Automated Seafloor Object Classification and Seagrass Coverage Assessment Based on AUV Simulation

Olaf M. Winkler (B.Sc.)

March 15, 2017

Master's Thesis

University of Bremen

Fachbereich 5 - Geowissenschaften

MARUM

Institute for Artificial Intelligence

Supervisors

Prof. Dr. Achim Kopf (MARUM)

Prof. Michael Beetz Ph.D. (Institute for Artificial Intelligence)

Mentor: Dipl.-Inf. Fereshta Yazdani

to my loving wife Anna

Abstract

Oceans are still largely inaccessible to humans and man-made devices, however, there is an increasing need for sustained, persisted and affordable presence that helps to understand and monitor amounts 96% of the Earth's living space. Marine robotics as an innovative approach for exploration the Oceans and ocean floor, but also as a link to existing systems (floats, drifters, cabled networks), represents an emerging field MARUM and the University of Bremen have been covering over the past decades, and is planning to do so even more rigorously in the future. Since very much knowledge and huge amounts of data are gained in marine sciences, it is important to connect robotic systems to that advancement in knowledge. In order to exploit these technologically high developed systems, e.g. for interactive science, video streaming, monitoring and long-term observations, it is mandatory that they are taught how to interact. This thesis is concerned with the interaction between an AUV (autonomous underwater vehicle) and the marine environment. The objective was to simulate a seagoing mission where the system needs to have knowledge to point and identify objects and annotate them by intertwining the disciplines of geosciences and artificial intelligence. By using the cloud-based knowledge service called Open-EASE it's possible to process and reason about activity as well as understand robots behavior. it is possible for such a system to identify objects of various characteristics in real time. In this work I limited the objects to rocks, animals and seagrass and kept the focus on their detection. One of the main software frameworks for robot software development is the Robot Operating System (ROS) that is hosting the underwater simulator (UWSim). It includes ROS messages to process the robots sensor signals and stores the found values in a file beside the geo location. During mission execution all robot sensors and activity data are recorded as an episodic memory and can be queried in Open-EASE afterwards. By using a modeling software, the scenery is produced as well as high quality render images of the scene to clarify subjects within this thesis. The simulated area is a geo location near the eastern coast of Australia and the robot is supposed to identify objects and recognize borders of seagrass meadows. Since the growth of the meadows represent a result of influences, they are used to indicate changes in the environment with the robot as an automated scientist who is looking for the cause. The work is held most generic and the key results attest that the same approach could be used in areas of seepage and methane hydrates, sediment coverage versus precious poly metallic deposits, or artifacts (pipelines, other construction) versus natural underground.

Contents

1 Introduction

"*No matter how far from the shore that you live, oceans still affect your life and the lives of your families and friends, classmates and colleagues. The air that you breathe, the water you drink, the food you eat, the products that keep you warm, safe, informed, and entertained — all can come from or be transported by the ocean.*" - *Protect Planet Ocean (2016)*

By keeping this statement above in mind, it is necessary to take a look at how to protect one of the most valuable environments of planet earth. One option is to use modern technology like robots to collect data in environment monitoring missions to determine possible hazards. Robots of this kind already exist and many marine disciplines are using AUVs but for other reasons than monitoring. Robots like AUVs are commonly used in geoscientific missions mostly for profiling or filming the seafloor [13] to take a video stream for the exploration of resources like poly metallic nodules and seafloor massive sulfides [15]. Many institutes like MARUM, DFKI, Jacobs University and some private institutes are working on automated systems for these underwater tasks and their simulation. This thesis is working on an own way of interaction between an AUV and the environment.

The investigation of the seafloor and its habitats is mainly done by expensive explorations and will mostly be executed via ships. This means a lot of costs for universities and labor for the ships crew. The most expensive explorations take place in areas that can be hazardous for human beings and need a special trained crew. For keeping the risk at a low level, the dangerous areas are observed by Remotely Operated Vehicles (ROV) or Autonomous Underwater Vehicles (AUV). As useful as these systems are, they rarely can be found in less dangerous areas like shallow marine environments due to high costs but some projects tried to take first steps into that field because of their economic value. The shallow environments need to be observed because of different ecological reasons. The "Project Seagrass" provides first information about occurrence of seagrass in a scientific meaning which occurs in the area of green island near the Australian coast as well as near the coasts of Great Britain but in this work the area of Australia is of interest. Habitat monitoring is a field of growing interest but also stress analysis becomes more and more important also to the industries. This work will focus on combining the disciplines of marine geoscientific technologies of AUVs with the methods of artificial intelligence. The main point is to create an approach to improve the existing geoscientific methods of habitat monitoring by simulating an AUV that is able to scan an environment with optical sensors and to record the gained data in geoscientific manner as well as in terms of artificial intelligence. While performing the mission, the entities given

in the environment are detected and stored in a hierarchically structure of semantic meaning. This enables to process the knowledge on the episodic memory (remembrance of a specific moment) of the robot to reason and understand its behavior. This makes it possible to reason for the robots findings and comprehend the robots behavior by extracting knowledge for humans.

1.1 AUVs in Geosciences

Manufacturer	AUV Name	Max. Depth
WHOI	SENTRY	6000 m
WHOI	Autonomous Benthic Explorer (ABE)	4500 m
WHOI	SeaBED	2000 m
NERC	Autosub6000	6000 m
IFM-GEOMAR	ABYSS	6000 m
MBARI	D. Allan B.	6000 m
MARUM	Seal 5000	5000 m
IFREMER	L'Epaulard	6000 m
IFREMER	AsterX	3000 m
JAMSTEC	Urashima	3500 m
ACFR	Sirius	700 m
Kongsberg *(commercial provider)*	REMUS *(various versions)*	100/600/6000 m
Kongsberg *(commercial provider)*	Hugin *(various operating depths)*	6000 m

Table 1: *collection of different AUVs and their manufacturers. Most of them are from the scientific field but AUVs are also build for commercial use (various sources)*

AUVs are supporting a wide range of geoscientific fields. It was possible to improve resolution of spatial data a lot since the first generation of geoscientific AUV missions from the early 1980's in which the L'Epauland AUV from the IFREMER was used to map deep-sea manganese nodule fields [12]. The first missions were about mapping and provided mostly video streams that were put into image mosaics to have a detailed map of the seafloor. The development has put a lot of effort not only to let the vehicle drive its path automatically but also to collect more data than just images. The seafloor-imaging tools improved, geo chemical, geophysical and oceanographic instruments were added to the payload of the vehicle [41]. Still some systems face problems of navigation drift influenced by tidal shifts, even at a velocity speed of 1.5 - $2.0^{m}/_{s}$ [40], which is a certain factor to path planning. The technical background improved a lot since the 80's and it gained focus by the early 90's. The AUV related publications raised ever since and also their field of use. From image mosaics in the 80's, over habitat monitoring in 2007 (done in [43]) to investigations of a young lava flow (done in [5]) and resource exploration in present times [15]. Even hydro thermal vents were investigated [18].

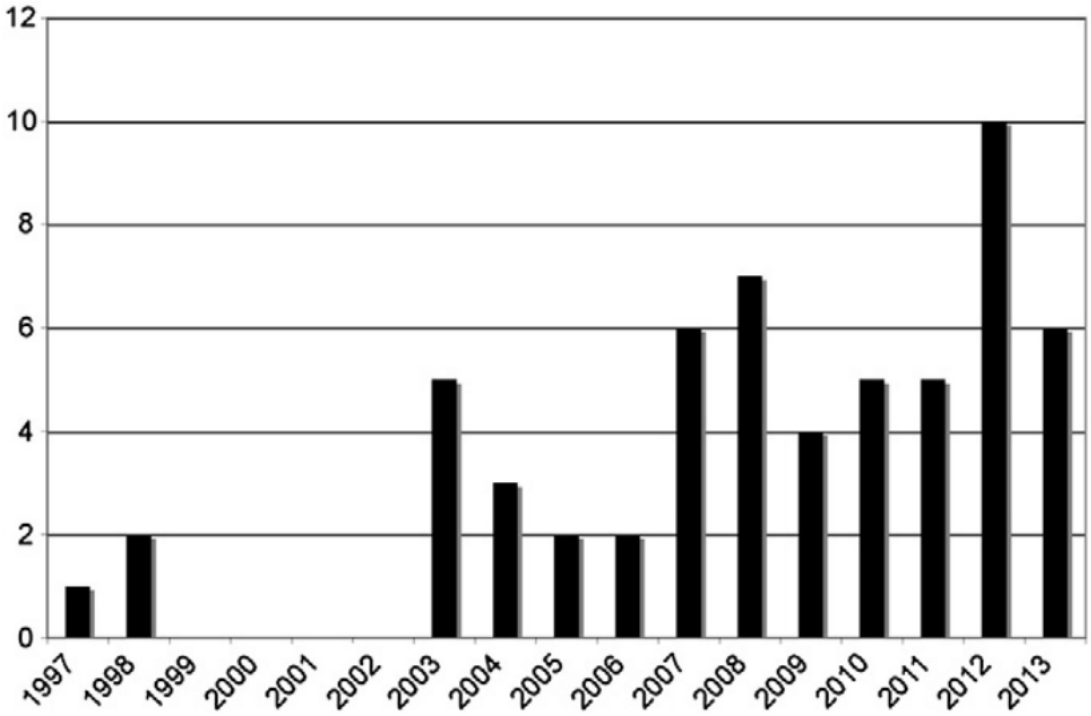

Figure 1: *Annual totals of reviewed papers that contain data, taken during AUV missions. The last decade shows an significan growth of AUV use (source: [41])*

In 2008, Newman et al. investigated giant km-scale pockmarks along a shelf edge by using the WHOI's SeaBED AUV at the east coast off the US. They discovered dissolved methane outflow along the pockmarks, carrying a chemical sensor. They covered many needs by carrying also other sensors to measure salinity as well as a camera for color photography during the mission. In the same year, Dupré et al. used the IFREMERs AUV AsterX, to map two mud volcanoes and resulted in a very high detailed map of them which could not be produced by a system from a surface vessel. The collected data provided an insight to the volcanoes formation and evolution [9]. Based on this data set, Foucher et al., found fluid escapes with a focus on the volcanoes center which were leading to a high density of seeps and to explosion of mud breccia blocks [10]. In 2012, Römer et al., used the MARUM Seal 5000 AUV to map Kerch seep communities (described in [41]) and associated it with the Blake Ridge Diapir off South Carolina [35].

1.2 Benthic Habitat Mapping

Benthic Habitat mapping started around 2007 with the REMUS AUVs mission to map eelgrass habitats in the Strait of Juan De Fuca in shallow waters of about 1 - 2 meters depth (mentioned in [41]). They used a Multi-Spectral radiometer to distinguish the eelgrass from other habitat members. A few years earlier, in 2004, Kennish equipped the REMUS AUV with a high frequency Side Scan Sonar to map the estuarine environment of Great Bay, New Jersy, USA [17], which wasn't a habitat mapping but led to technical implementation to do so. They were facing water depths of 50cm to 10.5 meters and a tidal current of up to $2^{m}/_{s}$ and were able to provide information about sediment characteristics like ripples, dunes and sand waves [17, 41]. Ruhl proved in 2013 [29], that it is possible to map and monitor open shelf environments by using the Autosub6000. The mission contained the data production of a seabed environment, compare results with those, captured by a surface vessel and to produce a data set that serves as a baseline for future monitoring missions. As every mission, also this one faces problems of exact localization, the AUV had to surface between each of the three dives, to refresh GPS position. The Autosub6000 mission served as a prove of concept that habitat mapping is possible even with near-bottom currents and turbidity to classify the seabed. Wynn et al. mentioned in 2014 that Wheeler [37] revealed habitat damages of mounds due to bottom trawling. This shows a need of an autonomous monitoring system, not only for this site but also for others.

The first mission of insights to a habitat performed with a color camera took place in 2012 to take a closer look at spatial distribution of deep-sea fauna at Porcupine Abyssal Plain in the north Atlantic [41]. The interests of such a mission is to map and measure individual organisms, geological features and to estimate organic carbon coverage. For this purpose, the percentage cover was analyzed automatically.

This is the connecting point to this approach as I am also working in shallow waters of ~3m depth with similar features as the Autosub6000 system and the Automated Benthic Explorer. As the Autosub6000 proved the concept to complete a survey and the REMUS completed an eelgrass mission, this work is going one step further by adding the artificial intelligence to the mapping purpose. I am also using a camera with optical features to scan the area, like the 2007's mission in the Strait of Juan De Fuca. By using an algorithm to filter every occurring color, the individuals are identified. This is done by direct threshold. Most coverage in the literature about environments are given as a value of percentage and for this purpose, I developed a projection to count the seagrass coverage of a certain area by its occurrence beside the direct geo location which makes it possible to determine C_{org} coverage.

For movement the typical AUV path planning is used to turn the vehicle by reaching a specific GPS position. For that purpose I used the automated robot control, developed by Srinivasan in 2016 [31]. I adapt it to fit the needs to the simulation borders so the robot acts completely autonomous without direct human interaction. This thesis's approach extends the existing approaches of mapping and produces data based on robot knowledge so the discipline of artificial intelligence is also able to work with it. This makes it possible for both fields to interact with each other.

1.3 Other Fields for AUVs

Geo hazards assessment in terms of oil and gas become more and more important for the industry. The focus on submarine maintenance and supervision gained importance since the Deep Water Horizon catastrophe in 2010. Autonomous systems also become an attractive addition to traditional vessels, since the fuel costs are increasing and the data collection needs to become more cost-effective. With an AUV in shallow waters it is possible to observe ecosystems like seagrass meadows in terms of growing rates, health and damage as well as observing objects on the visible seafloor. An approach for pipeline (Object) following already exists in simulators like UWSim.

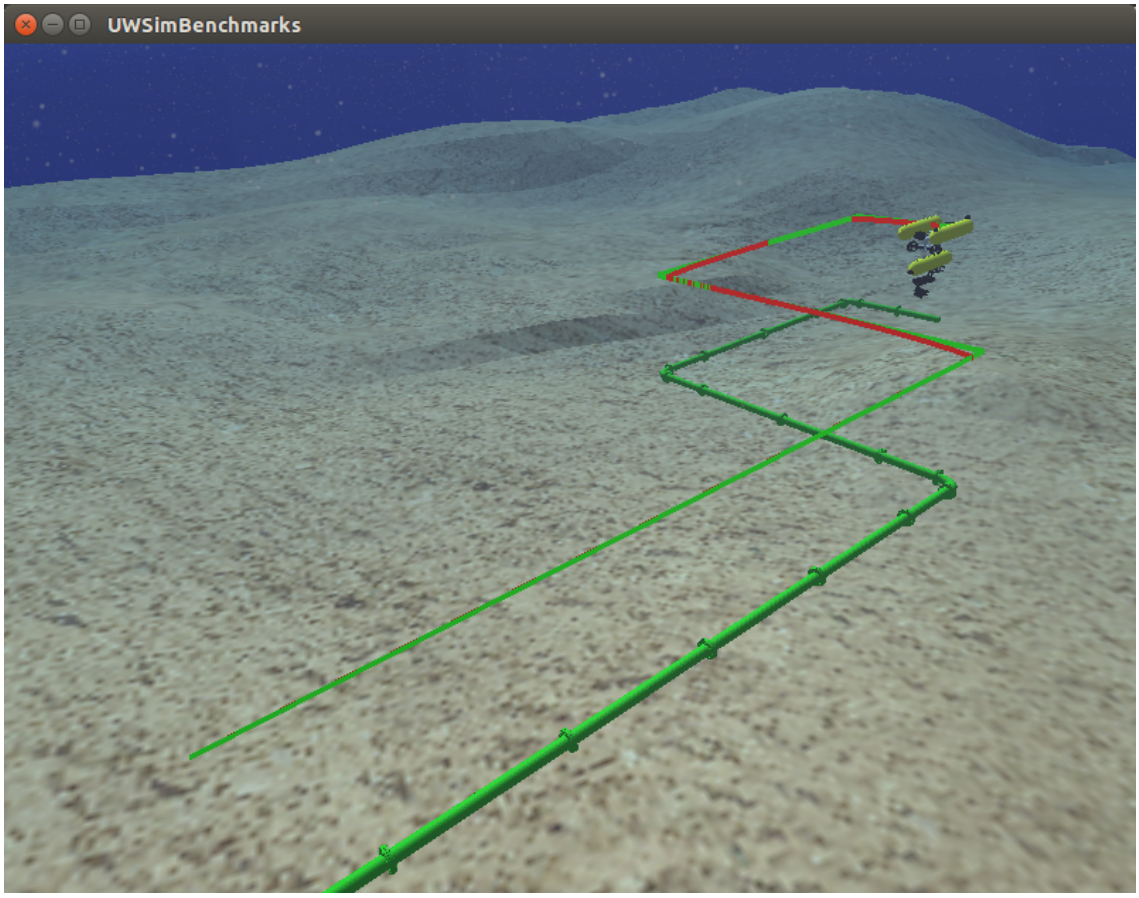

Figure 2: *Simulation to follow a pipeline automatically with an AUV, developed by the University of Girona (source: [25])*

For the resources industry it would give an opportunity to monitor a renaturation of a certain pipeline area or exploring the seafloor for new sources of material. Since the resource industry is focusing on underwater mining for like manganese nodes or methane hydrates, the field of seafloor harvesting is growing and the plan is to collect the nodes directly from the seafloor with humongous machines. This scenario truly is

in deep marine environments but for testing a robot to observe damages, it would be recommendable to run the machine in shallow environments. Also many installations for seafloor observations are fixed on one place (ESONET/EMSO) which represents "only" a static data source over a certain period of time and these systems also are limited to observe only a small area because of missing flexibility due to their lack of movement. An AUV with a basestation could expand the area a lot and could send data via the basestation on a regular basis so a larger area can be observed. This can also be broaden by not only using one robot but multiple. The usage of a robot is much cheaper than the planning and construction of a fixed station since many disciplines could share the costs and benefits. Due to multidisciplinary benefits and its potential, the systems need to be tested and simulated. A simulator also provides an insight to the possibilities for each partition.

It is surely effective to use an echo sounder in the deep [38] but in shallow purposes its cheaper to use optics like stereo cameras and laser scanners or to combine these sensors. Many sensors that can be found in modern AI robots are also available on underwater robots. It is likely to use similar software to get information off the interfaces that both fields of science are using, so both can work on the same framework, such as Robot Operating System (ROS). Sensors like these are available in most simulators. With the usage of ROS, Open-Ease and a geoscientific concern, this work will define a connection to use methods of habitat monitoring for the artificial intelligence (AI) to answer geoscientific questions and to get a cost effective connection of both disciplines. This will also broaden the possibilities of AUVs as the AI gets into a field of application. For that purpose an environment with geoscientific elements is loaded into the simulator to observe it with the methods from the artificial intelligence in the meaning of benthic habitat monitoring.

1.4 Robots Cognition

In the robotics, the missions/tasks environments are reproduced in Open-EASE. It is a framework to visualize semantic object maps, which are a subcategory of maps that store information about task-relevant objects of the environment [24]. The Open-EASE interface provides a tell-ask interaction of the robot, the map and the human. Tenorth and Beetz [33] have provided KnowRob as a concept of virtual knowledge base a technique to create a symbolic layer on top of the robots internal data structure [33]. This allows to keep original data structures in the background and to provide abstract concepts in low-level data. KnowRob comes with an internal design consideration that is following different rules:

- provide tell-ask service
- operate as part of the robots control system
- provide encyclopedic base to define and specify information for autonomous robot control
- etc ...

and other robot specific rules. As KnowRob developed separately to FieldLog (ontology based mapping tool), which comes with similar rules, it carves out that such a system can be self demounting if it is not flexible enough to operate with different data structures, so it can be adept to many systems. The demand of knowledge production, storage and representation is clearly given by the fact that similar systems develop in different disciplines.
KnowRob is designed in daily tasks and environments, such as kitchens with cooking devices within. The knowledge base is widely specified on objects like cupboards, spatulas, cutlery and food related devices in general. The state of the art to daily task robotics is very high developed as frameworks like RoboSherlock provide a possibility for mobile robots to operate in a human environment and facing the challenge of recognizing objects with different visual characteristics [2]. The artificial intelligence with robotics in daily tasks and environments is so far developed that robots are already able to learn about their surroundings and record their findings, movements and decisions semantically so the user is able to understand what the robot saw, did and why he did it. In this work, the robot is not learning or has to make complex decisions but by developing first step systems such as object recognition, it is likely to implement systems like RoboSherlock to it to provide a much higher chance to let a robot learn about the environmental systems.

2 Motivation

The motivation to this work comes from the claim to extend the methods of geoscientific monitoring and observation of environments with the methods of the artificial intelligence. Robotic systems are already used to a certain extend in geoscientific fields like the offshore industries for example. This industry uses it to do tasks like maintenance, exploration and setup [1]. The industries attempt to be eco-friendly but they influence the environment on the seafloor which takes them into the responsibility to monitor the affected area. To ensure this task on a reasonable level of costs, the traditional way of mapping and monitoring needs to use an advanced approach which has the potential to observe an environment over a long period of time and to notice and mark significant changes.

New technology brings opportunities for every field and so it is a chance for geoscientists to not only use parts of the robotics but also of the artificial intelligence. The needed database for storing the knowledge already exists and is used by other disciplines frequently. Unfortunately these database systems are rarely used by geoscientists or even associated with AUV robotic systems. Existing AUV simulators offer options to generate data which makes it possible to develop approaches for given cases.

To access the knowledge about environment, tools are ready to use from the side of the artificial intelligence. One of the tools is Open-Ease, whose web client interface which access the database of robotic knowledge to present it to the user who can query the available data sets to understand robots behavior in the environment. Due to the fact that the robots knowledge is represented within a database, it is possible to widen it by additional knowledge from the geoscientific field. This enables admission not only for experts but also gives the opportunity to others to extract knowledge from an environment.

