ACIT5930

MASTER'S THESIS

in

Applied Computer and Information Technology (ACIT)

May 2020

Applied Artificial Intelligence

Marker free 6D pose estimation for underwater vehicles with monocular RGB cameras

Håkon Weydahl

Department of Computer Science

Faculty of Technology, Art and Design

OSLOMET

Preface

I have always been fascinated by robotics for the interplay between mechanical design, software and electronics. All of which are aspects I enjoy separately. I knew I wanted to do an application oriented project and computer vision offers infinite depth. Coming from the specialization in artificial intelligence I also wanted to find a sensible application for AI methods. Machine learning has been hyped massively for the last decade and end-to-end neural nets are often presented as the solution for everything. In research novelty is king, even though the result might not be efficient, explainable or practical. With this in mind I have tried to design a pragmatic solution balancing model-based methods with machine learning tools.

I would like to thank Vahid Hassani for guidance and advice during this master thesis project. I will additionally thank Håkon Teigland for the initial problem statement and help along the way. Finally I want to thank my fellow students at Oceanlab for assistance with experiments and helpful discussions.

Håkon Weydahl
Oslo, May 2022

Abstract

Unmanned underwater vehicles (UUV) are important for industry, science and exploration. A key step in increasing their autonomy and lowering costs is performing 6D pose estimation on objects it interacts with. This report details pose estimation methods and the process of creating a dataset for training neural networks. Footage captured in the sea using a ROV is annotated with segmentation masks and poses. This is made public along with the tools developed to create it. The dataset appears to be unique in its kind, similar datasets were made in pools and also not published. Further, the dataset is used to train Mask R-CNN for 2D object detection.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

There is a large and growing number of inspection and interaction tasks in underwater environments [47]. Fish farming, public works, scientific discovery, the military and petroleum industry all rely on underwater vehicles. For instance, oil wells are now regularly established at depths far beyond the range of human divers. Underwater vehicles are used to inspect and maintain aquaculture fish cages [2]. Scientists even intend to search for life in extra-terrestrial ice-covered oceans and lakes [77]. These operations rely entirely on underwater vehicles.

These vehicles can be manned or unmanned. Unmanned vehicles are either autonomous or remotely operated. Remotely operated vehicles (ROVs) are tethered to a subsea installation or dedicated surface ship, and operated by a pilot. Then there are various degrees of autonomy between this and a vehicle operating without a tether or any communication with the surface. A hybrid underwater vehicle (H-ROV) might be configurable for both tethered and untethered operation.

Fully autonomous underwater vehicles (AUVs) are already used extensively for surveillance, mapping the ocean floor, monitoring water or wildlife [32]. And there have been several attempts to increase autonomy for intervention tasks since the 1990s [56, 60, 69, 70]. Some examples are connecting to or manipulating subsea panels or finding and retrieving objects from the seabed.

Although a ROV might still be necessary for the complete operation, certain tasks may be automated and consequently can reduce pilot fatigue. The pilot can issue tasks and supervise the ROV instead of controlling every action. Such as changing the tooling on the end effector of the robotic arm. If an operation could be done completely autonomously there would also be no need for a tether. ROVs rely on the tether and this necessitates a large and expensive supply ship with a tether management system. This expensive ship is manned by a crew who are all at risk in adverse weather or when operating near ice fields. Increasing autonomy has the potential to both decrease costs and increase safety.

But to interact with an object it is a prerequisite to determine the distance to and orientation of that object. This problem is called *pose estimation*. Previous UUV projects have developed custom pose estimation methods that are very particular to the problem they solve in their experimental demonstration. Meanwhile pose estimation for objects in air is a very active field of research, driven by interest in autonomous vehicles, augmented reality and collaborative robots. This project seeks to combine pose estimation and tracking methods using template matching to make a system capable of manipulating a wide variety of different objects in subsea operations.

Machine learning methods have been steadily out-competing traditional methods for the past decade. The state-of-the-art pose estimation methods today are neural networks [27]. However, these require massive amounts of data to train and there are few or none publicly available datasets for underwater pose estimation. Although some projects have applied machine learning techniques to UUVs [46, 49], these datasets were created in a pool environment and not made public.

The goal of this project is to create a dataset for the purpose of training neural networks to perform pose estimation. The dataset should be created in a way that does not require prohibitively expensive laboratory equipment or novel specialty sensors. Further it should be created in open water rather than a pool, as the poor visibility in a natural underwater environment must be regarded as the main challenge.

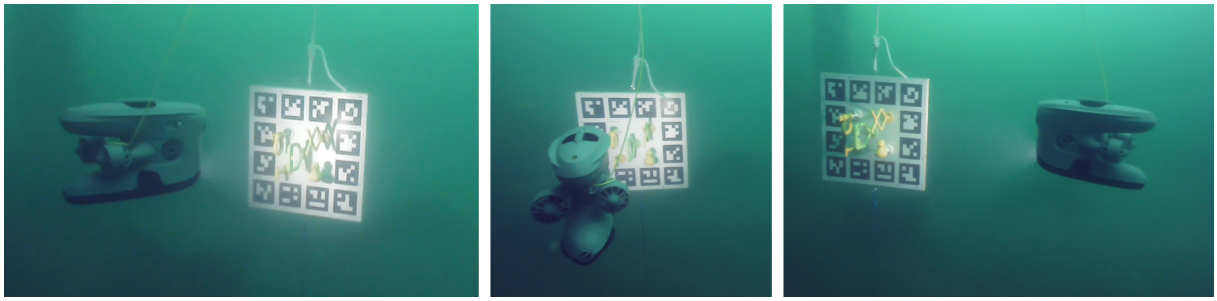


Figure 1.1: The Blueye X3 observed from a secondary ROV.

1.2 Operationalization

Figure 1.2 can be used to illustrate a typical use case. It shows a wellhead panel used in the oil and gas industry. ROV pilots need to interact with the valves and hot-stab connectors using the mechanical gripper very skillfully. Because of water turbidity, distortions and lack of depth perception this is very difficult. The goal here would be to use a simple color camera to infer the relative position and distance from the panel. This information could in turn also be used to automate the entire task.

Imaging sonar offer unique capabilities underwater, where visibility if often limited. No doubt why many marine mammals rely on their biosonars. But sonar still lack the spatial

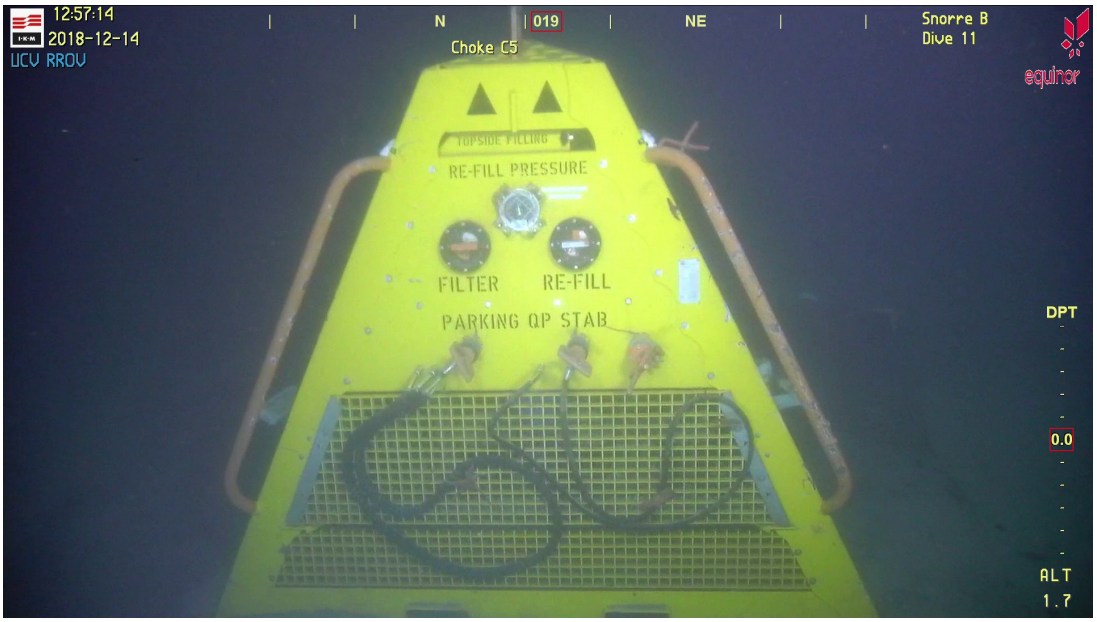


Figure 1.2: Subsea panel. Image courtesy of IKM

resolution necessary for manipulation tasks. And acoustic sensors are also vulnerable to noise in cluttered environments.

A pose estimation method for operational use should not rely on *fiducial* markers of any kind. A fiducial is something placed in an image to act as a scale or point of reference. In computer vision there are methods such as AprilTag [50, 78] and ArUco [20, 63] that utilize 2D barcodes as fiducials. These methods can exploit the known geometry of the barcode to estimate rotation and translation. Fiducials are commonly used in robotics for pose estimation in navigation and manipulation tasks. But placing fiducials on everything is impractical and sometimes impossible.

This project also only takes into consideration passive optical sensors, as opposed to active. Active sensor can be fiducials emitting light, or cameras emitting light to work as a 3D sensors. These use LiDAR (Light Detection And Ranging) or structured light to capture 3D shapes directly. LiDAR has become widespread in robotics in recent years and its use underwater have already been explored by several projects [24, 44, 52, 62, 65, 69]. There also appear to be commercial LiDARS available, but these appear to be intended for surveying. Passive sensors are a cheaper and more elegant solution, and should be perfectly sufficient for many tasks.

The focus is further constrained to methods using a single camera to the estimate the pose of an object. Previous projects attempting to use stereo cameras underwater were unsuccessful, despite significant effort in the design of the camera system [13, 38, 39, 41]. This project aims to test the techniques on ROV equipment without any special camera setup.

There is also an important distinction between pose *tracking* and pose *detection*. Tracking assumes a continuous video stream where the object moves little from frame

to frame. These methods are fast, but rely on being initialized with a pose, and must re-initialized every time tracking is lost. On the other hand, detection finds the pose from a single image. Though these methods also sometimes rely on additional information such as a bounding box. Detection is much more computationally heavy and it seems necessary to use tracking and detection methods complementary to have a robust realtime system.

The system should primarily rely on a more efficient pose tracking algorithm and only use pose detection whenever tracking is lost.

This report will evaluate methods that assume a 3D model of the desired object is available. Having to rely on a 3D model might seem unpractical, but in reality most objects an UUV might interact with are man made and created with computer-aided design. In the reverse case, 3D scanning is now commonly available as a service. These can create a model of pretty much any object with a normal surface, but might struggle with highly reflective or opaque objects.

1.3 Research contribution

- There are to my knowledge no datasets for underwater pose estimation publicly available. Similar datasets do exist, but are not published, and are made in indoor pool environments. This is captured in the sea at two different days and locations for varying visibility levels. This project details the making of such as a dataset, published free to use at https://github.com/wehak/ROV-handle-DB (GNU General Public License v3).

  - The dataset comes ready to use annotated in the COCO format, appended with ground truth pose.
  - It includes all the raw footage and calibration images if any researchers wish to utilize this.
  - All the tools used to create the dataset from the raw footage is published, free to use for anyone who wants to use a similar approach in creating their own dataset.
  - Finally it includes a Mask R-CNN model trained for object detection on the data.

- The report includes a unique review of how intervention-AUV projects in the last decade have done pose estimation.

- It outlines an architecture for at pose detection and tracking system combing existing methods. It is intended to work for any kind of object with a CAD-model available.

1.4 Overview

Chapter 2 highlights some general theoretical concepts relevant for the methods used later. Section 2.1 describes the consequences water have on robot perception. In section 2.2 some basic computer vision concepts are introduced. Further 2.3 present some mathematical representations of motion and rotation, along with methods to manipulate them. Lastly evaluation metrics for 2D object detection is discussed.

Chapter 3 maps out relevant research. Part 3.1 reviews pose estimation methods used in water and UUV projects that have tried to perform autonomous intervention tasks. In section 3.2 general pose estimation methods are discussed. Finally 3.3 present pose datasets that have been created specifically to train and benchmark neural nets for pose estimation.

Chapter 4 contains the methodology. Part 4.1 presents how I create a dataset for pose estimation in water. Part 4.2 detail how it is used to train a neural net. Lastly 4.3 describes details on the implementation of the methods.

Results are presented in chapter 5 and further discussed in chapter 6. Chapter 7 sums up the findings.

Chapter 2

Background

2.1 Constraints in the underwater environment

2.1.1 Imaging and computer vision

The methods for pose estimation discussed in this paper primarily rely on regular cameras. Pose estimation in air is a well studied subject in computer vision, but the same algorithms work less reliably in water. Water affects light differently than in air. Generally, water absorbs light very quickly, and different wavelengths are affected unevenly. As seen in Figure 2.1 green and blue travels through water easier than other wavelengths. This in turn results in the characteristic green and blue hue seen in most underwater images.

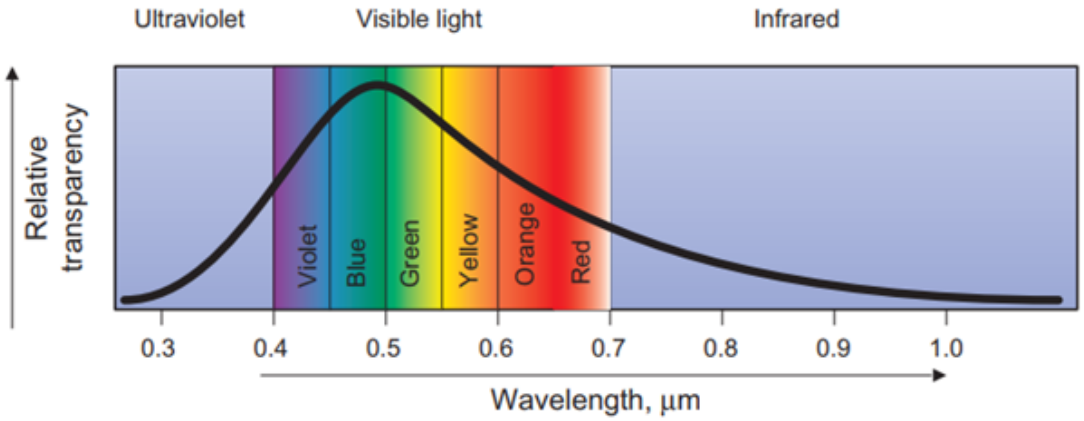
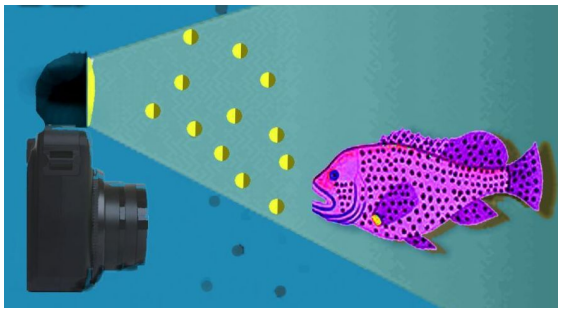


Figure 2.1: Water transparency as a function of wavelength [11]

Water will also contain a varying amount of suspended solids. These particles will scatter light moving towards the camera in an effect called *forward* scattering. This creates a blurring effect that reduce contrast, causing a varying degree of turbidity. In deep water, larger particles known as "marine snow" (Figure 2.2b) may partially occlude features and create a backward *scatter* effect. At these depths all natural light is gone, and vehicles rely on artificial lighting coming from the machine itself. This light is reflected in the marine snow, as illustrated in both figure 2.2a and 2.2b.



(a) Backscatter effect [79]

(b) Marine snow [67]

Figure 2.2: Noise in underwater images

Underwater images are also affected differently by refraction, as light travels more slowly in water than in air. Light passing from air into the lens has a different refraction index than light traveling from water into the lens. The camera housing on a deep water vehicle may be several centimeters thick and made of different materials. This creates a distortion that is largely range dependent and cannot be compensated for by using simple camera calibration [34].

Popular RGB-D cameras like the Microsoft Kinect is widely used in robotics, but does not work underwater. It relies on structured light in the infrared. As seen in figure 2.1, these wavelengths are quickly absorbed in water.

2.1.2 Communication

High-speed communication based on electromagnetic radio waves has long since matured to the point where it is taken for granted. Unfortunately it has never worked very well underwater. Attenuation in water is very high and its range is extremely limited [51]. Instead acoustic telemetry is the norm. It has a range up to 20 km, is energy efficient and low cost. On the downside latency is high and bandwidth very low. Data rates may only reach upwards of 38.4 kb/s and is vulnerable to a range of different environmental conditions [11]. This is totally insufficient for remotely controlling a robot via video. Instead ROVs rely on a tether than can both supply power and provide a high-speed data link.

2.1.3 Localization

Global navigation satellite systems (GNSS) is not directly available because of the high attenuation of radio waves in water, but a position can be established with the aid of acoustic positioning systems. The most common systems are long baseline (LBL), short baseline (SBL) and ultra-short baseline (USBL) [11].

USBL allows two vehicles to establish their position in relation to each other without deploying any beacons. A transceiver on one vehicle emits a ping and cause a transponder

on the other vehicle to respond. By measuring the time from emitting the ping and receiving a response, the transceiver can determine the range. The direction the response is coming from can be measured with an array of transducers on the transceiver. If one of the vehicles is a surface vehicle with a GNSS system, this can be used to indirectly infer the position of a UUV with a transponder [11].

LBL and SBL both rely on a mesh of transceivers and determine the angle by triangulation. SBL transceivers are wired together to a central hub, while LBL transceivers work independently of each other. LBL can both be placed on the bottom around the worksite or floating as bouys with GNSS.

The downside of acoustic positioning is the cost and their vulnerability to disturbances. The speed of sound is affected by temperature, salinity and pressure, and the ocean consists of layers of water with different characteristics. Waves are also deflected off objects and cause multipathing, which would be a problem for any ROV navigating an environment crowded with subsea structures.

In addition to aiding in manipulation of subsea panels, pose estimation can be a valuable as a tool in localization. Robot localization is the problem of orientation and position in relation to the world. Orientation is the rotation around the pitch, roll and yaw axes. This can be found using a magnetometer and accelerometer. Finding the position is much trickier. Establishing an exact position in relation to a known subsea structure could correct for drift or greater uncertainty in other methods.

2.2 Computer vision

2.2.1 Photogrammetry

By applying photogrammetric techniques, features can be used to extract information about the 3D structure of an object using 2D images. The pinhole camera model is used as a basis. Light rays from the outside are assumed to pass through a infinitely small focal point and onto a sensor. The model explain how 3D points in the world are mapped onto the 2D image plane. The parameters of the model are divided into *intrinsic* and *extrinsic* parameters.

The cameras intrinsic properties describes the lens distortion, how skewed the image is, the cameras focal length and the pixel size in world units. This is acquired by performing camera calibration using a checkerboard pattern.

The extrinsic parameters describe the position and orientation of the camera in the world coordinate system. The intrinsic parameters describe how 3D points are mapped into 2D points in the image plane. This process is lossy however and cannot easily be reversed. The problem of finding the camera extrinsics based on camera intrics, the image and assumptions of the 3D world is known as pose estimation.

2.2.2 Pose estimation

With pose estimation the goal is to find the camera extrinsic properties, that is the rotation and orientation of the camera in the world coordinate frame. In robotics the practical problem is usually finding the pose of an object in the coordinate frame of the camera, but this reversal is a trivial transformation. Pose estimation is crucial to enable robots to interact with the environment. It can also be used to create an augmented reality experience where 3D shapes are overlayed on the ROV pilot's screen. These shapes can be be annotated with properties such as distance to or the size of the target.

The classic technique is called perspective n-point and given at set of images it can determine the pose of the camera with a little as three *image features*. That particular variation is very brittle however, and in practice Szeliski [72] instead recommend using perspective six-point and refine that with an outlier detection algorithm such as RANSAC.

2.2.3 Image features

Given a scene with good features to track, PnP and RANSAC works well. But common feature detectors generally work less reliably under water [3]. Intuitively, *features* are things that sticks out in an image and can act as a point of reference when seeing the same scene from different angles. Features are a prerequisite for pose estimation and stereo vision. The ideal feature is a piece of information that is unique and tied to a specific thing in the image. Given a set of images of the same scene, the feature should be identifiable in all of them.

Features that stand out to humans are typically large and complex objects and shapes; A mountain silhouette, a door or a banana. Computer vision algorithms rely more on tiny patches of the image or edges and corners [72].

The feature *detector* is the specific method of finding features. A simple keypoint detector might be looking for an image patch with an extreme maximum. The suitability of the feature would then be evaluated by comparing its quantified properties with its neighborhood. Its suitability is primarily proportional to its uniqueness [72].

Other properties is invariance to changes in scale, rotation or luminosity. This would depend on the type of detector used. Detectors also consider different classes of features, the most common being keypoints, corner, edges or ridges.

