in this thesis still underperforms when compared to examples, such as in Lu et al. (2020). The most significant reasons for this were identified to be the incorrect parameterization of the used ZED camera, incapability to reach the required 30 fps capture frequency, and possible depth data unit conversion errors. Further evidence of the incorrect use of the ZED camera and possible data inaccuracies are displayed in Picture 17 and Picture 18, where the depth perception of the ZED camera seems to be confused by the monotonous white wall texture of the scanning environment. An example of the aggregation of this error is seen in Picture 26 where the final model contains the geometric error from all directions.

Other methods of data capture were examined, but chosen excerpts, such as Picture 15, display that data capture was problematic regardless of the software used. It is however

probable, that using OpenNI2 for collecting data instead of MATLAB would solve all possible depth data unit conversion errors. Loop closure, which is essential for room reconstruction was also not achieved. It is possible, that loop closure is a setting in InstanceFusion and needs to be separately enabled for closed models. It is also possible, that the data had too significant discrepancies due to the inaccuracy aggregations, as seen in Picture 26. InstanceFusion was deemed to be functioning correctly otherwise by making comparisons to example datasets.

Originally the reconstruction environment was intended to be of an empty room in the Hervanta campus of Tampere University, but result models from capture environments were too heavily occluded for good results. In one instance a wall was observed to make a smooth, almost 90° turn, to merge with the roof. This was one of the leading factors for the implementation of cropping in imagery.

Instead of the empty room at Tampere University in this thesis, the reconstruction environment was the author's apartment. This environment contains geometrically complicated surfaces, which might have been detrimental to the function of InstanceFusion or ElasticFusion. In research material, several examples of successful reconstruction in comparable environments were observed regardless, which means that most errors might be associated with the operator of the camera in this thesis. It is notable, that initial scans of the test area were unsuccessful partly due to having open windows in the real scene. Covering these windows with cloths improved reconstruction results significantly, especially regarding the room shape.

Object detection in scenes was partly successful, as most objects displayed are recognized to be one object. In the reconstruction scene, the only exception to this is the bed seen in Picture 24, in which said object is colored with a blueish and a reddish color. Multiple instances of single objects were typical in other scans when segmentation failed. The couch seen in Picture 23 was however recognized as a single object despite its erroneous shape.

Some scans were done as double loops as in displaying similar views a second time, but these scans resulted in unusable models with multiple floors and multiples of objects. This is possibly due to the unachieved loop closure, as areas are predicted to be inactive by InstanceFusion when they don't closely match previously stored views.

6. SUMMARY

The first research goal for this thesis was to create an overview of room reconstruction methods and utilize a modern reconstruction algorithm on a dataset. Results for the implementation were discussed, and the results were deemed poor in comparison to other sources. A secondary goal was to determine the usability of the ZED camera with the chosen algorithm, InstanceFusion.

6.1 Methodology and implementation

Semantic room reconstructions were achieved by utilizing InstanceFusion with a single handheld ZED camera wandering in a test environment. The ZED camera was successfully used with the MATLAB integration of the ZED API to capture data. This data was fed to InstanceFusion as a dataset converted to a suitable format. Results attained were poor, and possible reasons for this were assumed to originate from the incorrect parameterization of the camera, the incapability to reach the required 30 fps image capture frequency, and possible depth data conversion errors. To minimize errors, such methods as cropping, resolution switching, and various depth parameters were tested to minimize errors. Occlusion was effectively removed like this and models were improved. Further parameterization with more intimate knowledge of the MATLAB integration of the ZED API could yield better results.

6.2 Results and future works

A future study could be conducted on using the InstanceFusion interface in real-time as the foremost data-capture method with a ZED camera. This could only require the installation of suitable OpenNI2 drivers so that device recognition is automized but would probably not result in better results without further parametrization. Other potential future subjects include using the texture filtering or depth data filling capabilities of the ZED camera in similar reconstruction environments. Future considerations for improving the reconstructions with InstanceFusion also include comparison with other RGB-D capable cameras, such as a Microsoft Kinect. Future research could consider creating custom runtimes for InstanceFusion, as this thesis only used the default runtime provided with the InstanceFusion GitHub project.

BIBLIOGRAPHY

[1] Babahajiani, P., Fan, L., Kämäräinen, J. & Gabbouj, M. 2017, Urban 3D segmentation and modelling from street view images and LiDAR point clouds.

[2] Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A. & Silva, C.T. 2017, "A Survey of Surface Reconstruction from Point Clouds", Computer Graphics Forum, vol. 36, no. 1, pp. 301-329.

[3] Blaha, M., Vogel, C., Richard, A., Wegner, J.D., Pock, T. & Schindler, K. 2016, Large-Scale Semantic 3D Reconstruction: An Adaptive Multi-resolution Model for Multi-class Volumetric Labeling, IEEE.

[4] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. & Leonard, J.J. 2016, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age", IEEE transactions on robotics, vol. 32, no. 6, pp. 1309-1332.