Many institutions are already working actively on topics concerning AUV controlling, exploration and monitoring. The DFKI (Deutsches Forschungszentrum für künstliche Intelligenz) is one of the leading German institutes that is working on projects like AVALON which is an autonomously driving AUV that is helping during maintenance tasks. Industrial applications for such a system for pipeline inspections is accomplished by Prof. David Lane from Heriot-Watt University in Edinburgh, UK. Others, like the interactive and robotics systems lab of the university of Jaume, Spain, has developed a simulator for path planning and sensor communication tests [25]. By cooperation with the university of Girona, the systems lab has developed *Underwater Simulator* (UWSim) which is directly integrated into the *Robot Operating System* (ROS) to communicate with their developed AUVs *Girona500* (G500) as well

as *Sparus II*. The simulator is highly configurable and also allows the user to even implement real world physics that also includes current simulation and wave activity on the surface. It comes with many sensors like laser scanner, optical cameras, a stereo camera, range sensor and many others that allow a detailed seafloor scanning and makes it the optimal simulator for this work.

This approach of work provides a possibility to use it as an autonomous scientist that might operate in long-term missions to monitor environments of high interest and scientific importance. The field of robotics is a very fast growing and very interesting field of work and as it is reaching more and more into daily life, its time to implement it even more into the field of nature science and find tasks that can be accomplished by autonomous systems. The main motivation of this research is to learn more about how robotic systems are able to use knowledge and transport it as artificial intelligence to the real world to interact with it. Also to evolve a simulated long-term mission with nature scientific approach to develop a connection between the disciplines of the artificial intelligence and geo science.

2.1 Goals

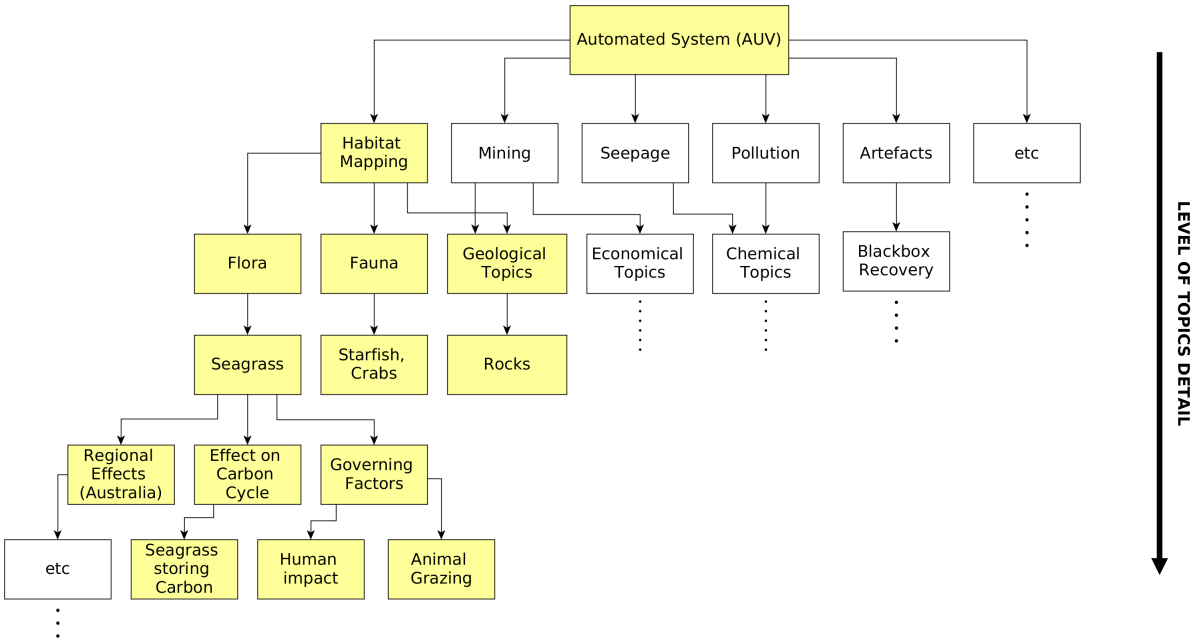

Figure 3: *Overview of this works objectives. The primary aim is to get an automated system for habitat mapping to identify flora and fauna as well as geological objects. This gives options for secondary aims of interpretative character like effects on the carbon cycle.*

The primary goals of this thesis are to develop an automated system that is able to collect data on its own and provide it as a service to state of the art methods of the artificial intelligence (AI) as well as geosciences. As most automated systems are developed to execute deep sea tasks, this approach is pointing more to shallower regions like typical onshore habitats. Like the seagrass Project, this work is interested in the health of the habitat but in terms of data mining instead of direct analysis of the individual plants. By using seagrass as a topic for the developed algorithm it shows how it handles the identification of objects. It is versatile enough to be used in other areas, like deep sea mining, detection of seepage, pollution or artifacts, depending on the need for the mission.

As the Habitat mapping normally contains the survey of the flora as well as the fauna, I decided to include starfish and crabs which are detected additionally to the seagrass. For reasons of geoscientific interest, I also added different kinds of rock that are identified as well. This gives the opportunity to step deeper into this topic later by refining the algorithm.

The detection of the seagrass leads not only to the overview of its spread but reveals also different insights to the habitats activity and influences affecting its growth. Simulated influences, like a human impact (anchoring and shadow casting due to

a walkway) or animal grazing are directly affecting the spread over the area and reducing the sizes of the meadows so the influence will be visible in the data files as well as in the resulting maps, generated after loading the discovery into a Geo Information System (GIS). As the data uptake is done in the way of traditional geoscientific methods (geo position + value), also AI methods will profit from this approach since it can be used to produce robot knowledge and log the mission is a hierarchically, semantic way. By logging the mission, the knowledge is observable in an external, cloud based web interface (Open-EASE).
On basis of processing the data in GIS, a secondary aim is to get information about the topic of seagrass spread due to its ability to fix carbon with up to $1.18 \frac{mg\,C}{g\,dry\,weight\,h^{-1}}$ [21] (depending on species and age of individual) or 83 to $138 \frac{g\,C}{m^2\,yr^{-1}}$ [47, 20](in general) within the sediment. This is an enormous number when it comes to big meadow coverage. By keeping in mind that the carbon is stored within the sediment in anoxic conditions, this makes a big difference to Rain forest (or typical terrestrial) carbon fixation. As the carbon is set free due to decomposition of the organic material or fire, the seagrass' carbon is stored in the sediment for millennia time scales, even if the plant becomes decomposed.
This work will compare the coverage of the seagrass meadows in an affected and unaffected situation and calculate the loss of carbon storage from the influencing object by simulating them, based on real world data, and the plants growth simulation. The data acquisition is done in an AUV simulator (UWSim) to run the algorithm on a real robotic framework and to be able to subsequently evaluate the mission. By collecting data, the robots knowledge and experiences can be accessed in different ways such as:

- CSV File for GIS to provide data for traditional map material and polygon meshes from the plants spread as well as object occurrence
- Photographs of detected objects
- Semantic knowledge and episodic memory from the mission
- Semantic maps that can be queried from Open-EASE web interface

For further purposes, objects of the surrounding will be identified and can be asked for in a web-based knowledge representation. An unexplored area with the AUV observation becomes a knowledge representation by viewing it in the web interface. By comparing episodic memory, the mission becomes a monitoring task.
The object recognition is done by color detection and placement on a map in GIS which is representing the position of the semantic object. An additional goal of this

thesis is to make the data that is stored available in Open-EASE to upport humans to understand the robots activity data as well as the resulting environment experiences. The result is a complete reconstruction of the scanned area as a traditional as well as a semantic map with knowledge representation and a comparison of different stages of the areas development.

2.2 Possible Use Cases

There are many opportunities to use this approach. One obvious case is to create and publish environment maps or data and if necessary to determine protected areas in which fishery is prohibited or anchorage might be forbidden. Furthermore, this approach is a chance for every land registry office to widen and expand their knowledge to the marine environment.

Because plants are setting their roots into the sediment, they are solidifying the underground. This is an indirect act of coastal protection which can be important to highly populated coastal regions. The high environmental stress that is pushing onto the coastline is mainly associated with anthropogenic influence. Increasing population, rising temperatures and acidification of the oceans already influencing the coastal ecosystems. Extreme consequences would cause kelp forests to disappear in the south and marl environments to vanish in the north due to warming and acidification. The influx of nutrients into coastal areas leads to more productive biomass and a rise of hypo toxic events especially for estuary systems [11]. To prevent or predict such events, a monitoring system is postulate.

Seagrass meadows occur on almost every shallow coastal line in the world [36] and play a valuable role in these ecosystems [7]. They are recycling nutrients, stabilize the sediment, provide habitat for survival and reproduction of vertebrate and invertebrate taxa and show a "nursery" function for some species [14]. As entire estuaries are identified as nurseries to protect individuals from predators and to provide food, it deserves to be protected. Because these environments are almost omnipresent in coastal areas, the treatment of them is also interesting for decision makers of projects. By monitoring, these decision makers have another component to look at when it comes to calculation of long-term cost estimations.

Not only seagrass is an important component to look at but the approach has its variability. It is important to change topics easily due to different interests components that are identified in this project, it is relatively easy to add or change the topics. This gives the opportunity to also use it in terms of exploration to identify resign oil, methane or to search for fields of gas hydrates. Beside the geoscientists it is also a chance for biologists to observe environments and engineers to test their algorithms.

2.3 Geoscientific Impact

With the given reasons, this is a chance for the geoscience to add another method to tasks of exploration and observation. By using robot knowledge, it is possible to generate a live detailed data set which can be made even more precise by a scientist who is evaluating this data by looking at pictures that are taken from the system when it comes to object detection. As AUVs are already in use in geoscientific missions, this approach widens the abilities to such systems. It enables primal autonomous data evaluation of in the area of interest. The gained data from this first set shows possibilities that the actual algorithm, that is present right now, can improve the methods of geo technologies. Robots already drive autonomously but the detection approach can be used in many fields to help scientists to calculate environment conditions.

Figure 4: *As the robot finnished scanning the environment, the recorded data is stored in different files and can be used to produce paper maps as well as resulting statistics. A scientist can also access the herarchically recorded data in a web interface, providing the background data base with knowledge and episodic memory*

Additionally to the traditional methods, comes the fact that this work provides a way to integrate methods from the artificial intelligence (AI) to access knowledge about the environment, the mission takes place in.

Because not every discipline is interest in a set of seagrass detection or topics of this kind, the approach is is highly variable. Limits that are set for the image threshold are providing levels in which objects are detected or ignored. This creates another possibility to change the approach to ones needs, as different systems may come with hardware of different optical noise behavior.
This work is focusing on the AUVs optical sensor and recognizes different colors. By taking a chemical sensor, this approach can be used to recognize chemical changes in the water column to record the occurrence of specific chemical compounds. Both sensors combined can provide an even more detailed picture to record occurring bacteria mats for example. The autonomous control of the robot is triggered by the systems positioning sensor so the change in direction is triggered via GPS. Other projects i.e. for underwater ice surface investigation may use a timed change in directions since the GPS signal becomes weaker in depth or under thick layers of ice. On the other hand it may be useful to surface the vehicle from time to time to refresh the geo position.
Due to a predefined path, it would be another chance to take the optical part to scan underwater ice shields for anomalies. The only limit is the mentioned missing GPS signal that could be defined by a signal that may come from another source. Since the incoming signal just has to be set to the GPS topic of the robot, it may be another future work to simulate such conditions with GPS stations or fixed orientations points.

3 Methods

The methods in this work are reaching from the usually used geoscientific tools of GIS over Blender to the more robotic tools of a Robot simulator, hosted by ROS to the mainly used robotic web interface of Open-EASE, which is based on KnowRob, a knowledge base of robots.
Geo Information Systems are widely used in geoscience to show positions of working areas or to indicate a bigger picture of i.e. an event influencing its surrounding environment. Very good examples are geological maps or urban planning, based on GIS data (Google Maps, OpenStreetMap).
Blender is well known in the community of photorealistic 3D artists, who are rendering stills or animations. As it is a free, open source software, with a humongous community and many tutorials on the Internet, it is a very advantageous way to produce data for the robotic field. Many fields, especially outdoor environments, that need to be explored by robots, depend on predefined simulation tests to lower costs and minimize the risk of a mission fail. To provide a connection between the robotic tools and the geoscientific tools, Blender is used to interact with both sides, as it is able to read exported data from geo oriented fields and provide it as data that the robotic field is able to handle.
The artificial intelligence in turn is working on a framework, that provide a wide range of possibilities when it comes to working with this gained data. The robotics side is providing a system to host the simulator while an intelligent recorder is producing knowledge that comes directly from the robots algorithms. This knowledge is accessable as epsodic memory within the cloud based web intercafe of Open-EASE, which rebuilds the environment as a semantic map whose information can be asked for in the interface.

3.1 Traditional and Modern Mapping Methods

Ontology based geological field mapping

The traditional way of field mapping contains a topographical paper map with surrounding key objects as reference points. While the mapping person identifies objects, the paper map is filled with colors or patterns that represent rock types and contains information about age, mineralogy, formation and other data of geoscientific interest. This technique of classical field mapping is still object of mapping classes in universities to train freshman to identify rocks by their appearance and mineral setting. The development of very accurate GPS modules in mobile devices and internet based services such as Google Maps or OpenStreetMap, it is possible to map digital and online at the same time, which provides a big benefit when it comes to collaborative mapping. This digital development lead to more improvement by building a database, holding the found information in matter of spatial appearance and geoscientific background information for individuals.

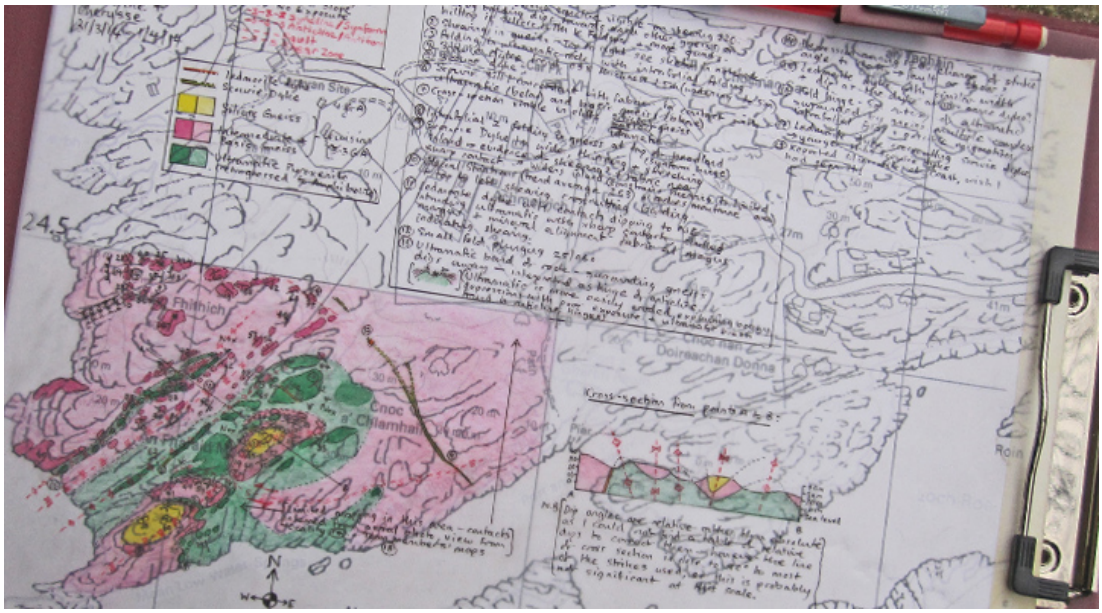

Figure 5: *Example of a partly colored geological map of a mapping field exercise*

The development of *FieldLOG*, a software with database structure, brings an early approach of using the collected data in an ontological way. The software is in development since the 90's in Canada and the developers have set some standards for a flexible database design for individual requirements and corporate mandates [Brodornic, 2004].

This explains that a flexibility is required due to varying identifications of different field scientists. Software and database flexibility with predefined schematics, defined in ontologies, bring a solution to such problems.
As *FieldLOG* provides an ontological basis, it is easy to connect others to it or to develop own ontologies to provide additional individuals and classes. As ontologies are a construct of philosophical disciplines and are a very old method to describe the surrounding, they are clearly a new field for computer science. Digital ontologies are developed since the 1990's and are written in an own "ontology web language" (owl) as standard. Many can be found over the internet and important ones are: SWEET ontology build by NASA as "Earth and Environmental Terminology" [27] or KnowRob as knowledge for robotic systems to achieve more flexible and general behavior and better performance [34] for such systems.
Ontologies provide a standardized as well as flexible way to contribute data to a field, that robots will become more and more important in the near future. This offers opportunities to interaction between human and robotic systems. Automated systems for information extraction already exist. They are able to extract the information to a small extend [16] such as rooftops of buildings, used to estimate the buildings area coverage. Information like these are valuable for projects like OpenStreetMap which represents a data source for outdoor robotic missions like in the SHERPA project which is used for human-robot interaction while performing a rescue mission in avalanche scenarios [42, 4].

Semantic mapping

Nugoeras-Iso [22] describes ontology approaches from the late 90's and shares Ideas of interoperabilities of systems like OBSERVER (1998) which provides an architecture for queries to global information systems. This system works with entities across ontologies and shares the approach as a new concept. Actually Berners-Lee [3] admits that "the semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation". Semantic mapping can be found in websites of the UK Office for Library and information Networking, which uses a standard of "crosswalk based semantic interoperability". This is an example of developing mappings among different standards and different domains that provides a wide flexibility. The Project MADAME (Methods for Access to Data and meta data in Europe) gives an example of geographic information meta data to this field. The Digital Geographic Information

Working Group offers crosswalks of ISO 19115 and CSDGM - a connection of both standards. Nogueras-Iso et al. provide an overview to the crosswalk construction of these standards as they follow certain rules:

1. Harmonization: Obtain a formal and harmonized specification of both standards
2. Semantic mapping: Mapping establishment between elements of the source standard and the target standard - deep knowledge of both standards is required
3. Additional rules for meta data conversion: Solving problems related to different levels of hierarchy, data conversions, etc. - eliminating domain and structural conflicts
4. Mapping implementations: Obtain a completely automated crosswalk by means of the application of some type of tools. Searches can be provided according to different families and meta data standards.

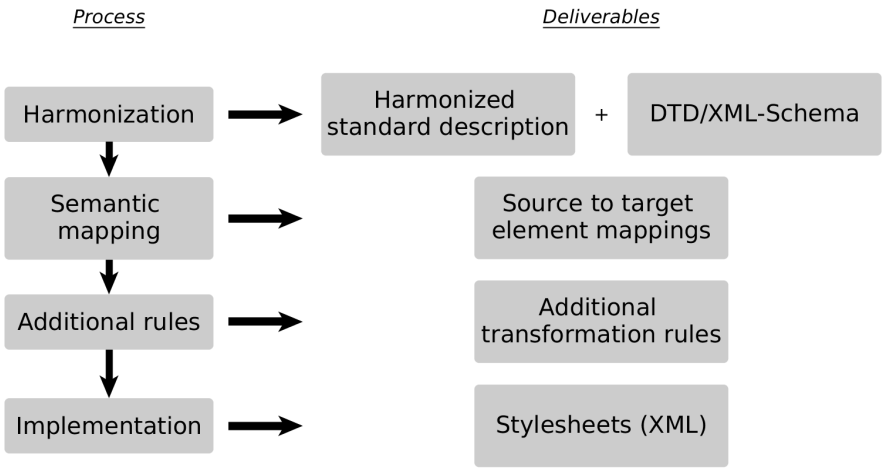

Figure 6: *Schematic representation of crosswalks process steps between meta data standards (source: [22])*

As semantics follow a Hierarchical and structural organization, also the semantics of this work are build up in such a way. But here, the elements are implemented into another ontology instead of finding a crosswalk of two existing ones. The semantics of the KnowRob ontology are extended by objects of interest.

A complete overview of how the crosswalk is build up can be read in “Geographic Information Metadata for Spatial Data Infrastructure” (s. [22]).

Seagrass mapping

While geological mapping is done by special extraction, traditional seagrass surveys are done by scientists on the computer with aerial images [28] or "in situ" by placing nets/squares over the seagrass, and taking photographs to estimate the density of species and maybe the algae coverage on the leafs. McKenzie [19] provides a guideline with standards that are useful for scientist in the field and give a possibility to identify species and provides an idea of the appearance of the seagrass itself. The mapping process can be an exhausting experience and needs several day sometimes.

