FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO


# 3D reconstruction in underwater environment using CAD model alignment with images

**Fábio André da Rocha Morais**

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Luís Filipe Teixeira, PhD

Co-Supervisor: Pedro Costa, MSc

July 29, 2021

# Resumo

O investimento em equipamentos de exploração, produção, armazenamento, transporte e distribuição na indústria petrolífera e do gás é elevado. Tipicamente, as operações de inspeção, manutenção e reparação (IMR) são realizadas por veículos operados à distância (ROVs) que são controlados por um piloto. Ao automatizar estes ROVs, o custo da operação pode ser significativamente reduzido e a sua qualidade de inspeção melhorada.

O objetivo da dissertação é fazer reconstrução 3D em unidades métricas de alguns elementos de interesse das estruturas subaquáticas. A abordagem para este objetivo foi segmentar vários elementos, tais como válvulas e *hot stabs*. Estes objetos segmentados irão fundir-se com modelos CAD 3D destes mesmos elementos. A segmentação e localização dos objetos pode ser útil para no futuro planear as ações necessárias para interagir com os equipamentos. Foi desenvolvido um modelo de segmentação e um outro modelo de estimação de profundidade, para ser utilizado num método de simultaneous localization and mapping (SLAM) que projeta em 3D e segmenta os objetos de interesse. Finalmente, foi criado e explorado uma variação do *Mask R-CNN*, que deteta, segmenta e prevê a pose dos objetos (rotação + translação), este método permite o alinhamento dos modelos CAD com a imagem.

O modelo 3D SLAM semântico alcançou resultados satisfatórios, apesar de não ter uma reconstrução 3D satisfatória. Para a estrutura, válvulas e *hot stabs* o modelo de segmentação obteve valores de *intersection over union (IoU)* de 0.88, 0.44 e 0.11, respetivamente. O método proposto para gerar mapas de profundidade a partir de modelos CAD melhorou a precisão do modelo de 0.45 para 0.64.

Uma vez que o modelo 3D SLAM semântico não conseguiu alcançar uma reconstrução 3D de alta qualidade, foi introduzida uma variação do *Mask R-CNN*, que prevê a pose dos modelos CAD para cada *frame*, que obteve um erro médio de distância para as válvulas de 0.5 metros e um erro médio de ângulo de 1.3º.

As nossas experiências demonstram que é possível melhorar o modelo de profundidade utilizando modelos CAD para gerar um mapa de profundidade mais denso. Além disso, quando os modelos CAD estão disponíveis, é preferível alinhá-los com a imagem em vez de prever a profundidade para ser utilizado nos métodos SLAM, principalmente em aplicações de robótica submarina que requerem objetos complexos e com fracas condições de visibilidade.

# Abstract

The equipment for exploration, production, storage, transportation, and distribution in the oil and gas industry are expensive. Typically, the operations of inspection, maintenance and repair (IMR) are carried out by remotely operated vehicles (ROVs) that are piloted remotely. By automating ROVs, the cost of operation can be significantly reduced while the inspection quality is improved.

This dissertation aims to develop a 3D reconstruction method on underwater structures, particularly in some objects of interest. This goal was approached by segmenting several elements, such as valves and hot stabs. These segmented objects will merge with 3D computer-aided design (CAD) models of these same elements. The segmentation and location of each object of interest may be helpful to, in the future, plan the necessary actions to interact with the given equipment. To achieve this, a segmentation and depth estimation model was developed to be used in a dense 3D semantic simultaneous localization and mapping (SLAM). Finally, a variation of Mask R-CNN was introduced to detect, segment and predict its relative position to the camera by adding a new predictor to return the pose of the object (rotation + translation); this method enables the alignment of CAD models with the image.

The proposed 3D semantic SLAM achieved satisfactory results, even though it does not have a satisfactory 3D reconstruction. For the structure, valves, and hot stabs, the segmentation model achieved intersection over union (IoU) values of 0.88, 0.44, and 0.11, respectively. The proposed framework for generating depth maps from CAD models improved the depth model, increasing the model accuracy from 0.45 to 0.64.

Since the 3D semantic SLAM was unable to achieve a high-quality 3D reconstruction, a variation of the Mask R-CNN was introduced, which predicted the pose of the CAD models for each frame with a distance mean error of 0.5 meters for the valves and an angle mean error of 1.3 degrees.

Our experiments demonstrate that it is possible to improve the depth model using CAD models to generate a more dense depth map. Additionally, when CAD models are available, it is preferable to align them with the image for underwater robotics applications that require complex objects and poor visibility conditions rather than predicting the depth to use in SLAM methods.

# Agradecimentos

Em primeiro lugar, gostaria de agradecer ao Professor Doutor Luis Teixeira e ao Pedro Costa pelo excelente apoio, orientação e por estarem sempre disponíveis para qualquer dúvida que tivesse.

Um grande obrigado à equipa *AIResearch* por me terem recebido tão bem, ao Filipe, ao Hugo e ao Pedro Rocha pelo acolhimento, companheirismo e pelos seus contributos basilares para o desenrolar deste trabalho.

Agradeço aos meus pais e irmã por todo o apoio que me deram ao longo destes anos, que fizeram de tudo para me darem o melhor.

Um enorme agradecimento à Rita, que esteve sempre do meu lado em todos os momentos mais importantes.

Um agradecimento especial ao Alberto, o meu eterno colega de grupo e ex-colega de casa.

Um enorme obrigado ao Marcelo, Armando e Xico por todos os momentos incríveis que passamos nestes 5 anos.

Aos meus restantes amigos que fizeram com que estes 5 anos fossem os melhores, por toda a entreajuda que houve ao longo do curso, ao Diogo, Ricardo, Samuel, Gonçalo, Bernardo, Rolando e ao Fernando.

Por fim, um agradecimento ao meu primo Flávio por toda a educação extra que me deu.

Fábio Morais

*"If you look at what you have in life, you'll always have more.*
*If you look at what you don't have in life, you'll never have enough."*

Oprah Winfrey

# Contents

# List of Figures

# List of Tables

# Abbreviations and Symbols

| | |
|---|---|
| ROV | Remotely Operated Vehicles |
| CAD | Computer-Aided Design |
| SLAM | Simultaneous Localization and Mapping |
| IoU | Intersection Over Union |
| SOTA | State-of-the-art |
| FCN | Fully Convolutional Network |
| CNN | Convolution Neural Network |
| R-CNN | Region Convolutional Network |
| ROI | Regions Of Interest |
| RPN | Region Proposal Network |
| MSRCR | Multi-Scale Retinex With Colour restoration |
| EKF | Extended Kalman Filter |
| IMR | Inspection, Maintenance and Repair |
| DoF | Degrees Of Freedom |
| SfM | Structure From Motion |

# Chapter 1

# Introduction

## 1.1 Context

The oil and gas industry equipment for exploration, production, storage, transportation, and distribution are expensive and essential assets. If they are not well maintained, their failures can have disastrous consequences for the natural world and human civilization [17]. As a result of these factors, the authors of [17] predict that the inspection robots market will continue to grow in the near future, as inspection robots are desirable for the industry by reducing human error and lowering the cost of inspections.

The most widely used inspection technologies can be roughly classified into four categories based on the type of sensor used, namely visual inspection, ultrasonic inspection, magnetic inspection, and eddy current inspection [17]. A central piece in most of the oil and gas fields is the subsea Xmas tree, as shown in Figure 1.1: a subsea structure that regulates the flow of pipes. Such structure is used for maintenance purposes.

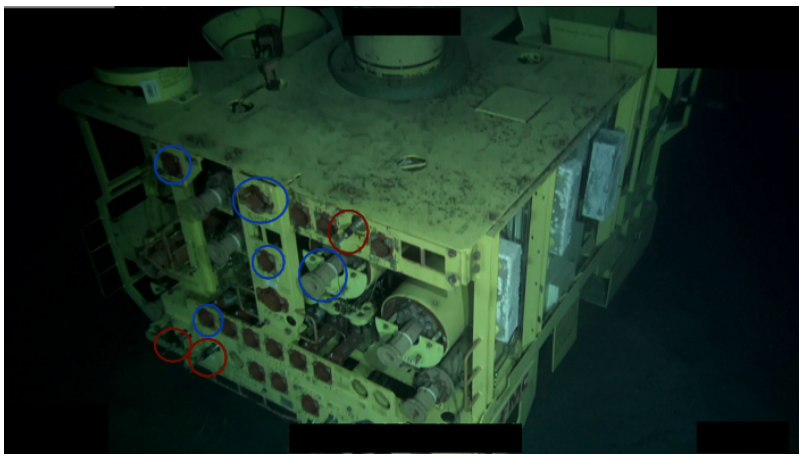


Figure 1.1: The Xmas tree that will be used. The blue circles represents valves, whereas the red circles represents hot stabs.

A subsea Xmas tree is made up of numerous valves that are used for testing, maintaining, regulating, or restricting the flow of produced oil, gas, and liquids coming up from the well below that are required to control hydrocarbon production. These complex assemblies of valves and other components are installed at the wellhead to monitor and control production flow and manage gas or fluids injection.

Subsea Xmas trees typically have a wide range of valves. These valves are typically configured with manual or actuated valves, either hydraulic or pneumatic. The configurations of subsea Xmas trees may vary depending on the needs of the projects and field developments.

Another important element is a subsea hydraulic connection device that transports hydraulic fluid from a topside hydraulic power unit to energize subsea equipment is known as a hot stab. It is essentially a hydraulic quick-acting connection developed for deep-sea use.

## 1.2 Motivation

Typically, remotely operated vehicles (ROV) perform inspection, maintenance and repair (IMR) operations under the control of a pilot aboard a ship. It would be interesting to control the ROVs on land to perform these operations safely and cheaply, dispensing hiring a boat and crew. By improving the automation of ROVs, the operation cost can drop considerably and improve the quality of the inspection [18]. The limitation of controlling ROVs from the ground is the communication latency, which is especially noticeable during operations requiring a high degree of precision, such as interacting with the structure. The goal is to automate these tasks that are currently impractical due to latency.

The project is lined up with Abyssal's projects, an independent privately-owned technology company that provides cutting edge 3D visualization, simulation, and digitalization capabilities for subsea operations.

The goal of Abyssal is to develop a system capable of making underwater vehicles semi-autonomous in tasks like inspection, maintenance and repair operations. As part of these operations, vehicles are common to perform high precision actions, such as turning valves. To develop a system capable of turning valves, we need a model of perception capable of identifying different objects of interest.

This dissertation project will develop a model for segmenting the valves and hot stabs. This segmentation will be merged with a 3D point cloud of the same objects, which will be created using CAD models. The segmentation and location of each interest element may be helpful to, in the future, plan the necessary actions to interact with the given equipment.

## 1.3 Problem Definition

As mentioned in Section 1.2 this work aims to merge 2D information from segmentation with 3D models of a Xmas tree, more precisely valves and hot stabs, using an unknown monocular RGB camera.

Most of the existing solutions for 3D reconstruction are limited to the indoor or outdoor environment; in underwater environments, the semantic segmentation, depth estimation and simultaneous localization and mapping (SLAM) are less advanced. Besides that, the state-of-the-art (SOTA) methods of 3D semantic segmentation use a stereo camera or an RGB-D sensor to accurately predict the depth map, as RGB-D cameras can get the pixel-level dense depth map of RGB image directly. However, RGB-D suffer from the limited measurement range, sunlight sensitivity and do not work underwater.

Finally, within the existing solutions, we have 3D reconstruction from multiple images that are based on feature matching. Although, for the underwater environment, extracting repeated features is challenging. The characterization of the environment is discussed in Section 2.1.

## 1.4 Objectives

The fundamental objective of this project is to implement a model that:

1. Segments a set of objects of interest in videos of underwater structures;
2. Fuses the segmentation with 3D models of these same objects;
3. Improves the 3D reconstruction by using CAD models;
4. Predicts the object pose and aligns CAD models into an image.

## 1.5 Document Structure

Chapter 1 focuses on the context, motivation and objectives for the proposed work.

Chapter 2 reviews the background and literature, starting by characterizing the environment in Section 2.1 and then Section 2.2 presenting the background about camera geometry. Section 2.3 presents Image Segmentation, the state-of-the-art architectures, as well as examples of Image Segmentation in underwater environments. Then, it proceeds to an introduction of depth estimation and the recent works on unsupervised depth estimation in Section 2.4. Section 2.5 introduces Simultaneous Localization and Mapping. Section 2.6 reviews 3D Reconstruction and 3D semantic mapping. Finally, CAD model alignment in RGB images was presented in Section 2.7.

Chapter 3 discusses the two models developed to perform 3D reconstruction and object recognition on each object. In Section 3.1 describes our approach to 3D semantic mapping, which combines depth estimation and semantic segmentation. In Section 3.2 our proposed approach for CAD model alignment with images are described.

Chapter 4 presents the various experiments and their results, as well as a discussion of those results and their implications for the system's limitations. Starting by describing the dataset in Section 4.1, the 3D semantic mapping are evaluated in Section 4.5 and the CAD model alignment with images in Section 4.6.3, the comparison between these two methods are presented in Section 4.7.

Finally, chapter 5 draws conclusions and future work.

# Chapter 2

# Background and Literature Review

## 2.1 Characterization of the Environment

Underwater images are usually more challenging due to the specific conditions, such as quality degradation due to absorption and scattering [19], thus causing low contrast, blurring and colour deviations. Moreover, organic particles such as sand, floating debris, dead or decomposed organisms, called "marine snow" affect the image quality [2]; examples of this degradation are shown in Figure 2.1.

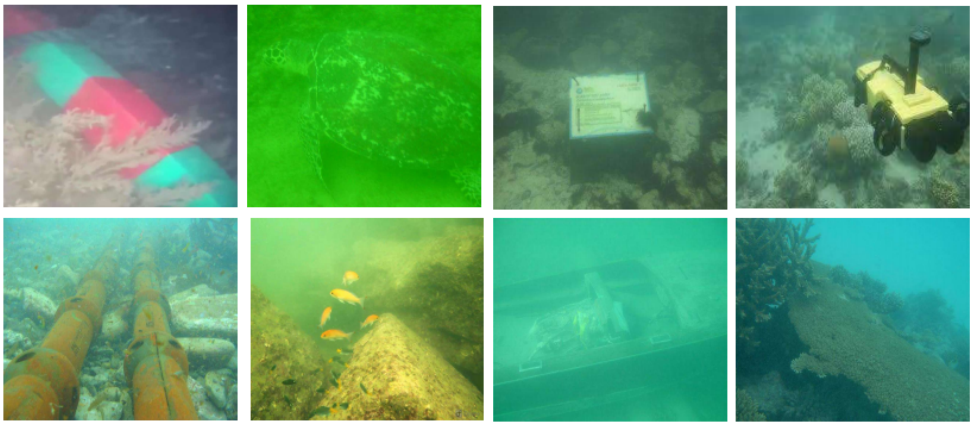


Figure 2.1: Examples of underwater image quality degradation (e.g., color casts, decreased contrast, and blurring details). Extracted from [1].

Figure 2.2 shows the *Jaffe-McGlamery* model; the light received by a camera is decomposed into three parts: the light reflected directly from the object; a forward scattering component that randomly deviates light to the camera; and a backscattering component that reflects the light in floating particles into the camera.

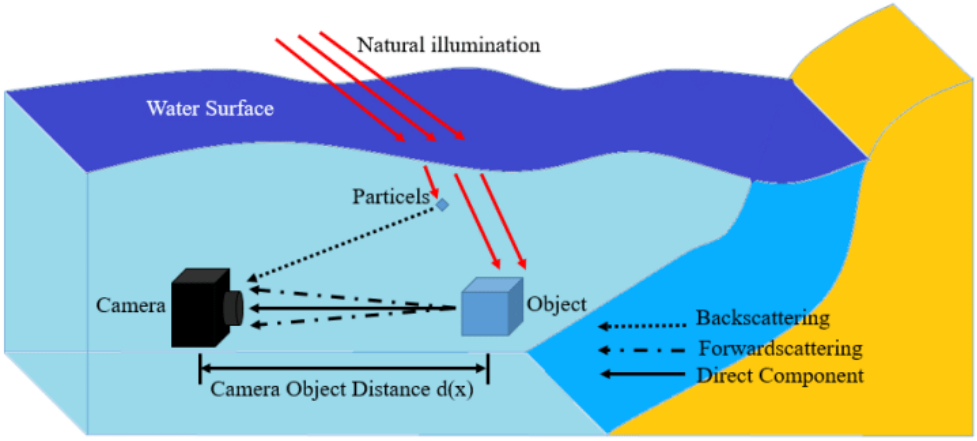



Figure 2.2: Under water model. Extracted from [2].