Some common detectors are SIFT, SURF, ORB, Hessian or difference of Gaussian. These features are "hand-crafted" in that they are not machine learning methods. The algorithm relies only on the information in the image. Machine learning feature descriptors learn to extract features from the training data, and applies this experience to the interpretation of a new image. A convolutional neural net (CNN) is any type of neural net that use convolution in its layers. Convolution is a mathematical operation on two matrices where a small matrix called a kernel slides or convolve across the bigger matrix. The output depends on the properties of the kernel and parameters to the convolution

operation, but generally it will reduce a large matrix into a smaller matrix. The purpose is typically to compact an image to a "feature map" that contains higher level features instead of just individual pixels. Convolution has been used for a long time in computer vision, for instance for performing Sobel or Canny edge detection. The difference in CNNs is that the algorithm learns to adapt the kernel itself. Dai et al. [14] compare features created by different CNNs and find they outperform traditional hand-crafted feature descriptors.

Structure from motion is a related technique for finding the 3D position of features. This can be used to create a sparse 3D reconstruction of an object when viewed from multiple angles.

Stereo vision

Stereo vision works because an object imaged by two cameras with overlapping views will be slightly offset in each, and this offset is inversely proportional to the distance from the cameras. Techniques that match the type of features described above are called sparse stereo matching. The converse is dense stereo matching that match pixel-to-pixel. Sparse stereo has some utility if the desire is to limit the number of matches to more robust features [72]. This could for instance be edge-type features, that are more prominent in underwater images of subsea structures. Subsea structures are often without texture and have a uniform color, which could make matching keypoint features difficult. Yet another alternative is image segmentation based techniques.

Stereo-vision can be used to gauge depth, but is generally harder to use under water. It will depend on the quality of the features it is tracking. Even if two identical cameras are calibrated perfectly, turbidity or floating particles may affect light hitting each camera differently.

The output of stereo matching is a 2D depth map of the original image. Given many images of the same scene, multi-view stereo techniques can be used to create dense 3D models of objects [72]. The 3D representation can be a point cloud, mesh or voxels. These are obviously valuable for robot navigation, but also more complex and computationally intensive than basic stereo vision or feature-based pose estimation.

2.2.4 3D sensors

LiDAR ("Light Detection And Ranging") and structured light sensors are capable of detecting 3D shapes directly, instead of restoring them from 2D images.

LiDAR works by sending a laser beams at a point in the scene and measuring the time it takes to return. These are used in the MERBOTS (section 3.1.4) and SEAVENTION projects (section 3.1.8).

The sensor used in SEAVENTION is called UTOFIA and is developed specifically for underwater use [44]. It integrates data from both a LiDAR and a regular camera. The range measurements are range-gated, meaning laser beams that bounce off particles too close to the camera are ignored. This has the effect of filtering out backscatter close to the camera. By increasing both laser beam intensity and the range-gate parameter, the instrument is capable of looking further ahead. The image from the regular camera is also enhanced using the depth map created by the LiDAR.

Structured light techniques project geometric patterns onto the scene and infer the shape of objects by how the shapes are distorted. Rishold et at [62] explores its use for underwater robotic perception.

Structured light generally offers greater spatial resolution and higher framerates than LiDAR alternatives, but at the cost of lower range.

2.3 Describing poses

There are many different mathematical representations of motion, each with their pros and cons. All methods described in this project describe rigid bodies. A rigid body can be a geometric primitive such as a cube or sphere, but also a complex 3D model. The point is that is does not take deformation into consideration. Rigid bodies would be a good starting point for viewing plants and creatures as well, as these could be modelled as several rigid bodies linked by joints.

Axis-aligned orientations are most intuitive when describing an object's orientation relative to the camera frame. These Euler angles α, β and γ correspond to rotation around the x, y and z axis [74]. When describing 3D motion these are typically called roll, pitch and yaw.

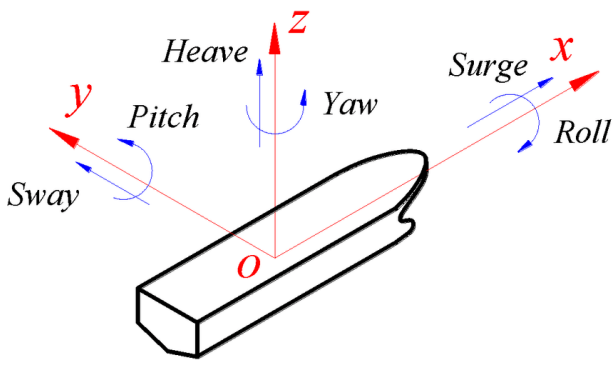


Figure 2.3: Euler angles in maritime terms

When the rotation is considered one axis at a time, rotation around each axis can be describes by the matrices below.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) \\ 0 & sin(\alpha) & cos(\alpha) \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) \\ 0 & 1 & 0 \\ -sin(\beta) & 0 & cos(\beta) \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 \\ sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler's rule tell us that the dot product of these three matrices correspond to a rotation matrix. The rotation matrix is a transformation from one coordinate system to another. A rigid body in an image can be seen as a coordinate frame with a rotation around the x, y and z axis, along with a position expressed in x, y and z coordinates. These 6 degrees of freedom define its pose in the reference frame of the camera. Mathematically the rotation is expressed as a rotation matrix R.

$$R(\alpha, \beta, \gamma) = R_x(\alpha)R_y(\beta)R_z(\gamma) \in \mathbb{R}^{3x3}$$

This describes one frame's rotation is relation to another, but not their relative position. Position or translation is expressed as a vector $t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T$. Rotation and translation is combined into a *homogeneous transformation matrix* T that act as a transform of both rotation and translation.

$$T = \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{4x4}$$

Homogeneous transformation matrices and rotation matrices typically have a subscript to tell us for which frames they can transform. If it transforms our model $\mathcal{M}$ in the camera $\mathcal{C}$ frame the subscript would be T_{cm}. Rotation and transformation matrices have the happy property that $T_{cm}^{-1} = T_{mc}$ meaning the choice of reference is usually arbitrary [74].

If describing motion a homogeneous transformation matrix captures the transform from one frame to the next. This link means that the trajectory is a sequence of transforms

$$T_{cm}(t_k), \quad t_k \in \mathbb{R}, \quad k = 0, ..., l,$$

And because transformation matrices have the other happy property that

$$T_{ab}T_{bc} = T_{ac}$$

The trajectory can be described as

$$T_{cm}(t_l) = T_{cc}(t_l, t_{l-1})...T_{cc}(t_3, t_2)T_{cc}(t_2, t_1)T_{cm}(t_1)$$

And give us the transform at any timestep k.

While all these happy properties make the homogeneous transformation matrix a convenient representation, it is not the most efficient computationally. The matrix has 16 numbers, but they are constrained such that it really only has 6 degrees of freedom. This matters in real time systems. Instead rotation can be represented as twist $\mathcal{T}$. $\mathcal{T}$ is a vector with angular w and linear v velocity in all three axes.

$$\mathcal{T} = \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} \omega_1, \omega_2, \omega_3, v_1, v_2, v_3 \end{bmatrix} \in \mathbb{R}^6$$

Rotation is also described Rodriguez vectors (axis-angle representation). Here a three part vector describe the magnitude of rotation around a given axis. Rotation can also be represented as quaternions. Quaternions is a type of complex numbers system and is defined at the quotient of to vectors in 3D space.

Axis-angle vectors and quaternions are not very intuitive representations of rotation, but more efficient than the homogeneous transformation matrix and more practical than Euler angles. For Euler angles rotating around X, Y and then Z yield a different rotation than Z, Y and then X. It can also suffer from singularities or "gimbal lock" where a degree of freedom is lost.

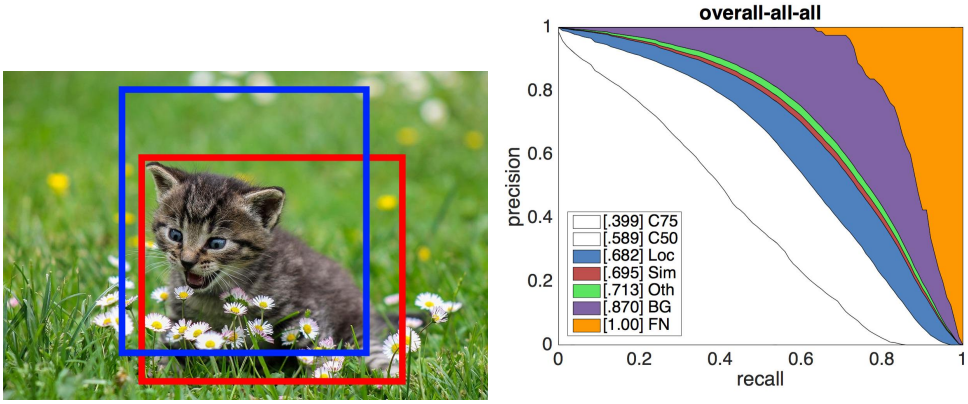
2.4 Accuracy metrics

The most fundamental metrics in machine learning are precision and recall. Precision is how many of you prediction are accurate. Recall is how many of the positives you are able to identify out of all actual positives.

$$Precision = \frac{TP}{TP + FP} Recall = \frac{TP}{TP + FN}$$

In the case of 2D object detection the goal is to draw bounding boxes around classes of objects in an image. Here accuracy is measured by the Intersection over Union (IoU). In figure 2.4a the blue square represent the predicted bounding box and red is the ground truth of the category "cat". The union is the area of both squares, while the intersection is the overlapping area. A perfect prediction would give a IoU of 1.0. In benchmarking

0.5 is generally the threshold between a true or false positive.



(a) IoU with ground truth in red and prediction in blue

(b) Example of a precision-recall curve (COCO website)

Here precision is the fraction how many of your classifications are correct. Recall is a measure of many you did right out of all possible classifications. There is a trade-off between the two. If you didn't care about false positives, had a low classification threshold and classified everything as positive then recall would be perfect, but accuracy very low. If you classify some as negative for accuracy, then you risk recall decreasing. In a good model both precision and recall are high. If plotted against each other the area under the curve should be as large as possible. This is the Average Precision (AP) and is calculated of each category. When averaged over all the different categories you have the mean Average Precision (mAP) which the most important metric for image classifiers.

Chapter 3

Literature review

3.1 Pose estimation in water

AUVs are already hugely important for gathering data in underwater environments. In oceanography their low cost and high endurance makes it possible to record very long time-series data that would otherwise be too expensive. Scientists can monitor areas and study the changes that would not be seen in data from a single mission [32]. This role could be enhanced further if they are able to interact the environment. Interaction is also the key for increased in use industrial application such as aquaculture or oil and gas.

Ridao et al. [59] summarize the earliest development of intervention AUVs (I-AUV). It began in the 1990s with projects focusing on development of the manipulator arm. Early examples are ODIN and OTTER. Most notably AMADEUS [8] and UNION [61]. AMADEUS was focused on developing a mechanical gripper and its control framework. UNION breaks down an imagined intervention mission into separate tasks and prototypes solutions to each. Cameras are discussed alongside other sensor types like laser and sonar. For instance, they simulate a scenario where the robot automatically follows a pipe using computer vision. Other computer vision techniques are mentioned as well, but they do not demonstrate any uses.

3.1.1 SWIMMER (1999-2001)

SWIMMER [18] represent the first field trial of a I-AUV. Their vehicle could autonomously navigate to and dock with a subsea structure. The SWIMMER vehicle carried a regular ROV on board and acted as the tether management system once docked. The dock would have a fibre connection and the pilot could control the ROV from onshore. Once deployed it did not need a supply vessel and was intended as a cost-saving concept. Long-range navigation was done using a mix of dead reckoning and acoustic beacons. Perception for the final docking maneuver relied on a custom sonar system and utilized no computer vision.

3.1.2 ALIVE (2001-2004)

SWIMMER's sister-project ALIVE performed a docking and valve-turning operation [17]. Their intention was to dock autonomously with a panel not specifically designed for AUVs. They used sonar and grayscale video to navigate the robot to within 10 cm of the panel and clasped onto rails intended for ROVs (figure 3.1). Sonar is used for pose estimation in the initial approach to the panel and video is used for the final part.

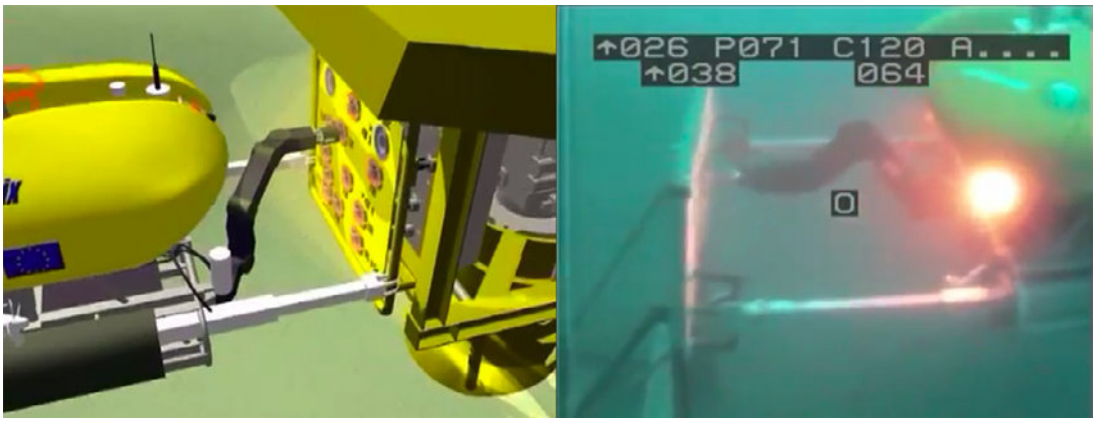


Figure 3.1: ALIVE docking and manipulation. Image from Zereik et al. [81]

Image features are extracted in three steps. First it performs Canny edge detection. The edges are then used to create a contour map. Third, circles are detected on the contour map. The circles represent the valves on the panel and is used for the pose estimation. The data is repeatedly matched with a CAD model of the panel until error is minimized. The authors present data showing that the method appears to be robust within 1.25-1.5 meters of the target. They mention that this method has "constraints", but nothing specific other than the limited field of view of 100 degrees. Finally, this pose estimate is fused with the estimate from the sonar module.

3.1.3 SAUVIM (1997-2009)

SAUVIM was a separate I-AUV project that performed a sea trial of a search and recovery mission, locating an object on the ocean floor and hooking a cable to it [43]. Their computer vision pipeline first performed Canny edge detection on a color image and then circle extraction on top of this. Two white balls were attached to the target and acted as fiducials. Because their size was known the pose could be estimated based on the size and position of the circles

3.1.4 GIRONA500

GIRONA500 is a I-AUV platform rather than a specific project. The platform appear to have spawned from the RAUVI project and capabilities have been added by later projects such as TRIDENT, TRITON and PANDORA [59]. GIRONA500 weighs less than 200 kg

and is much lighter than the UUVs used in previous projects. The ALIVE UUV weighed 3.5 ton and SAUVIM 6 ton. GIRONA500 also utilized an almost ready made manipulator arm, rather than going through a separate development process like AMADEUS. Most notably, the visual perception system of GIRONA500 is more elaborate than any of the previous projects. It is developed with three types of missions in mind.

1. Subsea-panel docking and fixed-base manipulation. The UUV uses only vision to track the panel and valves. The mission starts with the panel inside the field of view of UUV. Detection happens by template matching where the algorithm knows *a priori* what the panel looks like. Features are extracted from both the template and video stream and mapped to each other. Their relative pose is estimated using this mapping. Specifically they use the ORB feature extractor [64].

2. Free floating manipulation by learning. This was demonstrated by Carrera et al. [7] who perform a valve-turning operation in a pool environment. The authors adapt a learning by demonstration algorithm which learns from observing a ROV pilot. To detect the subsea panel they appear to use the same visual perception method as for fixed-base manipulation.

3. Search and recovery of an item. The UUV must survey an area for a certain object. Then it performs a maneuver to recover the object from the seafloor.

The visual perception system is implemented using ROS as middleware. It is a separate node publishing various topics. So is arm control along with UUV navigation, guidance and control. They all run independently as nodes and communicate using topics. This also means the visual system can run on separate hardware and improve system speed.

RAUVI (2009-2011)

The RAUVI project demonstrated an item recovery operation similar to SAUVIM. RAUVI however also added a surveying stage to the operation. This was then tested in a pool environment [58].

In the survey stage of the operation the goal was to locate the target on the ocean floor. The robot would move in a search pattern over the target area. It had two partially overlapping cameras in the front, one facing forward and the other downward. Only the downward facing camera was used to aid with the navigation. Using a novel algorithm it created a photo mosaic as the robot went along [19, 37]. Features were extracted from each frame and tracked from image to image so they could be stitched together. This was particularly useful whenever the robot crossed its previous path. Then the features could be used for a global localization estimate and correct for drift in the navigational system.

The cameras were also used to track the target in the intervention stage of the operation. The object they retrieved in their demonstration was a bright red box

resembling a flight data recorder. The targeting system was intended to be able to detect by texture, feature or color, but what they demonstrate is detection by color. For station keeping, a visual odometer module is able to extract features using SURF that act as a frame of reference.

In future work they mention their visual perception system cannot control for all degrees of freedom in station keeping. The visual localization also lacks full integration with inertial and acoustic systems.

TRIDENT (2010-2012)

RAUVI evolved into the TRIDENT project. The intervention-capable AUV was paired with an autonomous surface craft (ASC). The ASC would carry the I-AUV to site and assist in surveying the ocean floor [66]. In TRIDENT the visual perception system also appear to have matured into the "Fugu-f" system. Fugu-f is a system intended to be a complete visual navigation suite for UUVs performing a search and recovery mission [3]. It uses dual stereo cameras and is adaptable so to be mounted on different types of vehicles. It performs target pose estimation, as well as feature-based visual odometry and real-time 3D reconstruction in point cloud.

During the survey stage of the mission, the Fugu-f enabled UUV would scan the ocean floor. Once it returns to the ship, a operator can highlight a desired target in the images. For target detection and pose estimation during recovery, the UUV will search for features identified in the survey images. Experiments are carried out on two different kinds of features. The first method is color based. Some preprosessing is performed and the object identified by its signature in a hue and saturation histogram. The other method uses features extracted using the ORB algorithm. The method of feature extraction through color proved to the most robust method during trials. Both worked in the pool, but the method based on ORB-features failed in the sea trial [3].

In the recovery stage, the UUV will approach the target while extracting features. Once it has a set of features, the pose is estimated by translating the survey-stage image such that it matches the current. This works by assuming images captured in the survey stage are orthogonal to the seabed and the altitude is known. The pose estimation is performed using a perspective n-point algorithm [3].

TRITON (2012-2014)

This project demonstrated docking with a subsea panel and performing a fixed-base manipulation task. This is similar to the ALIVE project, but performed with a GIRONA500 UUV in a pool environment. The manipulation task was both turning a valve and couple/decouple a connector [53].

The docking and manipulation was done using visual perception. Similar to earlier projects it was assumed the AUV had approached the panel using some means of acoustic

perception. Once the panel is in view, template matching was used to do pose estimation. The AUV knew beforehand what the panel looked like. It used the ORB method for feature extraction on both the template of the panel and images coming form the camera. When it had detected a sufficient amount of features, the pose of the object could be calculated.

This pose estimation was also used in the localization module of the AUV. Localization relied primarily on a Doppler velocity log (DVL), attitude and heading reference system, and depth sensor. With a fix on a known object such the subsea panel, this point would then used as a landmark. This fix is what served as a reference in the docking maneuver.

The valves and connector was detected using two different methods. The valves has to be turned 90 degrees and it was necessary to know their orientation. A stereo camera was placed so it would directly face the panel once docked. The three tips of T-shaped valves were painted in a distinct color. This color was then detected by the cameras and each point triangulated.

On the connector an alternate method was used. An ARToolkit marker was placed on a known location close to the base of the connector. The authors report this method was more robust than the color detection used for the valves. Color detection required tuning as lighting conditions changed, even in the pool environment [53].

MERBOTS (2015-2017)

This project explored multi-robot collaboration and was the evolution of TRIDENT (section 3.1.4. It was aimed at a search and recovery mission for underwater archaeology. The system was intended to navigate an unknown and unstructured environment and pick up various artifacts

An AUV would initially map the target area alongside a ASC contributing global positioning through a USBL connection. This data would then be used to generate a 3D map of the area. In the second stage a H-ROV would be deployed to perform an intervention task with the AUV monitoring [65].

The project was split into subprojects MARMANIP, ARCHROV and SUPERION. As part of ARCHROV, Palomer et al [52] explore the use of a underwater laser scanner as sensor for manipulation tasks. The instrument works like a LiDAR and has sufficient speed and precision to be used in a manipulation tasks. Using a GIRONA500 they demonstrate picking up an amphora in a pool environment.

The SUPERION subproject was responsible for perception during the survey and intervention task. While the H-ROV manipulates the target object, the AUV hovers in place above to provide a alternative vantage point. The pose of the target in relation to the AUV is detected in the point cloud using Iterative Closest Point [57]. The coordinates of the target in the image plane are then sent to the second stage responsible for the AUV velocity control.

TWINBOT 2018-2020

This project explores using collaborating lightweight I-AUVs to do construction tasks [56]. The work mainly focuses on the control problem of coordinating two robots, but pose estimation is briefly mentioned. This is limited to the scenario they test in a pool environment. Two UUVs lift a yellow pole, do some maneuvering and put it back in place.