Figure 7: *Seagrass meadows on an aerial image*

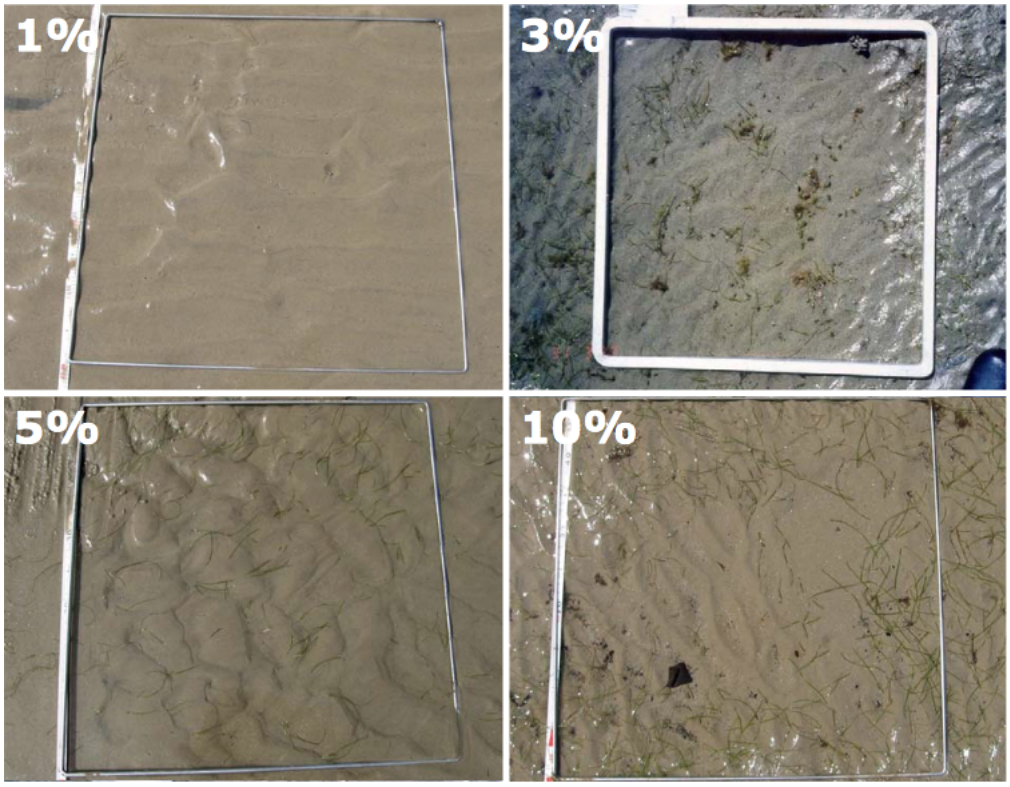

Figure 8: *A part of McKenzie's Guideline to seagrass coverage estimation (source: [19])*

The in situ type of mapping provides a very detailed picture of the flora and fauna but takes very long and is cost intensive since a group of scientists need to dive into the meadows itself to take a look at the plants and to shoot the pictures.

Figure 9: *Traditional in situ seagrass mapping by snorkeling*

The traditional mapping method faces similar problems to the mapping like the modern systems. The GPS module only works on the water surface and the mapping needs to be done as close to the GPS positioning device as possible to minimize the error which can be up to 3m depending on the depth in shallow waters [19]. The general field procedure is dependent on the tides of the area and is normally done by walking or diving around the perimeter and a grid pattern is placed where the seagrass occurs to take a georeferenced picture. The collected data will show gaps because the grid will be placed 100 to 1000 meters apart, depending on the size of the survey site [19]. Others see the benefits in coverage of large areas in short time ($200^{m^2}/_h$ depending on depth) and allows operation in areas where boating is limited [39]. These limits are also not present if an automated scientist like the AUV is taken for the task. Truly it can not be set up in an environment of quickly changing tides but other areas like the Baltic sea are a good point to start such a system. Comparing traditional methods to the automates scientist, the data from the robot does not have any big gaps because of continuous data allocation. Many regards have been made over the past decades to develop a monitoring and survey program to produce a cost-effective allocation of resources to conserve seagrass ecosystems [23]. There have been many attempts to reach that goal by examining areal images and satellite images over times of 10 to 20 years to get an idea of the trend the seagrass meadows are tending to.

In 1995 Kirkman gave a recommendation to such a task on a scale of 1:100.000 in areal Images to see changes in the meadows. The Program SeagrassNet from 2006 provides a protocol that can be used by international scientists to get and provide information about seagrass monitoring sites [23] and describes case studies of five locations across the American continent (two north temperate in the US and three tropical locations in Brazil). In two cases on the program, the loss of seagrass was caused by eutrophication and another two by climate change. The last one was influenced by interaction of the presence of a marine protected area. The sites were measured for four years, four times a year. The protocol provides measurements in changes of seagrass distribution, species composition and abundance. SeagrassNet is not used for mapping or for the characterization of region-wide trends but it can be used as part for other projects like this thesis to provide data for specific meadows that are in- or decreasing, that deserve to be surveyed. Another Story is that there are also observations of an increasing meadow spread over 20 years in Western Australia. Kendrick, Eckersley and Walker observed a growth of seagrass on the Success Bank, Australia with areal images from the 70's, 80's and 90's but came to the conclusion that studies still have to go on to understand the circumstances of the area. According to Orth [23], Tampa Bay (Northern US) and Hervey Bay (AUS) are also places where the Seagrass loss got reversed, but the number of decreasing seagrass coverage globally exceeds the reports of an increase. Therefore, the overall coverage of seagrass is decreasing over time and as one can read in almost every published data, the main cause of the decrease is of (direct or indirect) anthropogenic origin. Overall it is possible to get an idea of the current situation in a seagrass meadow but with an automated scientist it would be possible to get the information in a periodic matter which makes comparison and prediction possible in many cases. This work is taking optical occurrence but by using additional sensors, the mapping would provide additional information like chemical influences, salinity or currents. These information can also be visualized on a map. The interest of geoscientists is on the one hand to collect data about the environment and on the other hand to detect objects on the seafloor. By detection of objects and annotation of their characteristics, it is possible to let an automated scientist observe an area and provide data about them.

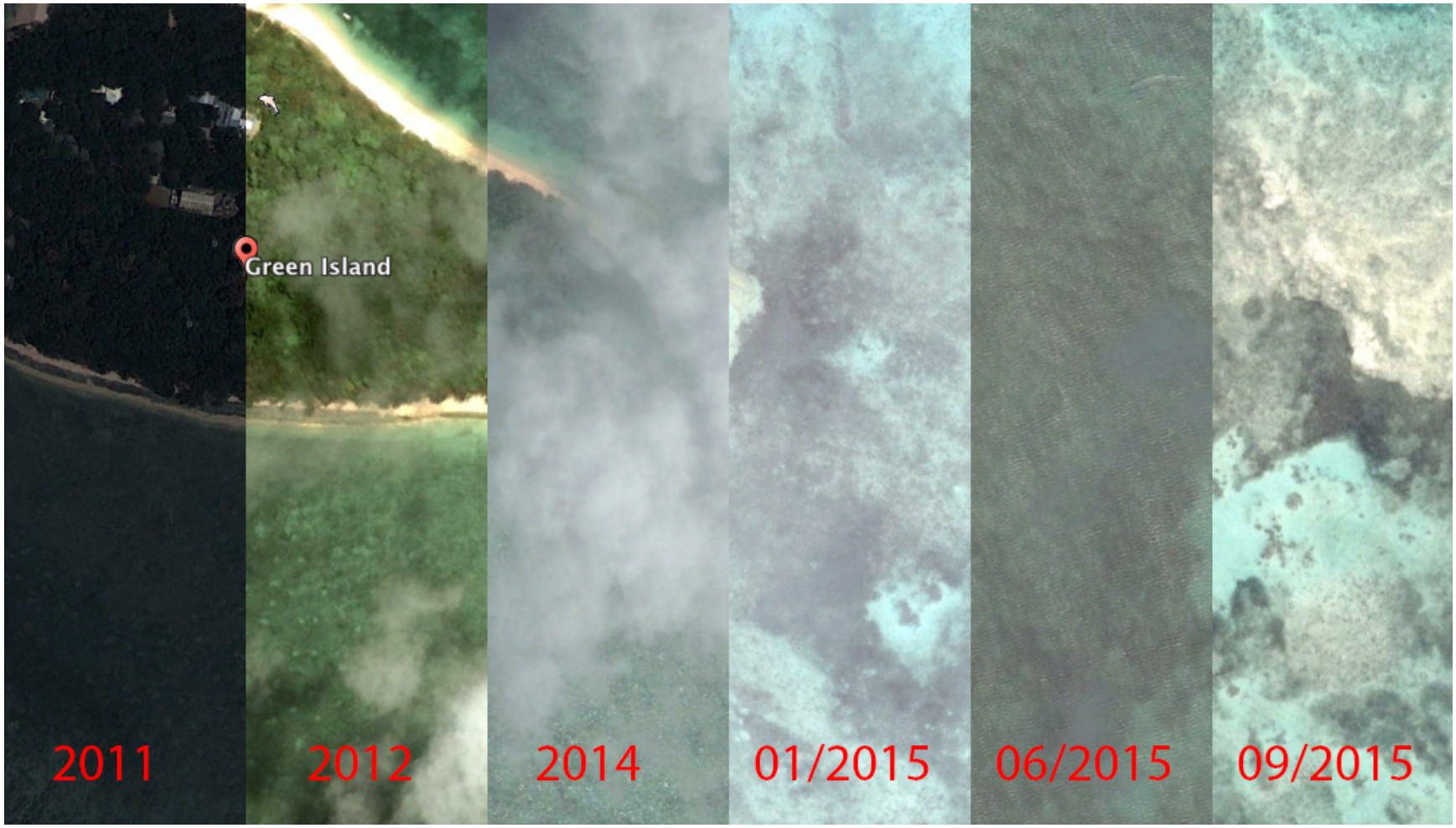

Figure 10: *Green island from 2011 to the present. The aerial images are taken from Google Earth. Only the images from 1/2015 and 09/2015 are reliable sources for traditional seagrass mapping. The others are useless due to cloud coverage and sun reflections on the water surface. An automated system, working only on these images, would have big problems using them to identify the seagrass. (images source: Google Earth (2017))*

3.2 Quantum GIS

Quantum GIS is an open source Geo Information System. It is mainly used for geo spatial analysis and is able to read and manipulate several data formats like ESRI Shape files, raw GPS data, XYZ files, ASCII formats, etc. The amount of formats can be widen by installing additional plugins provided by the open source community from a simple click in a menu within the software. The program is supported by many operating systems such as Linux, Mac OS, Microsoft Windows and partly by mobile operating systems. This makes it the top choice for low cost projects with the focus on mapping and geo spatial information representation. Like ESRI (the leading industrial GIS manufacturer), QGIS is using the .shp file format to save and manipulate data. It contains three different types of objects:

Points	represents an exact geoposition (XY(Z)) that can represent everything and contain any kind of digital information
Lines	representing at least two geoposition, connected to each other that can also contain any kind of information
Polygons	representing points that are connected via lines to each other and the last point will be connected with the first point of the order

Table 2: *Three main objects represent the data withing 2D or 3D space. A point can not only have a 2D (XY) position but also 3D (XY+Z) which makes it possible to represent every object in a 3D space*

Lines can also contain points of any kind of order but at least two. Like lines, polygons contain any order of points (at least three) but the first will always be connected to the last one. With these simple objects, it is possible to represent any kind of 2 dimensional information on a map.
The data can hold different kinds of information that are separated into Strings, Integer, Numbers (decimals) and datum. While strings, Integer and datum are variable in length, numbers can also have accuracy with a floating point.

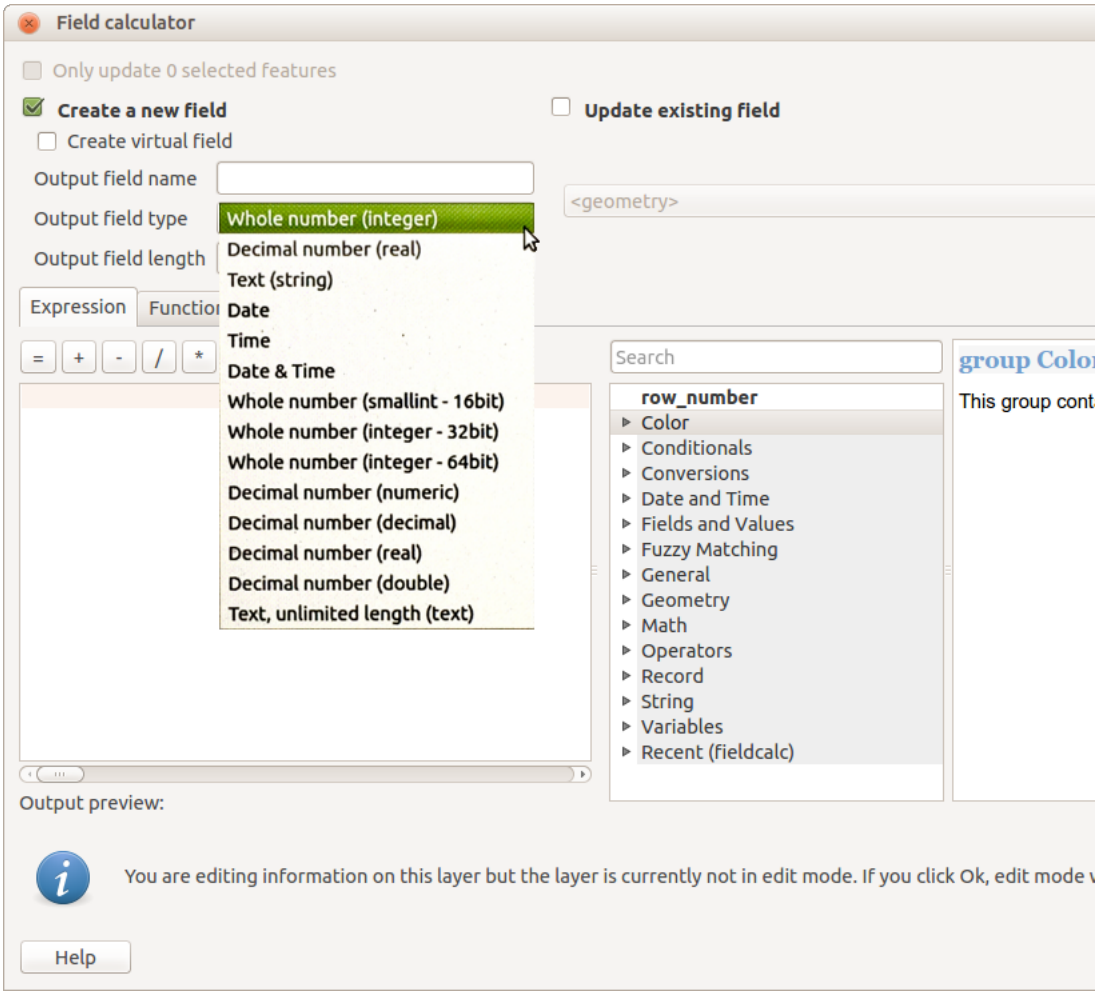

Figure 11: *The Field calculator shows different data types that can be used to implement data points (attributes) into a shape.*

Data storage

QGIS is saving the data of the polygons, Lines and points not only in one file but produces a stack of files that contain all needed information for different tasks as:

.dbf	attribute format: columnar attributes to the shape, stored in dBase IV format
.prj	projection format: coordinate system and projection
.qpj	same as .prj but with additional authority information
.shp	shape format: contains the 2D geometry features
.shx	shape index format: allows seeking in both directions quickly by indexing the shapes

Table 3: *The table shows the different data formats QGIS produces while saving the project information*

Beside of the standard traditional file formats, QGIS is able to read many different kinds of data holding files. Laser data, ASCII files and comma separated value (CSV) files are only a few to name. In this work, QGIS is used to read and work on CSV files, generated by the robots detection algorithm.

3.3 Data Extraction (Geoscientific)

As the traditional seagrass mapping is using aerial images, this work is using them to recreate the environment, since its impossible for me to perform a real world mission. As the chapter 3.1 is showing, not all satellite images are suitable for completing this task. Cloud coverage and the reflecting sun as well as low contrast are limits to identify the seagrass meadows. QGIS is used to create polygons that cover the seagrass and are exported to recreate the environment. This is done with the valuable plugin support of QGIS. One of the most useful plugins is "OpenLayers Plugin" which enables to load satellite images directly from a server into the mapping canvas and prepare data for the environment simulation. By downloading the satellite images and marking the seagrass borders, polygon shapes are created that represent the full spread of the plants. Another very useful Plugin is "Qgis2Threejs", which produces a polygon mesh file from the marked area that is used further more in Blender to replace the polygons with spread plant meshes (cf. Figure 15).

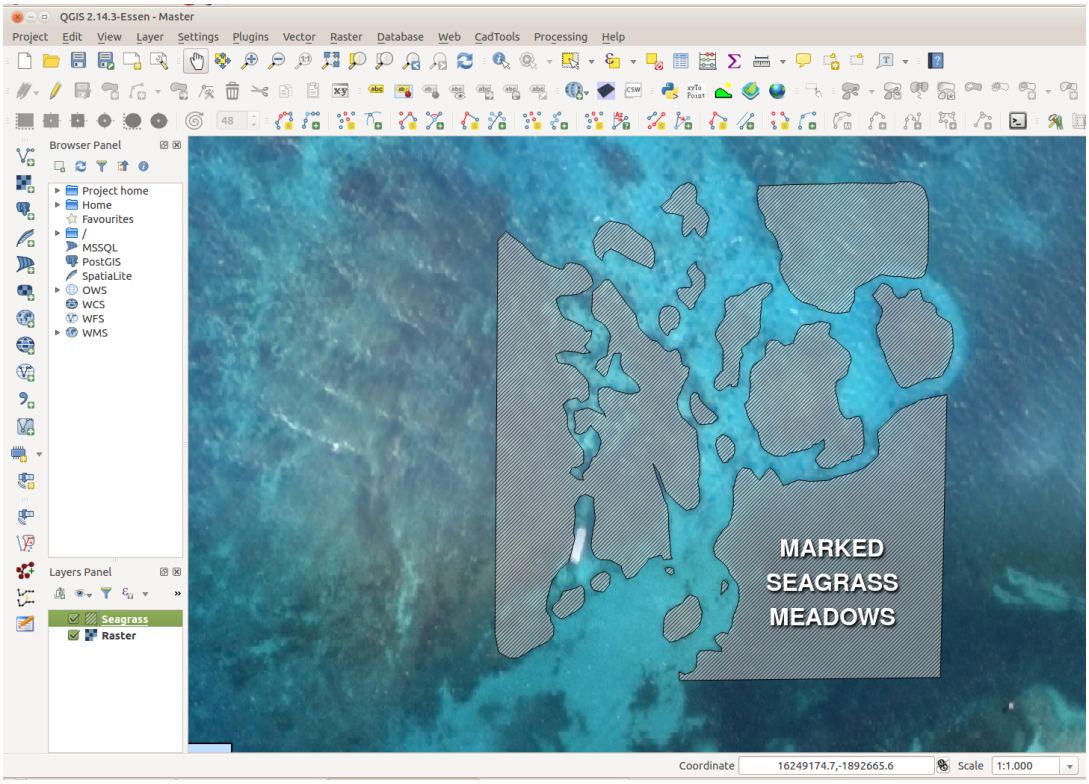

Figure 12: *Marked Seagrass meadows from the satellite image which is downloaded by the OpenLayers Plugin from the Google Maps Server. The polygon represents the border that will be extruded to a 3D mesh file.*

With the Qgis2Threejs plugin the polygon, that is covering the seagrass areas, is exported as a 3D object. Figure 12 represents a map canvas of the seagrass area near Green Island, Australia. Furthermore, the seagrass is marked with a line tool (red line in Figure 13) to indicate the simulator limits. The polygon is extruded to a given height as a 3D object and extracted in .OBJ or .DAE file format so it can be loaded into Blender. The 3D object is used to determine the spread of seagrass meshes with a "weight" tool (heavy weight for more spread) while recreating the environment.
For keeping the project as close to the methods of the robotics as possible, .OBJ format is chosen. The file is imported into Blender to to keep working on it for the simulation *(As the AI methods are capable of also reading .DAE files, the tool can also be used to directly provide data for Open-EASE)*. The output creates also a web browser based preview page so its possible to check the exported objects and their position on the ground plate (canvas).

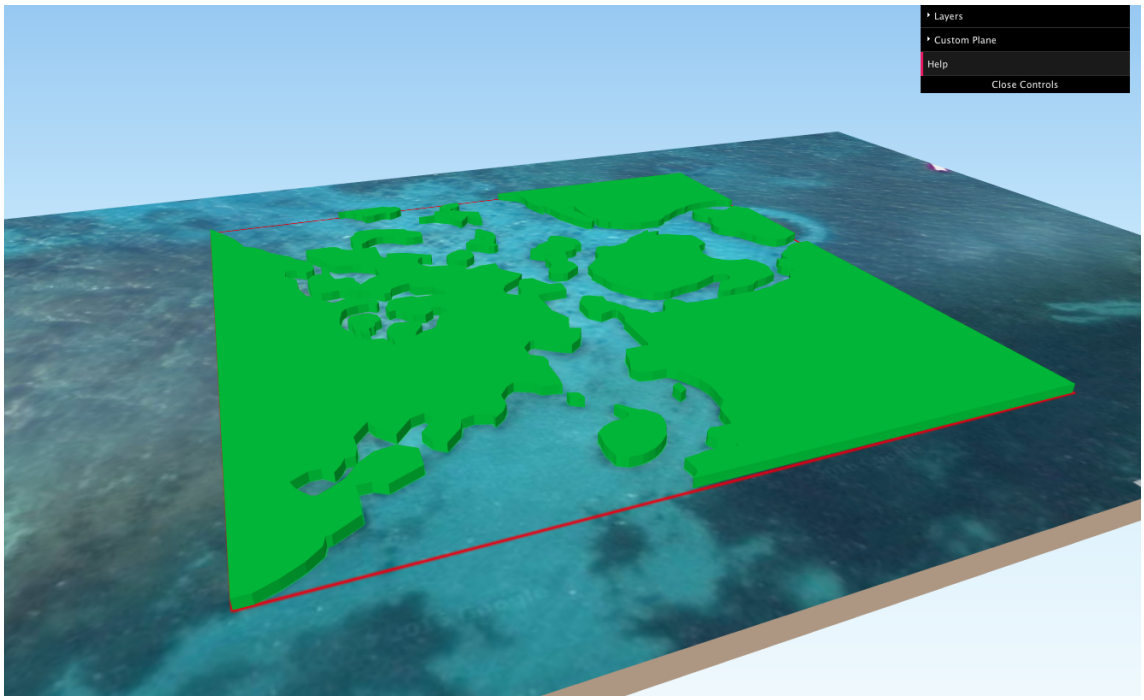

Figure 13: *Exported data of the working area that is represented in 3D. (Extruded) Polygons represent the appearance of seagrass. Color, height and transparency can be set manually or by a value within the data source*

3.4 Blender

Blender is a powerful open source 3D Editor which has the meaning of animation and photo realistic rendering for still images, video frames as well as complex game interaction. It is supported by almost any modern operating system like Linux, Windows and Mac OS. Many tutorials, well written books and even YouTube Videos give a good insight and practice to this complex and (in the beginning) difficult to learn software. Blender is capable to handle several 3D formats and fits perfectly the needs when it comes to modeling from simple to very complex shapes. Since there is a 3D environment which needs to be compiled, it is a very handy tool for this approach. Blender works with meshes that form the 3D objects which appear in the environment. The interaction between Blender and QGIS is given by the Qgis2threeJS plugin by exporting its meshes as collada DAE and OBJ files that Blender is able to read.