Recently Sea-thru [20] was presented to solve some of these issues. The authors propose to "remove the water" from images by estimating backscatter using the dark pixels and their known range information. The calculation of the spatial variability illuminant is then used to obtain a range-dependent attenuation coefficient, thus significantly improving the image's analysis.

## 2.2 Camera Geometry

In the scope of this dissertation, it is also necessary to specify what is meant by an object's or a camera's pose. Pose with 6 degrees of freedom (DoF) refers to an object's coordinates in 3D space, including translation and rotation around the x,y, and z axes, also known as pitch, roll, and yaw. The camera's extrinsic matrix describes the camera's location in the world and what direction it is pointing to.

### 2.2.1 3D Transformations

Different rotation representations can be used. As a result, the two most commonly employed representations, and thus of importance in this dissertation, are rotation matrices and Euler angles.

**Euler angles** explain orientation through a series of three rotations, each applied on the object's original axis and changing after each rotation (intrinsic rotations). Because any sequence of rotations is possible, the Euler angles representation is heavily reliant on conventions. As long as standards are followed appropriately, Euler angles are easy to use and do not suffer from over-parameterization.

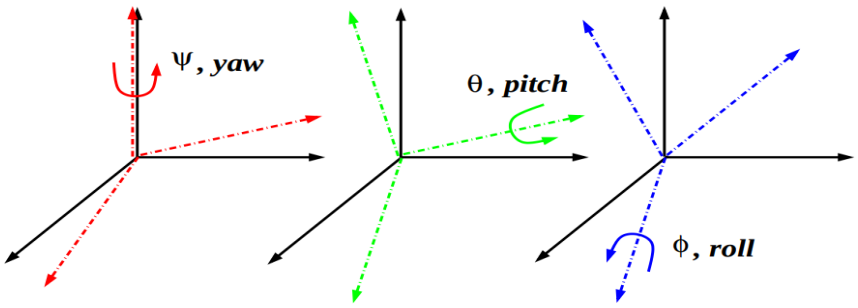



Figure 2.3: The yaw-pitch-roll convention for Euler angles [3].

**Translation** - 3D translations can be written as

$$x' = \begin{bmatrix} I & | & t \end{bmatrix} \bar{x} \tag{2.1}$$

Where $I$ is the identity matrix ($3 \times 3$), $t$ the translation vector, $\bar{x}$ is the initial homogeneous coordinates of the object and $x'$ new homogeneous coordinates of the object after translation.

**Rotation + translation** Also know as 3D *rigid body motion* or the *3D Euclidean transformation* [21] is defined as

$$x' = \begin{bmatrix} R & | & t \end{bmatrix} \bar{x} \tag{2.2}$$

Where $R$ is a $3 \times 3$ orthonormal rotation matrix

$$\mathbf{R} = \mathbf{R}_Z(\psi)\mathbf{R}_Y(\theta)\mathbf{R}_X(\phi) \tag{2.3}$$

$$= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{2.4}$$

Multiplying out the three matrices on eq. 2.4 , we obtain the Euler angle parameterization of the three-dimensional rotation matrix. Where $\psi, \theta, \phi$ represent the three Euler Angle axes, yaw-pitch-roll as shown in Figure 2.3.

$$= \begin{pmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi'\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{pmatrix} \tag{2.5}$$

### 2.2.2 Pinhole Camera

Understanding the parameters of a pinhole camera model that will be used during this project is crucial to comprehending the principles that will be thoroughly presented. As a result, this section will give a brief overview of that parameters used in this project.

**Camera Intrinsics**

The intrinsic matrix is parameterized by Hartley and Zisserman [22] as

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.6}$$

which uses independent *focal lengths* $f_x$ and $f_y$ for the sensor $x$ and $y$ dimensions. In a true pinhole camera, $f_x$ and $f_y$ have the same value, but it can differ for a number of reasons, such as flaws in the digital camera sensor, errors in camera calibration and the image has been non-uniformly scaled in post-processing.

The camera's *optical centre* is the intersection point between the pinhole and the straight perpendicular line to the image plane. The *optical centre* is the point at which it intersects the image plane. Usually $(c_x, cy) = (W/2, H/2)$, where $W$ and $H$ are the image height and width, and ($c_x$, $c_y$) is the optical centre.

**Camera Extrinsic**

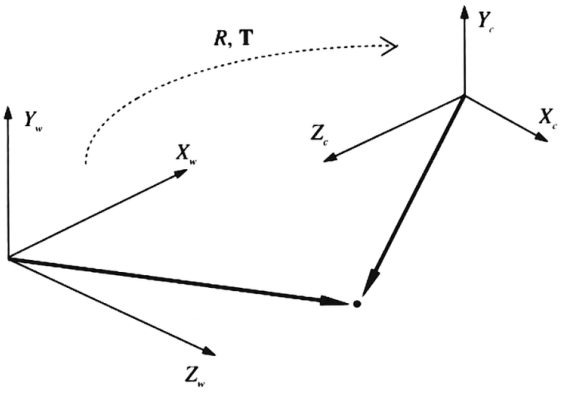



Figure 2.4: Extrinsic parameters and the relationship between a point in the world and camera coordinates.

Due to the frequency with which a camera might move, it is necessary to construct a new coordinate system that will be utilized as a reference frame, as shown in Figure 2.4. This new coordinate system will be referred to as the world frame in this context.

The extrinsic matrix is a rigid transformation matrix, consisting of a $3 \times 3$ rotation matrix in the left-block and a $3 \times 1$ translation column-vector in the right:

$$E = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

Combining equations 2.6 and 2.7 makes it possible to project a 3D point $X$ into a pixel $x$ using the formula $x = P \cdot X$, where $P = K \cdot E$.

## 2.3 Image Segmentation

Image segmentation is a classic subject and is one of the most studied in image processing and computer vision [23]. Image segmentation divides an image into distinct regions. It is often not interested in all the parts of the image, but rather in particular areas with semantically relevant classes. With many uses, such as robotic perception, augmented reality, medical image processing, image segmentation is an essential subject in computer vision [4]. It is also an essential topic in autonomous driving due to the capability of detailed scene understanding.

Image segmentation has been an important step in recent years, and a large number of papers have been carried out to develop new approaches. It started with the earliest methods, such as thresholding [24], k-means Clustering [25], watersheds [26]. New modern algorithms have emerged in computer vision due to the popularity of deep learning and machine learning techniques.

Image segmentation aims to obtain the mask of the interest object(s) by choosing the portions in the image that correspond to it. Two distinct types of segmentation can be formulated, the most classic version (semantic segmentation) and instance segmentation, represented in Figure 2.5.

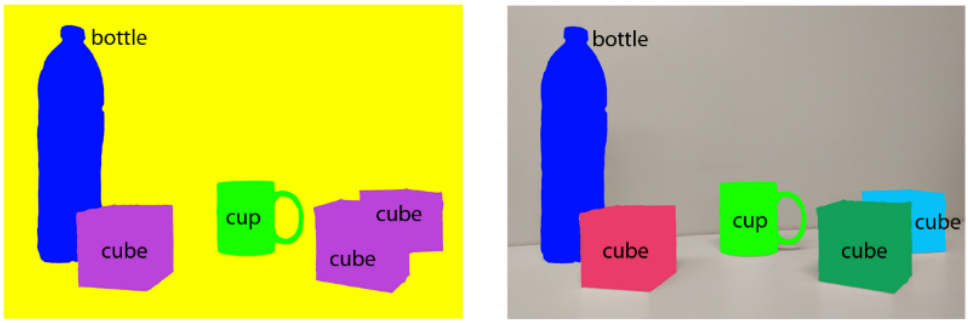



Figure 2.5: In the first image where semantic segmentation is applied, the category (cube) is one of the outputs, and all cubes are coloured the same. The image on the right, instance segmentation divides the instances (the cubes) apart from the categories (cube). Extracted from [4].

In semantic segmentation, the output is the category for every pixel in that image, labelling all the pixels independently with the respective category label. Consequently, it can not distinguish between two objects, as shown in Figure 2.5 . In contrast, instance segmentation outputs the

location and objects identities in that image, similarly to object detection. On the other hand, instance segmentation predicts all the segmentation mask for all the objects in the image.

One limitation on underwater images is on the semantic segmentation, due to almost no underwater public datasets to train the models. Thus, it is impossible to have a benchmark evaluation of segmentation models. Recently, in [7] the authors have proposed a large-scale dataset for semantic segmentation of underwater imagery that contains 1525 underwater images and the respectively ground truth semantic labels.

In the next section, a brief introduction to the deep learning architectures used for segmentation will be presented, such as Fully convolutional network, encoder-decoder based models and lastly an R-CNN based models (for instance segmentation).

### 2.3.1 State-of-the-art segmentation models

#### 2.3.1.1 Fully Convolutional Network

The authors of [5] proposed the Fully Convolutional Network (FCN), one of the first deep learning networks for image segmentation task. FCN uses and transforms the existing CNN architectures, such as VGG16, GoogLeNet, AlexNet and ResNet into fully-convolutional layers [4], replacing the fully-connected layers of that architectures. Consequently, the model outputs a spatial segmentation map instead of classification scores, as shown in Figure 2.6.

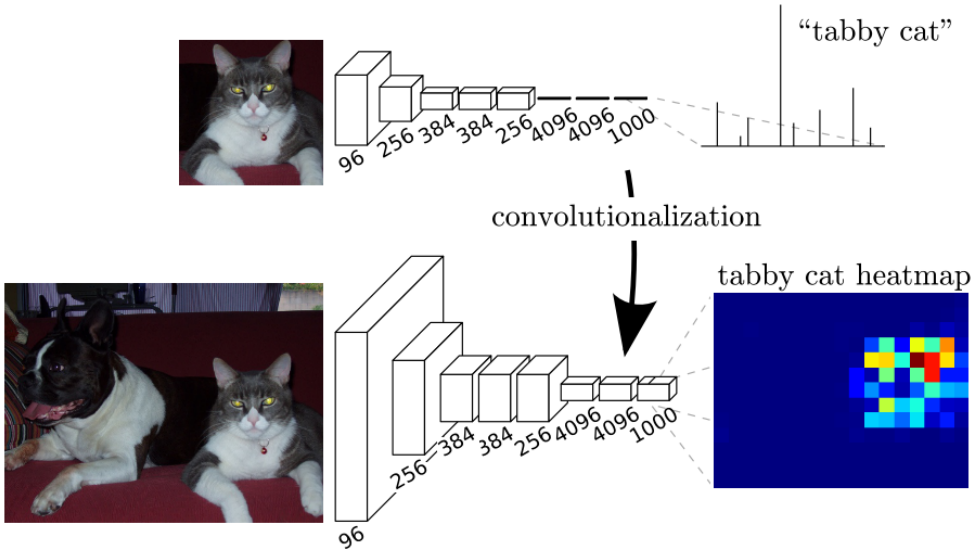



Figure 2.6: Replacing fully connected layers into fully convolution layers.
Extracted from [5].

This work is considered a milestone in image segmentation [27]. It showed how CNNs could be trained end-to-end, learning how to produce dense predictions for semantic segmentation on

variable-sized images. However, FCN has some limitations in the standard model. The first significant limitation is the actual speed being not fast enough to run in real-time. Lastly, the model can not be directly applied to unstructured data such as 3D point clouds.

#### 2.3.1.2 Encoder-Decoder Based Models

The encoder-decoder architecture comprises an encoder that takes an input image and generates feature maps or a low-resolution image with a sequence of convolutions and pooling layers, and the decoder that recovers the original image resolution using the low-resolution feature map to produce the pixel-wise segmentation, minimizing the loss. However, the decoder is a divergence point in some architectures. One advantage compared with other methods is the freedom of choosing the input size.

The encoder-decoder architecture also takes a network for classification, such as VGG16 which can be used as encoder.

The authors of [6] proposed a new approach using a different decoder stage, as described above, with a series of upsampling and convolution layers with a softmax classifier at the end to predict pixel-wise labels, as shown in Figure 2.7.

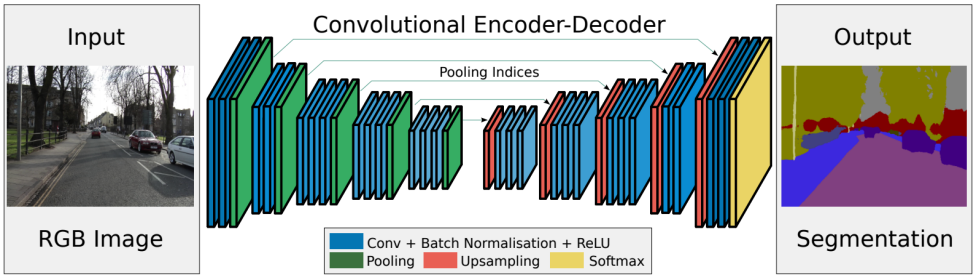



Figure 2.7: SegNet architecture. Extracted from [6].

A new fully convolutional encoder-decoder shown in Figure 2.8 was recently proposed by [7], with skip connections between mirrored composite layers. It provides competitive efficiency while maintaining fast end-to-end inference, which is fundamental for underwater robots autonomous navigation.

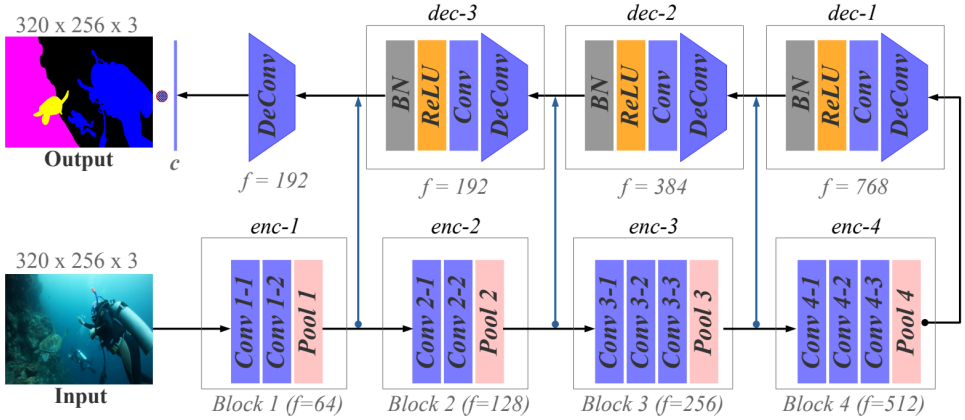



Figure 2.8: SUIM architecture. Extracted from [7].

#### 2.3.1.3 R-CNN based models

The initial purpose of region-based convolutional network (R-CNN) was to take an input image and output a set of bounding boxes, each of which contained an object and its class. The big problem with R-CNN is the computational cost, since it needs much time and cannot run in real-time. Many extensions resolved these issues, such as Fast R-CNN, Faster R-CNN, and Mask R-CNN. In testing times, Faster R-CNN is 213x faster than R-CNN with truncated SVD [28]. Faster R-CNN was the first perform a region proposal network to get the regions of interest (ROI) and the respective class label.

As illustrated in Figure 2.9, masked region-based convolution neural network (Mask R-CNN) works towards the problem of instance segmentation, the process of detecting and delineating each distinct object of interest in the image, it is a combination of two subproblems, object detection and semantic segmentation.

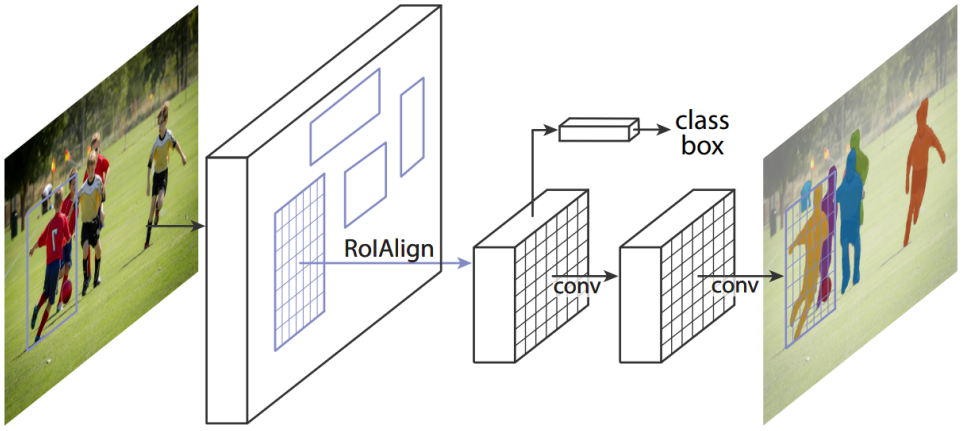



Figure 2.9: Mask R-CNN architecture. Extracted from [7].