In each frame, the pole is separated from the background by image segmentation in the HSV color space. Lines are fitted on top the segmented image to define a contour. Black stripes marks the spot where the pole should be grasped. As the dimensions of the pole and the black stripes are known beforehand, the size of this spot can be used to estimate the distance to the target.

As part of the TWINBOT project, Himri et al [24] also explore using a laser scanner. Object recognition and pose estimation is performed in the resulting point cloud. This is part of a SLAM algorithm where the pose estimate localize the robot in relation to recognized objects.

3.1.5 MARIS (2015-2018)

This project aimed at integrating technologies into a versatile I-AUV for the offshore industry, scientific exploration, and search and recovery. Goals included: guidance and control of a floating object, grasping and manipulation of objects, robot cooperation using underwater wireless communication, mission planning and lastly, object detection with pose estimation [9].

The various subsystems in the software architecture use ROS to interface it with each other. Some systems are fully implemented as ROS nodes, for instance the visual perception system. Other systems like the vehicle or gripper control run independently of ROS, but is bridged to a shell ROS node that acts as a interface [68].

The authors primarily emphasize developments relating to control in their summary of the project. Particularly the task priority system and manipulator arm. Their stereo camera rig is a notable element in the visual perception system. But to do pose estimation they use a type of template matching, similar to previous projects [68].

The visual perception system in MARIS is presented in [38, 39]. They investigate stereo-vision techniques capable of pose estimation without prior knowledge of the target. But apparently the accuracy and speed of this algorithm was insufficient for object grasping. Instead they rely on some prior knowledge about their target. They put special emphasis on detection and pose estimation of pipes. Pipes are very relevant for real world applications in offshore industry. But they are hard to detect accurately because of the lack of texture detail or corners. They are instead tracked using features that are more reliably detected in an underwater environment.

Man-made objects are identified by their contour of mostly straight edges and color uniformity. Color is detected by first performing the *grey-world* method of white

balancing. As the color of their target is known beforehand, it is easily detected if the white-balancing is successful.

Analysis is done in the hue-saturation-value (HSV) color space, which simplifies the process. In the HSV space, color ("hue") is a separate layer from brightness ("value"). This makes discrimination based on color more robust to differences in lighting conditions. Shapes are identified using a histogram of oriented gradients. Man-made objects appear as spikes in the histogram because they are generally made up of straight lines.

For pose estimation, a Sobel filter is applied on the image to extract edges. The edges of the pipe are identified on the resulting contour map. To average out noise, edges are tracked from frame to frame by a alpha-beta filter. The contour of the cylinder is a plane tangent to the camera. The planes are intersected to find the pipe's axis of symmetry. The ends of the pipe are used as frame of reference. With these assumptions pose estimation can be done with both mono and stereo vision, with stereo being slight more resistant to noise. This was tested in a pool environment, both in day and at night [38, 68].

Their system is more robust to occlusion than previous projects where manipulators had to stay out of the field of view. The kinematic system already knows the position of the manipulators. This information is translated into the visual system and is used to mask the area of the image where the arm is. This way, they can avoid a potential source of errors [68].

The authors suggest the method is transferable to other object givens a few conditions:

1. Target color must be distinct and detectable under varying lighting conditions.
2. Its shape must be regular so that contours can be geometrically defined.
3. It is a template matching technique and so the exact geometry must be known beforehand.

3.1.6 DexROV (2015-2018)

The DexROV ("Effective Dexterous ROV Operations in Presence of Communications Latencies") goal was to improve teleoperation of ROVs. Normally the staff controlling the ROV are on board the ship that deployed it. It would be cheaper to have these people on land, but remote operation is difficult because of bandwidth limitations and latency issues caused by satellite communication. Their idea was to let the staff on shore interact with a simulator environment and have the system translate this into high-level commands sent to the ROV. The ROV works in a semi-autonomous mode where it receives high-level commands from shore and has to translate this into low-level control on its own [1].

The basis of the perception system is a stereo camera rig that can be deployed with two or three cameras. Camera calibration is key in stereo vision, and the project developed a method for this they call the *pinax-model* [41]. It is intended for underwater cameras in housings with flat lenses and allows the calibration to be done in air.

The output of the stereo system is converted into a 3D occupancy grid. This is represented as a ochtree in the open source library OctoMap [29]. OctoMap also exist as a ROS package. The OctoMap data is integrated with the DVL and IMU data in a extended Kalman filter and used to aid in localization.

The voxels in the grid occupancy map is colored using information from the cameras. Given a set of observations of the same voxel, the one with the brightest color is selected. This guarantees each voxel was colored based on the image where the scene was best lit or camera closest [42]. They also explore using the dark channel prior method [21] for haze removal, adapted for underwater environments. But this has a high computational cost and the effect of dehazing is very varying. They do not test how it affect any computer vision algorithms [40].

The ROV can also perform semi-autonomous manipulation of a subsea panel. The spatial resolution of the 3D occupancy grid map is at best 2 cm. To do pose estimation of small parts on the panel they use an alternative set of methods. The algorithm knows beforehand the CAD model of the panel, and it is additionally outfitted with fiducial markers. Based on this, it selects regions of interest (ROI) containing a single circular handle (figure 3.2 a). The ROI is cropped from the current frame of the camera feed (b). An ellipse is then fitted on top of the handle using [33] (c). Inside the ellipse it performs Canny edge detection and Hough transform to identify the most dominant regular angle (d). To improve reliability, both cameras do this independently and the angle is averaged from one frame to the next [1, 48].

Their desire is to project real life conditions into the simulator environment. The simulator is intended to be the interface for the ROV pilot, but it is also used to validate subsystems of the complete DexROV system during development. As seen if figure 3.2 e-h, they perform the same pose estimation inside the simulator for validation.

The simulator can also aid in other stages of development. It can interact with either real or simulated subsystems. As each subsystem is developed independently, teams can test their subsystem on the complete system even though other compotents are incomplete [48]. To enhance the realism of the simulator, they also explore methods to project the current real world visibility into the simulator [16].

3.1.7 EU ROBUST (2016-present)

The goal of this project is to survey the seabed for mineral resources using I-AUVs [69]. The AUV must have some intervention capability in order to do *in situ* analysis of rock samples using laser-induced breakdown spectroscopy (LIBS). The task requires the AUV to identify potential rock samples, land in front of it and bring the LIBS instrument into close proximity with the sample.

They trained a convolutional neural net (CNN) to identify maganese nodules on the seafloor. The architecture name was "YOLOv3-tiny" and is fast enough to run real time

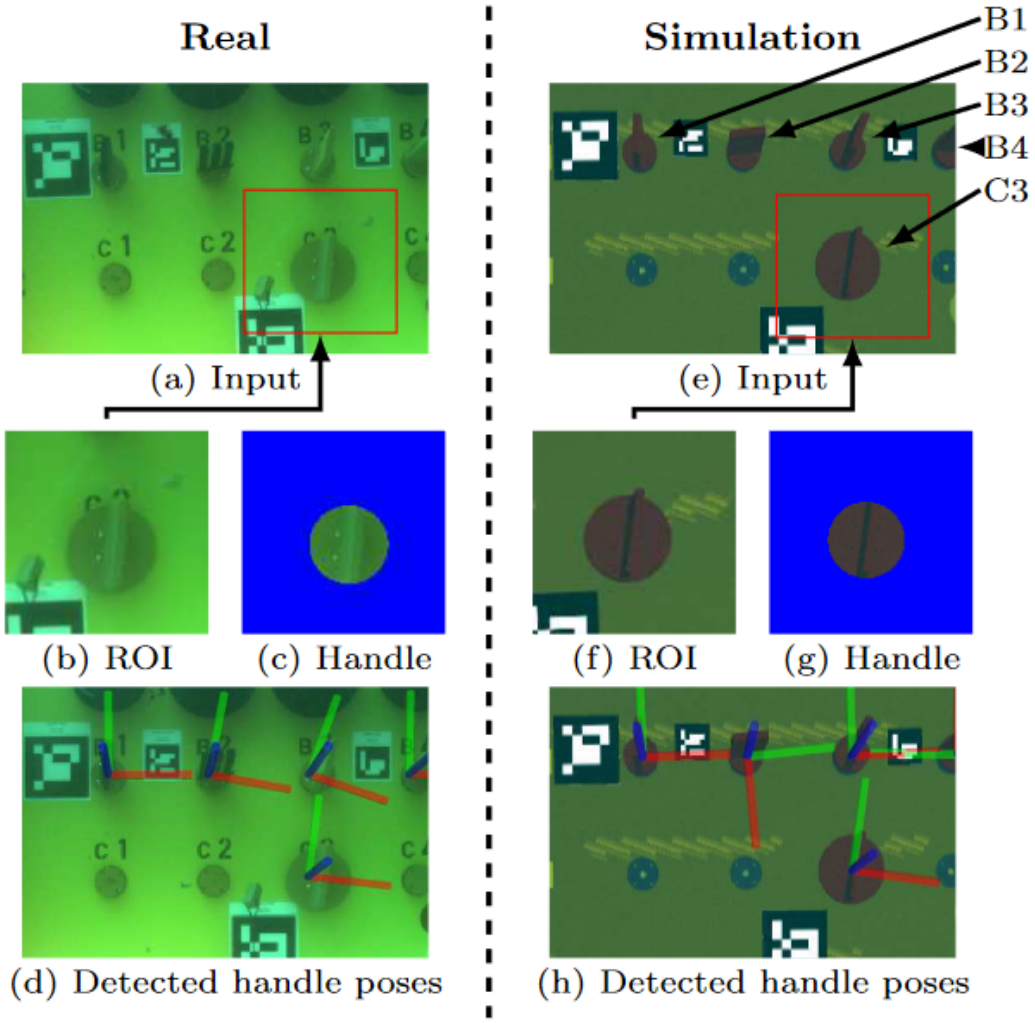


Figure 3.2: How pose estimation is performed by [48]

on a NVIDIA Jetson NX2 on board the AUV. The network draws a bounding box around each candidate and assigns a probability to it. Once a maganese nodule sample was identified, the AUV would land in front of it. This maneuver was achieved by holding the object in center view while decreasing altitude [69].

To do pose estimation of the sample they relied on a laser scanner to produce a point cloud. The nodules were identified in the point cloud using information from stereo cameras. The laser produced very good point clouds in the pool trials, but in the ocean it performed poorly. The sea trial was performed in shallow water, but the laser was not designed to work in sunlight. Additionally, they reported that particles in the water caused backward scattering of the laser beam, generating false points in the cloud [69].

Their fallback strategy was to use the stereo cameras for pose estimation. The CNN would draw a bounding box around the nodule on images from both cameras. The center point of the bounding box was then regarded as the target. Using both images, they could triangulate the position of the nodule, move the arm over and sample it with the LIBS instrument [69].

3.1.8 SEAVENTION (2018-2021)

This Norwegian project explores mission planning, perception and augmented reality for autonomous underwater operations [70]. As part of this, Nielsen et al. [49] use the PoseNet architecture [31] attempt to do end-to-end neural network 6 DoF pose estimation of a mock-up subsea panel connector in a pool environment. Training data is labeled using a system that relies on fiducials to visually track the ROV while it hovers in front of the mock-up. Accuracy is good on average, but suffers from outliers they suspect come from errors in the training data. It is also uncertain how well generalizes to new environments. Testing and training was collected from just two sessions with varying water turbidity. To track a different object it would have to be retrained, but could possibily utilize transfer learning.

Kedir et al. [46] also use a end-to-end neural net to perform pose estimation, but their net seem tailor-made for the UTOFIA sensor described in section 2.2.4. They also use fiducials as ground-truth, but as shown in figure 3.3 use a extra camera above the surface to detect fiducials as well. The UTOFIA sensor under water is rigidly attached to the camera over water with a 1 meter rod. Both capture frames of a poster that extends both above and below the water surface. Knowing the extrinsics of both cameras they can use rigid transforms to infer the position of the UTOFIA sensor, even in highly turbid waters. Their dataset is created in a pool environment where they can control the level of turbidity by dissolving clay in the water and measure it by recording the water's light absorption.

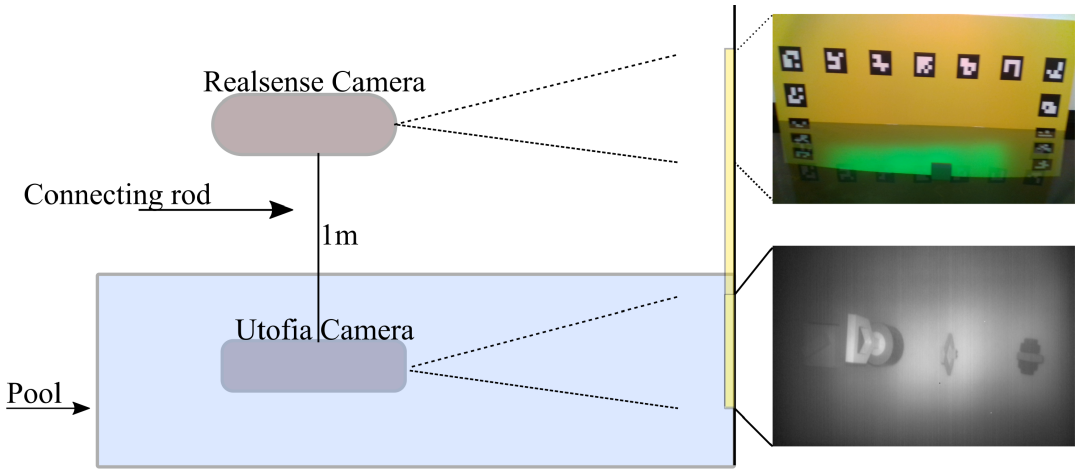


Figure 3.3: The setup used by Kedir et al. to capture fiducial pose estimates used as ground truth.

The neural net use the EfficientNet CNN architecture to create a feature map of both the depth and RGB image produced by UTOFIA. This is fed into a bi-directional feature pyramid (BiFPN) layer. Four separate networks each interpret the BiFPN to solve different tasks. One each for object detection, bounding boxes, rotation and translation. This was trained using roughly 30000 images captured at different turbidity levels. Their

sensor fusion approach give an average 6.49 cm translation error and 2.59 degree rotation error.

3.1.9 Other projects

Park et al [54] demonstrate pose estimation by template matching, integrated into a navigation system that use it to update its position. The CAD-model is projected onto the image and the moving edges algorithm [4] used for feature extraction. With this method they demonstrate circling and manipulating a mock-up subsea panel in a pool environment.

Cui et al [13] develop a stereo-vision system for picking sea urchins. They use a CNN to draw bounding boxes on the urchins and do feature extraction at each corner. If the corners of the bounding box in each stereo image have matching features it is a pair, and the object is tracked in the following frames using kernelized correlation filter [23]. Focus is on the accuracy of the detection and tracking, and they do not report on the accuracy of the pose estimate.

3.2 Pose estimation in air

3.2.1 Pose detection benchmarks

Pose estimation in air has been a very active field research in past decade. Many of the most successful methods were recently all benchmarked on a unified dataset in the BOP challenges. BOP stands for Benchmark for 6D Object Pose estimation and is a competition held in 2018, 2019 and 2020 [26, 27]. There is also an upcoming 2022 challenge. In the 2020 challenge eleven existing datasets were collected and offered to participants in a unified format, together with an evaluation metric. The competition encourage methods that don't use real training images. Methods should rely only on the 3D model and artificially generated renders of that model. This is because of the challenges in creating datasets, further discussed in in part 3.3.1.

Unfortunately most methods they benchmark somehow rely on images with a depth channel, captured by a 2.5D camera. RGB-D methods are difficult to use underwater. Widely available depth cameras use *structured light* to determine the distance to each pixel. But these does not work in water, which quickly absorbs the infrared light used for ranging. Only speciality RGB-D cameras such as UTOFIA can work underwater [44]. UTOFIA fuses data from a LiDAR and a regular camera to create a RGB-D image, but is not commercially available.

However the contestant in third place in 2020 is a method using purely RGB images. It is a variation of the same architecture that won the competition, that also added a pose refinement step relying on the depth channel. The method is named CosyPose [35]

and is described further in part 3.2.2. It also performed best when Yu-Wei Chao *et al.* evaluated it on their DexYCB dataset [10]. The only method that came close was the DeepIM variant using RGB-D.

A final remark about BOP is that it evaluates different categories of pose estimation methods, and only in 2020 did the learning-based methods jump to the top of the leaderboard. In previous years point pair methods working on 3D point clouds were dominant. These relied heavily on the RGB-D images, whereas learning-based methods tend to rely on RGB only. The success of learning-based methods is credited to a special rendering tool made available to contestants by the BOP creators.

The tool is called BlenderProc4BOP [15] and produces photorealistic images of 3D models that are used to train deep neural nets. The BOP team made available 50,000 renders of objects in the seven datasets that were used for evaluation. Comparison of results after training with both BlenderProc4BOP and traditional "render and paste" methods show very significant improvement [27]. For instance, the accuracy of CosyPose improved by 57.9 absolute percent on one particular dataset (T-LESS [28]). Figure 3.4 show a comparison of the rendering methods. BlenderProc4BOP images are in the bottom and are obviously more realistic than the old renders.

Figure 3.4: Old renders in the top row, new renders on in the bottom

A generic version of BlenderProc [15] is available on GitHub. It is a actively maintained project made specifically to create training images for various neural net architectures. As the name implies, its backend in the popular open source rendering software Blender. Blender is very much capable of creating underwater scenes, and both it and BlenderProc could be adapted to create training images.

3.2.2 Pose detection neural nets

These networks first use a separate object detection or instance segmentation network to identify a region of interest in the image. Object detection is drawing bounding boxes on each object belonging to a category. Instance segmentation draw a mask over each separate instance of an object in that category by classifying every pixel in the image.

The pose detection networks then generally use some kind of feature extractor like CNNs to find keypoints. Some also try to find a mix of different features, like HybridPose [71] which searches for keypoints and feature concepts like edge vectors and symmetry correspondance.

On the output of the feature detector is typically separate neural nets that each predict the translation and rotation of an object, but the methods vary slightly. For instance PoseCNN [80] directly regress a quaternion representation of rotation from image features. But for translation utilize the pinhole camera model: It only guesses the distance Z to the object center. From this they can infer the X and Y position via the camera model.

Most methods focus on the problem where there is single input image. CosyPose is a framework for detecting one pose from multiple views. With this approach they won the 2020 BOP Challenge as best overall method. It assumes it is given a set of images and a set of 3D models of objects in those images. It then first performs rough single view single object pose detection on all objects in all images. Then in the second stage these error-prone estimates are filtered, leaving only those that are consistent across images. This step is key, and the name CosyPose is a play on the word COnSistencY. In the final third step pose refinement is performed on the remaining estimates [35].

Exploiting information of the same object across several images make the method more robust and able to outperform singe view single pose methods. It might also a good fit in robotic applications, as multiple frames can be sampled from the UUV video feed over time.

The method was trained on 1 million synthetically generated images per dataset in the original paper, with each dataset containing 20-30 objects [35]. Roughly 50000 images per object. The method was trained again for the BOP Challenge [27], which in addition used BlenderProc images that improved results further.

Pix2Pose [55] was much less successful in the BOP challenge than CosyPose, but might still be a particularly good fit for use under water. CosyPosy rely on image features extracted by a CNN; the network tries to find correspondences between real life and a textured model. Typical subsea equipment has very little features however, and the CAD-models are generally designed without textures.

Pix2Pose instead works with textureless models and so belong to a different lineage of neural nets. It automatically generate a texture by using the X, Y and Z coordinate of that pixel's position. Because the RGB color space is a three element representation of color, RGB can also be used to visualize how Pix2Pose encode every pixel in XYZ

coordinate system. Figure 3.5 show model that is render such that pixel position in the X direction is expressed as "redness", Y-direction as "greenness" and so on. A generative-adverserial network is trained to create the coordinate-as-color images and after some error correction PnP and RANSAC is used to infer the 3D position of each pixel from their position in the 2D plane.

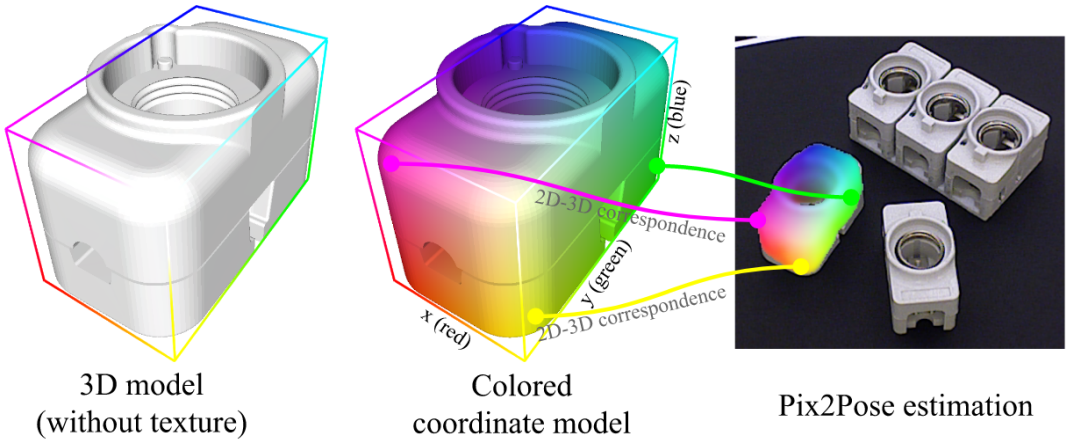


Figure 3.5: Visualization of how Pix2Pose automatically generate textures.