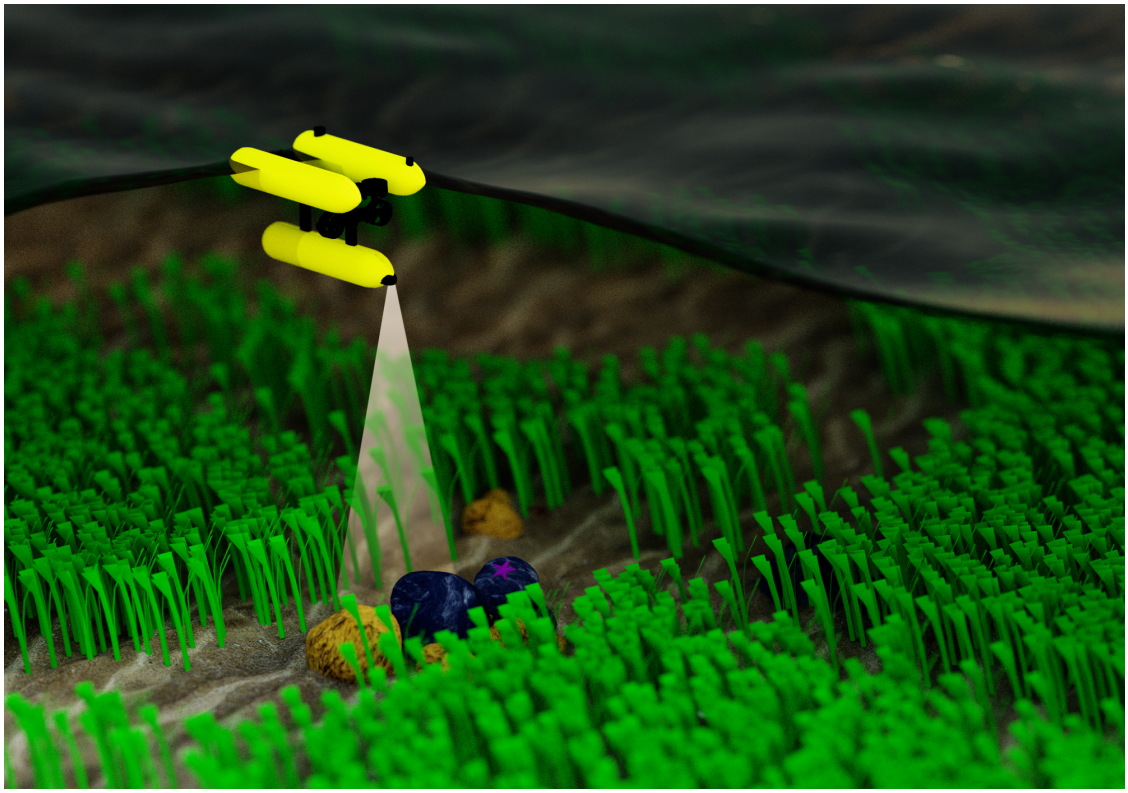

Figure 14: *Rendered image of Scene 2.4 in Blender*

3.4.1 Modeling a Geoscientific Environment (Geoscientific)

To recreate the environment of the satellite image, the mesh of the Qgis2Threejs Plugin is loaded into Blender. By importing the collada file (.DAE), the mesh is loaded into Blenders working area. It represents a 3D room with different tools and working modi. To prepare the plant spread inside of Blender, the imported mesh needs to be placed onto a plain mesh (2D plain). It needs to be subdivided several times due to vertex painting on the surface. As the simulation shouldn't be very resource consuming, the subdivision of the plain has to find a good comparison between realism and resource consumption of the simulator. The more complex the mesh becomes, the more resource consuming the scenery will be in the simulation. Somewhat between 50 and 150 is a good compromise, depending on the simulating machine. By setting Blender from "Blender units" to a "metric" system it is possible to use the same size as found in QGIS since the reference system is also using meters as a unit.

The imported mesh is positioned onto the plain mesh by moving it up and down on Z-Axis by grabbing the blue arrow indicator. The subdivided plain is now put into weight paint mode and the areas of the seagrass polygon is repainted as weight. Blue color indicates light weight while red color indicates heavier weight. green and yellow represent intermediate weight in between. this results in an area that a particle system is using to place plant meshes at a density with respect to the weight that was painted before.

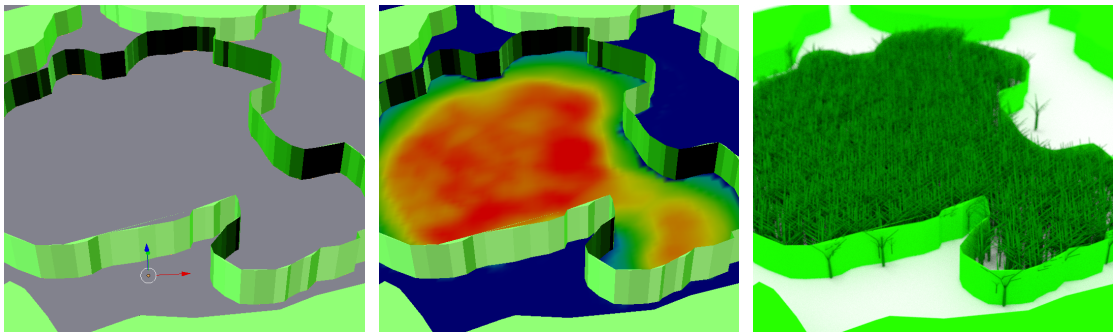

Figure 15: *The images from left to the right show the imported mesh from QGIS that is indicating the seagrass meadows. The second image is showing the weight paint result that is used to indicate the spread and density of the seagrass. The third image shows the result with seagrass within the borders of the meadow. The green walls are removed afterwards*

the whole environment is made this way and all scenarios have different weight profiles.

Modeling the seagrass plant

The seagrass mesh is held as simple as possible with the idea of "the less the better" in terms of verticies. The model doesn't need to be as realistic as the seafloor or the rocks as they appear in a big amount. The meadows are produced with a particle system so there will be ~1000 individuals with vertices. This means that there are

$$1000\,Individuals \times 45\,Vertices \;=\; 45.000\,Vertices$$

that need to be presented in UWSim as well as in Open-EASE. This needs a lot of machine power but makes sure that the robot has a realistic simulated environment. The simulation would run much faster with sprites (single planes that hold an image file) but would falsify the results as the plane doesn't contain any depth information that the robot can notice from above.
The plant itself is a model as simple as possible to prevent high resource consumption during simulation. It would be possible to just use one very simple mesh per meadow but that would make the object recognition too easy and wouldn't represent a simulation as realistic as possible in this simulator. The Mesh is build up with a plane that is subdivided in 4 sections, divided into a left and right part. That makes eight faces per leaf and 24 faces per plant. The sections are bent on one axis so it represents a leaf that is partly bent by gravity. The mesh is copied two times to build a whole plant. After the plant is finished, the meshes need to be grouped so they all can be used when the vertex group is referring to them as individuals.

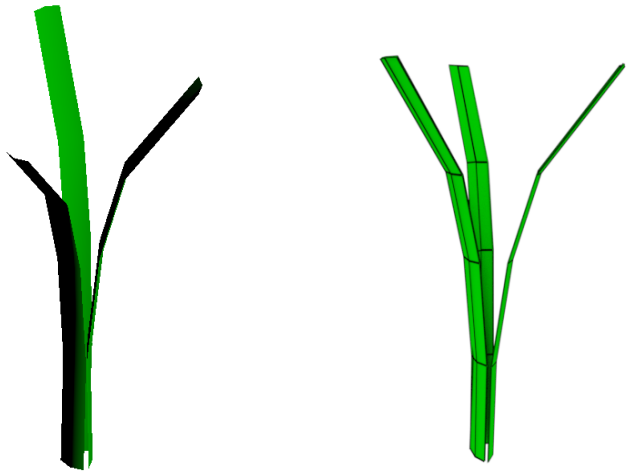

Figure 16: *Meshes of the Seagrass plants that are used in the simulation*

The Seagrass mesh is inspired by pictures that can be found all over the internet like at seagrasswatch.org or from the BBC. The structure is very diverse so I had to decide between several and chose the one that looked familiar to me. The sprout normally comes from the rhizome but as the robot doesn't need to see that it was not necessary to build the mesh that realistic.

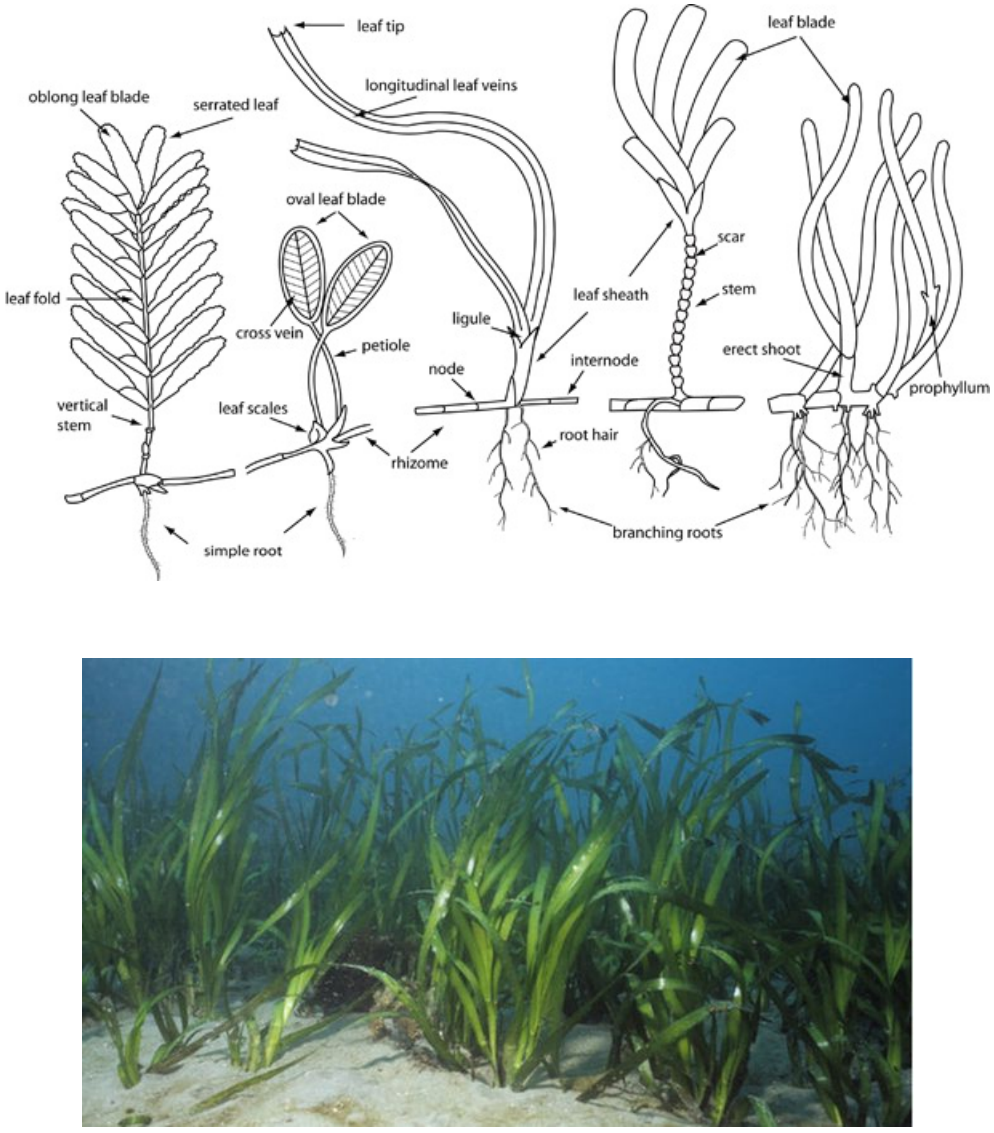

Figure 17: *The sketch shows that the leafs mostly come from the rhizome and the shapes differ a lot. The lower image shows that the leafs stick out of the sediment in groups and show a small amount of stem and sheath. This can be ignored as long as the leafs look almost like in the images*

Modeling the rocks

To model the rocks, it is necessary to start with either a spherical or a cubic mesh. Since the simulation takes place in marine environment, the rock will be roundish due to water column movement and influencing forces like currents and mechanical and chemical weathering by sand, water and other objects.

Figure 18: *Rounded rocks of a marine environment on the left and not rounded material from the field on the right.*

The modeling of the rock is done by using the sculpture mode in Blender and deforming a sphere. As the deforming process needs to be done with a fine meshed object, it produces many "faces" (planes to create the object) and this results in a big file size.
To discriminate both rocks, textures of blue and yellow color were taken. Both are free textures from the internet that are free to use and are manipulated in GIMP. Both are identified as rocks but with different attributes in the background knowledge.

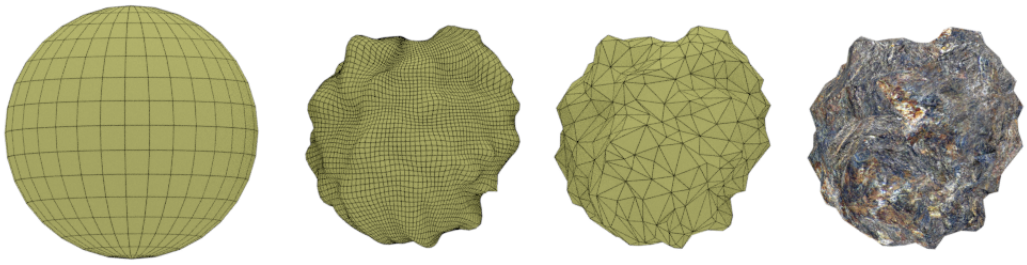

Figure 19: *The rocks are made from a sphere that is subdivided and sculptured. To reduce file size it is reduced with a modifier and finally textured*

The textures indicate many information, i.e. the color which enables the robots and semantic detection. By identifying these and give a link to the database, the rock can be asked for in Open-EASE.

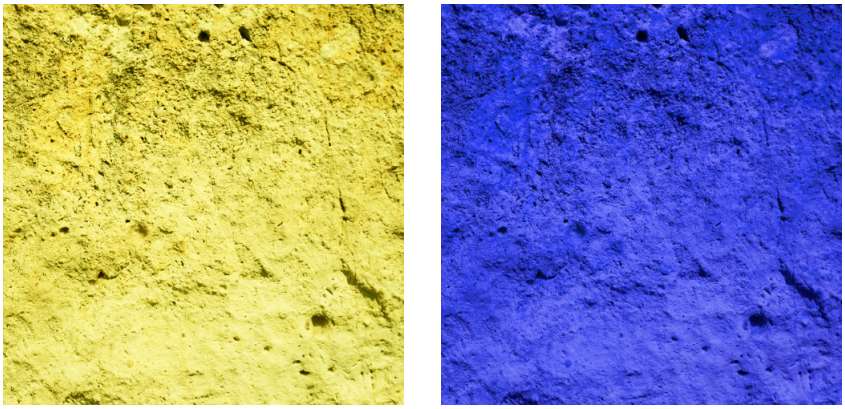

Figure 20: *The left texture shows the yellow rock that represents a link to the entries to the database that contain meanings to this kind of rock. The right texture does the same but links to entries of blue rocks. Both represent a semantic link to the database*

Modeling the seafloor

The seafloor is not that important for the detection algorithm but to be more realistic on the geoscientific side, it is modeled and many details are taken serious. Ripple structures often occur in shallow marine environments that contain tides and currents. The ripples can also be found in lakes with a sand floor and at beaches near the coast.

Figure 21: *Ripple structures on a beach of Borkum, Germany. These indicate a current that has stacked the sand grains*

In Blender the ripple structures are made with a displacement map that is used like a normal height map. The combination of the Texture with the displacement map results in a scaleable seafloor face that theoretically can also be scanned via laser scan which is also included into the simulator framework.
The ripple structure represent a small grain size which is easier to deposit by weak currents. The smaller the grain size, the likelier is the deposition in terms of ripple structures. The requirement is a force (current) that moves the particles.
To distinguish between grain sizes, it is possible to identify the ripple structures and suggest the grains by ripple formation of:

Normal structure	*coastal structure*
no transport	no transport
faint planar lamination due to poorly sorted sediment	ripples
dunes	upper planar lamination
upper planar lamination	antidunes
antidunes	

Table 4: *The normal structure can take place in non-marine environments as well but in modern environments the coastal structure is more likely (cf. Fig. 21). Different grain sizes need different forces to be moved so the deposition inference grain sizes (source: [45])*

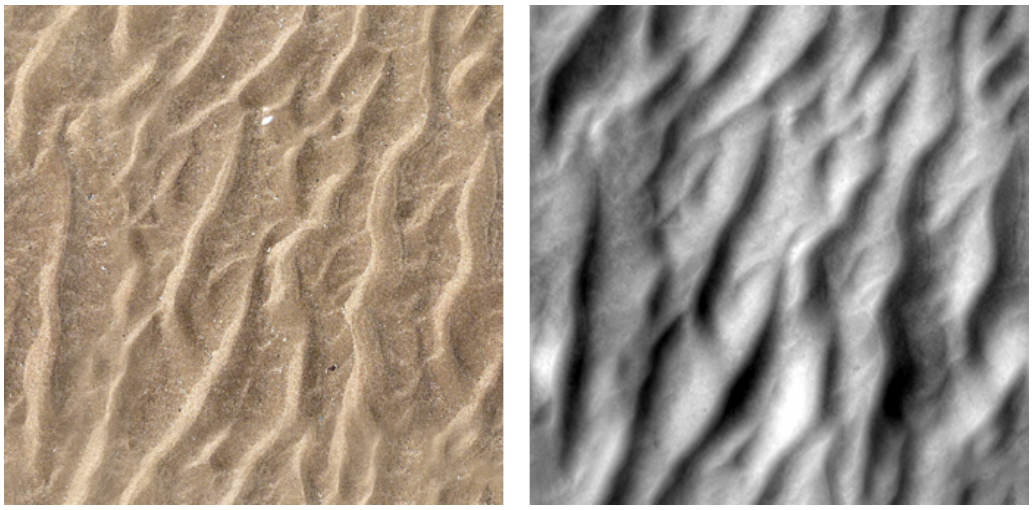

Figure 22: *The texture for the seafloor and the calculated height map/displacement map for the texture*

The setting of the seagrass appearance is taken via satellite image from Google Maps in QGIS and is imported as a rough mesh to Blender. The result is a rebuilding of the actual site to use it in the simulator. As said before - the simulator is used to simulate the robot, interacting with the environment. Its not meant to observe just in the simulator but as a representative it is good to keep it as close to the real world as possible.
The ripple structure of the seafloor is generated with a displacement map and the displacement modifier in Blender. The software „CrazyBump“ is able to calculate normal maps as well as displacement maps from a single diffuse texture by generating high quality height-maps from the darker and lighter parts of the texture itself. The result is a direct copy of the texture of the seafloor in black and white that can be used to generate heights and depths on the mapped texture.