Mask R-CNN [29] uses an architecture similar to Faster R-CNN, adapting it to instance segmentation, using fully convolution networks. Faster R-CNN performs object detection in two stages. First, it determines the bounding box and the regions of interest, which is done by the region proposal network (RPN). After that, for each ROI, it determines the class label of the object, which is done with ROI pooling. Finally, the proposals are passed to a fully connected layer to classify and output the bounding boxes for objects.

The authors of [8] proposed an integrated multi-scale Retinex with colour restoration (MSRCR) into the Mask R-CNN framework in underwater environments. The objective is to detect and segment three underwater creatures, namely echinus, holothurian, and starfish.

It verifies that Mask R-CNN has notable results in a challenging and complex environment; however, compared with Mask R-CNN, the mAP and Recall of the proposed method by the authors increases, the method has a recall of 94,52%, while the Mask R-CNN have 83.95%. Similar to Mask R-CNN, the method has a low number of pictures detected per second (FPS), compared with YOLOv3. They also demonstrate that proper image enhancement algorithms can improve deep learning models accuracy in a limited sample dataset underwater scenario.
Figure 2.10 illustrates one example of instance segmentation proposed by [8].

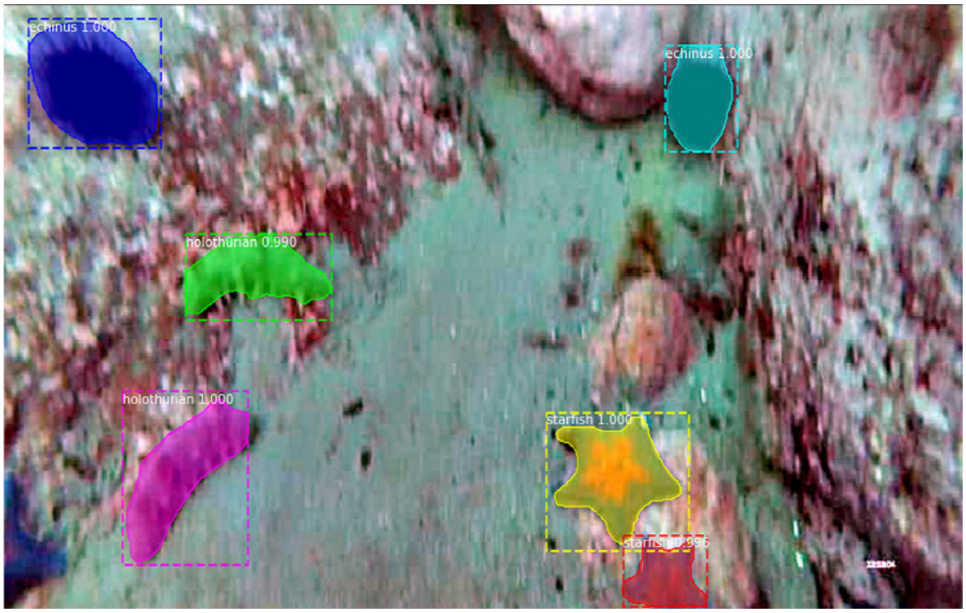



Figure 2.10: Result of instance segmentation proposed by the method.
Extracted from [8].

## 2.4 Depth Estimation

Predicting the depth of the scene from the input images is a challenge for robot navigation. Traditional depth estimating techniques, such as motion structure and stereo vision matching, are based on the correspondence of several points of view.

Depth estimation can be performed with different methods, such as supervised, unsupervised and semi-supervised. Inferring the depth of a scene and the ego-motion of a robot or autonomous vehicle is a critical problem in robotics and autonomous driving. Accurately estimating the 3D positions of objects and the scene's geometry is critical for motion planning and decision making. Several methods attempt to learn depth and ego-motion from monocular video. Figure 2.11 illustrates an example of unsupervised learning of depth and ego-motion from monocular videos.

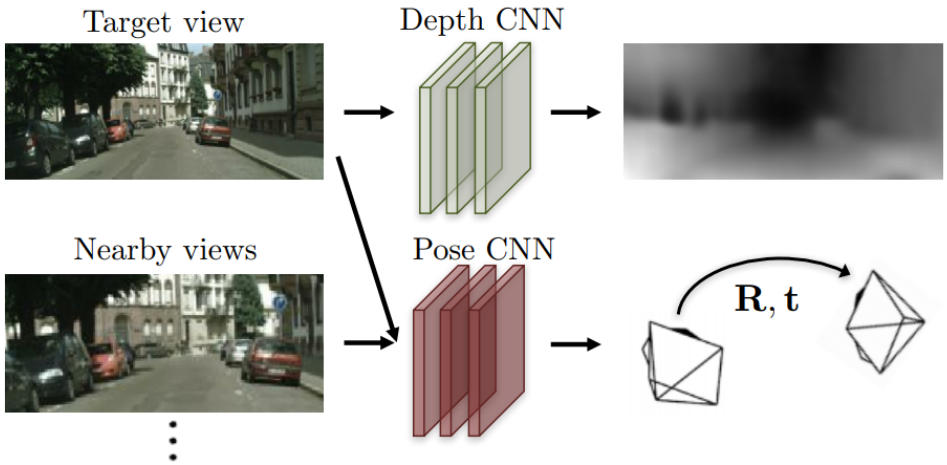



Figure 2.11: The authors of [9] proposes an architecture that employs a single-view depth and multi-view pose network, with a loss calculated by warping nearby views to the target using the computed depth and pose. A model that takes a succession of images and attempts to explain its observations by predicting the camera's expected motion and scene structure. Extracted from [9].

**Methods based on explainability mask:** The view reconstruction algorithm based on the projection function relies on the static scenario assumption, i.e. the location of dynamic objects on adjacent frames does not satisfy the projection function that affects the photometric error and the training process. Masks are also commonly used to reduce the effect of dynamic objects and occlusions.

Recent methods such as [9], [30], [31] and [32] use mask estimation based on deep learning, which reduces the effects of dynamic objects and occlusion.

**Methods based on traditional odometry:** Instead of using the pose estimated by a pose network, the pose regressed from traditional direct visual odometry helps in depth prediction [33]. Direct visual odometry takes the depth map created by the depth network and the three-frame snippet to estimate the poses between frames by minimizing the photometric error. Then, the measured poses are returned to the training system.

**Methods based on multi-tasks framework:** The latest methods have implemented additional networks for multi-task, such as object motion [10], optical flow [34] and camera intrinsics matrix [35].

*Casser et al.*[10] proposed unsupervised learning of scene depth and robot ego-motion, where supervision is provided by monocular videos. It also reveals that unsupervised depth estimation models can be more accurate than supervised sensor approaches due to sensor reading problems such as misses or noisy sensor values. It can also run in real-time on several GPUs.

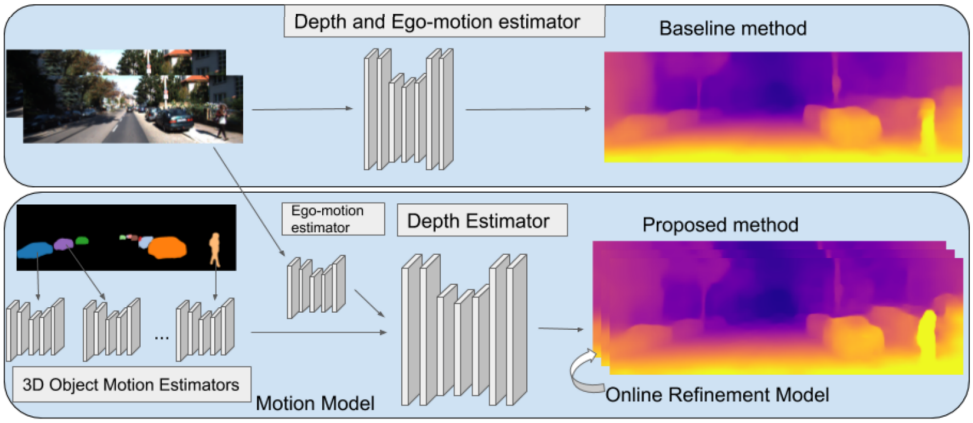



Figure 2.12: An illustration of the framework. Extracted from [10].

This approach enables migration between environments by moving models trained in one environment to another. It also enhances the pioneering work in unsupervised depth estimation [30], which in practice suffers from the fact that object motions in complex scenes are not handled.

In this improved version *Casser et al.*[10] considers the motion of dynamic objects in the scenes, as seen in Figure 2.12. An object motion network is implemented to simulate individual objects motion and uses segmented images as inputs. Since the above techniques are based on the pre-requisites of known camera parameters, this restricts the network's application to unknown cameras.

## 2.5 Simultaneous Localization and Mapping

Simultaneous Location and Mapping (SLAM) is a common technique for autonomous navigation of mobile robots in an unknown environment and aims to incrementally build a consistent map of this environment while simultaneously determining its location within this map. To drive precisely, a mobile robot must have an exact map of the environment [36].

Mobile robots may have internal and external sensors included. The internal sensors allow the entity to collect measurements such as velocity, position change and acceleration. The sensors that can be used for this purpose are the incremental encoders, accelerometers and gyroscope.

The external sensor contains a laser scanning sensor, an ultrasonic sensor and a vision sensor to obtain the robot's distance and position from the external environment relative to the external object [37].

When cameras are employed as the only exteroceptive sensor, it is called *visual SLAM*. *Visual SLAM* systems can be complemented with information from internal sensors to increase accuracy and robustness.

Monocular *visual SLAM* techniques are a challenge since they have problems such as necessary initialization, scale ambiguity and scale drift. However, this work focuses only on the *visual SLAM* with monocular and RGB-D data because of hardware constraints present; further ahead, it can be extended to other sensors.

*Visual SLAM* is classified into direct and feature-based (indirect) methods. The direct method leads to semi-dense and dense construction, while the indirect method causes sparse construction. Figure 2.13 illustrates this difference in a straightforward way. Both methods extract characteristics from images and connect them with descriptors. The direct method compares the intensity (depth can also be included) of a pixel in one image to its warped projection in another image. This is referred to as photometric error. On the other hand, the indirect method requires an additional step to extract features and point-to-point correspondence. Then, the reprojection error function compares the 2D position in one image to the 2D projection of the equivalent 3D point in another image. The additional step renders it indirect.

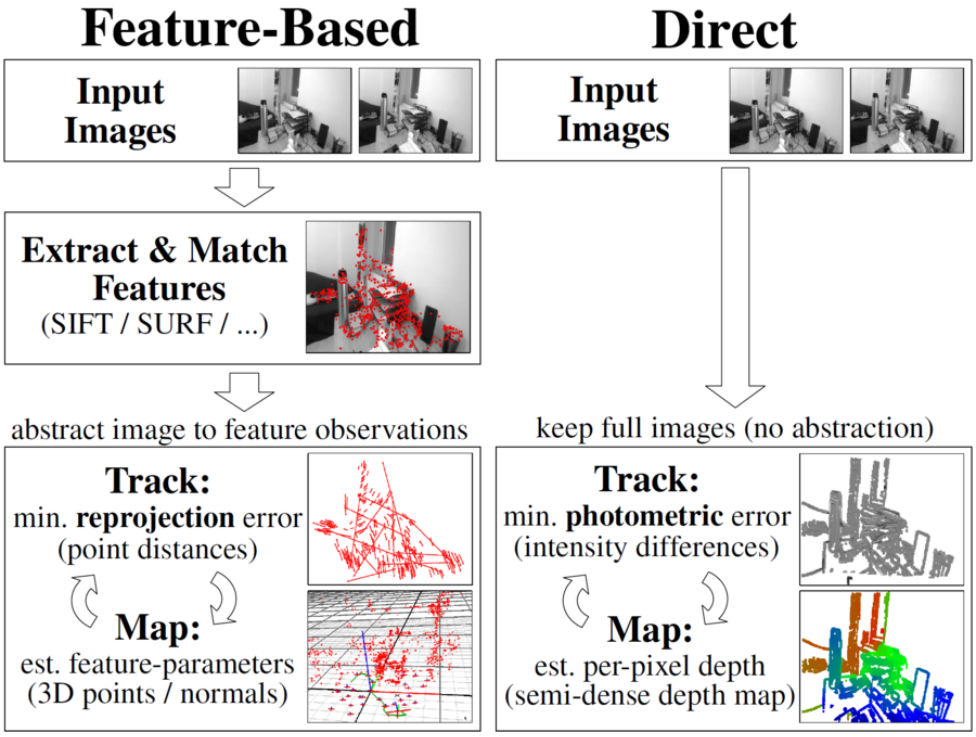



Figure 2.13: Difference between direct and feature-based methods. Extracted from [11].

### 2.5.1 Sparse Construction

- **MonoSLAM** [38] — was the first monocular SLAM system, based on extended kalman filter (EKF) and can perform in real-time.
- **ORB-SLAM** [39] — was the first to implement the concept of separating camera tracking and mapping in parallel, with three parallel threads: tracking, local mapping and loop closure.
- **ORB-SLAM2** [40] — introduces support with monocular, stereo and RGB-D cameras, including map reuse, loop closing and relocalization capabilities;
- **ORB-SLAM3** [41] — the most recent ORB-SLAM families, is 2 to 5 times more accurate than previous approaches, performing Visual, Visual-Inertial and Multi-Map SLAM with monocular, stereo and RGB-D cameras, using pin-hole and fisheye lens models. ORB-SLAM3 is as stable and substantially more accurate as of the best solutions available in the literature in all sensor configurations;

### 2.5.2 Semi-Dense Construction

- **LSD-SLAM** [11] — proposes a direct method which operates on Lie-Algebra, allowing to build large-scale and consistent maps of the environment with a monocular camera and runs in real-time;
- **DSO** [42] — is a new approach from the author of LSD-SLAM, a direct method that creates semi-dense models similar in density to LSD-SLAM, but much more accurate;

### 2.5.3 Dense Construction

- **ElasticFusion** [43] — proposes a method for performing dense visual SLAM in real time that is capable of capturing extensive and comprehensive dense globally consistent surfel-based maps of room-scale environments explored with an RGB-D camera;
- **BundleFusion** [44] — uses an RGB-D camera and estimates globally optimized poses in real-time, supports robust tracking with recovery from gross tracking errors and re-estimates the 3D model in real-time to ensure global consistency, within a single parallelizable optimization framework

### 2.5.4 Segmentation with SLAM

SLAM methods can include semantic information to increase the performance and provide a better understanding of the scene.

- **Fusion++** [45] — is a real-time object-level SLAM system that generates a stable and accurate 3D graph map of any reconstructed item. While an RGB-D camera navigates through

a cluttered indoor scene, Mask R-CNN instance segmentations are used to initialize compact per-object Truncated Signed Distance Function (TSDF) reconstructions with object size-dependent resolutions and a novel 3D foreground mask;

- **MaskFusion** [46] — is a real-time, object-aware, semantic and dynamic RGB-D SLAM system based on Mask R-CNN. It can also recognize, segment and assign semantic class labels to multiple objects in the scene, while tracking and reconstructing them even when in continuous and independent motion;

Other SLAM methods can address a variety of issues, such as **ScanComplete** [47], which takes an incomplete 3D scan of a scene and predicts a complete 3D model along with per-voxel semantic labels; **DenseFusion** [48] estimates 6D pose of a set of know objects from RGB-D images; **CNN-SLAM** [49] estimates the depth with CNNs from a monocular camera.

## 2.6 3D Reconstruction

With the latest developments in autonomous driving and augmented reality, a fundamental computer vision goal is the accurate 3D reconstruction of the surrounding world. This 3D reconstruction is typically achieved by fusing some sensors, such as LIDAR and Stereo, into 3D models. Although these sensors can be highly powerful, they require special hardware that will not be used in this project.