3.3 Pose estimation datasets

3.3.1 General considerations

For pose estimation in air the most commonly used dataset is LINEMOD created by Hinterstoiser *et al.* [25]. But this was not intended as a benchmark and has several biases. The creators of the BOP benchmarking framework [26] criticize LINEMOD for having constant lighting, targets that are unoccluded and always centered in the camera frame. These are some considerations when creating a new dataset.

Other factors emphasized by Kastman *et al.* [30] is including objects that has no 3D model and should not be tracked. The dataset should have objects that deviate slightly from their 3D model, as would be expected in real life. The dataset should also have multiple instances of the same object. There should be both CAD-designed products that are machined or 3D printed, and objects 3D scanned physically.

Most appear to include between 15-30 objects in a few different scenes. The HomebrewDB dataset [30] recorded each scene one time with a handheld camera and two times using a turntable, varying the camera angle between 35 and 45 degrees.

Occlusion was not a consideration when pose estimation methods were less mature, but in recent years it has become an important research focus. This is only for pose estimation in air though. Even the most recent projects exploring pose estimation underwater [49, 54] does not consider occlusion. Li *et al.* [36] also show that DNN architectures can be trained

on datasets where occlusion is added artificially to training data without occlusions. Hence having occluded objects might only be a secondary concern in a new underwater dataset.

3.3.2 Ground truth

Using a robotic arm

Articulated robot arms have also been to used to create pose estimation datasets. By using rigid body kinematics, the position and orientation of the end effector in relation to the base is very precisely known. However only speciality robotic arms would work underwater and the project does not have access to one.

Fiducial markers

As seen in figure 3.6 this method is used to generate several of the datasets that are being used to train and evaluate pose estimation methods in air.

Figure 3.6: HomebrewedDB [30], IC-MI [73], Linemod-Occluded [5, 6], T-LESS [28]

Henriksen *et al.* [22] investigated the accuracy of pose estimation using AprilTag in water. They performed an experiment in a pool with a separate tracking system that could act as ground truth. They found that the accuracy of the translation estimate was within a few millimeters. The yaw error rotation was reported to be less than one degree. These results encourage an attempt to use fiducials to generate an underwater pose estimation dataset.

Pose estimation with fiducials estimates rotation and translation of the tag in the camera reference frame. But the objective is know the rotation and translation of the object placed on the board. There is no obvious way of knowing the pose of this object in relation to any marker on the board. This matters for template matching methods that output the pose in the coordinate system of the object model.

Only HomebrewedDB [30] present a method for aligning the camera frame with the object model frame. But this method relies on RGB-D data to create a dense point-cloud.

Using acoustic positioning

Acoustic position is used by many [22, 49] for tests in indoor pools, but there does not any product any accurate tracking available for use outdoors. OceanLab has a short baseline underwater positioning system called "Waterlinked Underwater GPS G2". But

this appears to be designed for rough positioning at sea, with an estimated error of 1 meter at 100 meter range. This makes it too unreliable for this application, where accuracy should preferably be with a centimeter.

Chapter 4

Methodology

4.1 Dataset creation

To create the dataset I rely on a template-based pose tracking method called RBOT. This need to be initialized with a pose however, as it cannot detect without one scratch. To do this I use fiducial pose detection from OpenCV. The OpenCV pose detection is also used a sanity-check of the pose from RBOT once initialized.

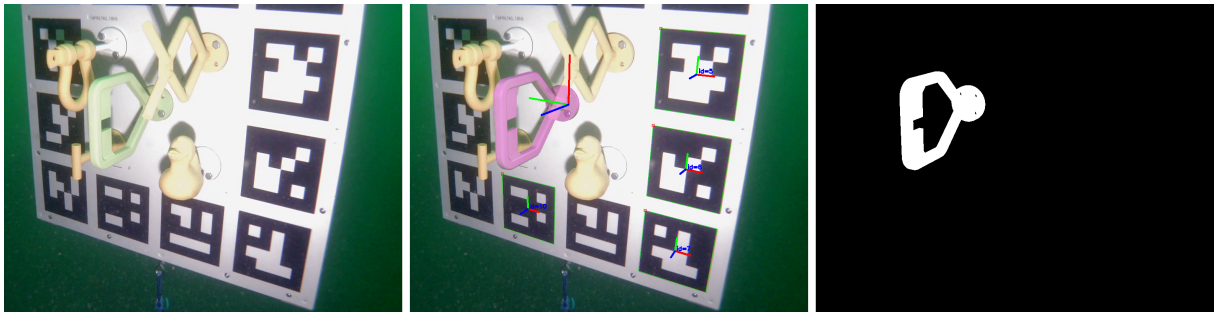


Figure 4.1: Both RBOT and the fidual pose estimator search the raw frame (left) and estimate poses. If the fiducial pose (large arrrows) and RBOT pose (pink model) align the frame is saved a binary mask (right).

The pose of the target model can be found by circling it with fiducials of a known size. Because the position of each tag in relation to the poster center is known, rigid body transforms can be used to find the pose of a target model. The poster used is shown in figure 4.2 with figure 4.3 showing the scale. Each tag is 14 x 14 cm. A large size is necessary for robust detection at range underwater when visibility it poor. The rings and dots in the center represent where targets should be bolted in place.

Generally RBOT has a good precision, that is it generates poses that are smooth and consistent. However its accuracy depend on the accuracy of the initial pose. Fiducial pose estimation has decent accuracy when averaged across many frames, but poor precision. Poses estimates has a large spread and looks jumpy when viewed in sequence as a video. To achieve both accuracy and precision the two methods are used in tandem, where the

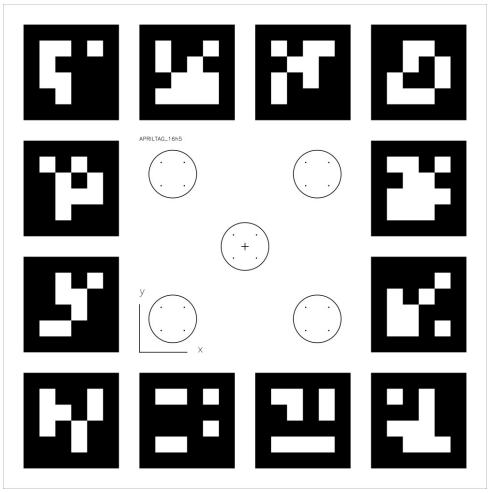


Figure 4.2: Poster with fiducials and sockets for models.

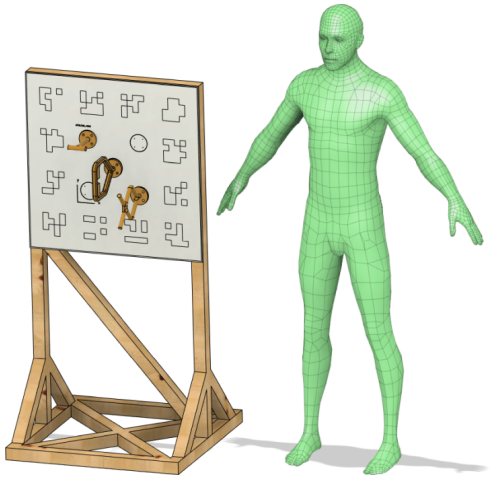


Figure 4.3: The poster with a human for scale

pose created by RBOT is set as ground truth only if it aligns with the fiducial pose estimate of that frame. That is, the pose is accepted as ground truth if both methods verify it. The rough process is illustrated in figure 4.1. In the center image the fiducial pose is drawn with large arrows and the RBOT pose as a pink model. If they align the resulting binary mask is saved. The contour of the binary mask is extracted and saved as polygon coordinates along with the raw frame.

To be a verified pose the rotation and translation error must be within certain limits. The limits used here are presented in table 4.1. If greater precision is required the percentage of valid images would obviously decrease. The parameters used here is a compromise between how much raw material is available and how many images I would need to train a neural net.

Rotation		**Translation**	
X	4°	X	5 cm
Y	4°	Y	5 cm
Z	4°	Z	10 cm

Table 4.1: Pose correspondence limits

4.1.1 Fiducial pose detection

The pose of each tag is extracted with generic functions from OpenCV that return the rotation and translation vectors describing each tag. These detect all square shapes in the image and look for a QR-code corresponding with a fiducial dictionary inside that square. If a square is recognized the algorithm knows beforehand both the size of fiducial and the camera intrinsics. With this information it can solve for the pose using PnP and RANSAC. The output describes the rotation and translation of each tag in the frame of reference of the camera. The implementation details are described further in 4.3.2.

Rigid body transformations

Rigid body transformations are used to convert the pose estimate of each fiducial tag to every model. The output of this process is illustrated in figure 4.4a. Here, the reference frames with short arrows are poses directly from OpenCV `estimatePoseSingleMarkers()`. The reference frames with longer arrows are transformed into the frame of the model in question. Orientation of the X, Y and Z axis depend on how the model was drawn in Fusion 360.

Object in the scene	Frame ID
The ROV camera	c
The pose of the fiducial tag	t
Origin is the center of the poster	o
The pose of the relevant model	m

Table 4.2: Frames in the scene

The relevant frames are summarized in table 4.2 and illustrated in figure 4.4b. What we are interested in is the pose of the model in the frame of the camera T_{cm}. But what we have is the pose of the fiducial tag in the frame of camera T_{ct}.

Because we know precisely how everything on the poster is laid out, we also have the pose of the origin in the frame of the fiducial tag T_{to}, and the pose of the model in the frame of the origin T_{om}. Using the properties of the homogeneous transformation matrix we can take the dot product so that

$$T_{cm} = T_{ct}T_{to}T_{om}$$

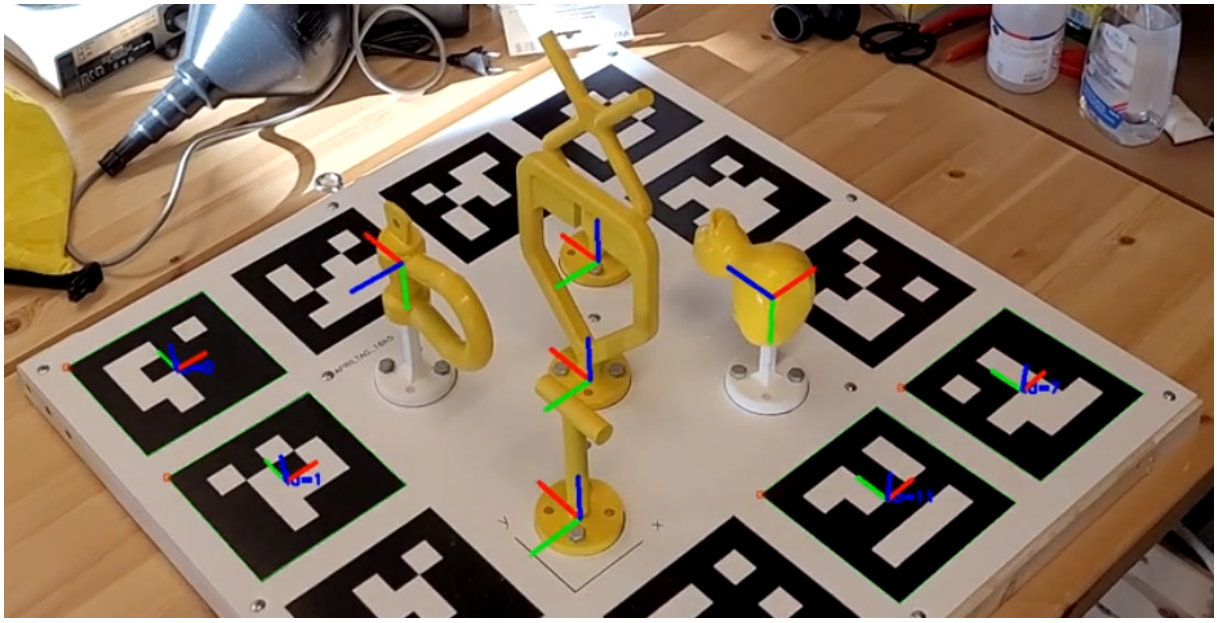
`MODEL_BASE_DISPERSION` from table 4.5 is important when defining the transformation from the image center to either of the four peripheral models sockets. This describes the diameter of a circle around the origin. The center of each socket is placed at the four corners of this circle. Using the Pythagorean theorem we find that the displacement in both the X and Y direction is

$$\frac{\texttt{MODEL\_BASE\_DISPERSION}/2}{\sqrt{2}} = \frac{\texttt{300}/2}{\sqrt{2}} = 106.07mm$$

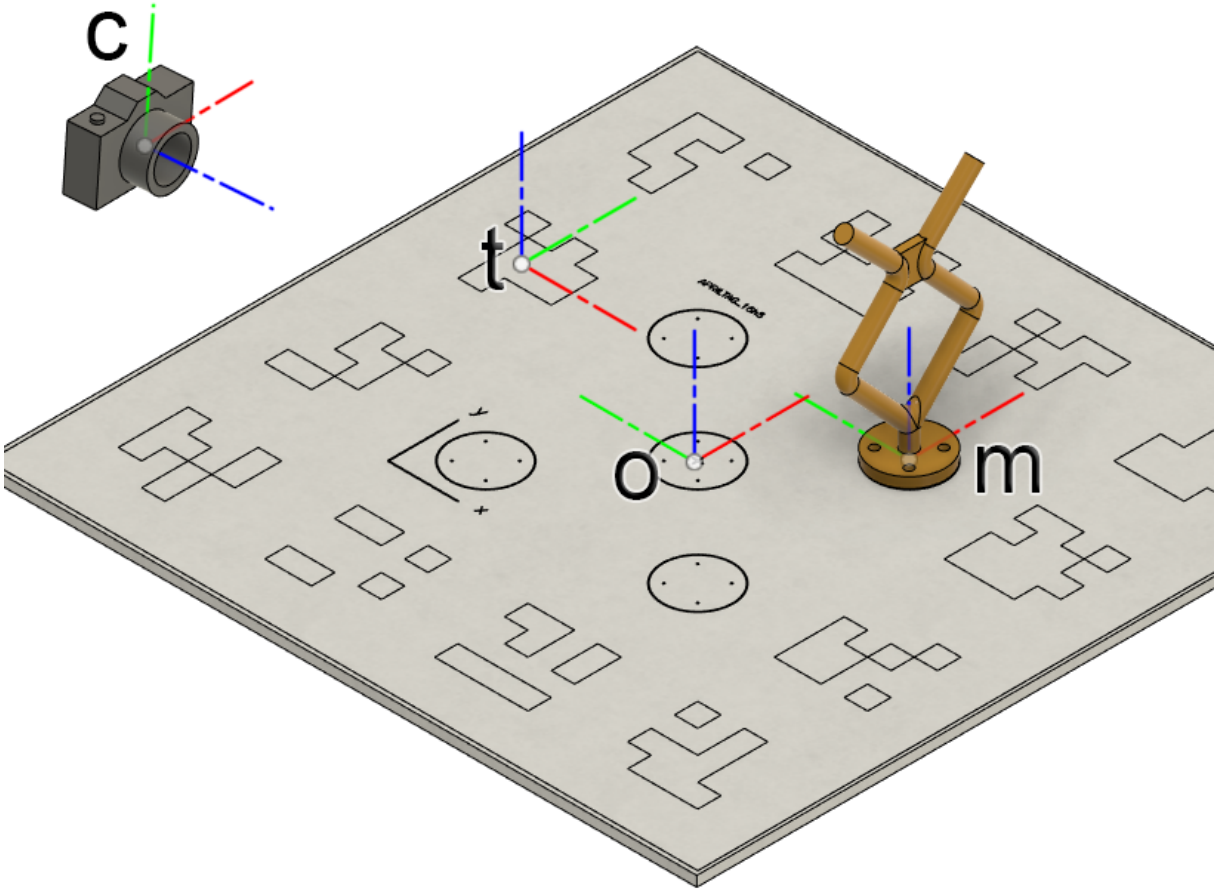
The transformation from origin to the fishtail handle then is

$$T = \begin{bmatrix} 0 & -1 & 0 & 0.106 \\ 1 & 0 & 0 & 0.106 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation element of the matrix reflect how the model is oriented relative to the origin, whereas the transformation vector show the model is moved 0.106 meters in the



(a) Reference frames with short arrows are from OpenCV directly, long arrows are transformed.



(b) The frame of the (c)amera, fiducial (t)ag, (o)rigin and (m)odel.

X and Y direction.

Pedestals

The shackle and ducky figure have the added element of being mounted on pedestals. This adds a transform in the Z direction as well. For the shackle for instance, this is 90 mm, as seen in figure 4.5. And so the tranformation matrix is

$$T = \begin{bmatrix} 0 & -1 & 0 & -0.106 \\ 1 & 0 & 0 & 0.106 \\ 0 & -1 & 0 & 0.09 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

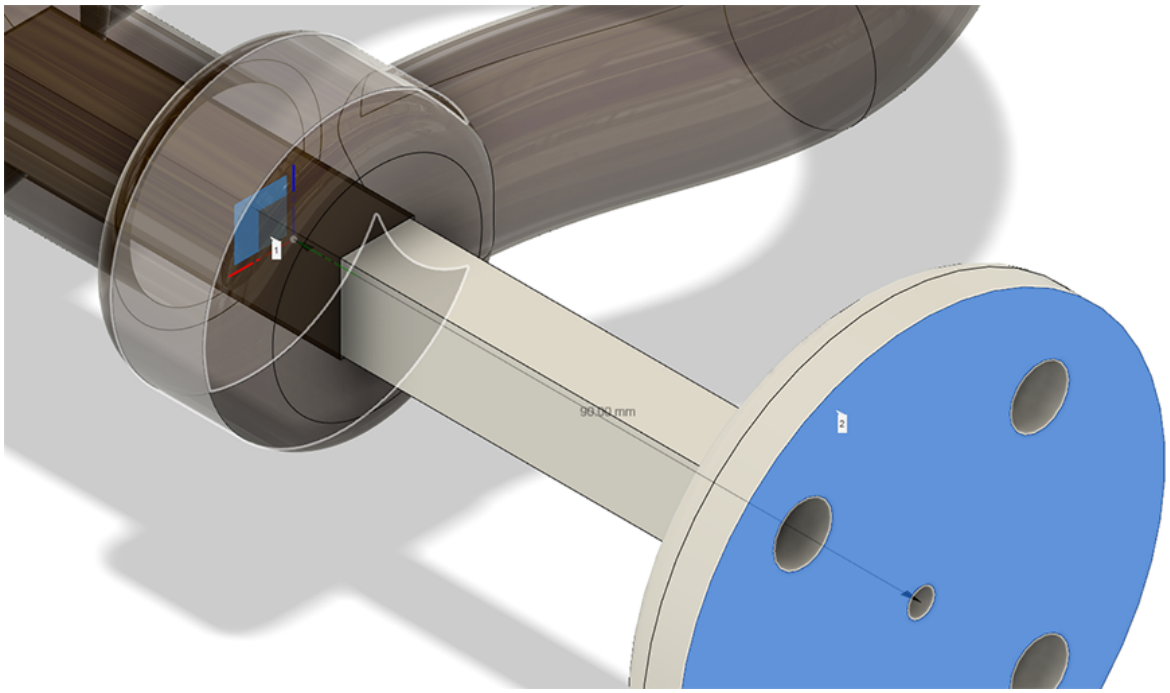


Figure 4.5: Distance from the model origin to pedestal base. The the origin of the model is shown as a red, green and blue frame.

The pedestals were necessary only because the models did not naturally feature a base. The models used by template matching algorithms does not feature this pedestal. As the pedestal is white, it blends into the white background on the poster and is ignored. A similar method was used on the duck model.

Outlier detection

The OpenCV fiducial pose estimation works by identifying squares with a valid QR code in the middle. The method is prone to error if the corners of the squares are occluded. It might still detect a square with a valid QR-code, but the estimated corner coordinates are wrong. Because the size and orientation of the square is used to determine the pose,

this results in a wrong estimate, as illustrated in figure 4.6. Here the bottom left corner of square on right is occluded by the duck. Resulting in a pose estimate that is wrong.

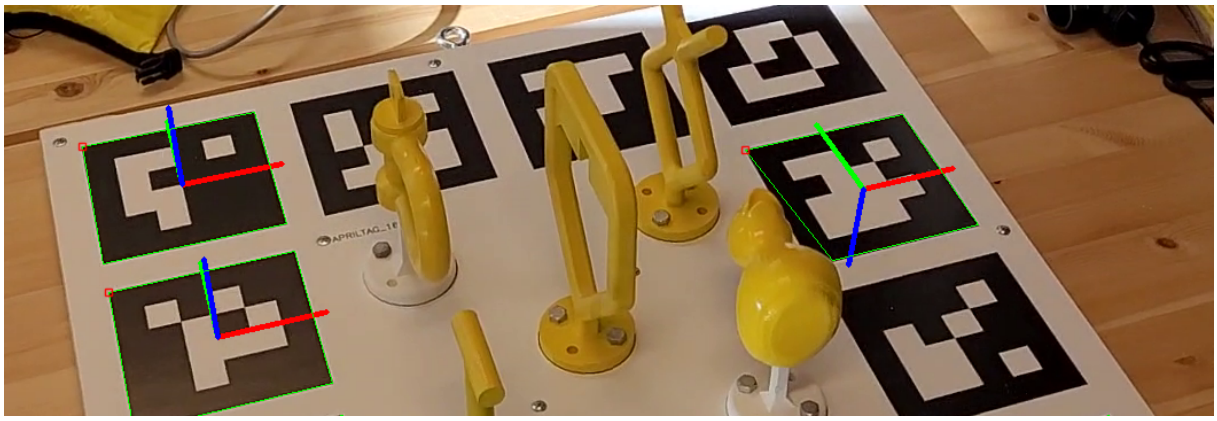


Figure 4.6: OpenCV incorrectly detect corner, resulting in a erroneous pose.