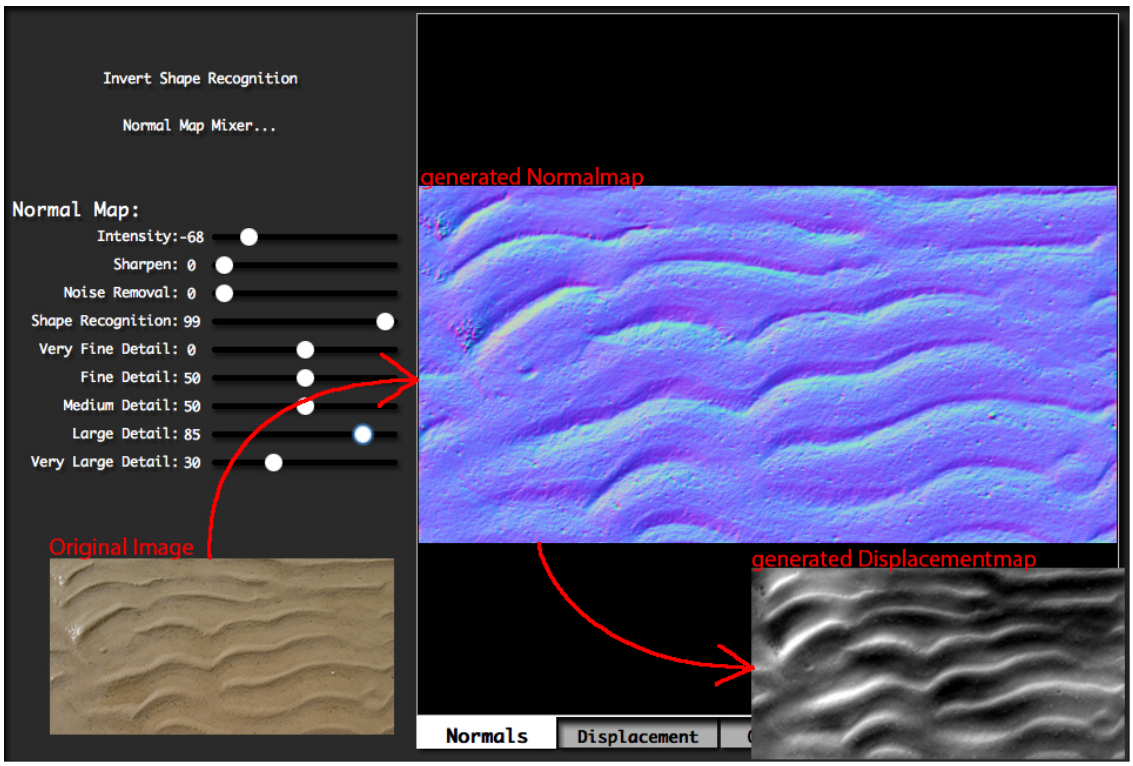

Figure 23: *CrazyBump is a free software to produce Height-map like images that are used to influence light behavior on surface textures or displace a mesh. In this work CrazyBump is used to produce the seafloor*

As the Displacement map is produced, the mesh inside blender needs to be subdivided several times. Blender will interpret the lighter areas as high and the darker areas as low profile (default setting) but this can also be manipulated in several ways. Here,

I have used just a few mm in hight to produce a seafloor. To give an idea to the color displacement I have produced a map from a red rock texture.

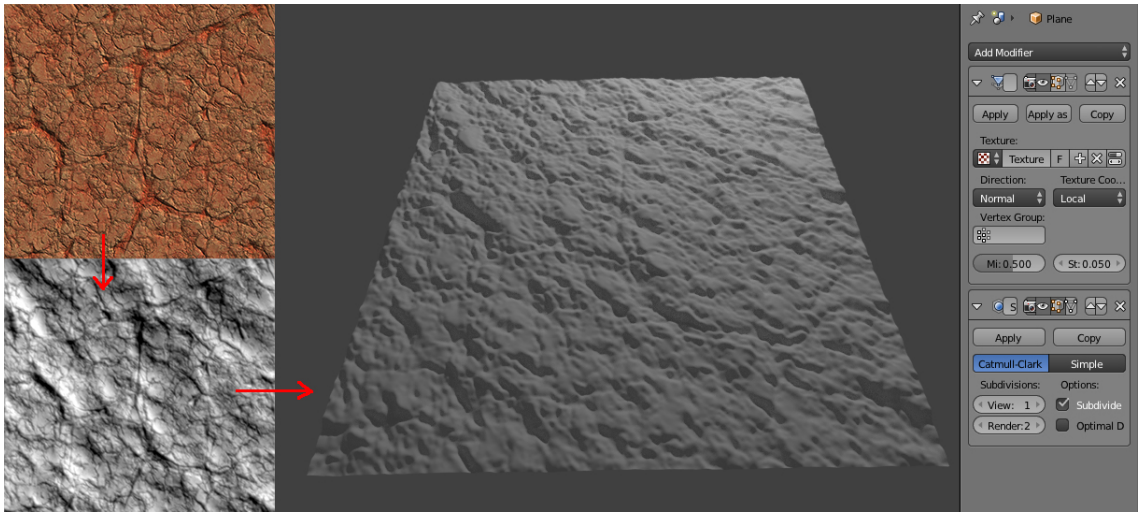

Figure 24: *The red rock texture is calculated into a displacement map and is used to create a surface with heights. The seafloor is made the same way*

The displacement map and the original texture are used on one mesh to create the seafloor. This method ensures that the texture looks realistic and the seafloor mesh results in a surface, visible to the laser scanner in the simulator.

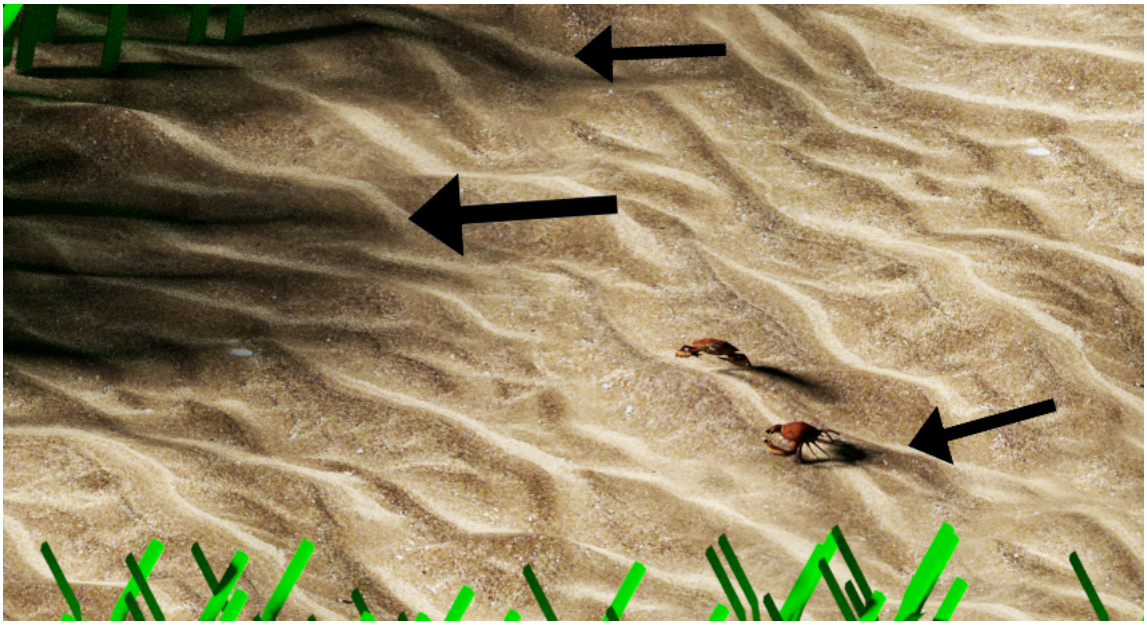

Figure 25: *The surface of the terrain mesh is made from the texture itself by calculating a displacement map from it and using both, the displacement map and the diffusive texture on one mesh. Shadows of objects indicate the three dimensional appearance*

3.4.2 Scenarios for Simulation (Robotics)

Once the meshes in QGIS are generated and saved, all information that are needed for the environment are gathered, the meshes are used to set areas for the appearance of the seagrass within the blender environment onto the seafloor. To do so, the weight tool is used because the appearance of the plants can then be manipulated easier. With this tool it is possible to set a density of objects to appear in the marked areas. In this project this is done on the seafloor mesh and the polygons from QGIS as borders for the appearance.

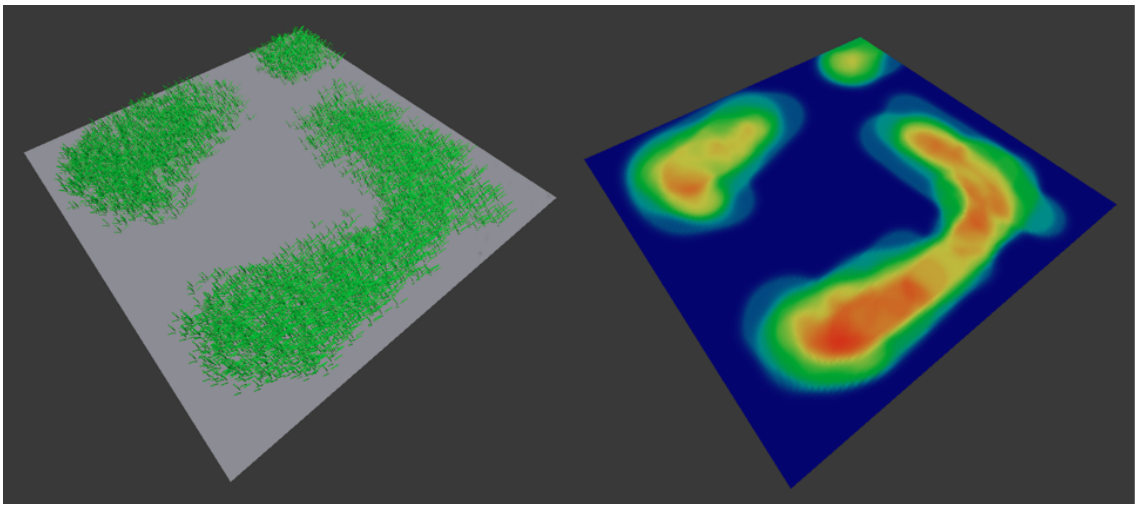

Figure 26: *Simulation in Blender is showing the density of occurring seagrass plant objects. Red indicates a high occurrence while light blue is the lightest occurrence or the lowest density. Darker blue is indicating no occurrence as one can see on the results in the left image.*

The upper image shows the principle of the weight paint tool. The right one is showing high weight in red color and blue with lighter weight. The left one is the resulting density spread of the seagrass object with respect to the weight.
The original density is measured by darkness of the appearing seagrass from the satellite image. Darker areas will have more weight and lighter areas will be "lighter" in density. This results in a real world reconstruction of the area that can be investigated in the simulator. The generated environment is exported in .OBJ file format so it can be used with any 3D program and system. This is important due to recent (open source) robotic systems are using open source mesh formats to be as flexible as possible and of course to be as cost effective as possible. This has a testing character to make sure that the robot is seeing the real world as a model. The identified field can later be used to calculate deviations.

First, the shape of the seagrass meadow is taken like it is done in traditional methods known from habitat mapping from aerial images. This step is described in chapter 3.3 and can be done with several plugins that can reach a maps server. The .DAE file is loaded into Blender and is used as a template to produce the weight field for seagrass meadow representation.

The weight field represents the exact area that is present in the QGIS section to reproduce the meadows. The only user interaction is a painting of the weight in different layers so the distribution of the plants differs in some spots.

The weight field is used within a particle system that refers to the painted weight as a vertex group which is representing the occurrence density of plant meshes. Because the density will be filled with objects, the type of the particle system need to be set as "hair" which doesn't mean that the seagrass will be as small as a hair but more that the appearing "hair" will be replaced by the plant mesh. By placing the meshes, the set of plants are exported as one object that is loaded into UWSim. It is important that only the selected group of plant objects are exported, otherwise the seafloor mesh. The seafloor is getting a ripple structure to indicate a shallow marine environment. The Texture is transformed into a height-map/displacement-map to simulate the ripples. The floor mesh becomes the ripple structure of the texture, which is indicated here with a light source:

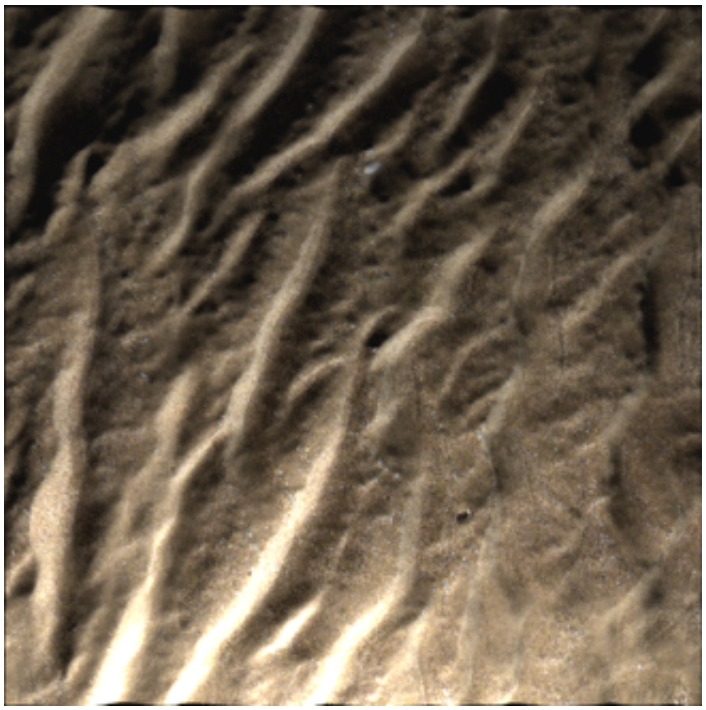

Figure 27: *The seafloor mesh with the displacement map (cf. Fig. 22) and diffuse image texture, illuminated by a light source from the lower left corner*

Other than the plants, the rocks in this environment are sculptured from a sphere. The rough shape is sculptured after subdividing the sphere several times and then used the sculpt tools. The reduce modifier minimizes the faces because a smaller mesh is positive for the simulation due to resource consumption. The mesh can be used for all occurring rocks. This makes it easier to place several rocks without producing many meshes. To spread diversity, the other rock objects are placed in groups and have different scales. The rocks differ to each other by taking up different textures. For a high contrast a yellow and a blue texture is used.
As the surface of the seafloor is produced, the weight tool (mentioned before) is used to place the meshes of the plants. Blender has a special mode for exactly this function that one has to activate. It is possible to change the size (Radius) and the intensity (Weight) of the brush (cf. Fig. 28). Dark blue color means light/no weight and dark red represents the heaviest weight. Green and yellow color are intermediate weight. The accuracy of the weight depends, like the displacement, on the subdivision of the mesh. The more subdivisions are present, the more accurate the tool can be used. Since subdivision means more faces and a bigger file size, it is necessary to find a reasonable compromise between accuracy and machine power because the bigger file size can later slow down the simulator a lot.

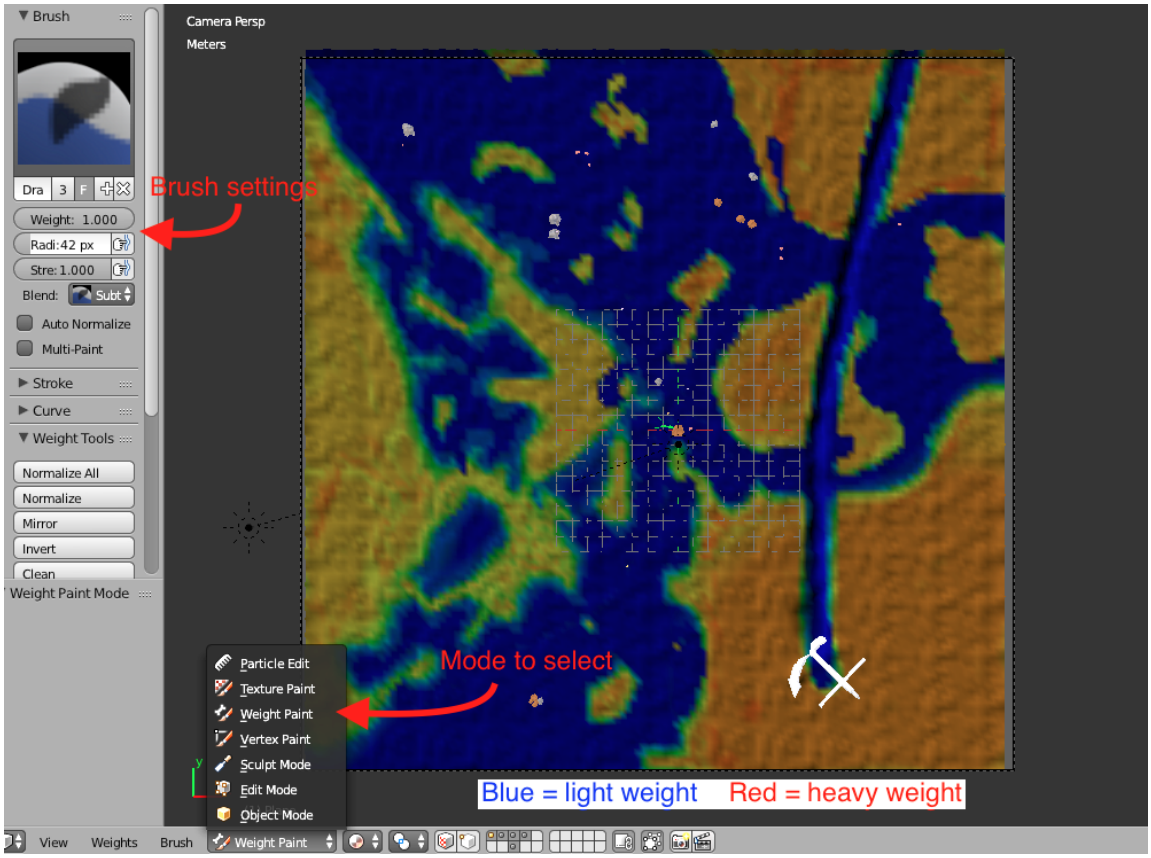

Figure 28: *The Blender interface with the option to choose "Weight Paint" as one of the modes and the brush to paint the weight can be changed in the upper left panel. In this image one can see the production of scene 2.2 in which an anchor ripped trench into a seagrass meadow. The trench has no weight because no plant will be present there. The modeling process can also be seen in Fig. 15*

As the weight and the borders of the seagrass are defined, the plant meshes are placed into the vertex group. As there are many plants that appear, the particle system needs to be set as "Hair" which enables to set the number and length of the "hairs"/plant meshes. These numbers are given in Table 6 as "Emission" in chapter 3.5. By using the "Object" option in the Render section, the hair placeholders are replaced by the plant meshes.

Figure 29: *Settings for the particle system in Blender with the number of hairs and their length*

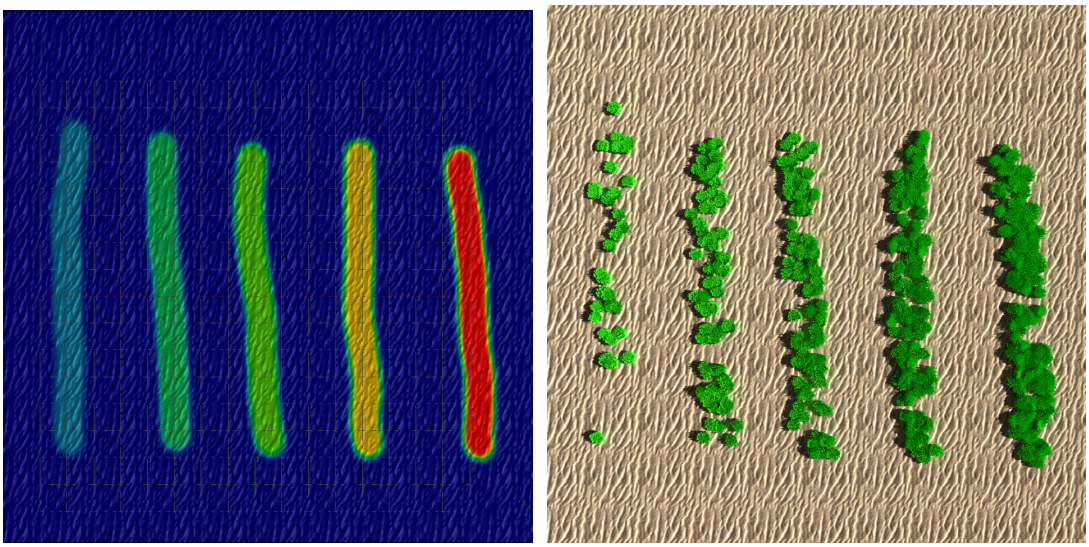

Figure 30: *Painted weight on the left image with weight values of: 0.2, 0.4, 0.6, 0.8 and 1.0 - The right image shows the rendered outcome of the particle distribution*

3.4.3 Exporting Environment Objects

Every set of objects needs to be exported from QGIS as well as Blender and than imported to the simulator again. This means they need to be exported in a file type that is supported either by QGIS and Blender as well as by the simulator. It is important to set the rendering type to „Blender render“ otherwise the textures will be messed up due to different methods of texture alignment in the different render engines.