Some approaches in 3D reconstructions require single-view, multiple-view or stereo reconstruction to perform the 3D reconstruction. The single-view reconstruction uses information from one view of the object or scene. In contrast, multi-view reconstruction uses information from two or more viewpoints of the object or scene. Stereo reconstruction is a particular case of multi-view reconstruction, and in this case, two cameras often placed close together concerning the object, are used.

The vision setup can be defined in three categories: a monocular camera [50, 51], a stereo camera [52] and an RGB-D camera [53]. Since the restriction of monocular camera use mentioned in chapter 1, this research will rely on the monocular and RGB-D approaches since it is possible to simulate RGB-D data predicting depth maps in monocular cameras.

The next two sections will concentrate on object reconstruction and 3D semantic mapping, respectively.

### 2.6.1 3D Object Reconstruction

Various approaches can be made to achieve a 3D shape representation, such as voxels, point-clouds, and meshes.

**Mesh R-CNN**[12] extends Mask R-CNN and outputs high-resolution triangle meshes of the objects with a single RGB image.

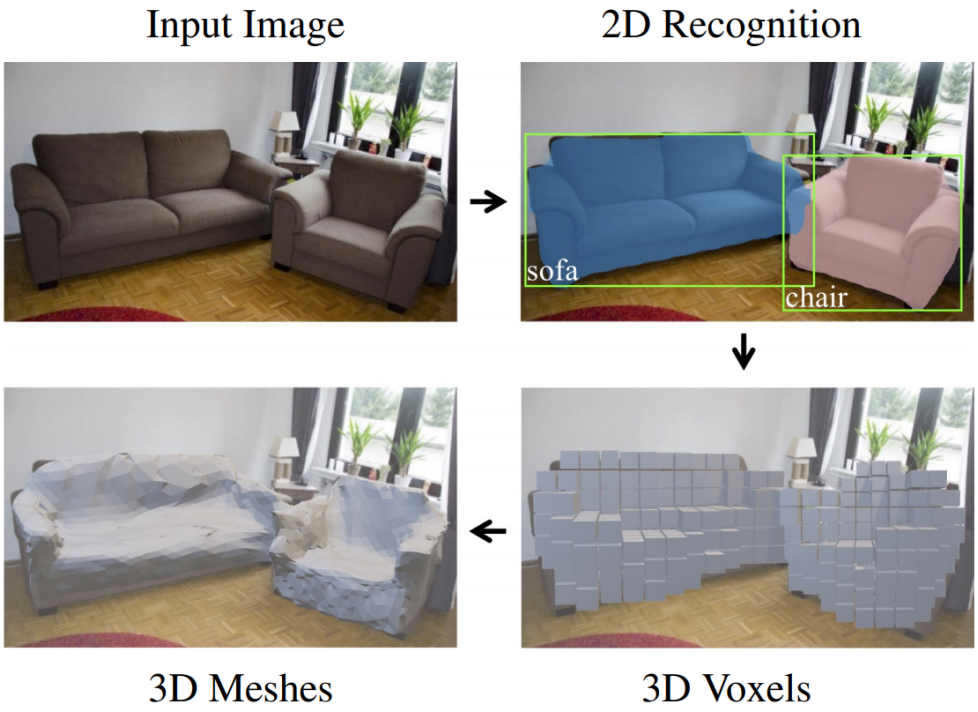



Figure 2.14: Predicting 3D meshes with an input image.
Extracted from [12].

Mesh R-CNN aims to build a system that inputs a single RGB image, detect all objects in the scene, and outputs a bounding box, category label, segmentation mask, and 3D triangle mesh for each detected object. Besides that, the system has to handle with cluttered images and be trainable end-to-end.

Figure 2.14 demonstrates that the system first detects and segments all the objects; then predicts coarse voxelized object representations which are converted to meshes.

The 3D shape process consists of a voxel branch and a mesh refining branch. First, the voxel branch calculates a coarse 3D voxelization of an object that is transformed to an initial triangle mesh. The mesh refinement branch then changes the vertex location of this initial mesh using a series of graph convolution layers on the mesh's edges.

Other methods, such as the latest PiFu[54], have introduced a model that uses occupancy fields to perform a highly detailed clothed human with a single input image or a multi-view input image. This recent work outperforms current solutions, as it can produce high-resolution surfaces, including unseen regions. PiFu demonstrates that it can be extended to general objects by explicitly combining global features and local features.

Some other methods are applied in multi-view shape prediction with modern learning techniques are [55, 56].

### 2.6.2 3D Semantic Mapping

3D reconstruction is desirable as a volumetric map, not just point clouds or surfels, since it is impossible to explicitly use point clouds and surfels for robot object collision detection or robot navigation. Densely labelled semantics is the secret to smart robot navigation, and it is also essential for robot-object interaction to distinguish between individual objects [57].

For the next stage of robot intelligence and intuitive user interaction, maps need to expand beyond geometry and appearance - They need to include semantics [13].

**SemanticFusion**[13] combines the geometric information from the ElasticFusion SLAM method with modern advancements in semantic segmentation using CNNs, allowing real-time processing at frame-rates of 25Hz.

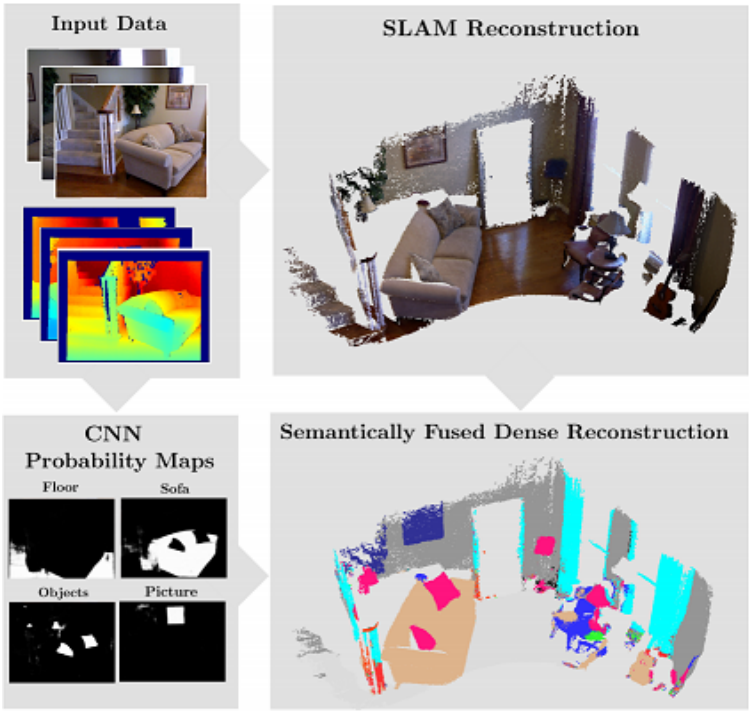



Figure 2.15: Overview of SemanticFusion pipeline [13].

As illustrated in Figure 2.15, the SemanticFusion pipeline consists of three different units: A real-time SLAM based on ElasticFusion, a CNN and a Bayesian update scheme.

The SLAM approach offers correspondences from a 2D frame to a globally compatible 3D map. Separately, the CNN receives an RGB-D image and returns a range of probabilities per pixel class. Finally, the Bayesian update scheme keeps track of the probability distribution class for each surfel, and uses the correspondence given by the SLAM to update those probabilities based on CNN's predictions.

**Voxblox++** [58] was proposed to overcome the previous method's limitation since it did not include details on the geometry and relative location of individual objects in the scene.

An approach to incrementally building volumetric object-centric maps during online scanning with a localized RGB-D camera was implemented. In each frame, the method performs geometric segmentation; semantic instance-aware segmentation refinement; data association and finally map integration.

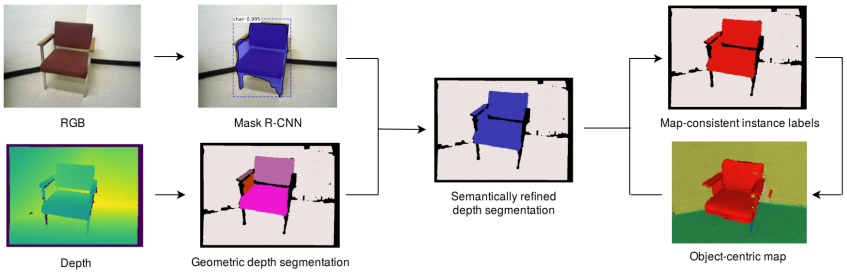



Figure 2.16: Individual stages of the system. Extracted from [13].

Figure 2.16 presents the individual stages of the *Voxblox++*. The RGB-D image is processed with Mask R-CNN to detect and predict the semantic mask. Simultaneously, the geometric segmentation decomposes the image depth into a collection of convex 3D segments. Consequently, the data association method defines the segments expected in the actual frame to their corresponding instance in the global map to be obtained for each map-consistent label. Lastly, dense geometry and segmentation information from the current frame is merged into the global map scale.

*Murez et al.* introduced Atlas [59], an end-to-end 3D reconstruction method, by directly regressing a truncated signed distance function (TSDF) from a collection of RGB images. They concluded that direct 3D regression is more successful than the intermediate representation of depth maps before calculating the scene's complete 3D model.

Additionally, the authors of [60] proposed a modern 3D semantic segmentation view-based approach that would use synthetic images made from "virtual views" of the 3D scene rather than limiting processing acquired by a physical camera. This method outperforms all prior multi-view approaches for 3D semantic segmentation.

## 2.7 CAD Model Alignment in RGB images

As seen before, we have many frameworks that can perform 3D reconstruction with RGB-D input; However, these reconstructions can be noisy and incomplete due to scanning patterns, motion blur or complex environment, like subsea images. As a result of these limitations, it is unsuitable for many applications, including real-time robots and augmented reality, since it is fundamental for the system to perceive the pose of the object.

One way to address this problem is to find and align CAD models to an input; with this solution, it is possible to have clean 3D reconstructions and each object's pose estimation, which serves the requirements for many applications described.

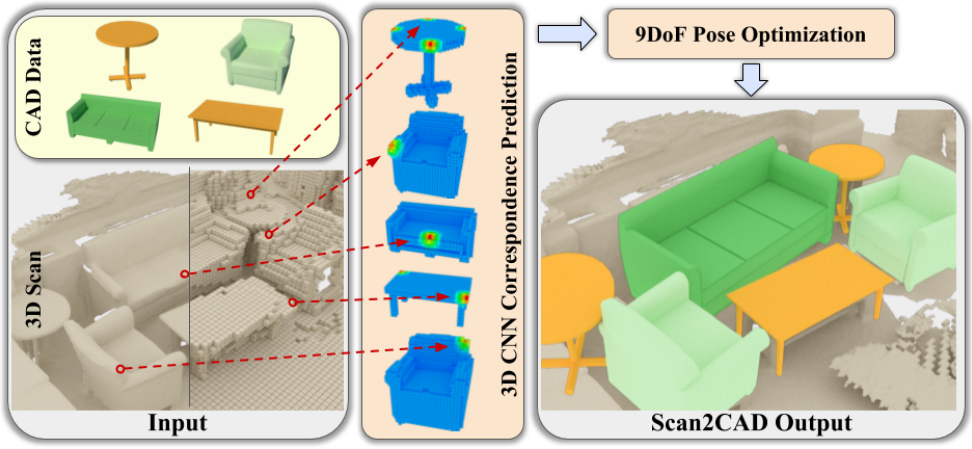



Figure 2.17: Scan2CAD takes an RGB-D input and align the CAD models by predicting heatmap correspondences and formulate an energy minimization to find the best alignment between CAD and the RGB-D input. Extracted from [14].

**Scan2CAD** [14] proposed a method to address the problems described before in RGB-D reconstructions. As illustrated in Figure 2.17, given the geometry of a noisy and incomplete RGB-D scan, Scan2CAD aligns clean and complete 3D CAD models to their counterpart objects in the 3D scan.

Scan2CAD proposed a variational 9DoF (3 degrees for translation, rotation, and scale each) optimization to produce final CAD model alignments. The method detects Harris keypoints and predicts correspondence heatmaps for each Harris keypoint and CAD model. It is possible to find optimal 9DoF transformations by using the predicted heatmaps.

Scan2CAD also introduced a new dataset from ScanNet [61], CAD models from ShapeNet [62] and oriented bounding boxes for each object.

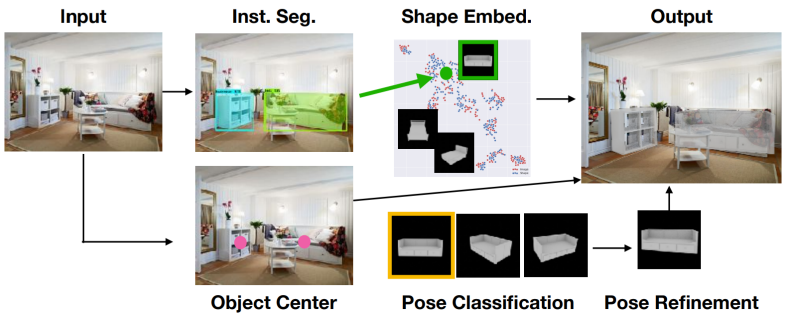



Figure 2.18: Representation of Mask2CAD system. Extracted from [15].

**Mask2CAD** [15] attempts to combine 2D recognition with 3D reconstruction capabilities by using CAD model representations of objects. The method takes an RGB image as input and predicts each object's bounding boxes, class labels, and segmentation masks. It learns to establish a shared embedding space between these recognized picture regions and 3D CAD models of objects from these detected image regions, allowing it to retrieve a geometrically similar model for the image observation, as illustrated in Figure 2.18.

For object detection, Mask2CAD uses a modified version of ShapeMask [63], which predicts bounding boxes, class labels, segmentation masks and also the object alignment to the image.

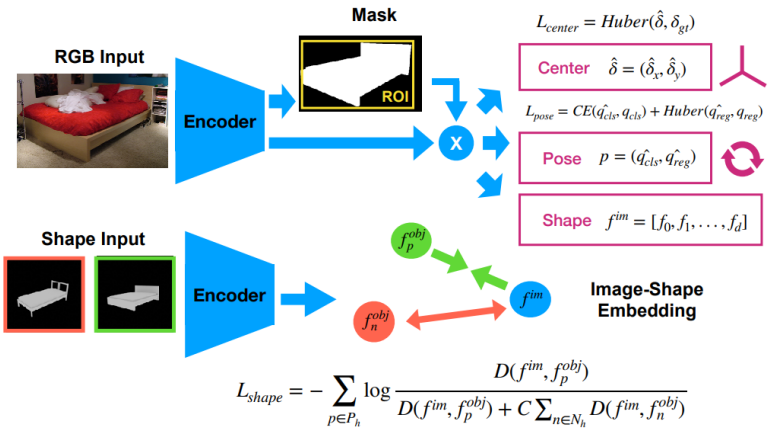



Figure 2.19: During training, an RGB image is used to perform object detection, which generates a bounding box, segmentation mask, and feature description for each discovered object. After that, the object feature descriptor is used to train for shape retrieval using an image-CAD embedding space, pose regression for object rotation, and center regression for object position. Extracted from [15].

As illustrated in Figure 2.19 the same regions of interest (ROI) feature predicts the rotation and translation of the object in the camera space. It initially performs k-medoid clustering on the quaternions of each class to obtain k-canonical poses for rotation.

Mask2CAD uses a contrastive loss for the pose and center regression with complex example mining since the contrastive loss is more stable than the triplet loss. Because the easy examples do not include much information to improve the model, mining complex examples helps their model to achieve better performance.

**Vid2CAD** [64] is based on Mask2CAD and addresses how to combine 3D shape retrievals and alignments from individual frames, such as those produced by Mask2CAD, during each video frame sequence to create a globally consistent 3D representation of the complete scene.

It solves the problem of matching CAD models to a video sequence of a complex scene with many elements. This integration process reduces scale and depth ambiguities in per-frame predictions and enhances the estimation of all pose parameters in general. The solution resolves

occlusions and handles items that are out of view in individual frames by using multi-view constraints, resulting in reconstructing all objects into a single globally consistent CAD representation of the scene.

# Chapter 3

# 3D Semantic Reconstruction

This chapter will discuss the two models that we developed for 3D reconstruction and object recognition. We will begin with our baseline method, which is based on dense 3D semantic SLAM, and then present a new methodology that utilizes pose alignment with CAD models to improve the precision.