Because the poster has many fiducial tags transformed into a pose of its center, outliers can be removed based on that average. Assuming the majority of estimates are correct, outliers are removed based on their difference from the average. For every frame, the average and standard deviation of each element in the translation vector is calculated. If any element is greater than two standard deviations from the average, the pose is classified as an outlier.

Calculating the average of the translation vector is straightforward, but the rotation vector is not. Translation simply represent position in meters perpendicular to the image plane (Z-axis) or along the image plane (X and Y). The rotation vector encode rotation along the XYZ axis and averaging it can create singularities. Instead rotation is averaged using a method from Markley et a. (2007) [45]. Given a list of quaternions, this method sums the dot product of each quaternion and its transpose as a 4x4 matrix. The resulting averaged quaternion is the eigenvector of this matrix. Outliers are sorted based on the translation vector only, but only inliers are included in the average rotation presented at the pose of the model.

Fiducial dictionary

ArUco and AprilTag comes with several different "dictonaries" that can translate a 2D barcode into a fiducial. These comes in sizes of 4x4 to 10x10, where the integer indicates the number or cells in the matrix. Figure 4.7 shows examples of tags in different sizes.

The detection rate of different dictionaries were tested in air with the purpose of determining which one was most suitable for underwater tests. For testing in air it is sufficient to print the tags using a normal printer. The tags are placed in the same arrangement that would be used in a pose estimation dataset, with space for the target object in the center. Results are presented in section 5.1.2 and show that "APRILTAG_16h5" is most suited.

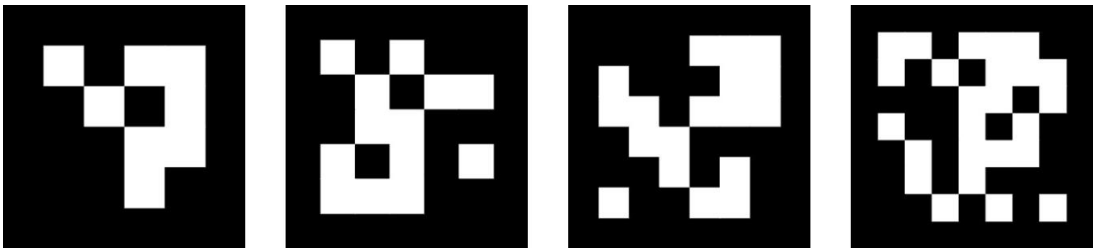


Figure 4.7: 4x4, 5x5, 6x6 and 7x7 ArUco tags

4.1.2 Template-based pose tracking

RBOT builds on a lineage of region-based pose tracking methods. These essentially work by first performing image segmentation to separate the object from the background. Then it project an artificial silhouette of the prior 3D model onto the segmented image. This silhouette is rendered in different poses to find the best fit. The silhouette of the best fit is in turn used to inform the segmentation process in the next frame of the appearance of the target. This loop relies on being given an initial pose to start the process [74, 75].

Formally a video is a sequence of images $I_c(t_k)$ where $t_k \in \mathbb{R}, k = 0, ..., l$ and $I_c(t_l)$ is the current frame. The method will calculate a sequence of homogeneous transformation matrices $T(t_k)$ to form a trajectory. It always assumes that $T(t_{l-1})$ is known and calculate $T(t_l) = \Delta T T(t_{l-1})$. Here ΔT is the exponential of the twist $\Delta T = exp(\mathcal{T})$. Using the twist is particular to RBOT compared to previous region-based methods. It allows more efficient optimization of the cost-function using a Gauss-Newton strategy instead of gradient decent [75].

The cost-function is an expression of how well the projected silhouette fits the actual estimated one. The cost is calculated as

$$E(\mathcal{T}) = -\sum_{\mathbf{x}\in\Omega} log(H_e(\Phi(\mathbf{x}(\mathcal{T})))\overline{P}_f(\mathbf{x}) + (1 - H_e(\Phi(\mathbf{x}(\mathcal{T}))))\overline{P}_b(\mathbf{x}))$$

Here $\mathbf{x}$ is a pixel in I_c and $P_f(\mathbf{x})$ or $P_b(\mathbf{x})$ determine the probability of it being either foreground (target) or background. $\overline{P}(\mathbf{x})$ is the extension to segmenting based on local histograms rather than global.

$\Phi(\mathbf{x}(\mathcal{T}))$ is a level-set method that define the contour of the projected 3D model prior. It's the squared euclidean distance between a pixel $\mathbf{x}$ and the closest pixel in the contour map. Being negative for foreground pixels and positive for background pixels. Figure 4.8 show $\Phi(\mathbf{x})$ applied to a contour $\mathbf{C}$ where 0 defines the border between foreground and background. This is finally wrapped by a smoothed Heaviside step function H_e that's ≈ 1 for negative Φ (foreground) and ≈ 0 for positive Φ (background).

The functions $P_f(\mathbf{x})$ and $P_b(\mathbf{x})$ rely on a model $P(\mathbf{y}|M_f)$ of the foreground and $P(\mathbf{y}|M_b)$ of the background to calculate the membership of a pixel. Here $\mathbf{y} = I(\mathbf{x})$. The models are 32-bit RGB color histograms of local regions around a projection of a model vertex $\mathbf{X}$. Figure 4.9 show examples of vertex $\mathbf{X_i}$ being projected onto the image in the

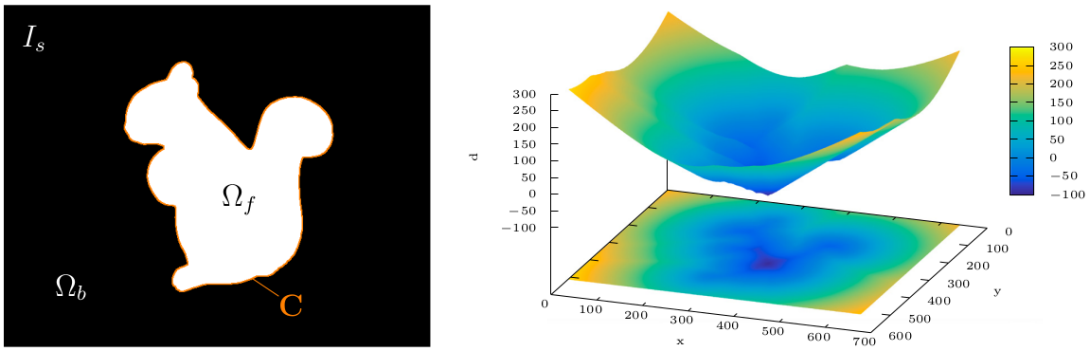


Figure 4.8: The $\Phi(\mathbf{x})$ level-set method applied on a rendered silhouette

center. The final segmentation mask on the right is the pixel-wise background-probability subtracted from the foreground-probability.

Figure 4.9: Illustration of temporally consistent local histograms. The small circles in the center image represent Ω_i.

A pixels probability of belonging to a certain region Ω_i is calculated as

$$P_{f_i}(\mathbf{x}) = \frac{P(\mathbf{y}|M_{f_i})}{\eta_{f_i} P(\mathbf{y}|M_{f_i}) + \eta_{b_i} P(\mathbf{y}|M_{bi})}$$

$$\eta_{f_i} = \sum_{\mathbf{x}\in\Omega_i} H_e(\Phi(\mathbf{x})), \quad \eta_{b_i} = \sum_{\mathbf{x}\in\Omega_i} 1 - H_e(\Phi(\mathbf{x}))$$

The method then assumes the previous segmentation mask is correct and updates the histogram models based on that mask. For the foreground model

$$P(\mathbf{y}|M_{f_i}) = (1-\alpha_f) P^{t_{l-1}}(\mathbf{y}|M_{f_i}) + \alpha_f P^{t_1}(\mathbf{y}|M_{f_i})$$

With α being a learning rate between 0.1 and 0.2 determined experimentally.

For the final cost function

$$\overline{P}_f(\mathbf{x}) = \frac{1}{\sum_{i=1}^{n} \mathbf{B}_i(\mathbf{x})} \sum_{i=1}^{n} P_{f_i}(\mathbf{x}) \mathbf{B}_i(\mathbf{x})$$

where $\mathbf{B}_i(\mathbf{x})$ is special masking function that is 0 for pixels not belonging to region Ω_i

and 1 otherwise.

If tracking is lost RBOT also has a strategy for performing pose detection. From a polyhedron with 20 faces (icosahedron) centered on the 3D model, 12 view ports are created from the corner vertices. Each view port is rotated four times and scaled in three different sizes. This results in $12 * 4 * 3 = 144$ base templates. These are searched for in a down scaled version of the image using a sliding window search. If a match is found it will begin a pose refinement strategy using less course poses and larger images.

4.1.3 Poster

The poster is generated using a custom python script that can easily change the size and layout of tags. The script takes a number of arguments, listed in table 4.5. These in turn change how the poster will look, illustrated in figure 4.10. Two different outputs are illustrated in figure 4.11 and 4.12.

The poster has $4 * n - 4$ fiducials tags and one or five "sockets" where models can be placed. Each tag has a unique ID and the distance between each tag and the image center is precisely known. This is stored in millimeters in a separate JSON text file created simultaneously with the poster. This makes the metrics easily accessible to any other software that will utilize the data.

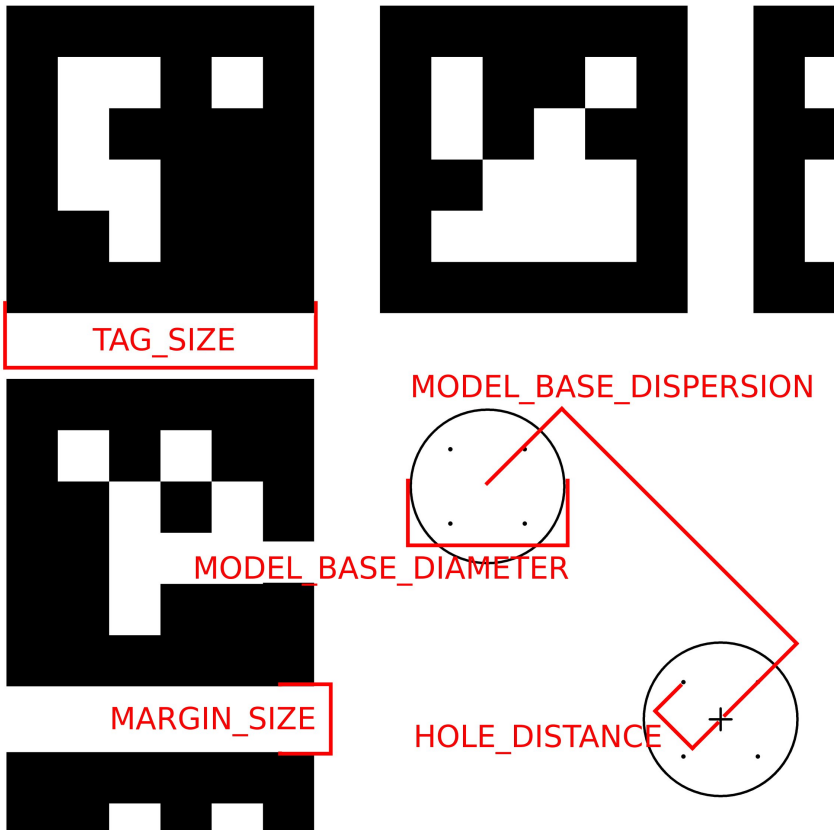


Figure 4.10: Illustration of how parameters affect the poster

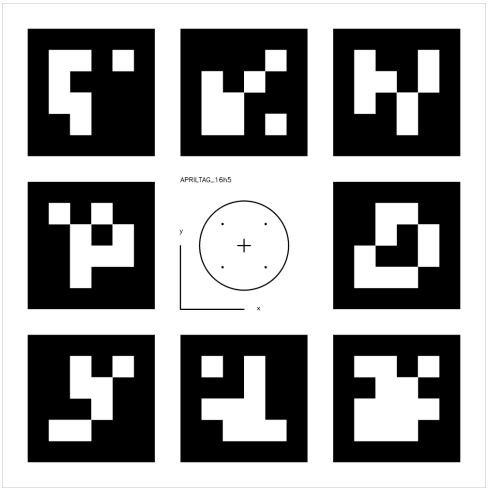


Figure 4.11: A generated poster with 3 fiducials on each side and a single socket.

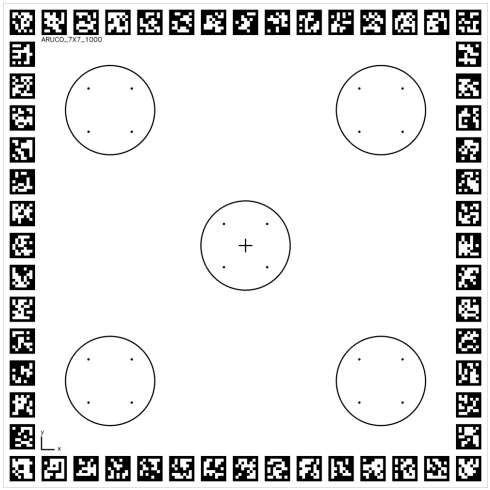


Figure 4.12: Another generated poster with 15 fiducials on each side and five sockets.

4.1.4 Models

The poster used in testing was populated with five different 3D models designed in Autodesk Fusion 360. These models were made to resemble realistic UUV "targets" such as shackles and handles.

The models were drawn on the basis of pictures and technical drawings from real manufacturers. Although the models used in this test will vary slightly from the real products, the difference is negligible, as seen in figure 4.14 and 4.15. The benefit of using self-made models is that these can easily be 3D printed to create a perfect replica for the template matching algorithm.

For template detection methods, the number of vertices matter for the performance. This is particularly true for RBOT with its region-based based approach, where it calculates histograms of regions around randomly sampled vertices. The number of vertices for each model is shown in table 4.3. The simplest model is the T-handle with 790 vertices. The most complex is the shackle with 7317. A maximum of 7000 vertices is recommended for RBOT.

Table 4.3: Model data

	Vertices	**Faces**	**Size [MB]**
D-handle	2312	4640	0,49
T-handle	790	1592	0,178
Fishtail	2588	5192	0,538
Shackle	7317	14615	1,5
Ducky	5482	10960	2,8

Figure 4.13: All models, rendered side-by-side in Fusion 360.

4.1.5 Underwater footage

ROV

A Blueye X3 was used to capture all the images. This is a mini-ROV intended primarily for inspection tasks. The image sensor is a Sony IMX290, a 1/2.8 inch Exmor R CMOS with 1920x1080 resolution at 30 FPS. The lens is fixed focus, with a minimum distance of roughly 20 cm. It has a 115°field of view.

Length	485 mm	Sensor	Sony IMX290
Width	257 mm	Resolution	1080 @ 30 FPS
Height	354 mm	Field of view	115°
Weight	9 kg	Fixed focus	
Rated depth	305 m		

Table 4.4: Blueye X3 specifications

The camera on board can be tilted up and down and this varies throughout the recording to make sure objects are captured from as many angles as possible. Similarly the lights vary in intensity from maximum to off.

Figure 4.14: A real fishtail handle made by the manufacturer Greenpin.

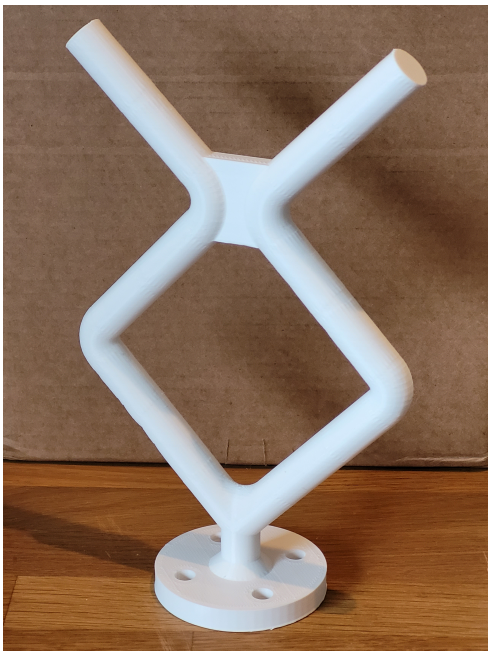


Figure 4.15: 3D printed fishtail handle drawn in Autodesk Fusion 360.

Figure 4.16: Blueye promotional pictures

Calibration

Camera calibration was performed in a water tank at Oceanlab, shown in figure 4.17. Calibration was performed under water because the refractive index of air is 1.000 and the index of saline water (3% NaCl) is 1.338.

The lab has a freshwater tank big enough to hold both the calibration pattern and ROV. This was preferred to capturing images outside. Foremost because the ROV could be held by hand and a good collection of calibration images be collected systematically. An added benefit was the increased visibility, both in terms of lighting and turbidity. The downside is the difference in the refraction index of fresh and brackish water, but this is negligible (1.338 vs. 1.333) for the purpose of this test and out-weighted by the benefits of the tank.

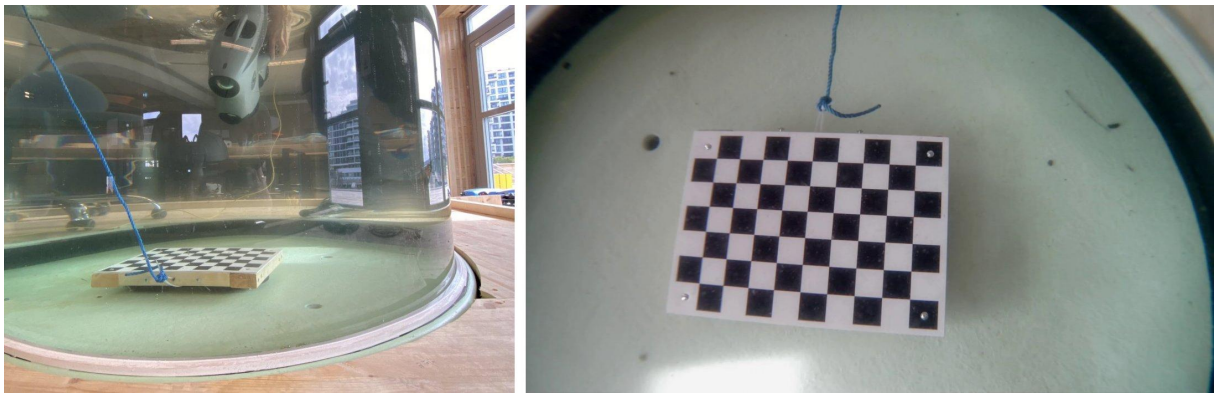


Figure 4.17: Blueye X3 during camera calibration in the indoor tank.

Location

The poster was filmed on two different days and two different locations. All recordings were done at approximately 8 meters depth. Recordings were made at Filipstad dock near Oceanlab, shown in figure A.1. The two locations are roughly 100 meters apart. Figure 1.1 show the ROV filming at location 2.

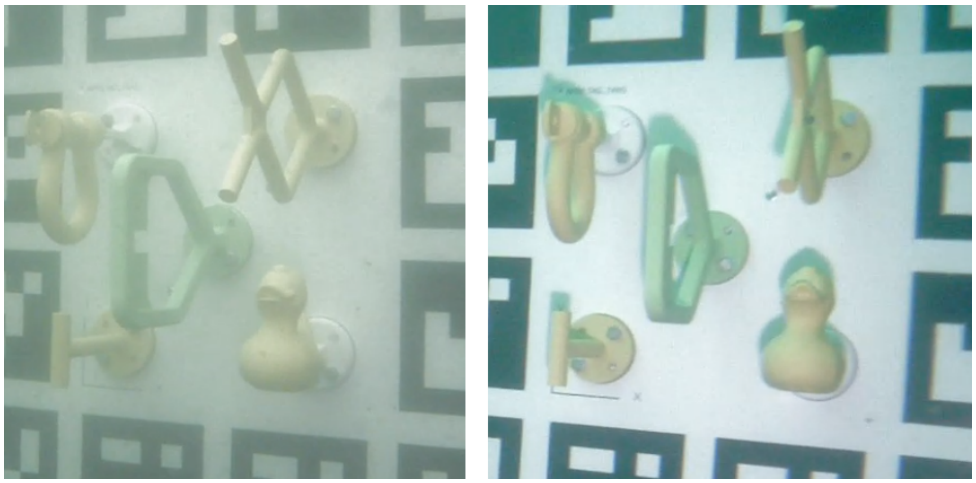


Figure 4.18: Visibility conditions at the two different locations

In total 64 minutes of raw footage were recorded from these locations. 49 minutes from location 1 and 15 minutes from location 2. In the data that is published with annotations 13 minutes are from location 1 and 8 minutes are from location 2. What varies between them is mostly the level of visibility. There is significantly more suspended solids in water at location 1, as seen in figure 4.18.