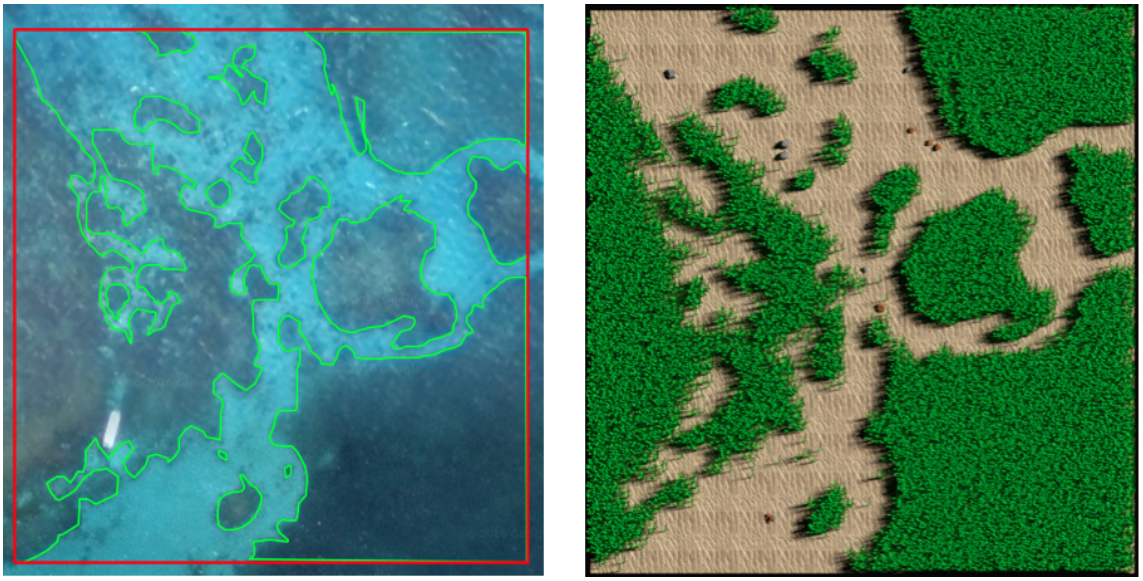

Figure 31: *The left image shows the real world situation with borders in QGIS. The green lines represent the borders of the seagrass meadows while the red one indicates the borders of the operation area. The right image shows the reproduced area in Blender as it is a 1:1 copy of the meadows. The meshes can be exported to be used directly within the simulator.*

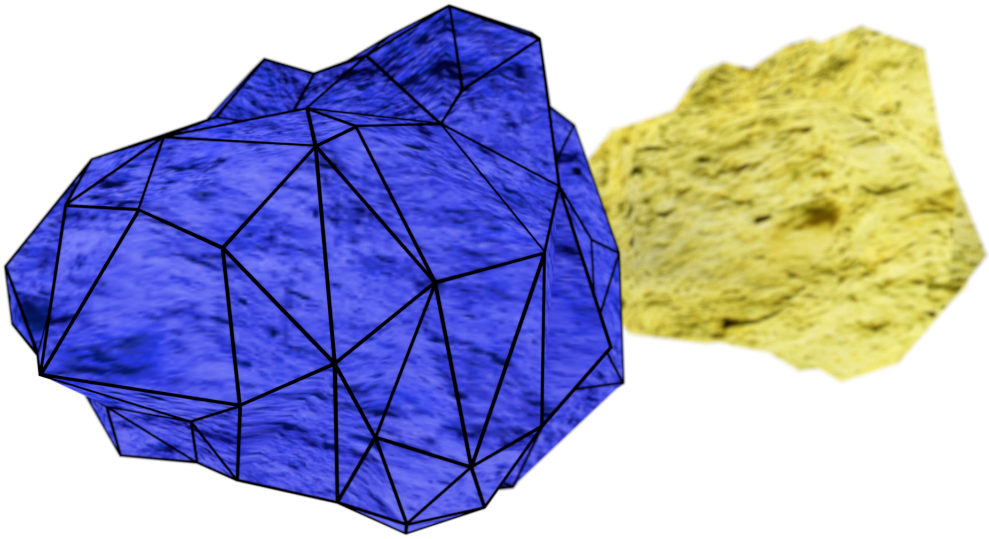

Figure 32: *Two of the rocks that are spread over the seafloor in the simulation. One is blueish, the other one reddish. They represent different kinds of rocks that the robot has to recognize.*

Cycles engine works in a different way for texture placement than Blender render and makes it impossible for the simulator to interpret the texture coordinates correctly so rendering for illustrations is done in cycles while the working process is done in Blender render. The objects are also reduced to a minimum of faces to reduce resource consumption during simulation. To do so, every model is filtered by a reducing „Decimate“ modifier that reduces the faces at a set factor. In this case the rock model is reduced by 0.05 to decrease the amount of faces a lot and end up with a model that almost looks the same but is dramatically reduced in size. For simulation this makes a very big difference when it comes to resources of the simulating machine. The vision of the robot makes no difference between the two versions of the rock but the simulation can be done on a weaker machine (cost effective) as well as in a more effective way on stronger machines because. It reduces the chance of run-time errors or memory outrun (time efficient). This is not only the point for the rock but when it comes to a high amount of objects in a complete system it can make a big difference so less faces are always better to reduce the resource consumption.

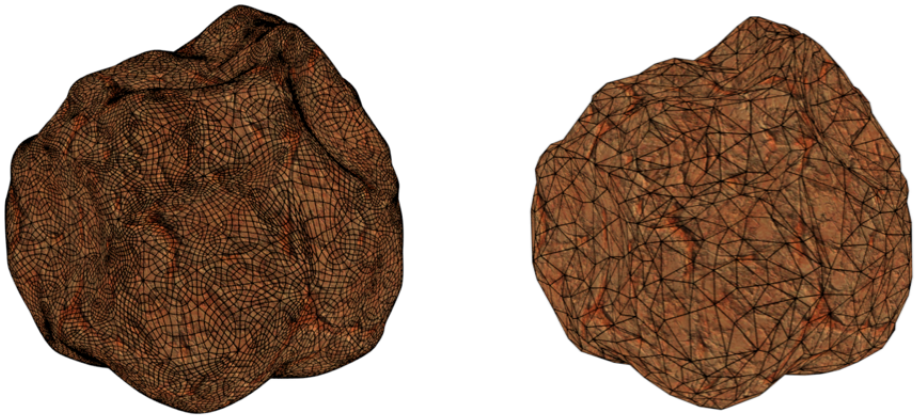

Figure 33: *On the left the rock with 18,064 faces and a file size of 2.7MB versus the smaller version on the right with only 1,463 faces and 130KB in file size. No difference for the detection algorithm but a big difference to the simulating machine*

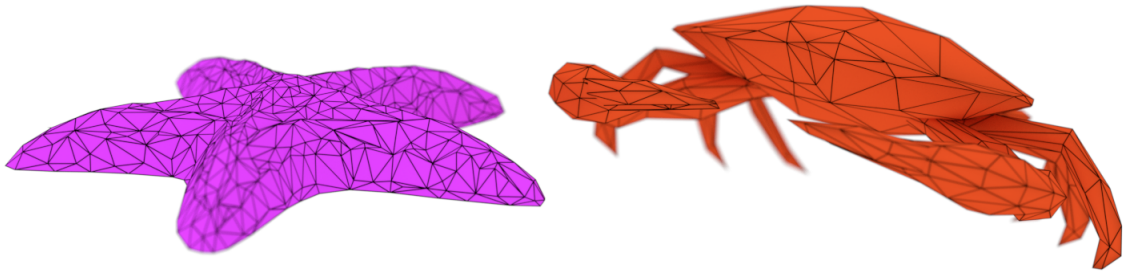

Figure 34: *The meshes of the crab and the starfish that are used in the simulation - Both are low polygon models that come with small file size*

3.5 Scenarios

The environment can be manipulated in the .XML files that the simulator is referring to while starting the simulation. It contains all necessary values to start the environment like coordinates of the meshes (objects) and location of them on the hard drive as well. Also all values for the robot and its sensors. The sensors need values, like sensitivity or orientation, and are stored as digits. They are known from the original parts i.e. the camera resolution comes in pixels, the orientation of the parts in meters and rads and the multi-beam angles in degrees. The header of the file also contains the direct, constant influence to the environment like strength of the wind from X and Y direction with seed, choppiness and scale of the waves, depth and initial height of the water surface. Fog, color and density of dirt in the water column can also be set within this file. The density of the dirt is simply set by one value between 0 and 1 while 0 is clean water and 1 is completely dense. The color of the water as well as the fog is set in a RGB range. Parameters of the simulated environments are:

Influenced	Value	-	-
Wind X	0.04	no unit	
Wind Y	0.04	no unit	
Windspeed	12 km/h		
Depth	1000 m		
Wavescale	1^{-7}	no unit	
Choppyfactor	2.5	no unit	
Ocean Surface Height	0 m		
Fog Density	0.1	no unit	
Fog Color	R: 0	G: 0.05	B: 0.3
Water Color	R: 0	G: 0.05	B: 0.3
Attenuation	R: 0.015	G: 0.0075	B: 0.005

Table 5: *Settings for each scenario within the XML files*

All scenarios are following the same schematic rule of development. The first of four steps is showing a first bloom of the plants while the following are showing increasing growth. Table 6 indicates that the first blooming starts in winter times, one of the blooming seasons of seagrass, and is also increased in summer times, the second blooming season. In spring and autumn times the growth is a bit reduced as indicated in „growth [x100]“. It starts by 0 because its refereeing to the scene before.

Truly these values are exaggerated and seagrass needs much more time to grow that much but in this work it is used to indicate a development that can be recorded autonomously by a robot scientist.

Scenario	Season	Emission	Hair Length	growth [x100]
1.1	Winter	500	0.2	0
1.2	Spring	800	0.3	3
1.3	Summer	1300	0.4	5
1.4	Autumn	1400	0.45	1
2.1	Winter	500	0.2	0
2.2	Spring	800	0.3	3
2.3	Summer	1300	0.4	5
2.4	Autumn	1400	0.45	1
3.1	Winter	500	0.2	0
3.2	Spring	800	0.3	3
3.3	Summer	1300	0.4	5
3.4	Autumn	1400	0.45	1
4.1	Winter	500	0.2	0
4.2	Spring	800	0.3	3
4.3	Summer	1300	0.4	5
4.4	Autumn	1400	0.45	1

Table 6: *The table shows the values used in Blender to represent a growth. Emission is the density of plants and Hair Length shows the size of the plants in Z-Axis.*

While the first scenario is showing no disturbance to the meadows, the others are showing anthropogenic and natural influence to the growth and spread. This work doesn't aim to show the exact seagrass behavior but it helps to avoid the use of an expensive real world robot mission. The single scenarios are representing different influences. While the first one is undisturbed and indicates a complete growing of meadows, the others are showing influenced scenes. Shadow casting, anthropogenic and animal disturbance are indicating that the meadows are damaged in some way.

3.5.1 Sequences 1.1 to 1.4 - Unaffected

1st sequence of Scene 1

The undisturbed scenario starts with a beginning of a growth and sets the first appearance of the meadows. As the growing continues, the meadows become more visible and bigger. They also develop borders so the robot is able to differentiate between the plants and the background.

Figure 35: *Scenario 1 indicates an undisturbed environment with normal growth. From slight coverage in 1.1 up to total coverage in 1.4. The first picture indicates winter times with high growth and first visible borders. A very slight increase of plants in the second picture indicates spring time with a growth rate not as high as in winter times. The third picture shows a very large coverage of the seafloor which isn't visible in the meadows anymore. This is caused by the high growth rates in summer times. The full coverage but a very slight increase can be seen in the fourth picture which represents autumn.*

The occurring spots of seagrass and seafloor are taken from the satellite images and are reproduced in Blender and QGIS to investigate it with the robot in the simulation. It can also be used as a reference to check if the algorithm is working properly. To make the area more interesting and to test the algorithm's detection function, some additional objects are added. As animals some Starfish and crabs are present with different colors so OpenCV can differentiate between the organism species. There are also rocks of different size, color and deposits. They are all typically rounded since they are lying in the ocean and represent different reflectance due to different colors of brighter and darker appearance. This is another indicator that an optical sensor can use to observe differences.
The spread of the plants are encountered with Blender values of the particles systems emission of:

Scene	1.1	1.2	1.3	1.4
Hair Length	0.2	0.3	0.4	0.45
Number	500	800	1300	1400

Table 7: *The table contains the values for Blender to produce the grown Seagrass. The first step in this scenario represents the first bloom of the meadows and continues over the next steps*

2nd sequence of Scene 1

The second scene is the same as the first scene but with increased growth of the seagrass in a healthy way. A single plant leaf of seagrass can grow up to 100cm and I want to see the maximum of the growth in a healthy way after one time period without any disturbance and let the robot take a closer look and compare it to the first scenario.

3rd sequence of Scene 1

The third scene represents a winter state for the seagrass since it has its growth maximum in spring and autumn but its minimum in winter and summer. So in winter times the stress for the plants is much higher and it shows some decreased growth rate [6].

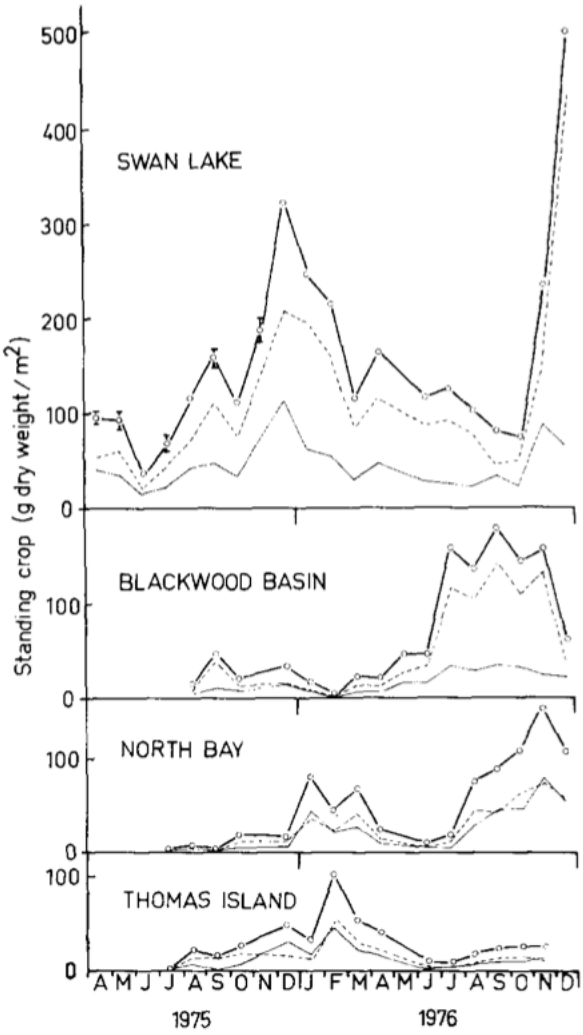

Figure 36: *Congdon and McComb found the leafs growing increased in winter times and saw a low appearance in autumn to winter with a rapid increase in spring to summer (source: [6])*

In QGIS it is possible to visualize the different growth rates and number them in a scientific way.

4^{th} sequence of Scene1

The last scene represents the autumn state in which the growth is reduced but almost completed for the year just before cease blooming. The spread reaches the maximum and represents the actual state that can be found in aerial images from Google Earth or other satellite image sources. The robot is able to see all plants and shows the complete coverage of the area but some objects like rocks and crabs are covered too

much with plants so the threshold of 0.28% doesn't get into use here so the robot ignores the object.
This scenario is used as a reference to identify the differences between the undisturbed area with influenced areas.

3.5.2 Sequences 2.1 to 2.4 - Anchoring

Figure 37: *The scenario is showing an anchor that destroys parts of the upper right to the lower right meadows. The growth in the trench is decreased and leaves a scar in the recovered field. 2.1 shows the initial situation that isn't different to the one in 1.1 but 2.2 contains a mesh of the anchor. 2.3 is showing a slightly recovered area while 2.4 has recovered completely but with a scar that indicates a former destruction.*

1^{st} sequence os Scene 2

The first scenario is not different to the first reference scenario but the second scene in here receives its first anthropogenic influence. The seagrass gets hit and partly destroyed by an anchor that came down from a ship and will remain only in the

second scene. The robot is supposed to identify the objects and show the trench within the seagrass in the influenced area. By comparing the undisturbed area with the area of the trench its easy to see the difference.
The anchor has ripped a trench in some of the meadow areas and leaves a destroyed area that the robot should be capable of to recognize. The lost of the seagrass mass can be reproduced in QGIS as the robot is recognizing less white pixels when it goes over the destroyed fields.

2^{nd} sequence of Scene 2

The second scenario shows the area influenced by the anchor. The robot is supposed to check the whole area and the data will point out the regeneration of the influenced area to let the user calculate the deficits of seagrass. Also it will represent the growth of the seagrass in such an influenced area. It is possible to calculate the trench as an object and to link it to the anchor in a semantic way. By looking at the area by comparing the coverage of the seagrass, the loss of seagrass is 2.09%. At the first look this is a relatively small number but if it would be more than just one anchor, the number would increase dramatically.

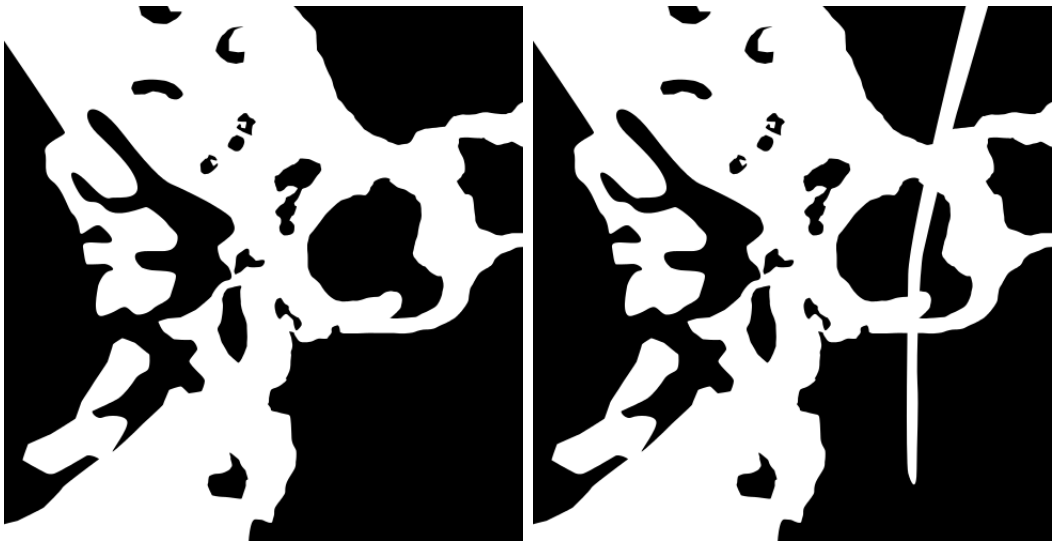

Figure 38: *The left image shows the complete coverage of the seagrass area while the right image is showing the same amount but with the trench which has been cut with the anchor. The left unaffected scene would make a coverage of 52.52% and the right a coverage of about 50.43% which makes a difference of 2.09% of coverage. Since this is a primary hypothetical prediction, it can also differ in the end but this is the way, the calculations are set.*

3^{rd} sequence of Scenario 2

The third scenario is supposed to show if the seagrass has recovered from the stress of the influence and to calculate a trend how long the plants will need to recover from such an influence. This will point out that the recovering is depending on the season, the impact takes place. As the seagrass has a reduced growth in some seasons, an area protection could possibly be set during certain month within the year. Summer is one of the most crowded seasons for people to use boats and the growth rate of the seagrass is relatively reduced during this time. This means that the recovering process would take place a few month later.

Figure 39: *The mesh of the anchor that appears in the simulation*

4^{th} sequence of Scene 2

The fourth scene is showing a complete recovered seagrass meadow with a scar in it that the robot is not able to identify. This means that it recovers to a normal state but got influenced during the year and the carbon uptake as well as the productivity was reduced in total.

3.5.3 Sequences 3.1 to 3.4 - Shadow Casting

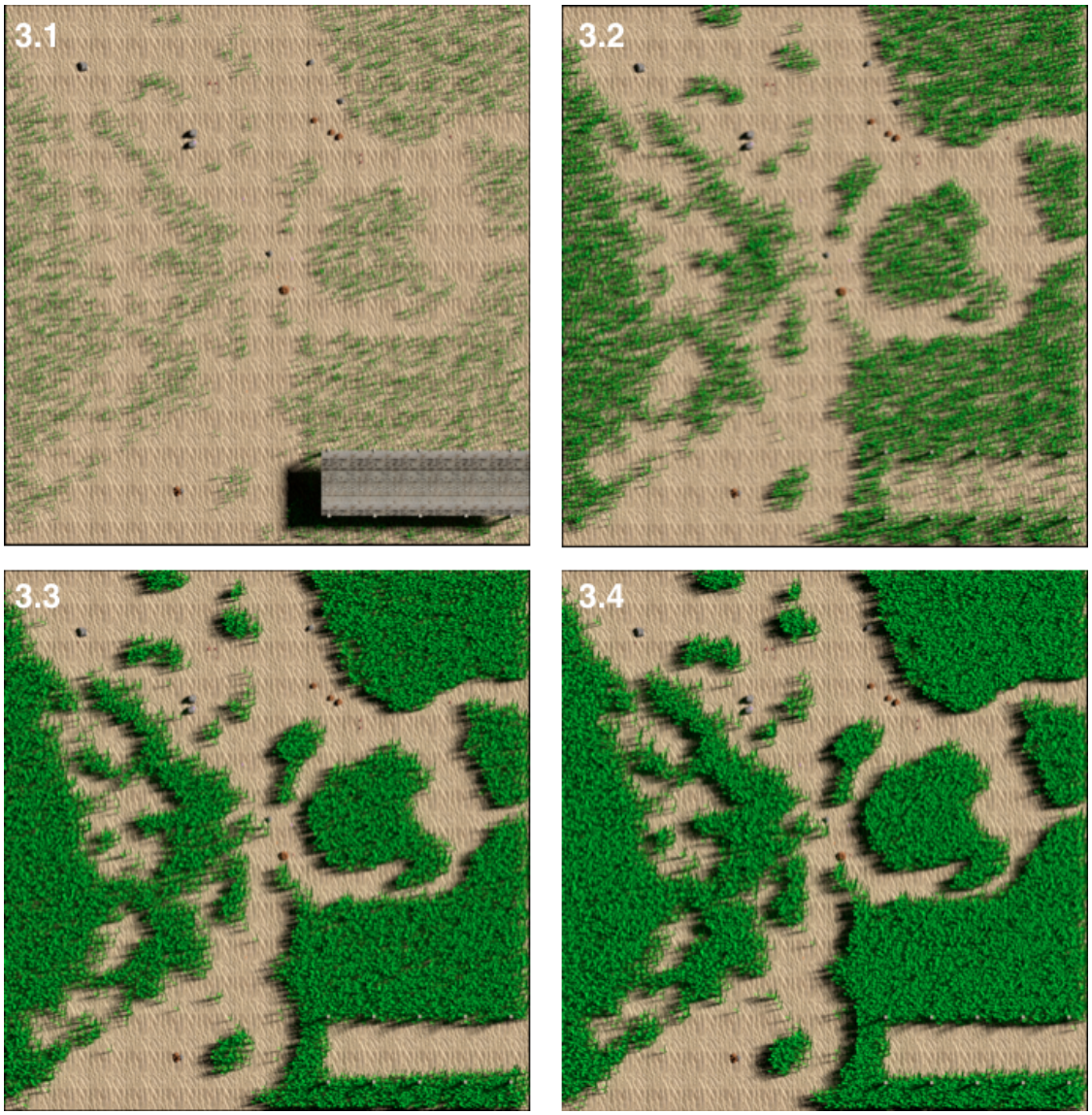

Figure 40: *The scenario is showing an anchor that destroys parts of the upper right to the lower right meadows. The growth in the trench is decreased and leaves a scar in the recovered field. 2.1 shows the initial situation that isn't different to the one in 1.1 but 2.2 contains a mesh of the anchor that can also be identified by the robot. 2.3 is showing a slightly recovered area while 2.4 has recovered completely but with a scar that indicates a former destruction.*

1^{st} sequence of Scenario 3

The first scenario shows a plank walk over a seagrass meadow. Due to the light deficits the seagrass will start to die and decrease in this area. This scenario is

the starting point like directly after the building progress has finished so the winter scenario will provide a few plants and will look like before but during the first change the material under the plank walk won't grow because its covered by the plank. after the first stage, the seagrass that is undisturbed, will grow like in the previous scenarios but with a decreasing size under the plank walk. This leads to a certain loss of mass material in the area of the plank walk object. So in the third situation the seagrass under the plank will be dead and the seafloor will be completely present. Since there is only one major influence to the scene, it is less time consuming to simulate and scan the area and only the SE section needs to be scanned four times.