[5] Corke, P. 2017, Robotics, Vision and Control Fundamental Algorithms In MATLAB® Second, Completely Revised, Extended And Updated Edition, 2nd edn, Springer International Publishing, Cham.

[6] Eigen, D. & Fergus, R. 2015, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture, IEEE.

[7] ElasticFusion (2021) GitHub page, Available (accessed 04/05/21): https://github.com/mp3guy/ElasticFusion

[8] Euclideon (2021), industry applications, Available (accessed 04/05/21): https://www.euclideon.com/industries/

[9] Evans, W. (2020) 'How Amazon hid its safety crisis', *Reveal*, Available (accessed 04/05/21): https://revealnews.org/article/how-amazon-hid-its-safety-crisis/

[10] Fairchild, M.D. 2013, Color appearance models, 3rd edn, Wiley, Chichester, West Sussex, U.K.

[11] Goodfellow, I., Bengio, Y., Courville, A. & Goodfellow, I. 2018, Deep learning : das umfassende Handbuch : Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze, 1st edn, mitp Verlags GmbH & Co, Frechen.

[12] Gorban, A.N., Mirkes, E.M. & Tyukin, I.Y. 2020, "How Deep Should be the Depth of Convolutional Neural Networks: a Backyard Dog Case Study", Cognitive computation, vol. 12, no. 2, pp. 388-397.

[13] Gordienko, P. 1993, Construction of efficient neural networks: Algorithms and tests, IEEE.

[14] He, K., Gkioxari, G., Dollar, P. & Girshick, R. 2020, "Mask R-CNN", IEEE Transactions on Pattern Analysis and Machine Intelligence; IEEE Trans Pattern Anal Mach Intell, vol. 42, no. 2, pp. 386-397.

[15] Lai, K., Bo, L. & Fox, D. 29 September 2014, "Unsupervised feature learning for 3D scene labeling", 2014 IEEE International Conference on Robotics and Automation (ICRA)IEEE, Hong Kong, China, 31 May-7 June 2014.

[16] Lu, F., Peng, H., Wu, H., Yang, J., Yang, X., Cao, R., Zhang, L., Yang, R. & Zhou, B. 2020, "InstanceFusion: Real-time Instance-level 3D Reconstruction Using a Single RGBD Camera", Computer Graphics Forum, vol. 39, no. 7, pp. 433-445.

[17] Mask R-CNN (2021), Mask R-CNN GitHub page, Available (accessed 04/05/21): https://github.com/matterport/Mask_RCNN

[18] McCormac, J., Handa, A., Davison, A. & Leutenegger, S. 2017, SemanticFusion: Dense 3D semantic mapping with convolutional neural networks, IEEE.

[19] Palla, A., Moloney, D. & Fanucci, L. October 2017, "Scene Reconstruction from a Single Depth Image Using 3D CNN ", International Conference on Computational Biology and BioinformaticsICCBB, , pp. 85.

[20] Pfister, H., Zwicker, M., van Baar, J. & Gross, M. 2000, Surfels: surface elements as rendering primitives, ACM Press/Addison-Wesley Publishing Co.

[21] Prasetyo, E., Suciati, N. & Fatichah, C. 2020, A Comparison of YOLO and Mask R-CNN for Segmenting Head and Tail of Fish, IEEE.

[22] Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. 2016, You Only Look Once: Unified, Real-Time Object Detection, IEEE.

[23] Schneider, S., Taylor, G.W. & Kremer, S. 2018, Deep Learning Object Detection Methods for Ecological Camera Trap Data, IEEE.

[24] Shaik, N., Lutz, M. & Schlegel, C. 2020, 2D Localization in Large Areas Using Inexpensive RGBD Camera Augmented With Visual Tags, IEEE.

[25] Stereolabs (2021a), Camera Overview, Stereolabs documentation, Available (Accessed 04/05/21), https://www.stereolabs.com/docs/video/

[26] Stereolabs (2021b), Depth Sensing Overview, Stereolabs documentation, Available (accessed 04/05/21), https://www.stereolabs.com/docs/depth-sensing/

[27] Stereolabs (2021c), Stereolabs ZED home page, Available (accessed 04/05/21): https://www.stereolabs.com/zed/

[28] Tepsa, J. 2020, AI-Based Object Recognition on RGBD Camera Images.

[29] Tombari, F. & Stefano, L.D. 2012, "Hough Voting for 3D Object Recognition under Occlusion and Clutter", IPSJ transactions on computer vision and applications; IPSJ Transactions on Computer Vision and Applications, vol. 4, pp. 20-29.

[30] Whelan, T., Salas-Moreno, R., Glocker, B., Davison, A.J. & Leutenegger, S. 2016, "ElasticFusion: Real-time dense SLAM and light source estimation", The International journal of robotics research, vol. 35, no. 14, pp. 1697-1716.