## 3.1 3D Semantic Mapping

We present a method for leveraging RGB-D dense SLAM algorithms such as ElasticFusion [43] or SemanticFusion [13] using only monocular cameras. Due to the fact that these approaches require an RGB-D input and we only have an RGB image, we must estimate a depth map.

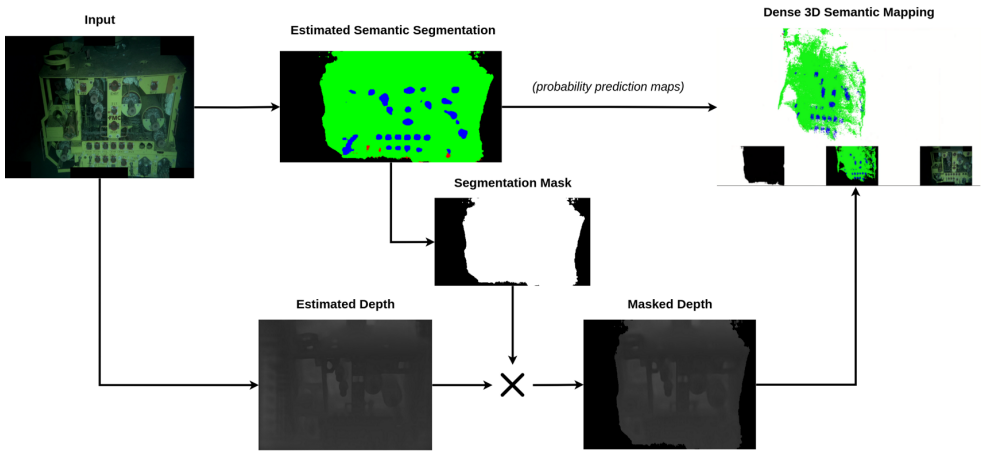



Figure 3.1: **An overview of our pipeline**: Input images are used to produce semantic segmentation and depth map estimations; These depth and segmentation maps are combined to create the final dense semantic map using SemanticFusion [13].

As illustrated in Figure 3.1, our pipeline is composed of three distinct units; 1) a model to predict semantic segmentation and the semantic mask segmentation; 2) a dense depth map prediction

model and finally, 3) the dense 3D semantic mapping, which merges these two predictions to build the semantic 3D map.

We want to distinguish three classes in this approach: structure, valves, and hot stabs. As illustrated in Figure 3.1, the segmentation model predicts the structure in green, the valves in blue, and the hot stabs in red.

### 3.1.1 Ground Truth Depth Maps Generation

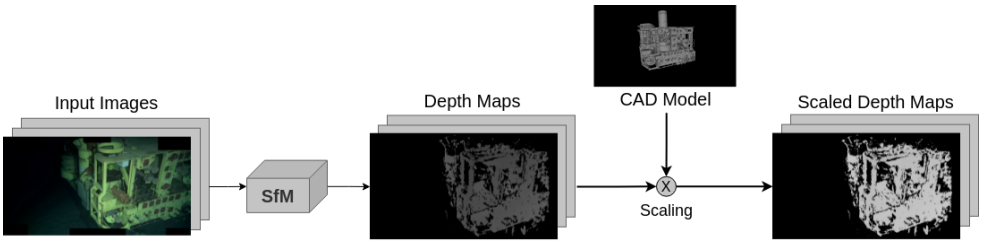



Figure 3.2: Pipeline proposed by [16] to create a dataset of semi-dense depth maps using structure-from-motion (SfM).

We used the methodology proposed in [16] , which presented a method for generating "ground truth" depth maps through the usage of the SfM framework, as illustrated in Figure 3.2. The purpose of SfM is to concurrently estimate the 3D model of a structure and the camera pose using image correspondences; thus, the environment must be static to obtain correct triangulation and matching points.

As the first step in constructing an SfM pipeline, a Match Graph must be constructed, with images working as nodes and an edge connecting two images if there is any point connection between them. We begin by identifying and reconstructing multiple disjoint Match Graphs individually, resulting in multiple independent point clouds. Along with the point clouds, the relative position of the camera used to capture each input image is determined. We obtain a depth map for each input image using this information in conjunction with the camera's intrinsic parameters [16].

To properly train and evaluate a CNN for depth prediction, we require a dataset with consistent scale across all examples; we scaled the different point clouds to metric units using 3D CAD models of known underwater structures.

To create our annotated dataset, we used OpenSfM as our SfM framework. OpenSfM is a Python-based Structure from Motion library built on top of OpenCV. It allows the construction of a high-quality point cloud of the entire structure at a high cost in terms of time. This software generates a JSON reconstruction file containing all camera poses, the 3D reconstruction and depth maps for all frames, which we used as our ground truth.

#### 3.1.1.1 Improved Ground Truth Depth Maps Generation

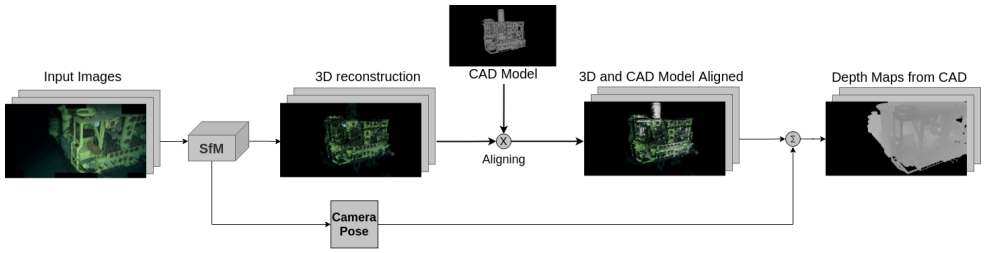



Figure 3.3: Our proposed pipeline to improve the framework proposed by [16]. We extend the framework by adding a new branch that generates synthetic views to capture depth maps from CAD using the camera pose.

The noise and incompleteness of the depth maps generated by the prior method can be a problem for our depth map model, as using semi-sparse ground truth depth maps makes it difficult to achieve a usable 3D reconstruction.

To tackle this issue, we proposed a new method to create the annotated dataset. As shown in Figure 3.3, we first aligned the CAD model with the 3D reconstruction generated by OpenSfM, and then we captured the depth of the CAD model per frame using the predicted camera poses by OpenSfM.

Due to the fact that we do not know the intrinsics of the camera, we empirically calibrated the camera parameters, assuming a pinhole camera model, to obtain the *focal length* and principal point.

This solution may have some disadvantages, such as alignment errors and the camera intrinsics not being accurate, but it produces more accurate depth maps for our model.

### 3.1.2 Depth Map Estimation Model

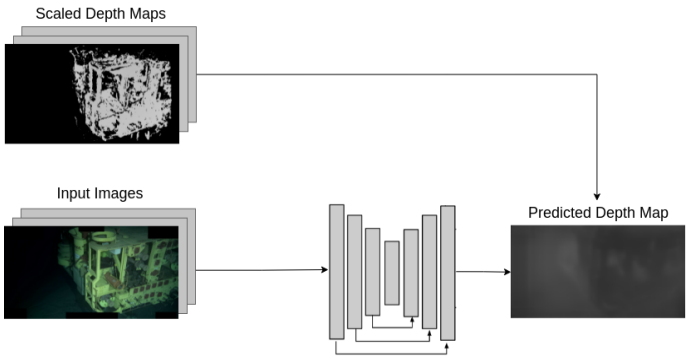



Figure 3.4: Our CNN based on [16] with skip-connections trained with semi-sparse ground-truth depth maps.

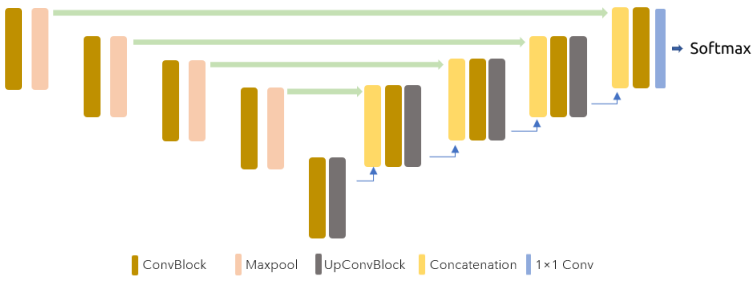
The architecture of our depth model, as illustrated in Figure 3.4, is inspired by the DispNet [65] architecture, which comprises an encoder-decoder architecture with skip connections between the encoder and decoder. Our network, on the other hand, does not perform multi-scale depth prediction. All convolutional layers are followed by ReLU activation, except for the prediction layers, which use $1/(\alpha * sigmoid(x)+\beta)$ with $\alpha = 7.203$ and $\beta = 0.935$ to constrain the predicted depth to be always positive within a reasonable range.

We used as loss the mean squared error (MSE) between the predicted depth and the depth ground truth. Because the ground truth depth maps are semi-sparse, some pixels in the input image do not have a ground truth value. As a result, we multiplied the ground truth depth map mask by the predicted depth map, ignoring the background and pixels with no value in the ground truth to avoid forcing the model to predict values close to zero. Therefore, the loss is applied only to the pixels for which y is defined:

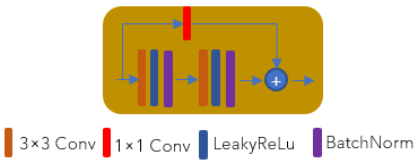
$$MSE(x,y) = \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i)\|^2 \tag{3.1}$$

Where N denotes the number of pixels in the input that contain ground-truth values, $x$ represents the input image and $f(x)$ the output of the current model, and $y$ denotes the ground truth depth map.

### 3.1.3 Semantic Segmentation Model




(a) Our ResUNet architecture based on [66].




(b) Convolutional block for the presented model based on [66].

Figure 3.5: Our ResUNET architecture and the convolutional block.

For this application we used ResUNet, an encoder-decoder architecture that combines the strengths of residual learning and the simpler model U-NET, as shown in Figure 3.5a. The residual unit simplifies the network's training, while the skip connections ensure that the data propagates without degradation, allowing the creation of a neural network with substantially fewer parameters and

improved performance. Our model differs slightly from ResUNet, consisting in five convolutional blocks (ConvBlock), each one followed by a maxpool layer of 2 x 2, except for the fifth ConvBlock. Each ConvBLock comprises two blocks of 3 x 3 convolutions (stride 1 and padding 1), a LeakyReLU with a slope of 0.2 and batch normalization, as represented in Figure 3.5b.

Because there are three classes (structure, valves and hot stabs), we used multiclass cross-entropy loss to classify them, whose output is a probability value between 0 and 1.

$$CE = -\sum_{i}^{C} t_i \log(s_i) \tag{3.2}$$

Where $t_i$ is the ground truth value (0 or 1) if class label is the correct classification; $s_i$ the predicted probability of class $i$, and $C$ the number of classes (structure, valve, hot stab).

### 3.1.4 3D Semantic Mapping

Our final step is to combine the segmentation and depth estimations described previously. To accomplish this, we modified the original SemanticFusion [13] to work with our models. The original SemanticFusion pipeline is composed of three separate units, 1) a real-time SLAM system, ElasticFusion, which provides dense correspondence between frames; 2) a CNN to output a dense pixel-wise semantic probability map; and finally, 3) a Bayesian update scheme that keeps track of the class probability distribution for each surfel[1], updating those probabilities using the correspondences provided by SLAM based on CNN's predictions.

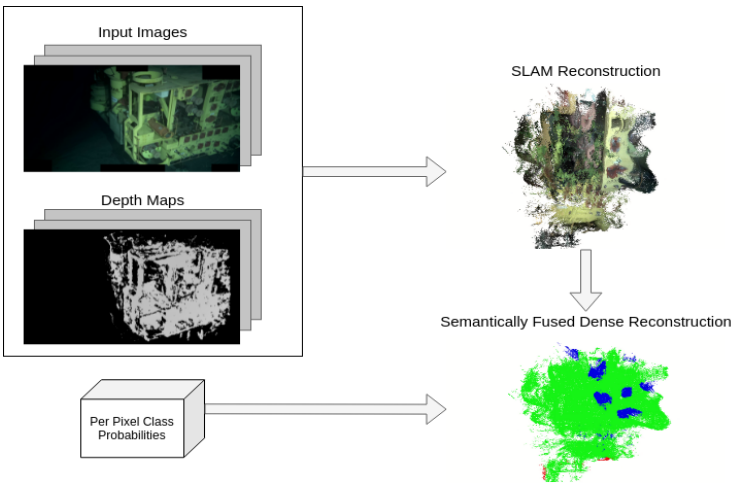



Figure 3.6: Our proposed architecture inspired by SemanticFusion [13]. Our model takes a set of RGB images and the correspondence depth estimation, which will be used for SLAM reconstruction. Our semantic segmentation model generates per-pixel class probabilities, which will be used to perform semantically fused dense reconstruction.

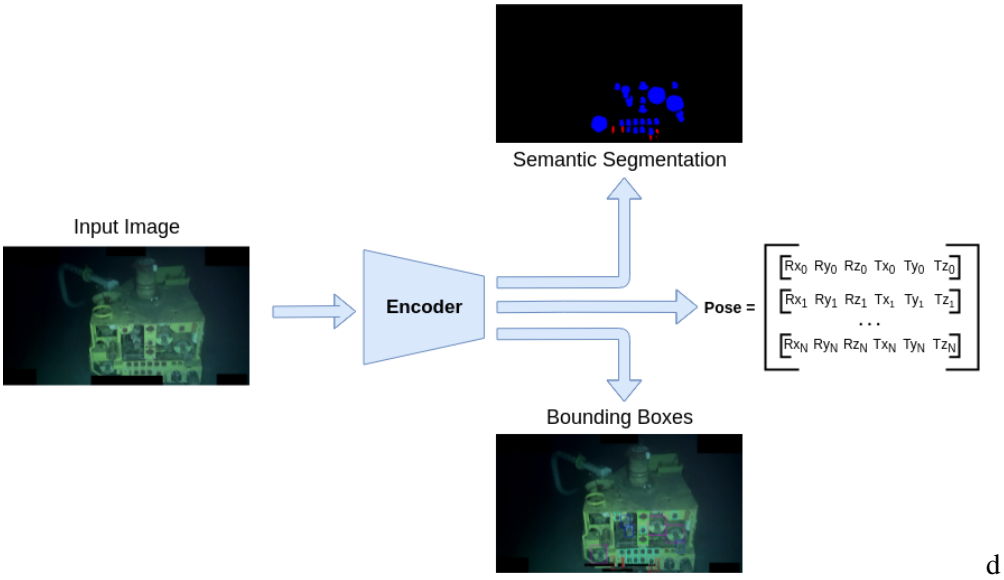
[1] Surface Element

As illustrated in Figure 3.6, our model removes the CNN architecture implemented by SemanticFusion, and replaces it with a per pixel class probabilities that we will read at the beginning of our method, making the method more adaptable to new models. Our method begins by reading all files containing the probabilities of each class in each pixel for each frame. This file has $C$ columns and $H \times W$ rows, where $C$ denotes the number of classes, $H$ denotes the image's height, and $W$ denotes the image's width. By loading these files at the start of the program, the CNN is no longer required to predict the per-pixel class probabilities allowing us to omit them from our method.

## 3.2 CAD Model Alignment With Images

We present our method for jointly detecting objects in real-world images and selecting the corresponding CAD model and pose for each detected object. This method produces a clean representation of the objects in an image; it ensures a correct, efficient shape representation for applications such as interactive scenarios.

As explained in section 2.7, techniques based on RGB-D SLAM generate geometry that may or may not reflect a natural shape, with a tendency toward noise or over smoothing and an excessive amount of tessellation. Due to these restrictions, these results are unsuitable for a wide variety of applications, including real-time robotics scenarios. Our proposed method addresses the limitations of our earlier method, which were discussed in section 3.1. In addition, this methodology attempts to accurately represent the objects of interest, such as valves and hot stabs; this is accomplished by employing the CAD model for each object.




d

Figure 3.7: Our proposed pipeline, which is based on Mask R-CNN, generates bounding boxes, 6dof poses, and semantic segmentation for each detected object. For each of the N bounding boxes, the encoder predicts a vector of rotation and translation along the three axes.

### 3.2.1 Proposed Architecture

As illustrated in Figure 3.7, we start by detecting all objects in the image domain by predicting their bounding boxes, segmentation masks and class labels from the input image. Next, we predict the object alignment to the image as 6 dof pose optimization (rotation + translation) based on these observed image regions.