2^{nd} sequence of Scenario 3

The second scenario represents the next season after the walkway was installed. It shows that the seagrass continues to die or doesn't continue to grow and the underground will be present. The mesh is removed in the visualization to make more clear what is happening underneath the walkway. Major differences are present in a larger scale. By scanning the area a point outcrop would not make a big difference but the whole picture shows first differences to the surrounding.

Figure 41: *The mesh of the walkway that covers parts of the seagrass. One walkway alone doesn't make a big influence but many of them would influence much more*

3^{rd} sequence of Scenario 3

The third scenario shows that the seagrass underneath the walkway is completely vanished and the seafloor is completely present to the robot. I will calculate if the growth of the surrounding seagrass can compensate the lost with respect to the storage ability of carbon of the seagrass.

4^{th} sequence of Scenario 3

As the seagrass is grown completely in this scene, it is possible to calculate the difference between an unaffected and this scene. As just one walkway does influence this area of 1 km^2it wouldn't influence the whole area of east Australia. But for sure it isn't the only walkway in east Australia so in total there should be an influence of shadow casting objects.

3.5.4 Sequences 4.1 to 4.4 - Animal Grazing

Scenario four represents a fictional shift of an animal habitat to the working area. Due to climatic changes, the animals are forced to search for another food source and chose to graze the seagrass. It is structured as an undisturbed beginning but ends with holes in the meadows and parts that are underdeveloped. Compared to the first (undisturbed) scenario, this is the worst case and is supposed to leave a much bigger impact. It shows that there are many spots of low or no seagrass that leaves a blank spot.

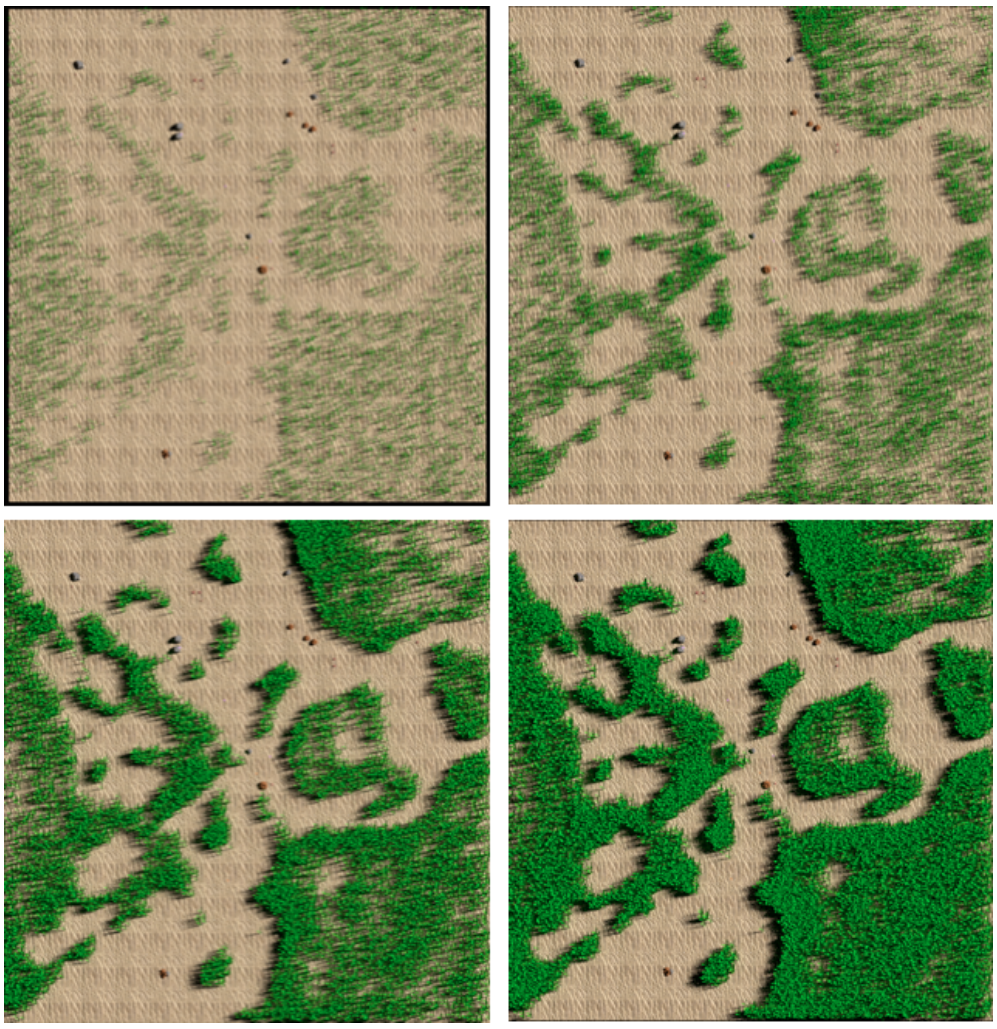

Figure 42: *Scenario 4 - A fictional habitat shift of animals forced them to search for another food source and they ended up grazing the area of seagrass. The first section is comparable to scenario 1.1 but at 4.2 the grazing started and the area ends with holes in the meadows as well as underdeveloped parts*

1^{st} sequence of Scene 4

As shown, the scene 4.1 is comparable to scene 1.1 to indicate a first upcoming seagrass. Manatees are also grazing small plants but I want to show a destruction that can be compared to the other scenes. Grazing of the plants not only happens by manatees but also by birds. The birds, mostly searching for other animals, plucking out the plants or at least damaging them. one bird would not make a big difference but many of them over a short time could be a problem.

2^{nd} sequence of Scene 4

Scene 4.2 represents the meadows directly after the animal grazing. The meadows are highly damaged and especially the one (skull shaped) in the middle contains a hole that will remain over the next scenes. As the robot will identify the new borders of the seagrass and the whole area is affected, the consequences of the grazing will be visible in the traditional maps as well as in the semantic maps.

Figure 43: *A manatee is a seagrass grazing animal that normally eats eelgrass and other water pants*

3^{rd} sequence of Scene 4

The third sequence shows a normal development of undisturbed plants but the overall image reveals the damage. Many plants are under developed or not present. The holes still remain as there are no more plants that can grow. In some cases, more objects like rocks can be identified easier as the plants don't outgrow their appearance.

4^{th} sequence of Scene 4

The plants are fully developed but holes are still present. The grazing damaged the whole scene and all meadows. The overall detection is expected to be much lower than in the undisturbed scene 1.4.

3.5.5 Alternative Scenarios

The approach of this work is not fixed to seagrass. It is possible to manage different scenarios and different sensors. As a user is able to add new sensors to the simulator (s. UWSim Wiki), the surface can be anything. The Robot can dive through chemical clouds or scan temperature - there is no limit for these sensoric missions. This approach manages to log the gained data and store it in a format to visualize it a Geo Information System and record it for Open-EASE. Therefore one can run a desired mission visualize it on a map or even in a 3D environment.
One possible scenario is to use a changing underground as can be found near coastlines. Most of the concrete undergrounds are used for coastal protection as they are lodging the coast.

Figure 44: Left: *Real world scene of coastal protection - For coastal protection, a structure of concrete cement is built and covers parts of the seafloor or beach site. This can also be found at steeper coasts, not only at flat beach sites. Right:Remake in Blender of the same scene as an alternative scenario*

Coastal protection has a very important meaning not only to the people living there but also for industries. Ports, farmland and touristic industries are located near the coast. As the population is growing and "approximately 3 billion people — about half of the world's population — live within 200 kilometers of a coast- line" [8] it is needed to intensify the focus on coastal protection. The interaction between global warming and a sea level rise are underlining the need. As some countries like the Netherlands have 1/4 landmasses below the sea level the need of coastal protection is of high interest.
One option is to let the robot scan a protection site by optics to check the upsanding of the material and monitor it by comparing the latest survey with older ones. Second

option would be to check for chemicals. As coastal protection is not only meant to concrete the coast but also to check for possible slides.
Example:
A sandy, steep coast or shelf is build up from sandy material. Sand in small grain size of down to $0.0025 - 0.0049in$ ($62.5 - 125$µm) is a very fine sand (mud, silt and clay is even finer). The sand particles are packed and perfectly aligned and the inter space allows only freshwater to pass trough. Permeability of such a system allows the water particles to diffuse but as salty particles are too big, they stay out. This brings a grade of salinity in front of the sand barrier. If the sandy material experiences a movement in pressure shift within the constructed system, the salinity gradient in front would fall due to freshwater inflow from the sandy system. A stationary monitoring system could permanently check the salinity in front of the sand barrier while an AUV drives by every now and then to check the whole site for optical changes and chemical differences as well as also salinity for reference reasons.

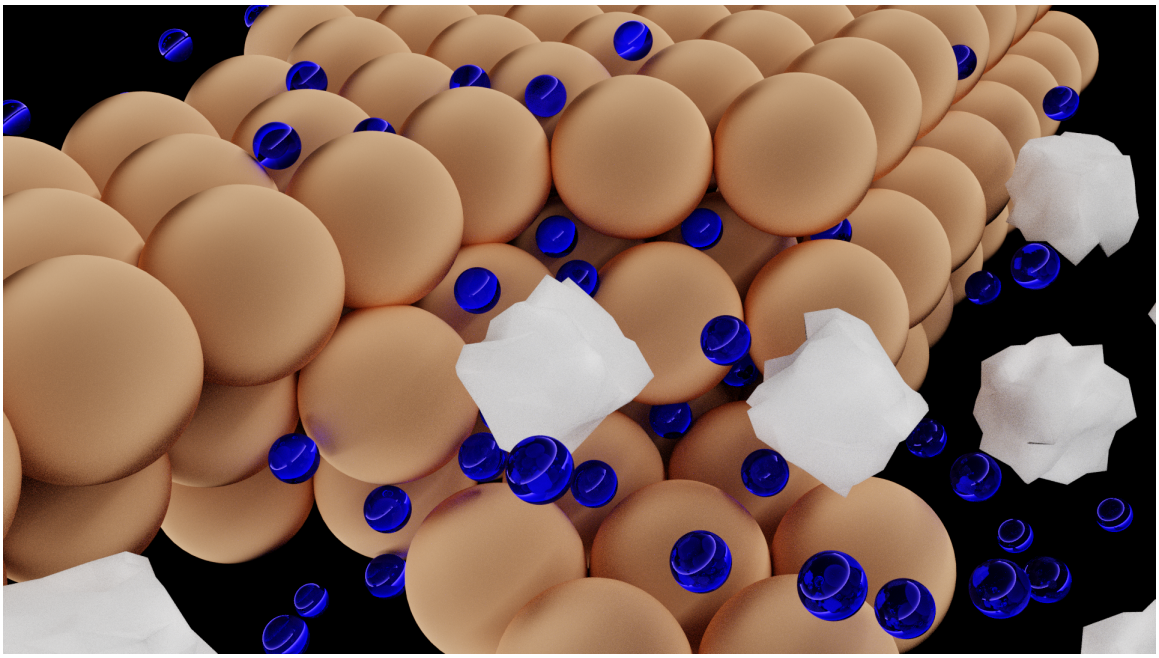

Figure 45: *Brown balls represent aligned sand particles while blue balls are water particles. White polygons are salty particles that can not diffuse through the sand barrier due to their size*

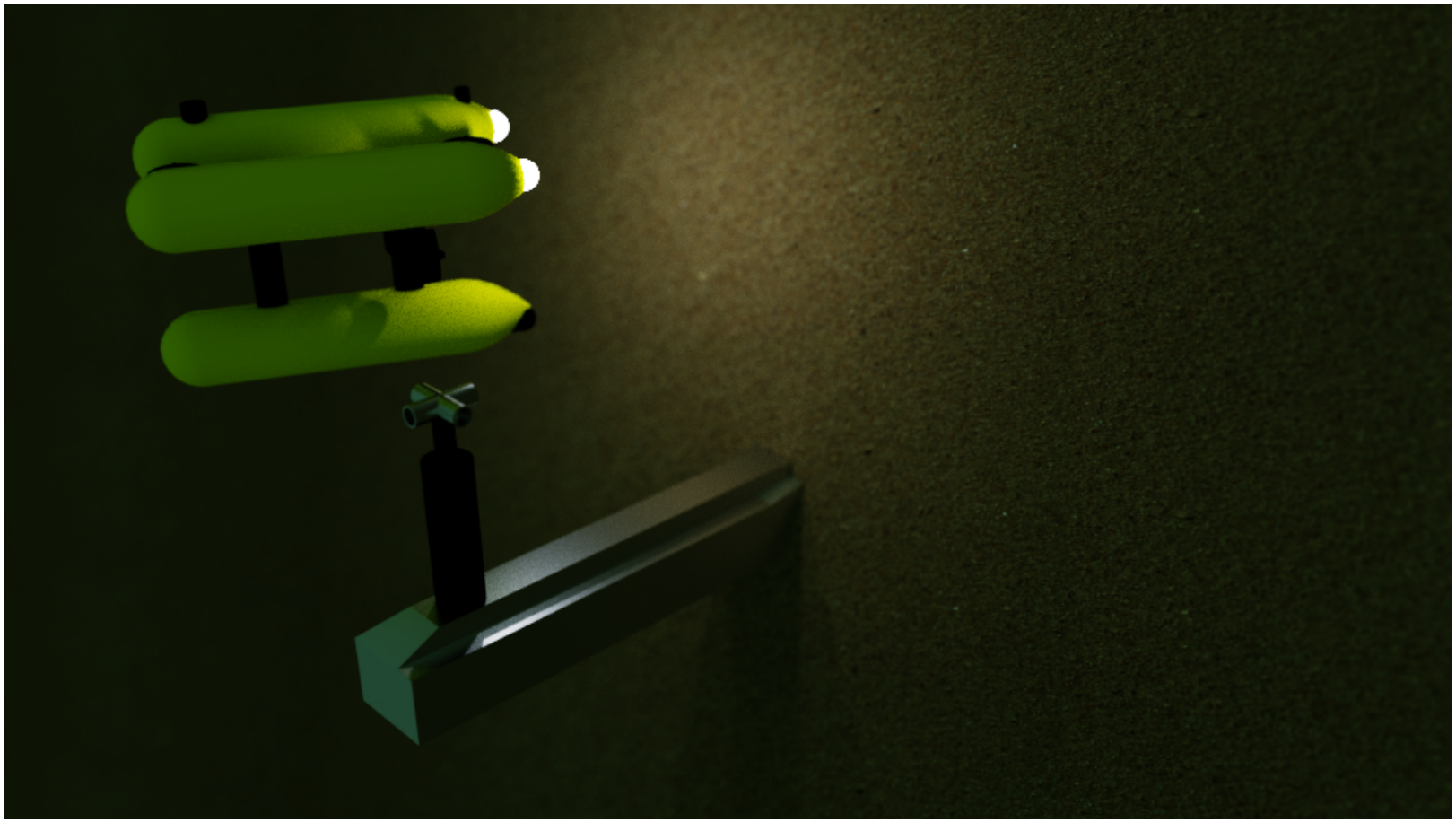

Figure 46: *A possible situation of the robot checking the sandy slope and the salinity measurement station attached to the slope*

3.6 Simulation and Data Acquisition (UWSim, OpenCV and Logging)

UWSim is an open source AUV simulator developed by the university of Girona. It started with the RAUVI and TRIDENT Project to simulate and test algorithms. It is written in C++ and uses OpenSceneGraph and its osgOcean libraries [25]. It is highly flexible and the scenes can be configured with standard third party editing software like Blender that also provides the file formats that are needed by UWSim. The complete scene can be configured in a XML file that also contains the characteristics of the objects like weight, spawning coordinates, color and if it is an obstacle that can be bumped into or not. The initial scenario represents the Girona test basin with a black box that can be interacted with. I used the scenario to set up my own. I used the given parameters and changed it to my needs and produced the meshes that represent the real world.

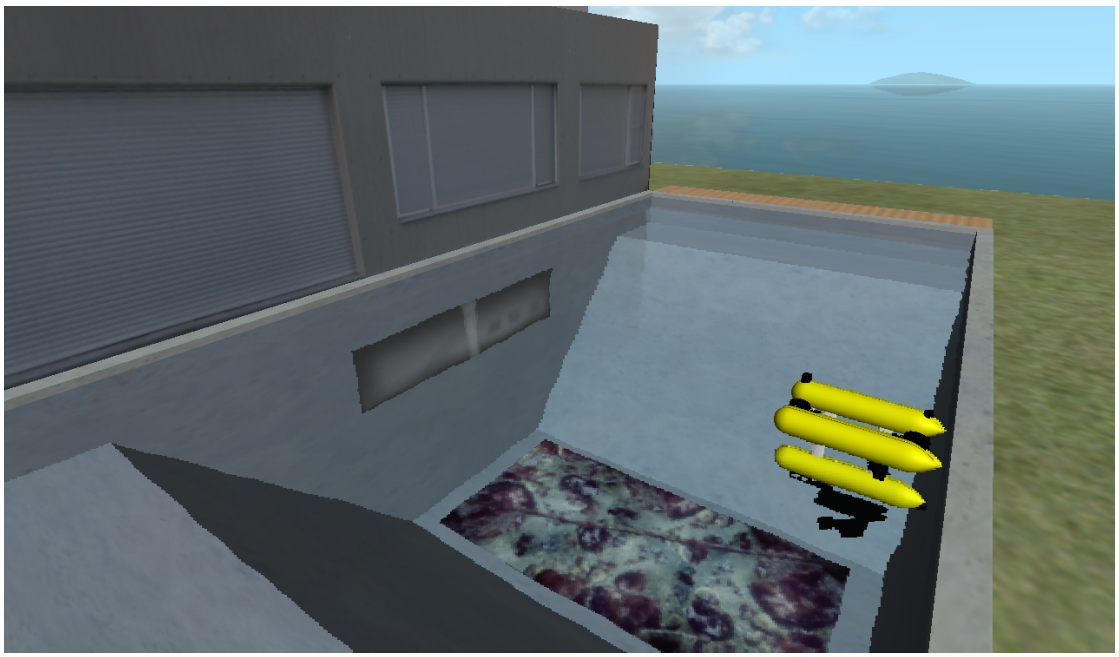

Figure 47: *Initial scenario in UWSim in the test basin of Girona University (source: [25])*

The Simulator comes with a texture that lies on the ground. In reality it is a plain mesh with a texture so I was able to do the same but with a little bit more representation of the reality since I was using a displacement map for creating ripples that are representing the seafloor (see chapter "Blender" in Methods). It is also possible for the robot to interact with the ripples like scanning them with the given laser sensor.