4.1.6 Labeling

Labeling is done in the COCO format for convenience. There are no universal standards for neither 2D detection or 3D pose detection datasets.

For 2D detection I have used COCO because it is widespread and can be read by

many libraries. The datasets are transformed into the COCO format using the scripts detailed in 4.3.3. This is also perform some data cleaning, removing images where the mask touches the outer boundary of the image. The result is a new folder containing images and a JSON-file containing all relevant information about the dataset, including object categories, masks and bounding boxes for each image. Figure 4.19 show a raw image from the dataset with annotation information drawn onto it.

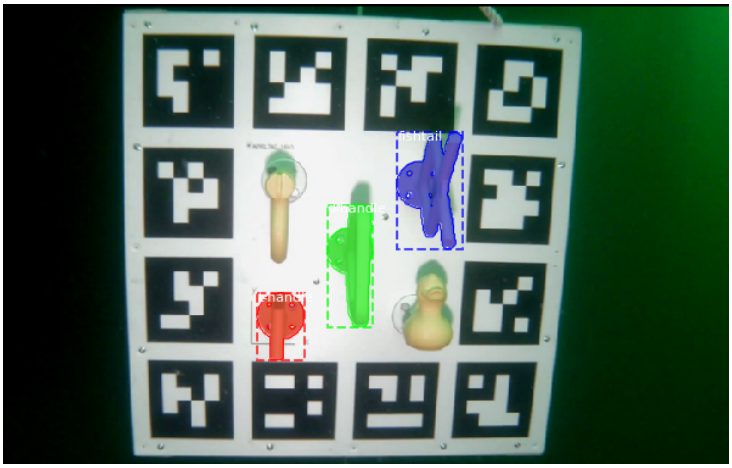


Figure 4.19: A image from the dataset with annotation information drawn on top.

For 3D detection the most widespread is the BOP format, but there is currently no script to this dataset to BOP. Instead the pose is added as a homogeneous transformation matrix "T" to the COCO annotations.

4.2 Training a pose estimator

The next step to realize the envisioned framework would be to a train a suitable neural network to perform pose estimation. I have trained a 2D instance segmentation network on the dataset, but no pose detection method due to time constraints. A 2D object detection or instance segmentation method is a necessary first step to use most pose estimation networks.

2D object detection

As mentioned in section 3.2, pose detection network generally take only a cropped region of interest as input. For this project Mask R-CNN is trained to find segmentation masks on the newly created dataset. To avoid having to train the network from scratch, transfer learning it utilized. Transfer learning is doing training on top of a already trained model. Typically, the last few layers of a model is reset and trained again, but the first layers in the pipeline reuse old weights. The benefit is that training require much less computation and training images. A neural net learn to extract image features through many different layers. The most fundamental feature-extraction properties are typically very general. In

this particular case transfer learning is performed on top a model trained on the official COCO dataset.

In this project I have used the "official" Mask R-CNN implementation from https://github.com/matterport/Mask_RCNN. Having a dataset annotated in the COCO format simplifies training somewhat, but it is still necessary to create a custom class that acts as the interface between the network and your dataset. This class is based on the template class `Config` from the repository and handles loading the image files along with annotations. Further details are given in section 4.3.4.

3D pose estimation

While there are ready made implementations of methods available, these all require some work to use. The first step is how the data is structured and annotated. The BOP competition offers a template for this that I would most likely have used.

Secondly this training requires significantly more computing resources than transfer learning of Mask R-CNN. The methods reviewed generally use large GPU clusters running for hours and days to train a competitive model for benchmarking. Oceanlab has a computer with a Geforce RTX 3090 that could most likely have been used. Alternatively a cloud computing service like Amazon Web Services.

4.3 Implementation details

4.3.1 Poster generator

The poster is generated by `generate_fiducial_poster.py` included in the appendix and Github repository. Parameters are detailed in table 4.5 and illustrated in figure 4.10. The user specifies everything in millimeters and sets the desired resolution (DPI). The script handles all conversions to make sure the end product is true to scale. The poster is first created as a raster image, as the script relies on a function from OpenCV to generate fiducials and this outputs raster images. But in order to be printed accurately to scale the poster must be converted to vector graphics. Professional printers will only guarantee that prints are to scale if vector graphics is used. Vector graphics is fundamentally different format than raster graphics. It stores graphics as geometric shapes defined by their coordinates on a Cartesian plane. This conversion is performed automatically using the open source tool Potrace. The accuracy of the conversion is verified in vector graphics editing software such as Inkscape.

4.3.2 Fiducial pose estimator

A C++ implementation of RBOT by the author is available on GitHub under the GNU public license. This implementation is used as a basis for creating the hybrid fiducial

Table 4.5: Fiducial poster script parameters

Parameter	**Description**	**Default**
N_TAGS	Number of tags on each side	4
TAG_SIZE	Size of the fiducial	140 mm
TAG_KEY	What fiducial dictionary to use	APRILTAG_16h5
MARGIN_SIZE	Distance between fiducials	30 mm
HOLE_DISTANCE	Model screw hole distance from model base	24 mm
HOLE_DIAMETER	Size of dot marking screw hole	1 mm
MODEL_BASE _DIAMETER	Size of the outline marking the model base	70 mm
MODEL_BASE _DISPERSION	If five sockets, distance of model socket from center	300 mm
FIVE_SOCKETS	One or five sockets	True
LINE_THICKNESS	Line thickness	1 mm
BORDERSIZE	Thickness or poster crop border	0.2 mm
font_scale	Font size	30
DPI	Dots per inch	1440

pose estimator. The `main.cpp` is heavily modified, but mainly does function calls to the library. Otherwise it is almost unchanged. The only difference is a change to `model.cpp` to shift the reported pose of a model to its origin, as to match how the fiducial pose estimator does. The RBOT default is to report the pose of a model from its 3D bounding box center.

A flowchart of how RBOT and fiducials are used to extract poses is given in figure 4.20. A configuration file in the JSON format specifies details such as what model to track, which video to use and where to save the data. For each frame in the video both RBOT and the fiducial pose estimator will attempt to get the pose of a selected model. If the poses are roughly equal then both the raw frame and binary segmentation mask is saved. The segmentation mask is created using the function `renderSilhouette()` inside RBOT that utilize its 3D rendering engine to mask a silhouette without shading on top of a black image. If there is a mismatch between the poses then no images are saved, but the pose estimates are still logged. If there has been 10 consecutive pose mismatches then it will attempt to reset RBOT to the pose produced by the fiducial pose estimator.

RBOT is implemented in C++ which necessitates that the fiducial pose estimator must be as well. This is the code in the `fiducial_pose` folder in the GitHub repository. The pose estimator is a class and its flowchart is seen in figure 4.21. To initialize it needs JSON files containing camera intrinsics, measurements of the poster (created by

`generate_fiducial_poster.py`) and what fiducial dictionary to use. It then creates an array the T_{to} transformation matrices for each tag.

For each frame `detectMarkers()` takes the current image frame as input and returns a list containing the corner coordinates of detected of markers and their ID. This is further given to `estimatePoseSingleMarkers()` which return translation and rotation vectors that is transformed into the transformation matrix T_{ct} describing the pose of each tag. Each tag is then transformed in the model pose using the method described in section 4.1.1. After this it will do the outlier detection method described in section 4.1.1. The inliers are averaged and it returns a new pose object containing the transformation matrix T_{cm} and the number of fiducials detected in the current frame.

All files in the folder `fiducial_pose` is custom except `json.hpp`. JSON files are read using the open-source library "JSON for modern C++".

4.3.3 Dataset annotation

Dataset files are handled and annotated using the scripts in the `dataset` folder. `make_coco.py` will parse the output of the fiducial pose estimator and create a dataset annotated in the COCO format, slighty alterted to accommodate pose detection. This is done by using the binary segment mask created by the RBOT 3D engine. This contour is tracked by OpenCV `findContours()` which returns a list of pixel coordinates forming a polygon in the shape of the mask. By default this is then converted to a different binary mask in the run-length encoding format using the `pycocotools` library. Alternatively the script can also output the polygon coordinates as segment mask. The script will also add the saved pose to the annotation. This is not part of the official COCO format, which is designed with 2D object detection in mind.

`split_coco.py` will take a dataset saved in the COCO format and split samples of it into fractions. The purpose is to create a train, validation and test set used to evaluate and train neural networks. The size of each is set as parameters to the script. The last script `unify_coco.py` will join two or more COCO datasets into a unified set. Old image filenames are lost as it will create new ones, 1000000+i for images in the first series, 2000000+i for images in the second and so on.

4.3.4 Training Mask R-CNN

The implentation I use require that the `Config` class in `coco.py` is customized to handle loading of images and annotations, running training and evaluation. The result is the `handles.py` script.

The script was modified for transfer learning by excluding layers "mrcnn_class_logits", "mrcnn_bbox_fc", "mrcnn_bbox", "mrcnn_mask" when loading COCO weights. Further training was restricted to the top layers of network.

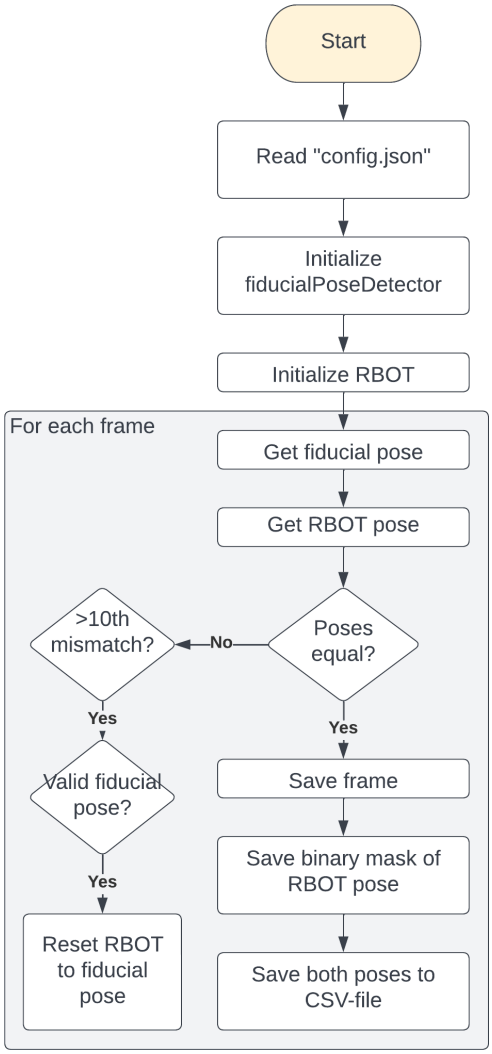


Figure 4.20: A flowchart of `main.cpp`

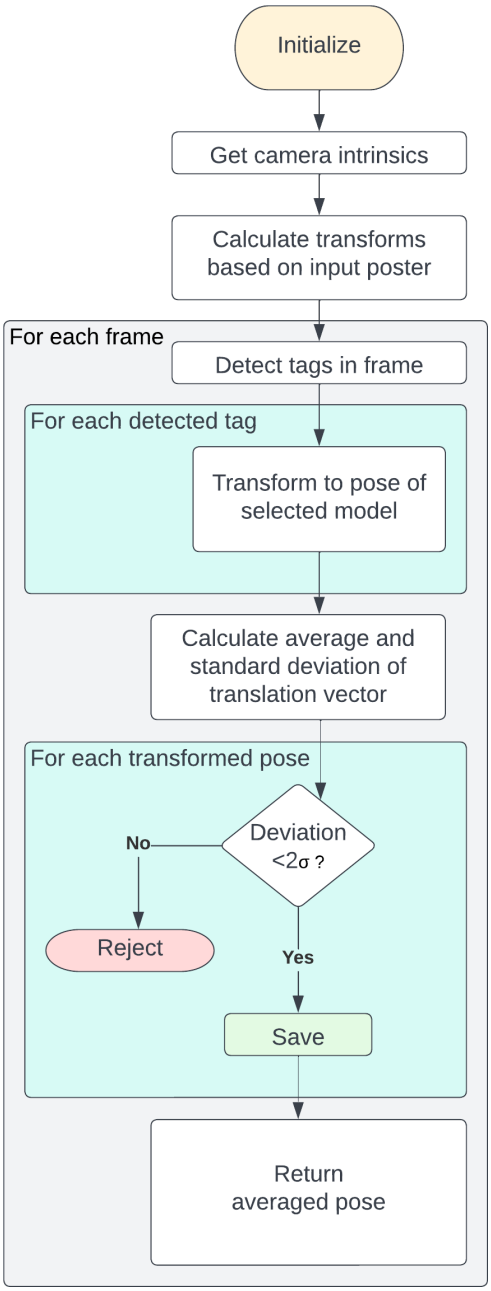


Figure 4.21: A flowchart of how the `fiducialPoseDetector` class works.

Other changes were mostly setting directories and changing hyperparameters such as number of categories and training duration. Function headers and calls also had to be changed as many had parameters that are specific to the COCO-dataset. Specifically calls to use special subsets of COCO based on year.

Training was done using CPU only for roughly six hours. 30 epochs with 60 iterations each.

4.4 System specifications

All testing and training was performed on a "Dell XPS 15 9500" laptop, specifications in table 4.6.

Table 4.6: System specifications of laptop used for testing and training.

Specifications
Intel Core i7-10750H
2x8GB DDR4-2933MHz
NVIDIA GeForce GTX 1650 Ti 4GB

Chapter 5

Results

Results are reported with poses of the shackle and duck model. While they worked well on plain RBOT, they no longer work when paired with the fiducial pose detector. Most likely because it takes up too much memory and the operating system kills the process automatically. This might work on a more powerful computer, but I have not been able to test it.

5.1 Dataset

5.1.1 Fiducial pose estimate accuracy

Transforming all pose estimates into the poster center show a certain spread even in air. In water the spread is generally worse, and dependant on visibility conditions. This is shown in 5.1 where the transformed poses are drawn on the image frame. As a frame of reference, the base of the model is 7 cm in diameter. While these images are only snapshots, the general tendency is a spread a of a few centimeters. Rotation is harder to gauge, but is clearly "wobbly" when frames are viewed in sequence.

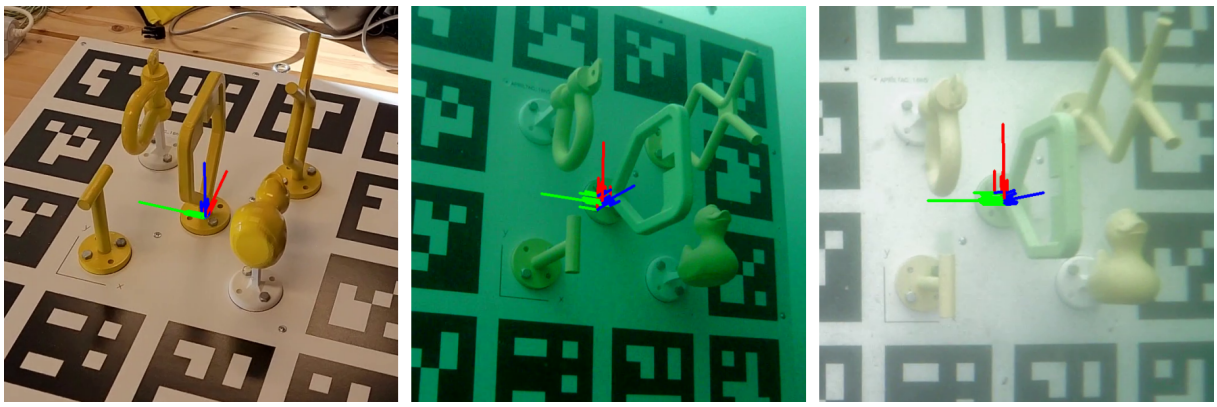


Figure 5.1: All transformed poses drawn onto the image frame. Frames with short lines represent individual transformed fiducial poses. The frame with long lines represent the average.

Table 5.1 and 5.2 summarize the deviation in translation and rotation between the fiducial poses and RBOT. Rotation is represented as Euler angles in degrees. In each cell the median is on the left and standard deviation on the right. The fishtail for location 1 is left out because of errors in the data. Deviation here is defined as

$$T_{error} = T_{fiducial} - T_{RBOT}$$

Figure 5.2 show five minutes of translation and rotation pose estimates from location 1. Blank spaces are where either pose estimation method draws a blank.

It appears that translation is fairly consistent in the image plane, that is the X and Y axis. In the Z-axis, representing distance to, the fiducial markers generally overestimate in comparison to RBOT. A difference in 5 centimeters is consistent between the locations. This is even larger for the T-handle which might be harder for RBOT to track due to its small size and simple shape.

For rotation along the X-axis spread is much greater than either Y og Z. Again the T-handle appear to be harder to track than the D-handle or Fishtail.

Table 5.1: Summary of metrics from location 1

	Translation [cm]						Rotation [degree]					
	X		Y		Z		X		Y		Z	
D-handle	-1.0	6.7	-0.0	4.0	5.2	20.4	-6.0	114.1	0.9	6.8	1.8	22.0
T-handle	-1.2	7.8	1.8	6.0	11.6	18.3	-23.5	100.0	1.7	9.7	2.5	23.7

Table 5.2: Summary of metrics from location 2

	Translation [cm]						Rotation [degree]					
	X		Y		Z		X		Y		Z	
D-handle	1.5	8.3	-0.3	2.1	6.3	11.4	-2.9	39.9	-0.8	5.2	1.5	15.5
Fishtail	2.5	12.4	-0.9	3.7	8.0	15.7	-7.3	54.7	-1.5	7.8	2.8	18.9
T-handle	1.4	9.3	0.5	3.2	17.9	29.4	-27.3	98.1	-4.9	13.3	3.9	25.3

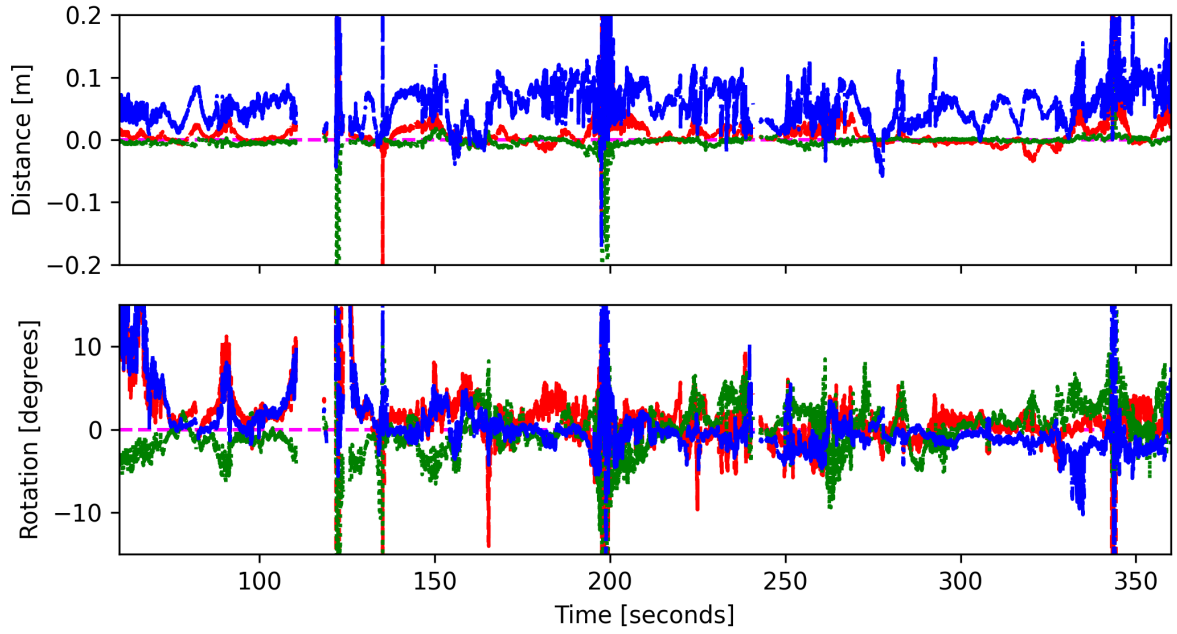


Figure 5.2: The deviation in translation and rotation between RBOT and fiducials for the D-handle at location 1.

5.1.2 Fiducial dictionary

Figure 5.3: Frame 100, 300 and 500 from experiment #5

Tests were also made to determine the fiducial dictionary with the best detection rate. Figure 5.3 show frames from the video used in experiment #1. A4 sheets with different dictionaries was printed using a normal laser printer. These were hung on a wall and filmed while the camera moved backwards.

Results from the test is presented in figure 5.4. On the y-axis is the number of detected tags. The x-axis is cropped to the interval where detection rate started to decline and reached zero. The x-axis unit is the frame number in the video, but also reflect increasing distance from the tags.