We used Mask R-CNN for object detection since it is a simple, flexible, and generic framework for object instance segmentation. Mask R-CNN takes an RGB image as input and produces detected objects with their bounding boxes, class labels, and segmentation masks. As described in section 2.3.1.3, Mask R-CNN uses the same RPN stage as Faster R-CNN to propose candidate object bounding boxes. Additionally, it introduces a new branch for predicting the class, box offset, and binary mask for each RoI.

We modified this framework to learn features for our pose estimation by adding a new predictor to output the pose (rotation + translation) for each RoI.

We used the average binary cross-entropy loss proposed by the original method for the $L_{mask}$, the classification loss $L_{cls}$ and bounding-box loss $L_{box}$ are indentical as those defined in [67], for the $L_{pose}$ we used the simple L1 loss specified in Equation 3.3.

$$L_{pose} = \frac{1}{N}\sum_{i=1}^{N} |(y_i - \hat{y}_i)| \tag{3.3}$$

Where N represents the number of bounding boxes predicted, $y$ represents the ground truth pose and $\hat{y}$ the prediction pose, both of which are a $6 \times 1$ vector denoting rotation and translation along each axis. $y = [R_x, R_y, R_z, T_x, T_y, T_z]$.

### 3.2.2 Generation of Ground Truth Object Poses

To train the method proposed, each object and frame must have a ground truth pose (rotation + translation). Since we have the camera pose for every frame generated by OpenSfM, mentioned in section 3.1.1.1, we can compute the pose for every object using the equation:

$$O/_c = C^{-1} \times O/_W \tag{3.4}$$

Where $O/_c$ represents the object pose relative to the camera, $C$ denotes the camera pose and $O/_W$ denotes the object pose relative to the world, all the variables are defined as a matrix of $4 \times 4$ $[R \mid t]$.

To obtain the object's pose relative to the world, the software MeshLab was used to align each object in its respective point cloud position manually. The point cloud was extracted by OpenSfM described in section 3.1.1.1. The camera pose for each frame was extracted by OpenSfM, as mentioned in section 4.

By applying equation 3.4 to the object's pose relative to the world and the camera pose, we obtain the object's pose relative to the camera. We applied this equation to each frame, using the camera pose for each frame and the object's pose relative to the world, a constant matrix; this resulted in an object pose relative to the camera for each frame.

Rather than training our model with a $4 \times 4$ matrix and then forcing it to predict 16 values, we convert the rotation matrix to Euler Angles, resulting in a pose vector of six elements $[R_x, R_y, R_z, T_x, T_y, T_z]$.

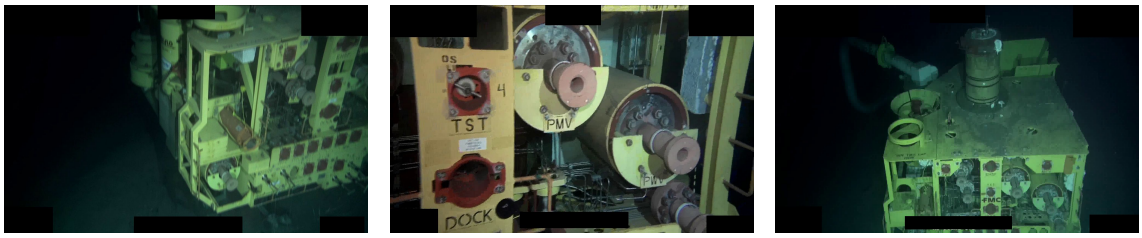
# Chapter 4

# Experiments and Results

## 4.1 Dataset

We used two different videos to implement the proposed methods, each with a unique scenario. After extracting one frame per second from each of the two videos, we obtain a total of 213 high-definition images ($1920 \times 1080$ resolution). These images were divided into 180 images for training, 33 images for validation. To evaluate our model, we used a distinct video from the others; we then extracted 1 frame per second from the video to evaluate our model, and 20 frames per second for dense 3D semantic mapping.

To evaluate our model, we used a unique video; we then extracted one frame per second from the video and used it to evaluate our segmentation and depth models, resulting in 99 images. We extract 20 frames per second to run the dense 3D semantic mapping described in section 3.1.4, resulting in 1980 images.

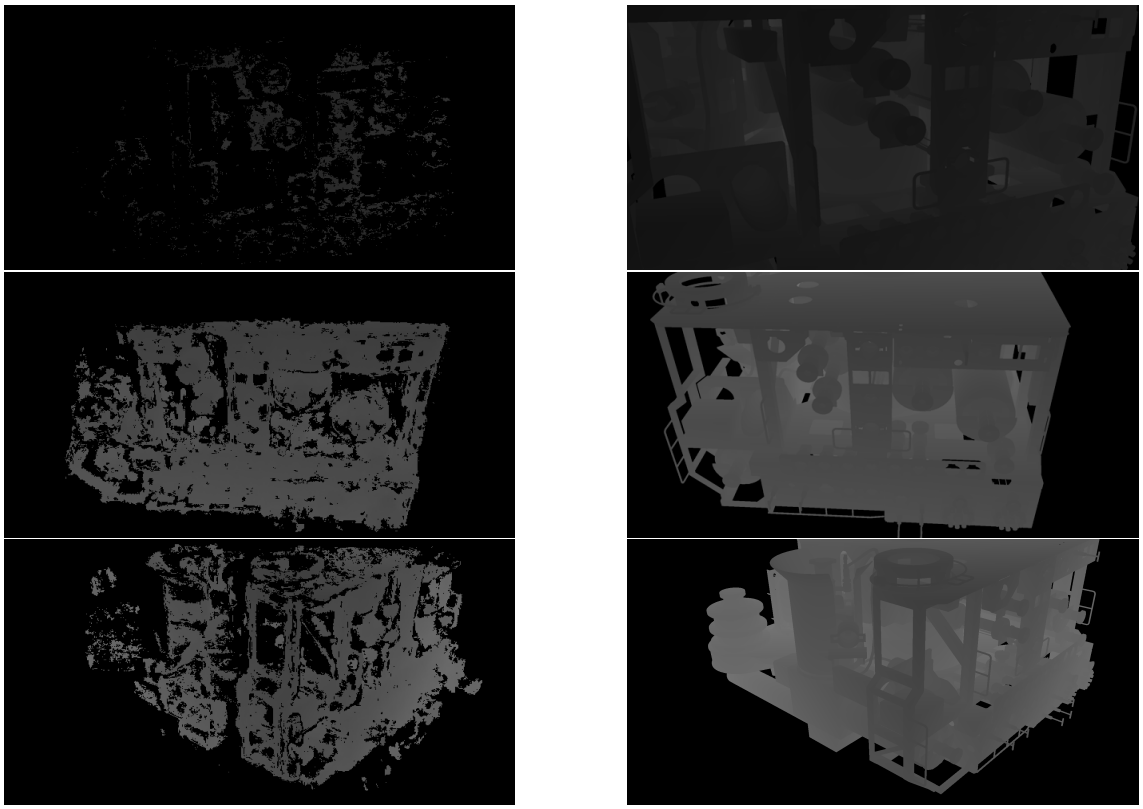



(a) Video for training purposes. (b) Video for training purposes. (c) Video for test purposes.

Figure 4.1: Comparative analysis of the videos included in our dataset's.

Example frames of the three videos are shown in Figure 4.1. The first Figure 4.1a combines multiple angles and shots of the structure from various distances; the second Figure 4.1b combines multiple perspectives and shots of the structure from a close distance ; the final Figure 4.1c combines different perspectives and shots from afar, solely for the purpose of testing the model.

## 4.2 Improved Depth Maps



(a) Depth Map Generated by OpenSfM. (b) Improved Depth Maps with CAD alignment.

Figure 4.2: Comparison between the depth maps generated by OpenSfM [16] and our improved method for generating ground truth depth maps.

As illustrated in Figure 4.2, the method described in section 3.1.1.1 significantly improved the ground truth depth maps; the depth maps are now cleaner and contain additional structure information, whereas the other method produces semi-sparse depth maps with a significant lack of structure information.

Because OpenSfM does not find correspondences between some images, the method proposed in [16] generates some ground truth images that are devoid of values. As a result, the depth estimation model must reject these depth maps. By contrast, our model has no such issues, as we obtain the depth from CAD rather than OpenSfM. Furthermore, both methods assume that the world is static.

## 4.3 Segmentation Model

We trained our model using the ResUNet segmentation model on an NVIDIA TITAN V GPU with 512x256 images and cross-entropy as a loss. It takes one hour to train 200 epochs with a batch size of 16.

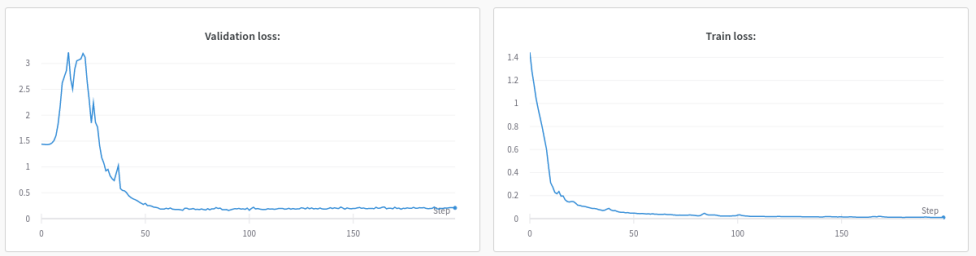



Figure 4.3: Loss of the training and validation data during the model's training phase.

We completed our training with a validation loss of 0.16 and a train loss of 0.02 using 188 training images and 33 validation images, as shown in Figure 4.3.

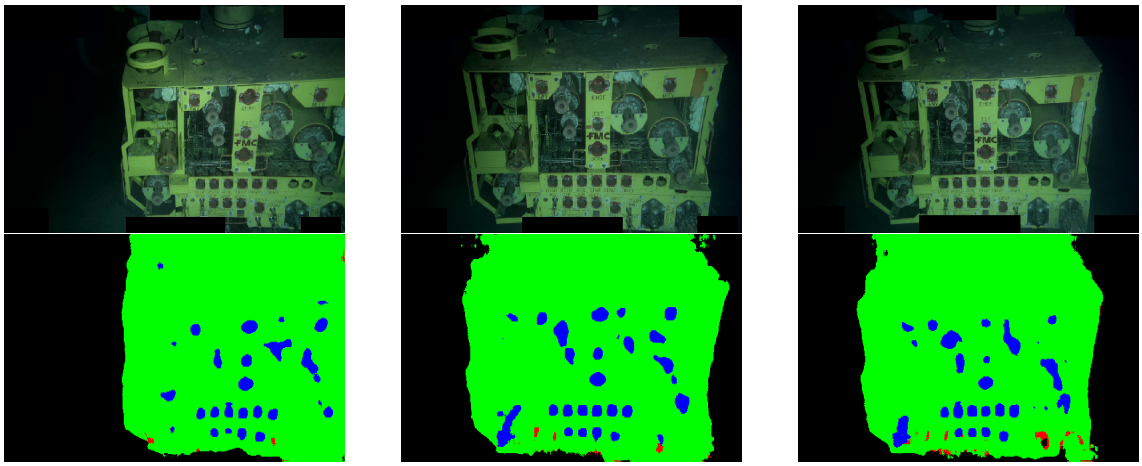


Figure 4.4: A selection of the frames generated by our segmentation model.Whereas green represents the Xmas tree, blue represents the valves, and red represents the hot stabs.

Figure 4.4 shows three examples of the model's output to a single image from the test dataset. The model predicts too many false positives for the valves, predicting valves rather than structure.

Table 4.1: Intersection Over Union (IoU) for each object.

| Object | IOU |
|---|---|
| Structure | 0.88 |
| Hot Stab | 0.11 |
| Valves | 0.44 |

Table 4.1 shows the evaluation per object; the hot stabs have the lowest IoU since they are one of the smallest objects in the image with poor image quality and illumination, and it is tough for the model to predict it; otherwise, the model predicts the tree reasonably well, as expected.

There are a few ways to improve the model's performance, including the addition of a larger dataset to enable a better generalization and a fusion of the segmentation and depth estimation to improve the segmentation. We focused on improving the segmentation of these small objects, which is accomplished by the approach described in section 3.2.

## 4.4 Depth Model

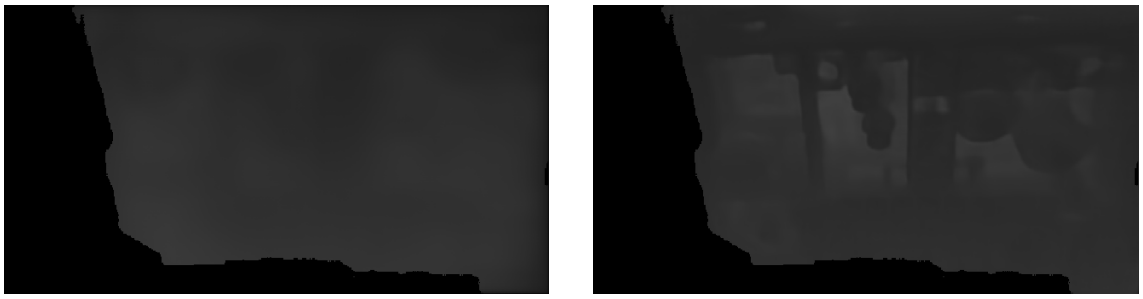
To evaluate the depth estimation, we follow the evaluation performed in [68]. The evaluation is based on measurements for error and accuracy. Absolute relative difference (Abs Rel), squared relative difference (Sq Rel), root mean squared error (RMSE), and root mean squared logarithmic error (RMSE log) are all error metrics. Three types of accuracy metrics are available: a1, a2, and a3, representing the percentage of pixel depth prediction reasonably near the ground truth. The a1 metric is the most exacting, while a3 is the least exact. For this purpose, we only consider the a1 and a2 metrics, as we want to compare the methods with strict accuracy to choose the most accurate.

We trained our depth model in 500 epochs, with a batch size of 4 and a learning rate of 0.0001. It takes about 400 epochs to converge and get $a1 = 0.93$ and $Absrel = 0.077$ in validation mode.

Table 4.2: Depth evaluation metrics over the 2 proposed methods for generating ground truth. The SfM GT Depth Maps method was described in section 3.1.1, while the CAD GT Depth Maps was described in section 3.1.1.1.

| **Method** | **Abs Rel** ↓ | **Sq Rel** ↓ | **RMSE** ↓ | **RMSE log** ↓ | **a1** ↑ | **a2** ↑ |
|---|---|---|---|---|---|---|
| SfM GT Depth Maps | 0.144 | 0.751 | 2.730 | 0.180 | 0.750 | 0.886 |
| CAD GT Depth Maps | 0.192 | 0.776 | 2.964 | 0.217 | 0.638 | 0.955 |
| SfM GT Depth Maps* | 0.302 | 1.776 | 3.564 | 0.517 | 0.454 | 0.755 |

* Evaluated on all pixels that belong to the Xmas tree. For pixels where the SfM method did not compute depth values, we use the CAD generation method.



(a) Depth map prediction generated with the method of SfM GT depth map.

(b) Depth map prediction generated with the method of CAD GT depth map.