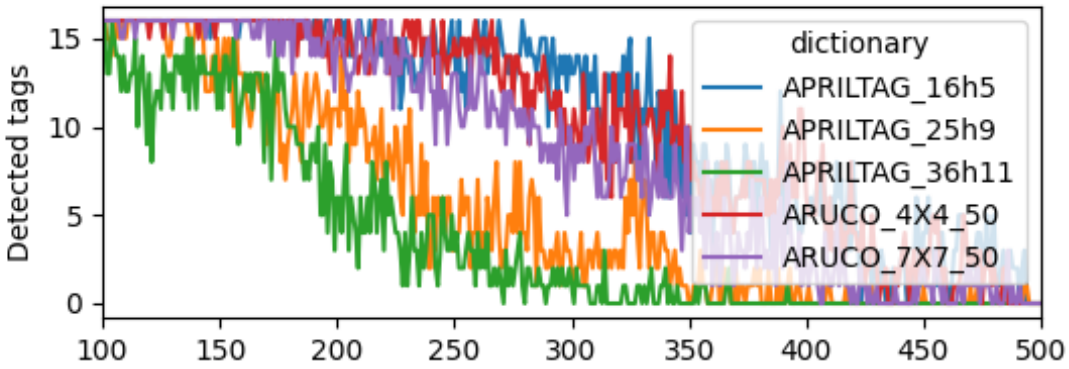


Figure 5.4: Number of detected tags over time in experiment #5

Mean detection rates from the six trials are presented in table 5.3. "APRILTAG_16h5" appear to be the most robust, closely followed by "ARUCO_4X4_50". These are both 4x4 matrices. Overall detection rate is presented in table 5.4. For both ArUco and AprilTag it is clear that matrices with fewer cells are more easily detected. The downside of a smaller matrix though is that it gives fewer unique identities. This does not really matter in this particular application however. For performing pose detection, all markers could have the same identity.

Table 5.3: Average detection rate in each experiment

Experiment	Dictionary	N tags
1	APRILTAG_16h5	**10.41**
	APRILTAG_25h9	7.02
	APRILTAG_36h11	6.24
	ARUCO_4X4_50	10.32
	ARUCO_7X7_50	8.94
2	APRILTAG_16h5	11.65
	APRILTAG_25h9	8.41
	APRILTAG_36h11	7.81
	ARUCO_4X4_50	**11.67**
	ARUCO_7X7_50	10.94
3	APRILTAG_16h5	**13.35**
	APRILTAG_25h9	7.95
	APRILTAG_36h11	7.19
	ARUCO_4X4_50	12.79
	ARUCO_7X7_50	11.83
4	APRILTAG_16h5	**11.95**
	APRILTAG_25h9	7.15
	APRILTAG_36h11	6.77
	ARUCO_4X4_50	11.28
	ARUCO_7X7_50	10.19
5	APRILTAG_16h5	**7.52**
	APRILTAG_25h9	4.85
	APRILTAG_36h11	4.03
	ARUCO_4X4_50	7.33
	ARUCO_7X7_50	6.50
6	APRILTAG_16h5	**7.47**
	APRILTAG_25h9	3.39
	APRILTAG_36h11	2.77
	ARUCO_4X4_50	6.15
	ARUCO_7X7_50	6.03

Table 5.4: Average detection rate across all experiments

Dictionary	N tags
APRILTAG_16h5	10.14
ARUCO_4X4_50	9.69
ARUCO_7X7_50	8.84
APRILTAG_25h9	6.32
APRILTAG_36h11	5.65

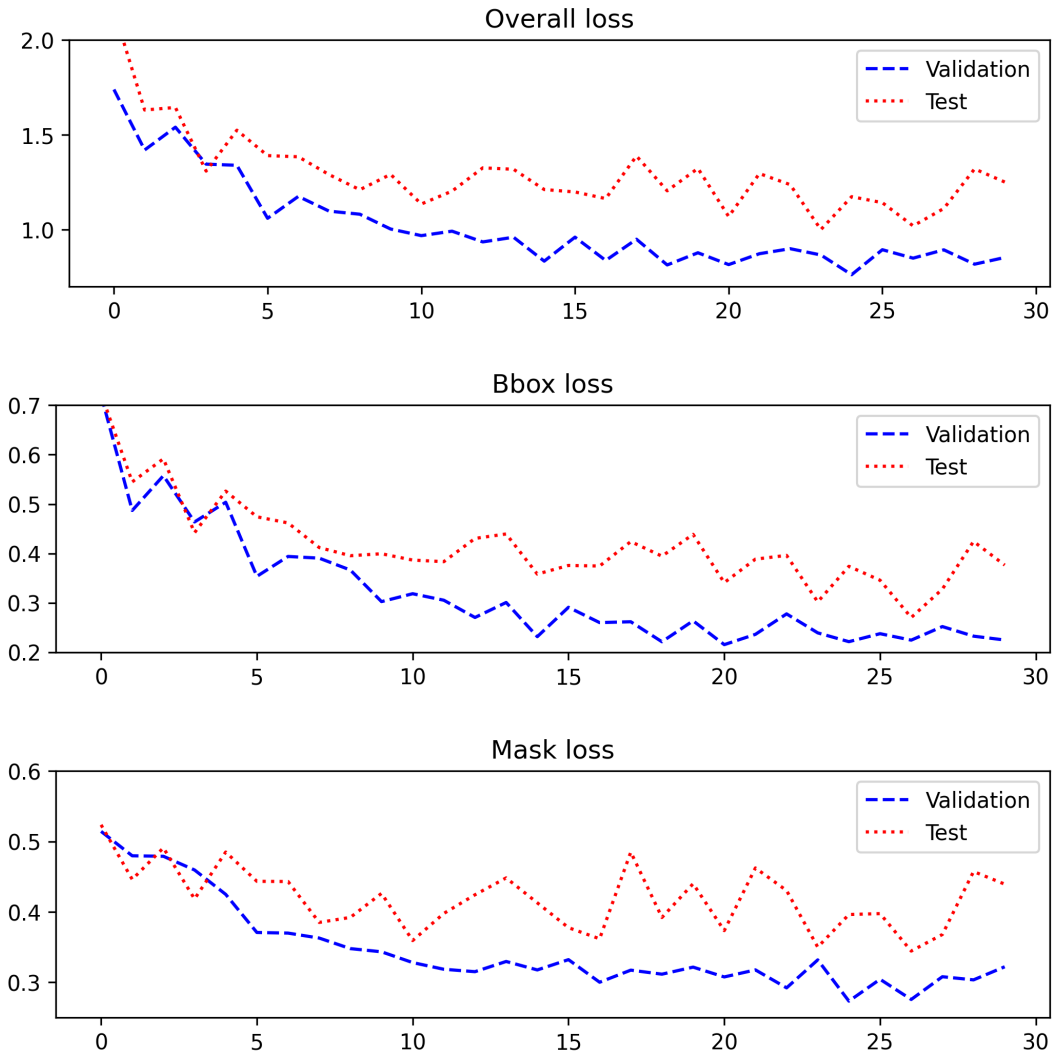


Figure 5.5: Loss on validation and test data after each epoch of training.

5.2 Object detection

Mask R-CNN was trained for 2D object detection using the dataset. The network predicts segmentation masks and create a bounding box based on that. Loss across epochs are presented in figure 5.5. It calculates loss for segmentation masks and bounding boxes separately. Bounding box loss appear to improve more than mask loss. Furthermore validation loss is significantly better than training loss. The validation set is used during training for parameter tuning, while the test set is used to evaluate the model afterwards.

Average precision and recall is presented in table 5.5. This the stats for the highest scoring model, which occoured in epoch 26.

Figure 5.6 present a few examples of failed predictions. Here the model is able to predict all instances of objects, but create a false positives as well. Either the shackle or duck model is falsely classified as handles.

Some examples of good predictions can be seen in figure 5.7. Here there are no

Table 5.5: Mean Average Precision and recall after training

	IoU	Score
mAP	0.50 : 0.95	0.354
	0.50	0.687
	0.75	0.280

	IoU	Score
Average Recall	0.50 : 0.95	0.584

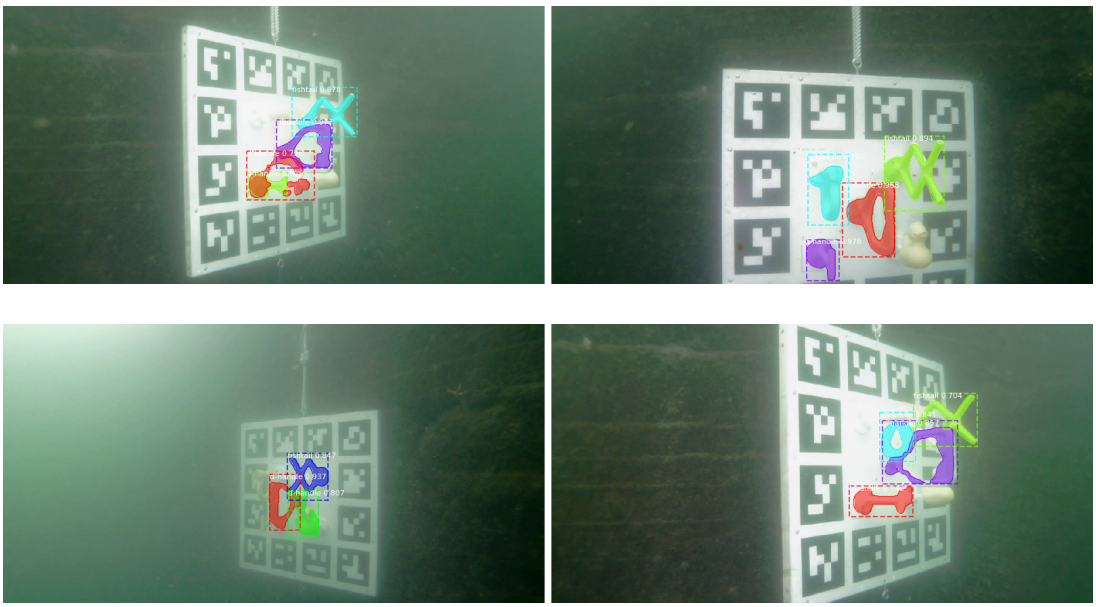


Figure 5.6: A sample of failed mask predictions from the test set.

false predictions, classifications are correct and although masks are sometimes patchy the bounding box fits precisely around the model.

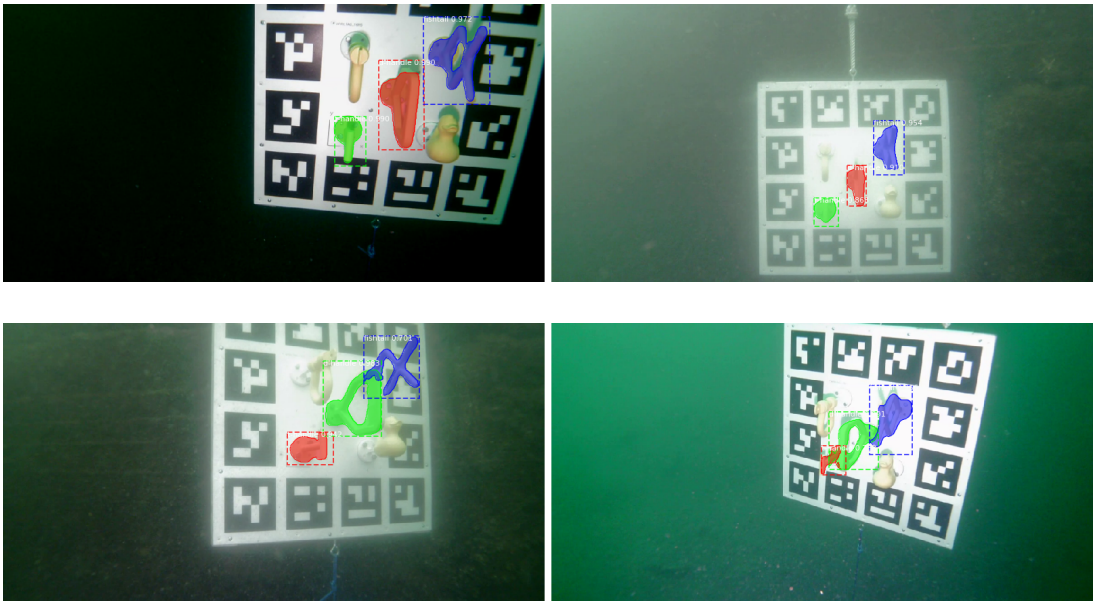


Figure 5.7: Successful mask predictions.

Chapter 6

Discussion

The initial idea was to use the fiducial markers as ground truth to create a dataset for training neural nets and evaluate RBOT. This based on the results reported by Henriksen et al [22], where displacement had millimeter accuracy and yaw was within one degree. They used acoustic positioning as ground truth to measure the accuracy of pose estimation with fiducials. While the current project does not have access to a sufficiently accurate positioning system to compare results quantitatively, it seems obvious that these fiducial pose estimates are far less accurate. This led me to turn things around and use RBOT as ground truth in the dataset when the fiducial poses roughly correspond.

6.1 Dataset

There are systematic biases in the dataset that could reduced in a future version. Because the ROV has a limited capability to look downward the poster was designed to hang vertically. This should be balanced out by footage of the poster lying horizontally on the ground.

A significant flaw in the current dataset is that a pose is recorded only when RBOT and the fiducial pose overlap. This means that even though all models are visible in a given frame, there might only be a pose for one or two models. This greatly skews the accuracy metrics of the 2D detection network. If for instance a frame has the D-handle, T-handle and fishtail all visible, but only a pose for the D-handle. The classifier might correctly guess the mask of the T-handle and fishtail as well. However these will be counted as false positives in the evaluation metric.

Evaluation of the 2D object detection network also revealed a large gap between scores on the validation and testing data. This could be caused by several factors, but is often a symptom of overfitting. That is when the model is not capable of generalizing to new cases. This could be caused by unbalanced training data. There might be more samples of the handles in certain configurations. The idea was to move back and forth in front of the poster in a random fashion and get a nice distribution of poses. However the two

pose estimation methods correspond much more under certain conditions. At 30 frames per second it quickly builds up a large amount of images if poses align all the time. In other situations there might be no correspondence for a long time.

6.2 UUV pose estimation framework

Using a method like RBOT that focus on silhouette and regional features rather than keypoint features appear to work very well under water. The method is also very efficient and can run at easily run real time on a laptop. In comparison neural nets are significantly slower. Although some papers report they can run up 10-15 Hz this relies on a powerful GPU. For ROVs this might be possible as they are connected through the umbilical to the surface. Most AUVs will not be capable of this however, although this might be changing with small embedded computers such as the Nvidia Jetson.

Both RBOT and the most suited 6D pose estimation neutral net should be implemented as ROS (Robot Operating System) nodes. ROS is popular robot "middleware" operating system. It lets nodes run independently of each other and facilitates communication between them. Nodes can run on separate machines and communicate via network. Although both RBOT and neural nets are computationally heavy, this burden could be distributed between two machines. This would let them work in tandem, such that the neural net would supply RBOT with a initial pose whenever tracking was lost.

Chapter 7

Conclusion

The literature review performed in section 3.1 focused on intervention-AUV projects from the last two decades. Most projects used pose estimation methods that were very particular to a certain task. For instance, the RAUVI project [58] primarily rely on color histograms to retrieve a red box. The TWINBOT [56] and MARIS [68] projects focus on manipulation of pipes and rely on assumptions of their particular color and geometry. The more versatile methods used feature descriptors like ORB in combination with traditional 2D-3D algorithms such as perspective n-point. In the TRIDENT project [3] ORB performed reasonably well in pool trials, but not in the ocean with realistic visibility conditions.

This initially led me to the idea of developing a method using perspective n-point and RANSAC, but by using learned feature descriptors rather than ORB. Learned feature descriptors are trained in neural nets and have proved to be more effective than hand-crafted descriptors such as ORB. However, similar methods has already been explored by Crivellaro *et al.* [12] and Tremblay *et al.* [76]. Point-based methods also appear to perform poorly in pose estimation benchmarking tests [26].

Finally it has some fundamental issues when applied in an underwater environment. ROVs are mostly used to manipulate man-made objects that are simple shapes with uniform color and very little texture. Hence there is little for a feature descriptor to work with. Methods that rely on local features such as keypoints are likely to perform poorly. And while these objects are designed with CAD tools, the models might not have texture added. Even if textures were added, equipment could be painted with a different color during repairs. For this application then, methods that rely on 3D CAD models without texture are more apt.

The methods reviewed here are either pose detection or pose refinement methods. Only pose detection methods can find the pose from a single image, but are too slow to run in realtime. Pose refinement methods make assumptions based on the pose in the previous frame and rely on being initialized. But this makes refinement methods more efficient and able to run in realtime on simple hardware. A flexible and robust pose

estimation system would probably have to combine estimation and tracking.

The dataset created in this project can be used to this purpose. Although generating synthetic images is most definitely more efficient than creating a real dataset, methods will have to try on the real world eventually. Also augmenting a synthetic training set with images from such a dataset can help lessen the reality gap.

7.1 Future work

- The database should be used to train a number of existing 6D pose estimation neural nets and evaluate how they perform in water. Results from the BOP challenge hint that machine learning methods are improving at a faster rate than traditional methods. However, neural nets and pose detection will forever be more computationally heavy than tracking. The best immediate solution seems to be a hybrid of a neural net pose detector and template based tracker. Neural networks not relying on textures seem like the best option on subsea equipment, for instance Pix2Pose. CosyPose performed much better in BOP, largely because it integrates pose estimates from many different views. The authors further claim this method works independent of what exact pose estimator is used (they use a variant of DeepIM). A textureless method like Pix2Pose then could conceivably be integrated with CosyPose.

- RBOT is fundamentally a pose tracking method and relies on initialization. In the lab this can be done manually, but an application usable in real life would have to do this automatically. By using a separate object detection algorithm, RBOT could be supplied with either a bounding box or even segmented image to restrict its search. And because objects used in subsea industry are generally yellow or orange, the histograms could be initialized with these colors, or by user choice. When using color histograms deep underwater one might be able to make an assumption that would be unreasonable in most other environments. That is if the UUV is deep enough that most sunlight is absorbed and it primarily relies on the artificial lighting coming for the UUV itself. If the color temperatures of the UUV lights are known, one could use that information to perform white balancing on the captured images. That would mean the hue of an object never changes, only the luminosity. This is particularly useful if using the HSV color space rather than RGB. HSV is an acronym for hue, saturation and value. If imagining a surface with a certain hue lit by a white light, "value" is the strength of that light. Unlike RGB where the intensity of red, green and blue is represented individually, in HSV hue and value are separate properties. HSV histograms of oriented gradients were used successfully for image segmentation by [38, 39] in the MARIS I-AUV project.

- Models occluding the fiducial markers and causing erroneous box detections (for

instance figure 4.6) could be fixed in a future project by either making the poster less crowded or using another fiducial detection method. If there was more space around the models they wouldn't occlude the markers in the first place. But other detection algorithms such as Olson [50] claim to do robust marker detection despite occlusion. I could also have made a custom algorithm that performs some kind of error correction. For instance, all the boxes are approximate parallelograms because of how perspective works. All the horizontal lines in all of the boxes should be more or less parallel on an undistorted image. Undistoring an image is not a problem as the camera intrinsics are necessary for pose detection anyway. And so the average gradient of each horizontal line could be used to identify outliers, instead of the each element in the translation vector. Likewise with vertical lines.

- Given the poor visibility conditions, "active" markers might create more reliable fiducial pose estimates. Active fiducials use LED diodes to create patterns the algorithm can use as feature points when solving the PnP for pose. A custom active marker poster could be produced using a CNC milling machine creating precise holes for diodes and screws that attach the model. Using the entire area of the poster as a square would create more reliable poses. The poster could also be divided into sub-squares that each create a recognizable pattern.

Bibliography

[1] Andreas Birk et al. 'Dexterous Underwater Manipulation from Onshore Locations: Streamlining Efficiencies for Remotely Operated Underwater Vehicles'. In: *IEEE Robotics Automation Magazine* 25.4 (2018), pp. 24–33. DOI: 10.1109/MRA.2018.2869523.

[2] Magnus Bjerkeng et al. 'ROV Navigation in a Fish Cage with Laser-Camera Triangulation'. In: *Journal of Marine Science and Engineering* 9.1 (2021). ISSN: 2077-1312. DOI: 10.3390/jmse9010079. URL: https://www.mdpi.com/2077-1312/9/1/79.

[3] Francisco Bonin-Font et al. 'Visual sensing for autonomous underwater exploration and intervention tasks'. In: *Ocean Engineering* 93 (2015), pp. 25–44. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2014.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0029801814004090.

[4] P. Bouthemy. 'A maximum likelihood framework for determining moving edges'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.5 (1989), pp. 499–511. DOI: 10.1109/34.24782.

[5] Eric Brachmann. *6D Object Pose Estimation using 3D Object Coordinates [Data]*. Version V1. 2020. DOI: 10.11588/data/V4MUMX. URL: https://doi.org/10.11588/data/V4MUMX.

[6] Eric Brachmann et al. 'Learning 6D Object Pose Estimation Using 3D Object Coordinates'. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 536–551. ISBN: 978-3-319-10605-2.

[7] Arnau Carrera et al. 'An Intervention-AUV learns how to perform an underwater valve turning'. In: *OCEANS 2014 - TAIPEI*. 2014, pp. 1–7. DOI: 10.1109/OCEANS-TAIPEI.2014.6964483.