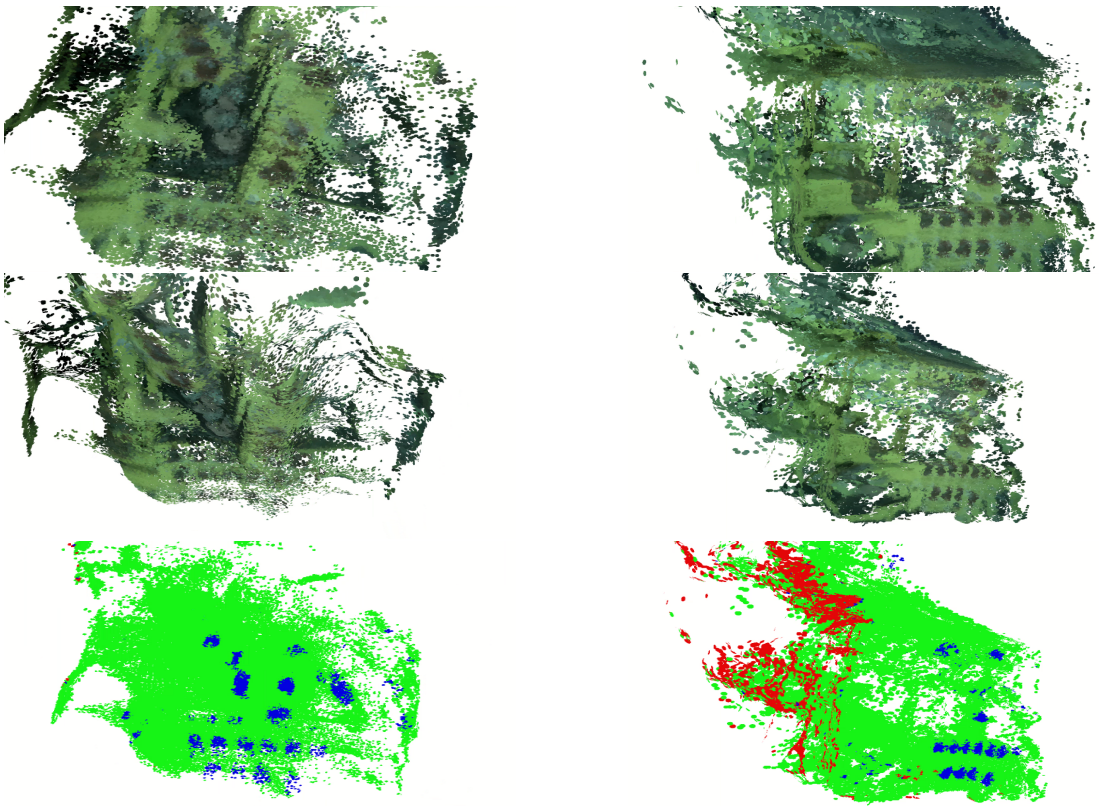
Figure 4.5: Comparison of the two previously described methods. In both methods, a segmentation mask was used to reject the background.

As shown in table 4.2, the SfM ground truth depth maps have a better accuracy metric than our CAD ground truth depth maps because we use a mask on our prediction to consider only the pixels for which we have a value. Due to the fact that the SfM depth maps contain significantly fewer pixels than our CAD version, we cannot determine which depth map performs better in 3D reconstruction by evaluating the depth map metrics. We applied the CAD GT depth map as a mask to our SfM GT depth map to determine which method produces the best values and has the

lowest error. Based on the results presented in Table 4.4, we determined that our CAD version mentioned in section 3.1.1.1 contains additional information about the structure, valves, and hot stabs, allowing the model to predict a precise output easily, as shown in Figure 4.5.

## 4.5 Dense 3D Semantic Mapping

For this evaluation, we used the test video described previously in this section and illustrated in Figure 4.1c; The video was divided into 20 frames per second rather than one frame per second for this application because 20 frames per second is the minimum frame rate required to run the video smoothly on SemanticFusion. We used video frames to predict segmentation and depth for all frames in order to generate data for SemanticFusion. All the videos of this section are available on our website[1].



(a) Dense 3D semantic mapping from the original ground truth depth maps.

(b) Dense 3D semantic mapping from the improved ground truth depth maps.

Figure 4.6: A sample of frames from our model that demonstrates the 3D reconstruction and object segmentation. The complete videos are available on our website.

The image results of our reconstruction presented in Figure 4.6 are aligned with the results presented in table 4.4, because our depth model has difficulties predicting complex objects with different shapes.

[1] https://paginas.fe.up.pt/ up201504257/dissertacao

While our improved method contains more detail of the structure than the original, both methods fail to reconstruct the structure, valves and hot stabs.

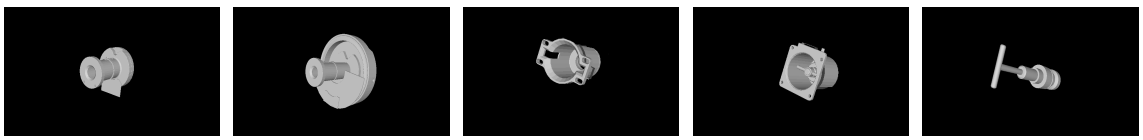
As a result, we can conclude that our method is incapable of generalizing complex objects in this environment, and thus that another approach focusing exclusively on the objects of interest is required.

## 4.6 CAD model alignment with images

We used only the test dataset 4.1c rather than the three presented in section 4.1 to simplify our evaluation; this eliminates scale ambiguity and incoherence poses along with different datasets.The dataset consists of a 1:44 minute video that generates 104 images at a frame rate of 1fps. It was divided into 89 training images (85%) and 15 test images (15%).

The CAD model of the entire structure is on the correct scale, measuring $9 \times 10 \times 6$ meters; the CAD model was subdivided into objects of interest: valves and hot stabs. The dimensions of the valves are $0.5 \times 0.7 \times 0.5$ meters or $0.5 \times 1 \times 0.5$ meters, depending on the type of the valve; the hot stabs are $0.05 \times 0.75 \times 0.3$ meters in size.

We used Mask R-CNN for instance segmentation, with the ResNet-50 as backbone and a gradient descent optimizer with a momentum of 0.9, a learning rate of 0.0005, and a decay rate of 0.1 after 100 epochs.



(a) Valve 1 (b) Valve 2 (c) Valve 3 (d) Valve 4 (e) Hot Stab

Figure 4.7: The CAD models that were used in this approach are shown above. There are four different types of valves and one hot stab.

We evaluated the bounding boxes, semantic segmentation mask and pose estimation for each object estimated by our model.

### 4.6.1 Bounding Boxes

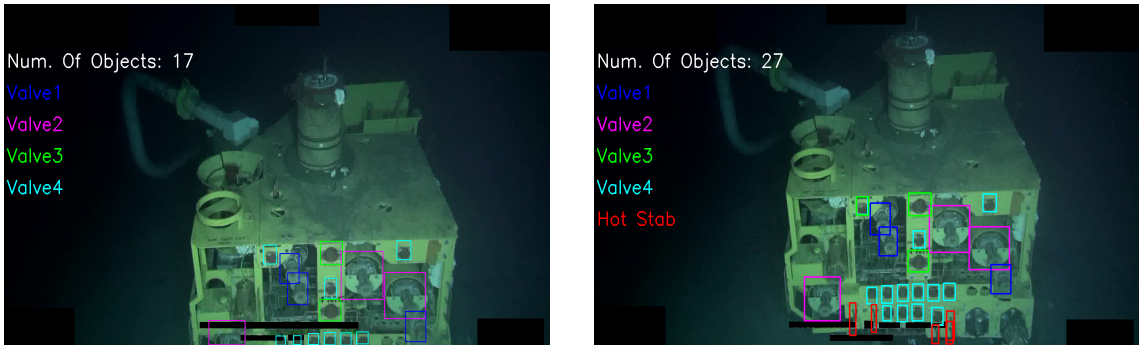



Figure 4.8: Two examples of bounding boxes estimation.

Table 4.3: Object detection results for all images.

| Object | IoU | Precision | Recall | F1 |
|---|---|---|---|---|
| Valves | 0.932 | 0.963 | 0.950 | 0.955 |
| Hot Stabs | 0.869 | 1 | 0.853 | 0.906 |

As illustrated in Figure 4.8 and Table 4.3, we obtained satisfactory detection results; this is critical for pose prediction, which is dependent on the detected object and its associated label class. Occasionally, our model predicts more than one bounding box for an object; to address this issue, we used a non-max suppression technique, which is typically used in object detection and aims to select the best bounding box from a collection of overlapping boxes.

### 4.6.2 Semantic Segmentation

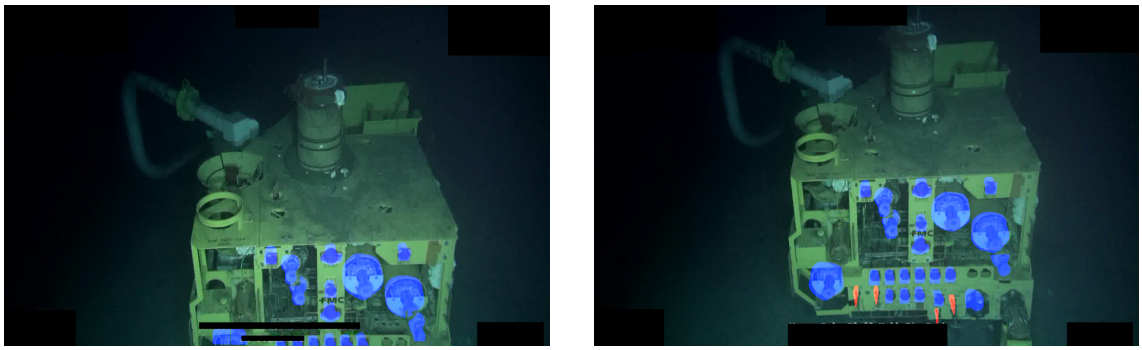


Figure 4.9: Two examples of semantic segmentation estimation.

Compared to the previous method, which had a high rate of false positives, the semantic segmentation estimation by Mask R-CNN is precise with a low rate of false positives. Our model does

not have a precise mask due to the small size and complexity of the hot stab, but it performs well in general. We assume that we have only one class for the valve; in contrast to the bounding box detection, where we had four classes for the valves, we assume that they are all identical, as we will not be using these masks to predict the pose.

Table 4.4: IoU metric evaluating the semantic segmentation results.

| **Object** | **IoU** |
|---|---|
| Valves | 0.871 |
| Hot Stabs | 0.474 |

### 4.6.3 Pose Estimation

The pose estimation model was trained over a period of 1000 epochs with a batch size of 2 and a learning rate of 0.0005 with a decay of 0.9 for every 100 epochs. The model was trained using degrees rather than radians for Euler Angles in order to obtain a more accurate result in terms of angle error rather than distance error. The angle values range are between 80 and 100, while distance values range between -20 and 10; when using the L1 loss discussed in section 3.2, the loss has a greater tendency to converge to the angle ground truth due to its higher values in comparison to the distance.

The model's metrics are summarized in Table 4.5; this model has a mean error of one meter due to the outliers, as shown in Table 4.5a. As expected, the angles are highly accurate; the maximum error is about 4 degrees. Again, due to the small size of hot stabs, the model is not very effective, in comparison to the valves.

Table 4.5: Results for pose estimation. All error distance measurements are in meters, while angle measurements are in degrees. The vector displayed in the column of angles has three axes [x,y,z] and represents the mean and standard deviation for each axis.

| | | Distance | | | | Angle | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Object** | **Number of Objects** | **min** | **max** | **mean** | **std** | **min** | **max** | **mean** | **std** |
| Valves | 281 | 0.116 | 6.740 | 0.813 | 0.962 | 0.00001 | 4.430 | [0.621, 0.635, 0.376] | [0.6264, 0.688 , 0.353] |
| Hot Stabs | 50 | 0.363 | 7.0294 | 2.665 | 2.228 | 0.003 | 2.676 | [0.782, 0.733, 0.434] | [0.671, 0.686, 0.351] |
| **Total** | **331** | **0.116** | **7.029** | **1.093** | **1.406** | **0.001** | **4.430** | **[0.645, 0.650, 0.385]** | **[0.636, 689, 353]** |

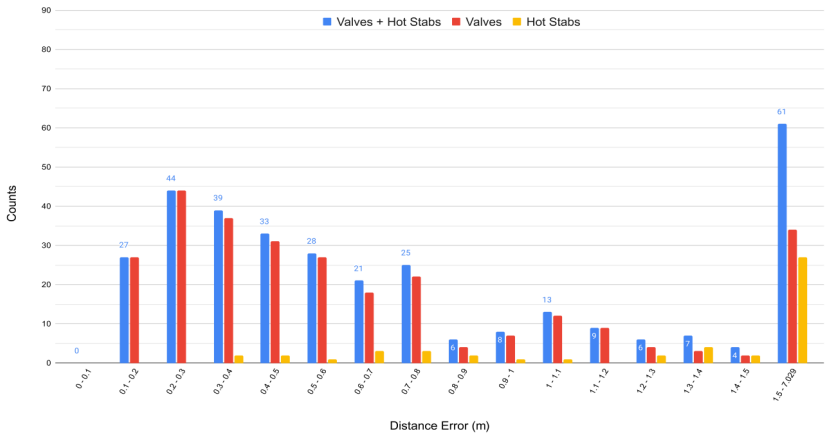
(a) Baseline with a high variance in the distance error.

| | | Distance | | | | Angle | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Object** | **Number of Objects** | **min** | **max** | **mean** | **std** | **min** | **max** | **mean** | **std** |
| Valves | 266 | 0.046 | 5.996 | 0.547 | 0.688 | 0.004 | 10.203 | [1.300, 1.076, 0.699] | [1.195, 0.845, 0.505] |
| Hot Stabs | 44 | 0.150 | 1.945 | 0.390 | 0.427 | 0.003 | 7.23 | [1.209, 1.124, 0.801] | [1.619, 0.889, 0.555] |
| **Total** | **310** | **0.046** | **5.996** | **0.525** | **0.602** | **0.003** | **7.203** | **[1.287, 1.084, 0.715]** | **[1.032, 0.853, 0.514]** |

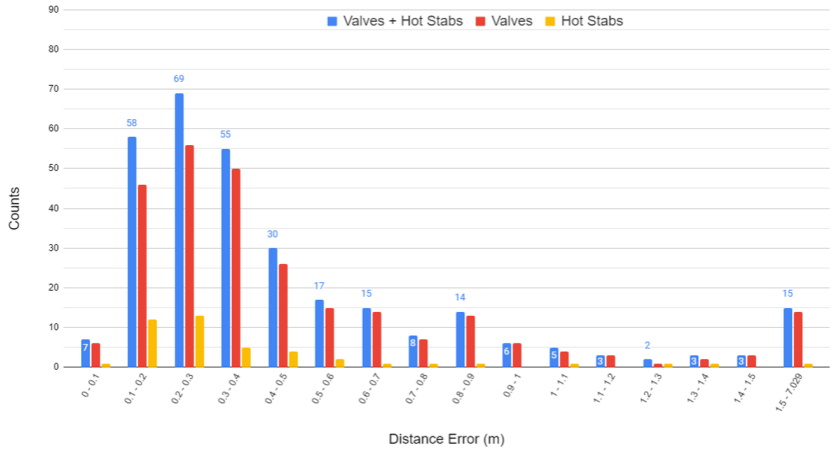
(b) Improved version that places more emphasis on the translation vector than angles, resulting in a reduction in mean distance error.

To attempt to improve our solution, we increased the score threshold to filter the boxes with a higher degree of confidence; we also emphasized the translation vector by increasing its weight in the loss function to obtain a more precise prediction. As expected, when we emphasize the

translation vector, the mean distance error decreases by 0.2m and the angle error increases by 0.5 degrees, as shown in Table 4.5b.




(a) Baseline with a high variance in the distance.It was used the Euler angles expressed in degrees without a score threshold.



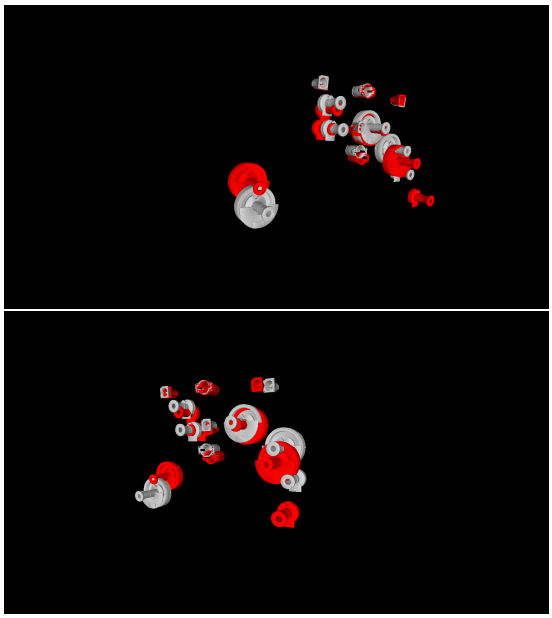

(b) Improved version that places more emphasis on the translation vector than angles. Using Euler angles are expressed in radians, and a score threshold of 0.5 is used.

Figure 4.10: Distribution of errors for each detected object. The sum of the two objects is indicated in blue. Hot Stabs have a high proportion of objects with errors greater than 1.5m, whereas valves have a higher proportion of objects with errors between 0.1m and 1m than with errors greater than 1.5m.

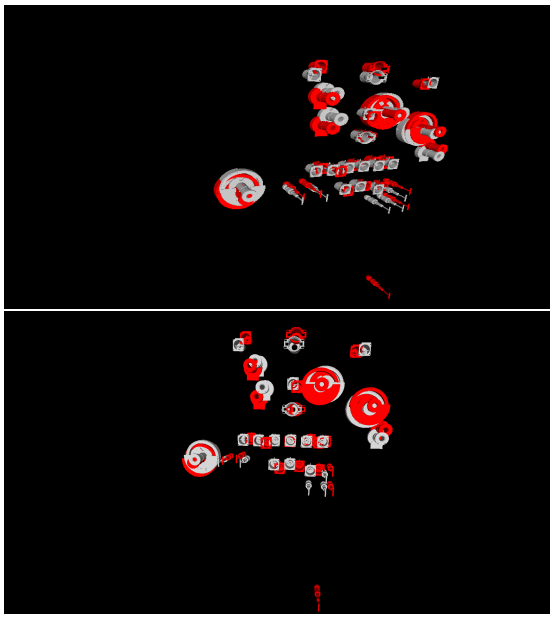
To help visualize how the model works, we depicted the distribution of errors for each detected object in Figure 4.10. The majority of valves have errors of less than 1m; on the other hand, the majority of hot stabs have values greater than 1m.

As illustrated in Figure 4.10b, the majority of valves are between 0.1m and 0.5m, which is a reasonable result in general. In comparison, we continue to have a high error value in hot stabs.

In Figure 4.11, two examples of the prediction pose are represented in red, and the ground truth pose is represented in grey. Figure 4.11b demonstrates a high precision in the distance error, but one outlier was detected.



(a) An example of a frame with only valves detected.



(b) An example of a frame with valves and hot stabs detected.

Figure 4.11: Two distinct frames, one with the prediction pose highlighted in red and the other with the ground truth pose highlighted in grey. Each object has the correct rotation, but some objects have an incorrect translation vector.

## 4.7 Comparison With The Baseline

The datasets must be the same in order to compare our proposed model to the baseline properly. As a result, we conducted an additional experiment using the 3D Semantic Mapping technique described in Section 3.1. This new experiment used the same dataset as the method proposed in Section 3.2; both used a 1:44 minute video shown in 4.1c divided into 89 images for training and 15 images for testing. We compared these two models by comparing their semantic segmentation and the final reconstruction.

| **Object** | **IoU** |
|---|---|
| Valves | 0.705 |
| Hot Stabs | 0.521 |

(a) Baseline.

| **Object** | **IoU** |
|---|---|
| Valves | 0.871 |
| Hot Stabs | 0.474 |

(b) Adapted Mask R-CNN.

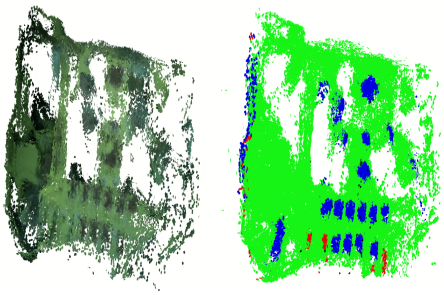
Table 4.6: Comparison between the semantic segmentation performance.

As illustrated in Table 4.6, the baseline method has a slightly better IoU for hot stabs, whereas our proposed method has a considerably better IoU for valves. Overall, we can conclude that the adapted Mask R-CNN performs better.

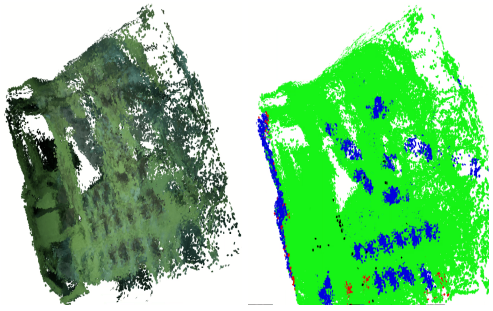
Table 4.7: Depth evaluation metrics of this experiment.

| **Abs Rel** | **Sq Rel** | **RMSE** | **RMSE log** | **a1** | **a2** |
|---|---|---|---|---|---|
| 0.129 | 0.03 | 0.739 | 0.043 | 0.87 | 0.97 |

As shown in Table 4.7, the baseline achieved an acceptable overall result for depth estimation.



(a) Result for frame 550.



(b) Result for frame 1500.

Figure 4.12: Reconstruction results. As shown in *a)*, the 3D reconstruction begins with a satisfactory reconstruction, but the reconstruction degrades in later frames, as shown in *b)*.

However, as shown in Figure 4.12 the method fails one more time in the 3D reconstruction, with a lot of over smoothing and imprecise position of the valves and hot stabs. The video of the 3D reconstruction with this method is presented on our website [2].

Finally, we can infer that our proposed method produced superior results, as combining depth map estimate into SLAM methods is highly imprecise, lacking in information, and vulnerable to semantic segmentation outliers.

[2] https://paginas.fe.up.pt/ up201504257/dissertacao

# Chapter 5

# Conclusions and Future Work

Depth estimation is a difficult task in the subsea environment, not only due to the physical properties of the water, but also due to a severe lack of datasets for deep learning methods.

This dissertation makes use of Abyssal's dataset, which contains 320 images with no depth map ground truth and semantic segmentation that has been manually annotated for each image.

Our initial approach to this problem was using a dense 3D semantic SLAM method, which joins depth map and semantic information predicted by our models. Unfortunately, this methodology failed to produce satisfactory results in 3D semantic SLAM due to the imprecision of the prediction depth maps. In comparison, the segmentation model produced an acceptable overall result. We attempted to improve the precision of the depth map model by optimizing our proposed ground truth depth maps; this improved the depth model's estimation but was still insufficient to provide a satisfactory 3D reconstruction.

Our second approach attempts to address these concerns by predicting the pose of CAD models for each image; since our problem was with predicting good depth maps, this methodology does not use depth maps. We modified Mask R-CNN by adding a new predictor to output the pose (rotation + translation) for each RoI. We achieved a high level of accuracy in predicting each object's bounding boxes and pose; for object detection, the valves and hot stabs had an F1 of 0.955 and 0.906, respectively. Additionally, for pose estimation, we obtained a mean distance error of approximately 0.5 meters for the valves and 2.6 meters for the hot stabs. Pose estimation is related to bounding box prediction; if the model predicts the incorrect class for the object or the bounding box of a non-object, the pose prediction will fail. Rejecting some outliers, we believe that it is possible to obtain a perfect 3D reconstruction and alignment using this method, allowing the ROVs to interact with the objects.

Due to the time constraints, the Mask R-CNN with pose prediction was developed using a single dataset. The main goal of future work is to evaluate the model's performance using all three datasets. Our model could be improved by increasing the number of images, which would allow it to generalize more confidently. We ignore occlusion and scale ambiguity in our environment, but it is critical to address these issues in the future. In addition, the proposed model is a single-frame representation; a step forward would be to resolve scale and depth ambiguities using multi-view

consistency constraints.

# References


[1] Chongyi Li, Chunle Guo, Wenqi Ren, Runmin Cong, Junhui Hou, Sam Kwong, and Dacheng Tao. An underwater image enhancement benchmark dataset and beyond, 2019. arXiv:1901.05495.

[2] Miao Yang, Jintong Hu, Chongyi Li, Gustavo Rohde, Yixiang Du, and Ke Hu. An in-depth survey of underwater image enhancement and restoration. *IEEE Access*, 7:123638–123657, 2019. doi:10.1109/ACCESS.2019.2932611.

[3] H Alemi Ardakani and TJ Bridges. Review of the 3-2-1 euler angles: a yaw-pitch-roll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK, Tech. Rep*, 2010.

[4] A. Garcia-Garcia, S. Orts-Escolano, S. O. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv*, pages 1–23, 2017. arXiv:1704.06857.

[5] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[6] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017. arXiv:1511.00561, doi:10.1109/TPAMI.2016.2644615.

[7] Md Jahidul Islam, Chelsey Edge, Yuyang Xiao, Peigen Luo, Muntaqim Mehtaz, Christopher Morse, Sadman Sakib Enan, and Junaed Sattar. Semantic segmentation of underwater imagery: Dataset and benchmark, 2020. arXiv:2004.01241.

[8] Shaojian Song, Jingxu Zhu, Xiuhua Li, and Qingbao Huang. Integrate MSRCR and Mask R-CNN to Recognize Underwater Creatures on Small Sample Datasets. *IEEE Access*, 8:172848–172858, 2020. doi:10.1109/access.2020.3025617.

[9] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video, 2017. arXiv:1704.07813.

[10] Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. *33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019*, pages 8001–8008, 2019. arXiv:1811.06152, doi:10.1609/aaai.v33i01.33018001.

[11] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham, 2014. Springer International Publishing.

[12] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. *arXiv*, pages 9785–9795, 2019.

[13] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. SemanticFusion: Dense 3D semantic mapping with convolutional neural networks. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4628–4635, 2017. arXiv:1609.05130, doi:10.1109/ICRA.2017.7989538.

[14] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X. Chang, and Matthias Nießner. Scan2cad: Learning cad model alignment in rgb-d scans, 2018. arXiv:1811.11187.

[15] Weicheng Kuo, Anelia Angelova, Tsung-Yi Lin, and Angela Dai. Mask2cad: 3d shape prediction by learning to segment and retrieve, 2020. arXiv:2007.13034.

[16] Filipe Marques, Filipa Castro, Manuel Parente, and Pedro Costa. A hybrid framework for uncertainty-aware depth prediction in the underwater environment. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 102–107, 2020. doi:10.1109/ICARSC49921.2020.9096196.

[17] Leijian Yu, Erfu Yang, Peng Ren, Cai Luo, G. Dobie, D. Gu, and Xiu-Tian Yan. Inspection robots in oil and gas industry: a review of current solutions and future trends. *2019 25th International Conference on Automation and Computing (ICAC)*, pages 1–6, 2019.

[18] Christian Mai, Simon Pedersen, Leif Hansen, Kasper Jepsen, and Zhenyu Yang. Modeling and Control of Industrial ROV's for Semi-Autonomous Subsea Maintenance Services. *IFAC-PapersOnLine*, 50(1):13686–13691, 2017. URL: https://doi.org/10.1016/j.ifacol.2017.08.2535, doi:10.1016/j.ifacol.2017.08.2535.

[19] Chong Yi Li, Ji Chang Guo, Run Min Cong, Yan Wei Pang, and Bo Wang. Underwater image enhancement by Dehazing with minimum information loss and histogram distribution prior. *IEEE Transactions on Image Processing*, 25(12):5664–5677, 2016. doi:10.1109/TIP.2016.2612882.

[20] Derya Akkaynak and Tali Treibitz. Sea-THRU: A method for removing water from underwater images. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:1682–1691, 2019. doi:10.1109/CVPR.2019.00178.

[21] Richard Szeliski. Computer vision algorithms and applications, 2011. URL: http://dx.doi.org/10.1007/978-1-84882-935-0.

[22] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[23] Nida Zaitoun and Musbah Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 12 2015. doi:10.1016/j.procs.2015.09.027.

[24] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979. doi:10.1109/TSMC.1979.4310076.

[25] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764 – 771, 2015. doi:https://doi.org/10.1016/j.procs.2015.06.090.

[26] Laurent Najman and Michel Schmitt. Watershed of a continuous function. *Signal Processing*, 38(1):99 – 112, 1994. Mathematical Morphology and its Applications to Signal Processing. URL: http://www.sciencedirect.com/science/article/pii/0165168494900590, doi:https://doi.org/10.1016/0165-1684(94)90059-0.

[27] S. Prince Mary, Ankayarkanni, Usha Nandini, Sathyabama, and S. Aravindhan. A Survey on Image Segmentation Using Deep Learning. In *Journal of Physics: Conference Series*, volume 1712, pages 1–22, 2020. arXiv:arXiv:2001.05566v5, doi:10.1088/1742-6596/1712/1/012016.

[28] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi:10.1109/ICCV.2015.169.

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018. arXiv:1703.06870.

[30] Alisha Sharma and Jonathan Ventura. Unsupervised learning of depth and ego-motion from cylindrical panoramic video. *Proceedings - 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality, AIVR 2019*, pages 58–65, 2019. arXiv:1901.00979, doi:10.1109/AIVR46125.2019.00018.

[31] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video, 2017. arXiv:1704.07804.

[32] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints, 2018. arXiv:1802.05522.

[33] Chaoyang Wang, Jose Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods, 2017. arXiv:1712.00175.

[34] Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose, 2018. arXiv:1803.02276.

[35] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation, 2019. arXiv:1805.09806.

[36] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006. doi:10.1109/MRA.2006.1638022.

[37] J. Li, L. Cheng, H. Wu, L. Xiong, and D. Wang. An overview of the simultaneous localization and mapping on mobile robot. In *2012 Proceedings of International Conference on Modelling, Identification and Control*, pages 358–364, 2012.

[38] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. doi:10.1109/TPAMI.2007.1049.

[39] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015. URL: http://dx.doi.org/10.1109/TRO.2015.2463671, doi:10.1109/tro.2015.2463671.

[40] Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, Oct 2017. URL: http://dx.doi.org/10.1109/TRO.2017.2705103, doi:10.1109/tro.2017.2705103.

[41] Carlos Campos, Richard Elvira, Juan J. Gomez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020.

[42] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry, 2016. arXiv:1607.02565.

[43] Thomas Whelan, Stefan Leutenegger, Renato Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. 07 2015. doi:10.15607/RSS.2015.XI.001.

[44] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics 2017 (TOG)*, 2017.

[45] John McCormac, Ronald Clark, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger. Fusion++: Volumetric object-level slam, 2018. arXiv:1808.08378.

[46] Martin Rünz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects, 2018. arXiv:1804.09194.

[47] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans, 2018. arXiv:1712.10215.

[48] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion, 2019. arXiv:1901.04780.

[49] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction, 2017. arXiv:1704.03489.

[50] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020. arXiv:1907.01341.

[51] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation, 2018. arXiv:1806.02446.

[52] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis, 2018. arXiv:1804.00650.

[53] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network, 2018. arXiv:1807.08563.

[54] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Hao Li, and Angjoo Kanazawa. PIFu: Pixel-aligned implicit function for high-resolution clothed human digitization. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:2304–2314, 2019. arXiv:1905.05172, doi:10.1109/ICCV.2019.00239.

[55] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. *IEEE Robotics and Automation Letters*, PP:1–1, 12 2016. doi:10.1109/LRA.2016.2634089.

[56] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine, 2017. arXiv:1708.05375.

[57] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. PanopticFusion: Online Volumetric Semantic Mapping at the Level of Stuff and Things. In *IEEE International Conference on Intelligent Robots and Systems*, pages 4205–4212, 2019. arXiv:1903.01177, doi:10.1109/IROS40897.2019.8967890.

[58] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart, and Juan Nieto. Volumetric instance-aware semantic mapping and 3D object discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044, 2019. arXiv:1903.00268, doi:10.1109/LRA.2019.2923960.

[59] Zak Murez, Tarrence van As, James Bartolozzi, Ayan Sinha, Vijay Badrinarayanan, and Andrew Rabinovich. Atlas: End-to-End 3D Scene Reconstruction from Posed Images. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12352 LNCS, pages 414–431, 2020. arXiv:2003.10432, doi:10.1007/978-3-030-58571-6_25.

[60] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual Multi-view Fusion for 3D Semantic Segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12369 LNCS, pages 518–535, 2020. arXiv:2007.13138, doi:10.1007/978-3-030-58586-0_31.

[61] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[62] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015. arXiv:1512.03012.

[63] Weicheng Kuo, Anelia Angelova, Jitendra Malik, and Tsung-Yi Lin. Shapemask: Learning to segment novel objects by refining shape priors, 2019. arXiv:1904.03239.

[64] Kevis-Kokitsi Maninis, Stefan Popov, Matthias Nießner, and Vittorio Ferrari. Vid2cad: Cad model alignment using multi-view constraints from videos, 2020. arXiv:2012.04641.

[65] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016. URL: http://dx.doi.org/10.1109/CVPR.2016.438, doi:10.1109/cvpr.2016.438.

[66] Hugo Miguel Miranda Barros. Aquaculture fish quality control using synthetic data. 2020.

[67] Ross Girshick. Fast r-cnn, 2015. arXiv:1504.08083.

[68] Yuanzhouhan Cao, Zifeng Wu, and Chunhua Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks, 2017. arXiv:1605.02305.