[8] G. Casalino et al. 'Dexterous underwater object manipulation via multi-robot cooperating systems'. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*. Vol. 4. 2001, 3220–3225 vol.4. DOI: 10.1109/ROBOT.2001.933114.

[9] Giuseppe Casalino et al. 'Underwater Intervention Robotics: An Outline of theItalian National Project MARIS'. In: *Marine Technology Society Journal* 50.4 (2016), pp. 98–107. ISSN: 0025-3324. DOI: 10.4031/MTSJ.50.4.7.

[10] Yu-Wei Chao et al. 'DexYCB: A Benchmark for Capturing Hand Grasping of Objects'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[11] Robert D. Christ and Robert L. Wernli. 'Chapter 2 - The Ocean Environment'. In: *The ROV Manual (Second Edition)*. Ed. by Robert D. Christ and Robert L. Wernli. Oxford: Butterworth-Heinemann, 2014, pp. 21–52. ISBN: 978-0-08-098288-5. DOI: https://doi.org/10.1016/B978-0-08-098288-5.00002-6. URL: https://www.sciencedirect.com/science/article/pii/B9780080982885000026.

[12] Alberto Crivellaro et al. 'A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images'. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4391–4399. DOI: 10.1109/ICCV.2015.499.

[13] Shaowei Cui et al. 'Real-Time Perception and Positioning for Creature Picking of an Underwater Vehicle'. In: *IEEE Transactions on Vehicular Technology* 69.4 (2020), pp. 3783–3792. DOI: 10.1109/TVT.2020.2973656.

[14] Zhuang Dai et al. 'A Comparison of CNN-Based and Hand-Crafted Keypoint Descriptors'. In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2399–2404. DOI: 10.1109/ICRA.2019.8793701.

[15] Maximilian Denninger et al. *BlenderProc*. 2019. arXiv: 1911.01911 [cs.CV].

[16] Tobias Doernbach et al. 'High-Fidelity Deep-Sea Perception Using Simulation in the Loop**TD, formerly known as Tobias Fromm 0000-0001-6488-8211, AGC 0000-0002-7132-1026 and CAM 0000-0003-1895-987X contributed equally to this work and share first authorship. The research leading to the presented results has received funding from the European Union's Horizon 2020 Framework Programme (H2020) within the project (ref.: 635491) "Effective dexterous ROV operations in presence of communication latencies (DexROV)".' In: *IFAC-PapersOnLine* 51.29 (2018). 11th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2018, pp. 32–37. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.09.465. URL: https://www.sciencedirect.com/science/article/pii/S2405896318321542.

[17] J. Evans et al. 'Autonomous docking for Intervention-AUVs using sonar and video-based real-time 3D pose estimation'. In: *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492)*. Vol. 4. 2003, 2201–2210 Vol.4. DOI: 10.1109/OCEANS.2003.178243.

[18] J. C. Evans et al. 'Docking techniques and evaluation trials of the SWIMMER AUV: an autonomous deployment AUV for work-class ROVs'. In: *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*. Vol. 1. 2001, 520–528 vol.1. DOI: 10.1109/OCEANS.2001.968776.

[19] J. Ferrer et al. 'Large-Area Photo-Mosaics Using Global Alignment and Navigation Data'. In: *Oceans Conference Record (IEEE)* (Jan. 2007). DOI: 10.1109/OCEANS.2007.4449367.

[20] Sergio Garrido-Jurado et al. 'Generation of fiducial marker dictionaries using Mixed Integer Linear Programming'. In: *Pattern Recognition* 51 (Oct. 2015). DOI: 10.1016/j.patcog.2015.09.023.

[21] Kaiming He, Jian Sun and Xiaoou Tang. 'Single Image Haze Removal Using Dark Channel Prior'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.12 (2011), pp. 2341–2353. DOI: 10.1109/TPAMI.2010.168.

[22] *Vision Based Localization for Subsea Intervention*. Vol. Volume 7A: Ocean Engineering. International Conference on Offshore Mechanics and Arctic Engineering. V07AT06A023. June 2017. DOI: 10.1115/OMAE2017-61773. eprint: https://asmedigitalcollection.asme.org/OMAE/proceedings-pdf/OMAE2017/57731/V07AT06A023/2533707/v07at06a023-omae2017-61773.pdf. URL: https://doi.org/10.1115/OMAE2017-61773.

[23] João F. Henriques et al. 'High-Speed Tracking with Kernelized Correlation Filters'. In: *CoRR* abs/1404.7584 (2014). arXiv: 1404.7584. URL: http://arxiv.org/abs/1404.7584.

[24] Khadidja Himri et al. 'Object Recognition and Pose Estimation using Laser scans For Advanced Underwater Manipulation'. In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018, pp. 1–6. DOI: 10.1109/AUV.2018.8729742.

[25] Stefan Hinterstoisser et al. 'Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes'. In: *Computer Vision – ACCV 2012*. Ed. by Kyoung Mu Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 548–562. ISBN: 978-3-642-37331-2.

[26] Tomas Hodan et al. *BOP: Benchmark for 6D Object Pose Estimation*. 2018. arXiv: 1808.08319 [cs.CV].

[27] Tomáš Hodaň et al. 'BOP Challenge 2020 on 6D Object Localization'. In: *European Conference on Computer Vision Workshops (ECCVW)* (2020).

[28] Tomáš Hodaň et al. 'T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects'. In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017).

[29] Armin Hornung et al. 'OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees'. In: *Autonomous Robots* (2013). Software available at http://octomap.github.com. DOI: 10.1007/s10514-012-9321-0. URL: http://octomap.github.com.

[30] Roman Kaskman et al. 'HomebrewedDB: RGB-D Dataset for 6D Pose Estimation of 3D Objects'. In: *International Conference on Computer Vision (ICCV) Workshops* (2019).

[31] Alex Kendall and Roberto Cipolla. 'Modelling uncertainty in deep learning for camera relocalization'. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4762–4769. DOI: 10.1109/ICRA.2016.7487679.

[32] B. Kieft et al. 'How Autonomous Vehicles are Enhancing Ocean Science'. In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018, pp. 1–5. DOI: 10.1109/AUV.2018.8729705.

[33] Daniel Köhntopp et al. 'Segmentation and classification using active contours based superellipse fitting on side scan sonar images for marine demining'. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 3380–3387. DOI: 10.1109/ICRA.2015.7139666.

[34] Kevin Köser and Udo Frese. 'Challenges in Underwater Visual Navigation and SLAM'. In: *AI Technology for Underwater Robots*. Ed. by Frank Kirchner et al. Cham: Springer International Publishing, 2020, pp. 125–135. ISBN: 978-3-030-30683-0. DOI: 10.1007/978-3-030-30683-0_11. URL: https://doi.org/10.1007/978-3-030-30683-0_11.

[35] Y. Labbe et al. 'CosyPose: Consistent multi-view multi-object 6D pose estimation'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020.

[36] Zhigang Li et al. *Robust RGB-based 6-DoF Pose Estimation without Real Pose Annotations*. 2020. arXiv: 2008.08391 [cs.CV].

[37] Diego Lirman et al. 'Development and application of a video-mosaic survey technology to document the status of coral reef communities'. In: *Environmental monitoring and assessment* 125 (Feb. 2007), pp. 59–73. DOI: 10.1007/s10661-006-9239-0.

[38] Dario Lodi Rizzini et al. 'Integration of a stereo vision system into an autonomous underwater vehicle for pipe manipulation tasks'. In: *Computers & Electrical Engineering* 58 (2017), pp. 560–571. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2016.08.023. URL: https://www.sciencedirect.com/science/article/pii/S0045790616302233.

[39] Dario Lodi Rizzini et al. 'Investigation of Vision-Based Underwater Object Detection with Multiple Datasets'. In: *International Journal of Advanced Robotic Systems* 12 (June 2015), pp. 1–13. DOI: 10.5772/60526.

[40] Tomasz Łuczyński and Andreas Birk. 'Underwater image haze removal with an underwater-ready dark channel prior'. In: *OCEANS 2017 - Anchorage*. 2017, pp. 1–6.

[41] Tomasz Łuczyński, Max Pfingsthorn and Andreas Birk. 'The Pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings'. In: *Ocean Engineering* 133 (2017), pp. 9–22. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2017.01.029. URL: https://www.sciencedirect.com/science/article/pii/S0029801817300434.

[42] Tomasz Łuczyński et al. '3D grid map transmission for underwater mapping and visualization under bandwidth constraints'. In: *OCEANS 2017 - Anchorage*. 2017, pp. 1–6.

[43] Giacomo Marani, Song Choi and Junku Yuh. 'Underwater Autonomous Manipulation for Intervention Missions AUVs'. In: *Ocean Engineering - OCEAN ENG* 36 (Jan. 2009), pp. 15–23. DOI: 10.1016/j.oceaneng.2008.08.007.

[44] Patrizio Mariani et al. 'Range-Gated Imaging System for Underwater Monitoring in Ocean Environment'. In: *Sustainability* 11.1 (2019). ISSN: 2071-1050. DOI: 10.3390/su11010162. URL: https://www.mdpi.com/2071-1050/11/1/162.

[45] F. Landis Markley et al. 'Averaging Quaternions'. In: *Journal of Guidance, Control, and Dynamics* 30.4 (2007), pp. 1193–1197. DOI: 10.2514/1.28949. eprint: https://doi.org/10.2514/1.28949. URL: https://doi.org/10.2514/1.28949.

[46] Ahmed Mohammed et al. '6D Pose Estimation for Subsea Intervention in Turbid Waters'. In: *Electronics* 10.19 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10192369. URL: https://www.mdpi.com/2079-9292/10/19/2369.

[47] Daniel Motta et al. 'Challenges for Deepwater Operations: An Industry Perspective'. In: *AI Technology for Underwater Robots*. Ed. by Frank Kirchner et al. Cham: Springer International Publishing, 2020, pp. 37–48. ISBN: 978-3-030-30683-0. DOI: 10.1007/978-3-030-30683-0_3. URL: https://doi.org/10.1007/978-3-030-30683-0_3.

[48] Christian A. Mueller et al. 'Robust Continuous System Integration for Critical Deep-Sea Robot Operations Using Knowledge-Enabled Simulation in the Loop'. In: *CoRR* abs/1803.02127 (2018). arXiv: 1803.02127. URL: http://arxiv.org/abs/1803.02127.

[49] Mikkel Cornelius Nielsen, Mari Hovem Leonhardsen and Ingrid Schjølberg. 'Evaluation of PoseNet for 6-DOF Underwater Pose Estimation'. In: *OCEANS 2019 MTS/IEEE SEATTLE*. 2019, pp. 1–6. DOI: 10.23919/OCEANS40490.2019.8962814.

[50] Edwin Olson. 'AprilTag: A robust and flexible visual fiducial system'. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2011, pp. 3400–3407.

[51] A. Palmeiro et al. 'Underwater radio frequency communications'. In: *OCEANS 2011 IEEE - Spain*. 2011, pp. 1–8. DOI: 10.1109/Oceans-Spain.2011.6003580.

[52] Albert Palomer et al. '3D Laser Scanner for Underwater Manipulation'. In: *Sensors* 18.4 (2018). ISSN: 1424-8220. DOI: 10.3390/s18041086. URL: https://www.mdpi.com/1424-8220/18/4/1086.

[53] Narcís Palomeras et al. 'I-AUV docking and intervention in a subsea panel'. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2279–2285. DOI: 10.1109/IROS.2014.6942870.

[54] Jisung Park, Taeyun Kim and Jinwhan Kim. 'Model-referenced pose estimation using monocular vision for autonomous intervention tasks'. In: *Autonomous Robots* 44 (Jan. 2020), pp. 1–12. DOI: 10.1007/s10514-019-09886-9.

[55] Kiru Park, Timothy Patten and Markus Vincze. 'Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation'. In: *CoRR* abs/1908.07433 (2019). arXiv: 1908.07433. URL: http://arxiv.org/abs/1908.07433.

[56] Roger Pi et al. 'TWINBOT: Autonomous Underwater Cooperative Transportation'. In: *IEEE Access* 9 (2021), pp. 37668–37684. DOI: 10.1109/ACCESS.2021.3063669.

[57] François Pomerleau, Francis Colas and Roland Siegwart. 'A Review of Point Cloud Registration Algorithms for Mobile Robotics'. In: *Foundations and Trends® in Robotics* 4 (May 2015), pp. 1–104. DOI: 10.1561/2300000035.

[58] Mario Prats et al. 'Reconfigurable AUV for intervention missions: a case study on underwater object recovery'. In: *Intelligent Service Robotics* 5.1 (2012), pp. 19–31.

[59] Pere Ridao et al. 'Intervention AUVs: The Next Challenge'. In: *IFAC Proceedings Volumes* 47.3 (2014). 19th IFAC World Congress, pp. 12146–12159. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20140824-6-ZA-1003.02819. URL: https://www.sciencedirect.com/science/article/pii/S1474667016435494.

[60] Pere Ridao et al. 'Intervention AUVs: The next challenge'. In: *Annual Reviews in Control* 40 (2015), pp. 227–241. ISSN: 1367-5788. DOI: https://doi.org/10.1016/j.arcontrol.2015.09.015. URL: https://www.sciencedirect.com/science/article/pii/S1367578815000541.

[61] V. Rigaud et al. 'UNION: underwater intelligent operation and navigation'. In: *IEEE Robotics Automation Magazine* 5.1 (1998), pp. 25–35. DOI: 10.1109/100.667323.

[62] Petter Risholm et al. 'Adaptive Structured Light with Scatter Correction for High-Precision Underwater 3D Measurements'. In: *Sensors* 19.5 (2019). ISSN: 1424-8220. DOI: 10.3390/s19051043. URL: https://www.mdpi.com/1424-8220/19/5/1043.

[63] Francisco Romero-Ramirez, Rafael Muñoz-Salinas and Rafael Medina-Carnicer. 'Speeded Up Detection of Squared Fiducial Markers'. In: *Image and Vision Computing* 76 (June 2018). DOI: 10.1016/j.imavis.2018.05.004.

[64] Ethan Rublee et al. 'ORB: An efficient alternative to SIFT or SURF'. In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[65] Pedro J Sanz et al. 'Merbots project: overal description, multisensory autonomous perception and grasping for underwater robotics interventions'. In: *Actas de las XXXVIII Jornadas de Automática* (2017).

[66] Pedro J. Sanz et al. 'TRIDENT: A Framework for Autonomous Underwater Intervention Missions with Dexterous Manipulation Capabilities'. In: *IFAC Proceedings Volumes* 43.16 (2010). 7th IFAC Symposium on Intelligent Autonomous Vehicles, pp. 187–192. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20100906-3-IT-2019.00034. URL: https://www.sciencedirect.com/science/article/pii/S1474667016350546.

[67] Marshall Shepherd. *How 'Marine Snow' Keeps Ocean Life Healthy*. 2016. URL: https://www.forbes.com/sites/marshallshepherd/2016/11/17/what-is-marine-snow/?sh=490fd25d7fd2 (visited on 06/06/2021).

[68] Enrico Simetti et al. 'Autonomous Underwater Intervention: Experimental Results of the MARIS Project'. In: *IEEE Journal of Oceanic Engineering* 43.3 (2018), pp. 620–639. DOI: 10.1109/JOE.2017.2733878.

[69] Enrico Simetti et al. 'Sea Mining Exploration with an UVMS: Experimental Validation of the Control and Perception Framework'. In: *IEEE/ASME Transactions on Mechatronics* PP (Sept. 2020), pp. 1–1. DOI: 10.1109/TMECH.2020.3025973.

[70] SINTEF. *SEAVENTION - Autonomous underwater intervention*. 2018. URL: https://www.sintef.no/publikasjoner/publikasjon/1635277/ (visited on 07/06/2021).

[71] Chen Song, Jiaru Song and Qixing Huang. 'HybridPose: 6D Object Pose Estimation under Hybrid Representations'. In: *CoRR* abs/2001.01869 (2020). arXiv: 2001.01869. URL: http://arxiv.org/abs/2001.01869.

[72] Richard Szeliski. *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag, 2010. ISBN: 1848829345.

[73] Alykhan Tejani et al. 'Latent-class hough forests for 3D object detection and pose estimation'. In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 462–477.

[74] Henning Tjaden. 'Robust Monocular Pose Estimation of Rigid 3D Objects in Real-Time'. eng. PhD thesis. Mainz, 2019. DOI: http://doi.org/10.25358/openscience-2815.

[75] Henning Tjaden et al. 'A Region-Based Gauss-Newton Approach to Real-Time Monocular Multiple Object Tracking'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.8 (2019), pp. 1797–1812. DOI: 10.1109/TPAMI.2018.2884990.

[76] Jonathan Tremblay et al. *Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects*. 2018. arXiv: 1809.10790 [cs.RO].

[77] C. Waldmann et al. 'Search for life in ice-covered oceans and lakes beyond Earth'. In: *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*. 2018, pp. 1–7. DOI: 10.1109/AUV.2018.8729761.

[78] John Wang and Edwin Olson. 'AprilTag 2: Efficient and robust fiducial detection'. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2016.

[79] Wikipedia. *Creating or avoiding photographic orbs*. 2010. URL: https://en.wikipedia.org/wiki/Backscatter_(photography)#/media/File:Orb_photographic.jpg (visited on 06/06/2021).

[80] Yu Xiang et al. 'PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes'. In: 2018.

[81] Enrica Zereik et al. 'Challenges and future trends in marine robotics'. In: *Annual Reviews in Control* 46 (Oct. 2018). DOI: 10.1016/j.arcontrol.2018.10.002.

Appendix A

Miscellaneous figures

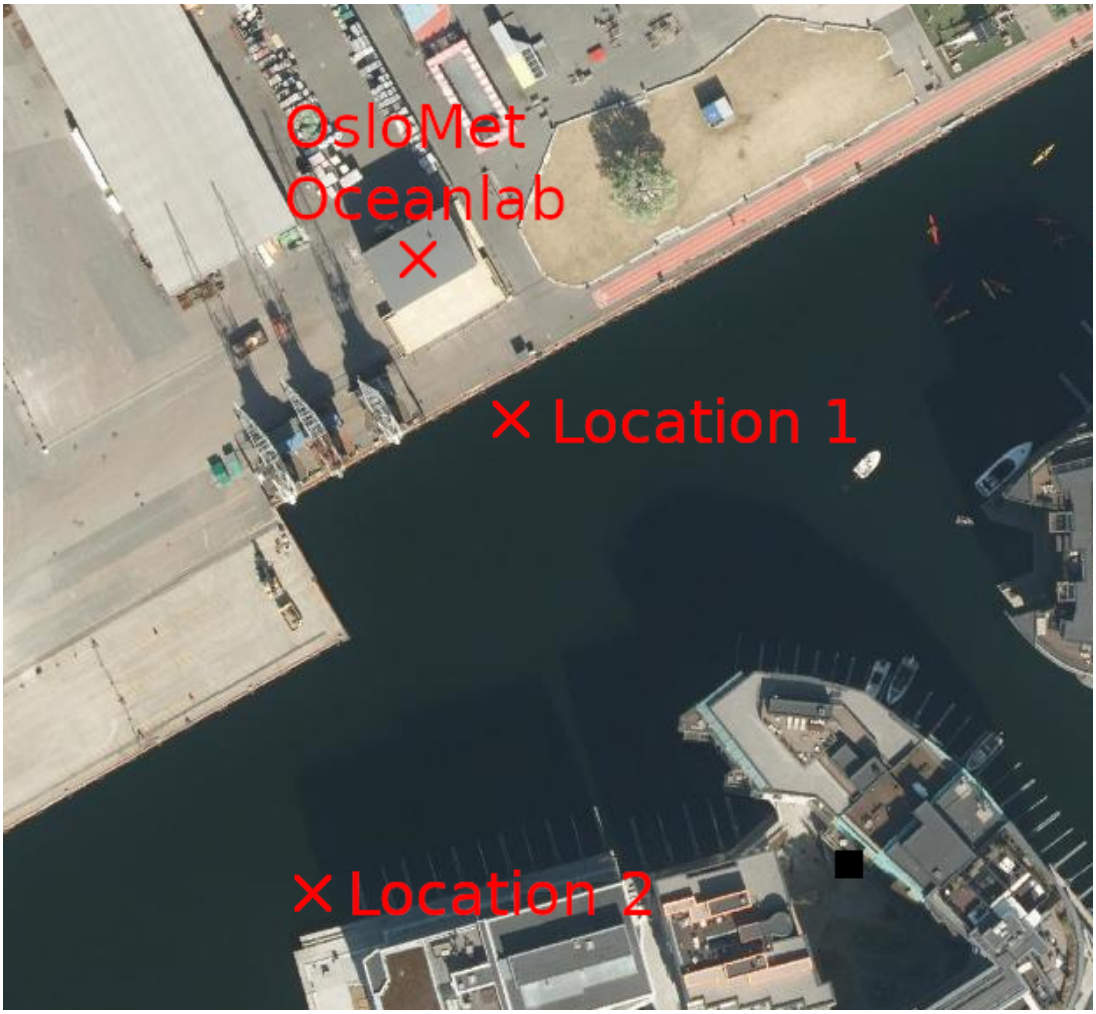


Figure A.1: Locations for underwater filming.