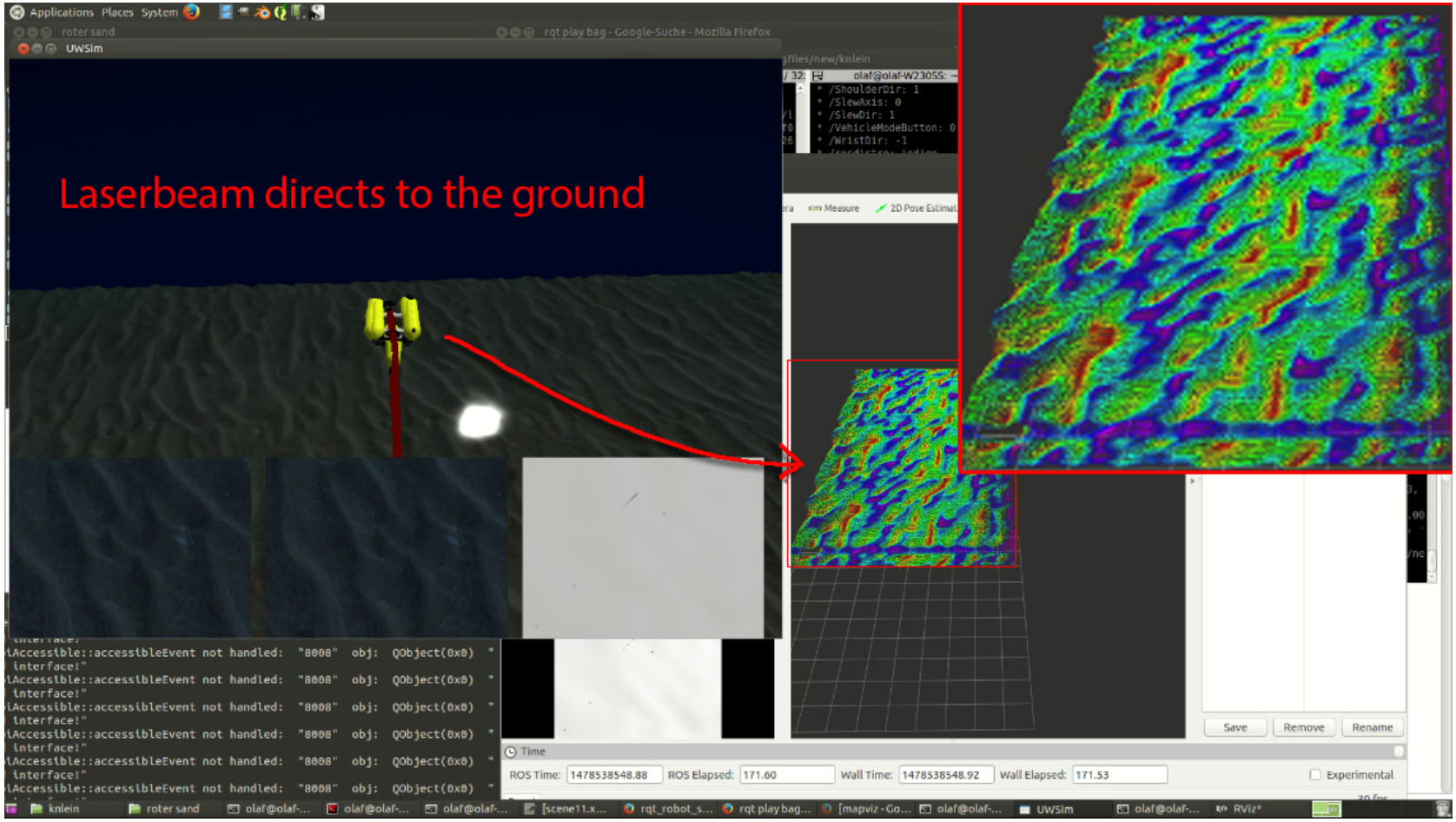

Figure 48: *The yellow robot is the sensor equipped Girona 500. It is scanning the underground with the laser beam that is visualized in RViz with a vision decay of 1000 ms. Higher areas are represent in red color while the lowest points are purple and green, yellow, orange and blue are values in between. The information is directly grabbed by the published topics of the sensors. The results are on the right with a 2x magnification.*

The Simulator provides a few sensors that are important for this work. The dataNavigator is used to give the robot automated commands to move into a certain direction. The Camera sensors are used by my algorithm that thresholds the image and measures the appearance of the seagrass and the GPS sensor provides the position of the value for my algorithm. Most of this can be seen in Figure 49:

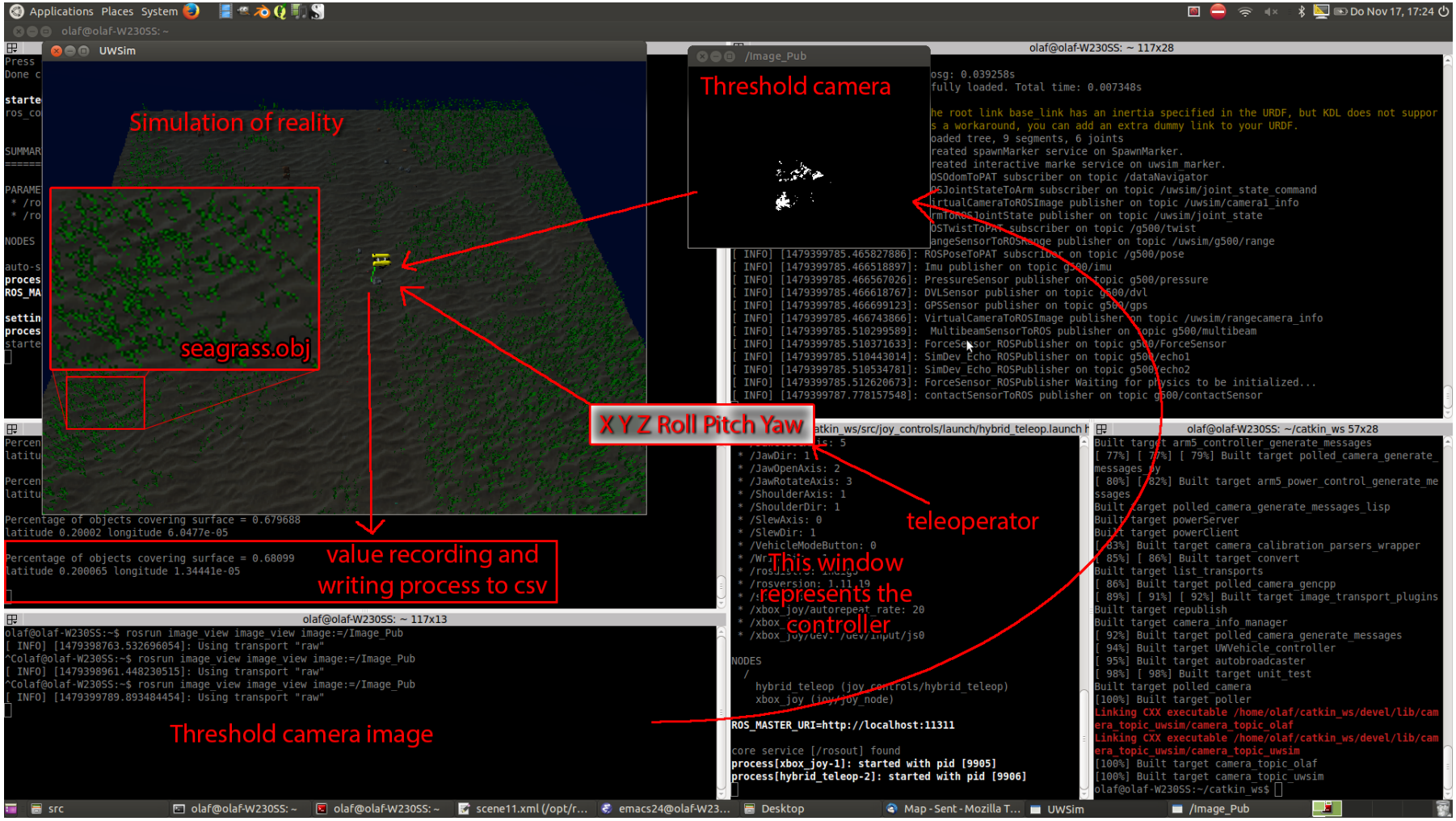

Figure 49: *The image shows the simulator while it is started and is simulating the working area with seagrass and objects. The processes that are running in the background are given within the specific windows and the arrows are indicating their influence. The window with the threshold camera is just for visualization and doesn't need to be opened, the process is running in the background. While the controller can provide a movement message, the algorithm is writing continuously the values and GPS information into a CSV file*

After setting everything like the underground and all values within the XML file, the scenarios are just slightly different. The rocks ans animals are static on the ground while the the distribution of the seagrass changes. It is always a different mesh that is representing the growth and spread. Like the description in the motivation chapter, the seagrass has fixed values of growth which are once undisturbed and influenced by different kinds of threats. Each scenario comes with four different stages (sequences) and each stage is scanned two times (W, E,) which makes in total 32 scans. While the robot is running through the scenario autonomously, this makes the most convenient part of the work. This should represent the reality because the robot would do the same while scientists would just wait for the data.
The robot in the simulation comes with twelve different sensors and a manipulator that all can be moved around 6 degrees of freedom:

X	Y	Z	Roll	Pitch	Yaw

Table 8: *6 degrees of freedom of the robot that is used in the simulator*

Sensor	Description
Camera	Provides virtual images that can be used for developing vision algorithms
Range Camera	Depth image of the camera
Range Sensor	Measures the distance to the nearest obstacle along pre-defined directions
Object Picker	Fakes object grasping when the object is closer to a pre-defined distance
Pressure	Provides a pressure measure
DVL	Estimates the linear speed at which the vehicle is traveling.
Imu	Estimates the vehicle orientation with respect to the world frame
GPS	Provides the vehicle position with respect to the world, only works when the vehicle is near the surface
Multibeam	Simulates an array of range sensors, providing distances to nearest obstacles in a plane at constant angle increments
Force	Estimates the force and torque applied to a vehicle part
Structured light projector	Projects a laser or regular light on the scene
Dredge	Dredges mud from buried objects

Table 9: *Twelve different sensors that come with the simulator. The Sensors are fixed with the robot but can rotate with the robot itself. Source of sensor documentation: UWSim Wiki - Main characteristics of UWSim [25]*

For the use of this project the simulator brings several sensors that are used for optical reasons and for position reconstruction. The simulator gives the possibility to interact with the simulated world and interact with objects. The prepared meshes from Blender and QGIS are loaded into UWSim and represent the rebuilding of the reality. Information from the robot are given as topics that one can subscribe to or interact with these values like publishing them into a comma separated value (CSV) file. As part of ROS (s. chapter "Hosting the Simulator") the libraries of OpenCV are

used to get an idea of the appearance of objects on the simulated seafloor. The whole environment is stored in the given XML file that provides the paths and values for the mesh files. In this case its all objects like crabs, rocks, starfish and the seagrass meadows. Also some of the values of the sensors and the robot are stored in this file. As the meshes are only the seafloor and objects, the water surface is provided by the simulator itself. Some of the values contain the position of the sun, water depth and dirtiness, wind speed and direction, physics parameters, initial position of the robot and the position of the sensors with respect to the robots position.

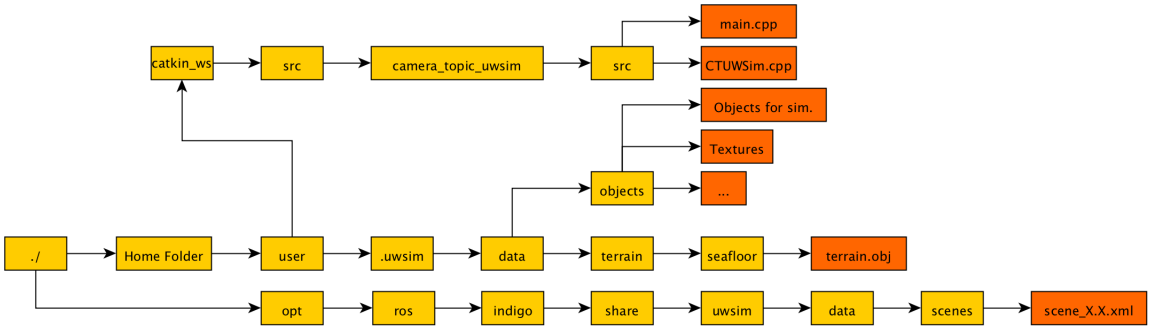

Figure 50: *The structure how the simulator is organized on a Linux machine with my method of scanning the seafloor (CamerTopicUWSim -> Camera_ topic_ detection)*

The terrain (seafloor) is stored as a OBJ File in a specific location so that the simulator can find it. To make the work easier, I chose to work with the seafloor mesh as a "Trimesh" that doesn't collide with objects because I was first facing problems with objects that are just disappearing into the the void (empty space) because of collision. Since the robot is only using visual response it is not necessary to have a collision model for robot and the environment. It also saves resources that otherwise would be used. As I am not using all sensors, I am also only recording only the sensor messages that are responsible for the robot movement and detection.

3.6.1 Hosting the Simulator (ROS)

The Robot Operating System (ROS) is an open source operating system that is used in almost all robots. It is not an operating system that is managing processes, it provides communication layers that allow processes to communicate with each other [26]. It is highly reliable and flexible since it allows the user to publish and subscribe to self made topics. The used simulator UWSim runs completely in ROS and provides published data that can be subscribed to and the gained data is written automatically into a file which is ready to read by QGIS.

The system is working with a „talker“ and „listeners“. It can be referred to like in a conversation where one has to imagine someone who is talking and others that are listening like in a lecture with a teacher and students.

The Simulator gets information from the camera node (provided data from the optical sensor) which contains the image information as a ROS message which is present as a combination of numbers (ROS shares information in form of messages). The code interprets the numbers to serve an image that OpenCV (an image manipulation framework) can work with so the given information will be separated into different tasks.

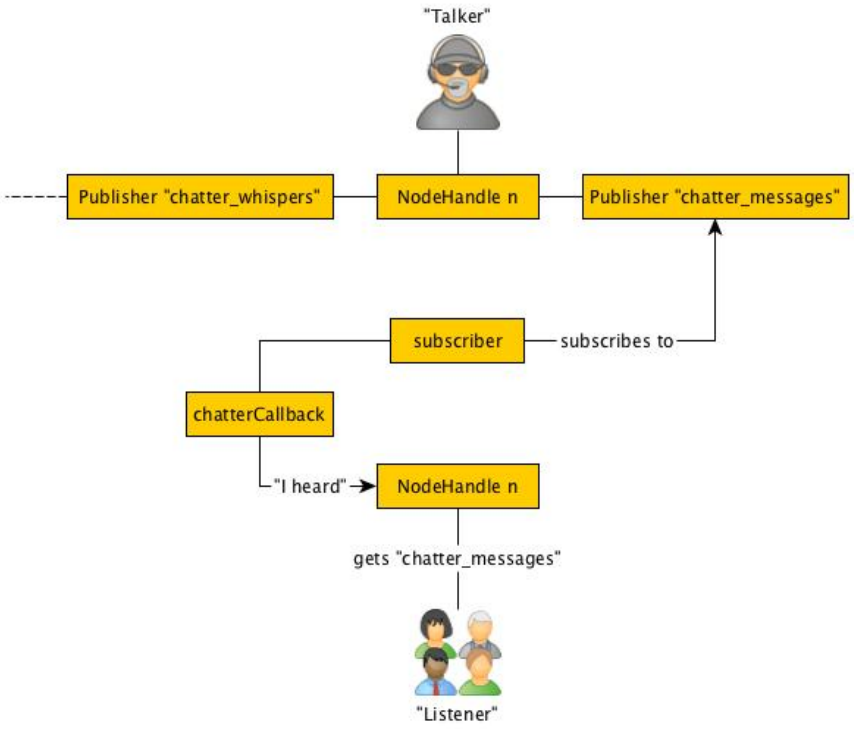

Figure 51: *The "Talker" is providing information while the subscriber is listening to it. The chatter Callback is taking the information from the subscriber and the "Listener" gets the information that was heard by the chatterCallback*

The publisher is "talking" the content of the camera image. It needs to be converted because the image processing application OpenCV does not directly work with messages from ROS but with images. Therefore the message is converted into an OpenCV readable format (BGR8) before performing any manipulation.
CV_Bridge (a plugin for ROS) helps to form the message into an image while one is subscribing to the topic. In this work this is done with the cv_bridge that makes it possible to work with images to perform thresholding and create a binary image. The gained data is published into an external file by a publisher that can be found in the code snippets.

Algorithm 1 *The code represents the internal structure of receiving, thresholding and publishing the data of the green channel. This is performed with all colors that appear in the robots camera.*

```
1  ofstream myfile_red("green_plants.csv", std::ios_base::app); //gained data will be
       written into a csv file
2
3  cv_bridge::CvImagePtr cv_ptr_green = nullptr;
4  cv_bridge::CvImagePtr cv_ptr_foto_green = nullptr;
5
6  if(sensor_msgs::image_encodings::isColor(imgData->encoding))
7    {
8     cv_ptr_green = cv_bridge::toCvCopy(imgData, sensor_msgs::image_encodings::BGR8);
9     cv_ptr_foto_green = cv_bridge::toCvCopy(imgData, sensor_msgs::image_encodings::
          BGR8);
10   }
11
12   if(cv_ptr_red)
13   {
14    cv::Mat imgMat_red = cv_ptr_green->image;
15    cv::Mat red_foto = cv:prt_foto_green->image;
16    cv::Mat hsv_image_green;
17    cv::Mat lower_green_hue_range;
18    cv::Mat upper_green_hue_range;
19    cv::Mat green_hue_image;
20
21    ros::NodeHandle n_green;
22    ros::NodeHandle n_green_string;
23
24    cv::cvtColor(imgMat_red, hsv_image_green, COLOR_BGR2HSV);
25     cv::inRange(hsv_image_green, cv::Scalar(42.5, 50, 50), cv::Scalar(67.5, 255,
           255), lower_green_hue_range);
26     cv::inRange(hsv_image_green, cv::Scalar(42.5, 10, 35), cv::Scalar(67.5, 255,
           255), upper_green_hue_range);
27     cv::addWeighted(lower_green_hue_range, 1.0, upper_green_hue_range, 1.0, 0.0,
           green_hue_image);
28
29   cv::cvtColor(green_hue_image, cv_ptr_green->image, CV_GRAY2BGR);
30
31   ros::Publisher imgPub_perc_green = n_green.advertise<sensor_msgs::Image>("
       Image_Perception_green", 100);
32   ros::Publisher imgPub_perc_green_string = n_green.advertise<sensor_msgs::Image>("
       Image_Perception_green_string", 100);
33
34   imgPub_perc_green.publish(*(cv_ptr_green->toImageMsg()));
35   }
36
37   ros::rate Loop_rate(0.3);
```

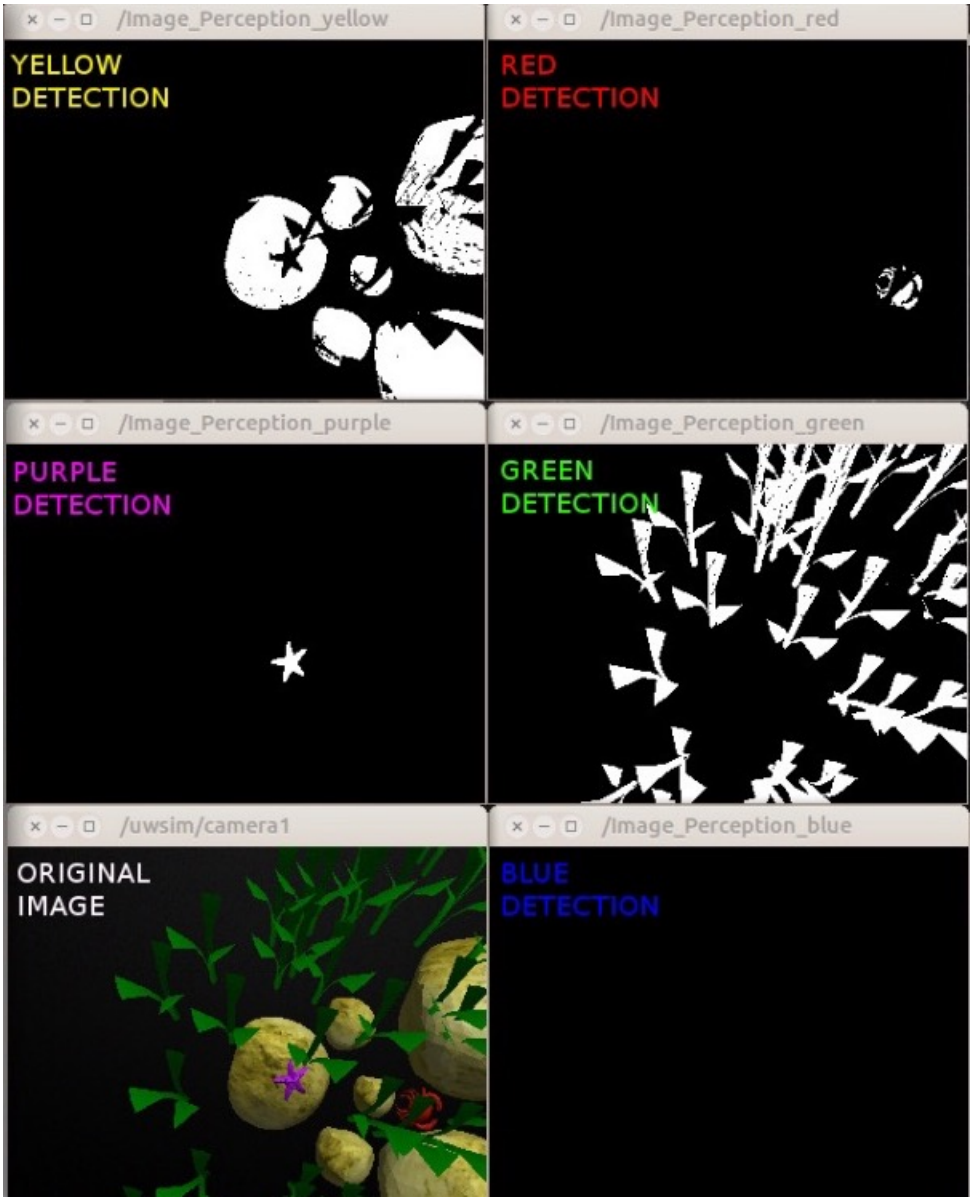

Figure 52: *Results of the code: The original image gets separated into color channels so the occurring white pixels can be interpreted as percentage of the whole image. This separates the objects and results in an occurrence density*

ROS is hosting the simulator and everything it is publishing. Other topics can subscribe to the output of the system. This means that the published data from the simulator can be used in other topics that are provided by ROS. In this work I am using it not only to threshold the received camera image but count the white pixels in the resulting binary output. This is used to calculate the percentage of appearance of different colors so the density of color specific objects can be calculated. Also the GPS locations from the positioning node within the simulator are written beside the pixel count within the output files.

Structure of CSV file:

Percentage [%]	Latitude	Longitude	File name with time stamp
1.45182	74.6445	2.03604	PLANTS_23-02-2017 03:27:23.png
6.23438	39.556	-2.03632	PLANTS_23-02-2017 03:27:39.png
42.2188	33.5495	-2.03629	PLANTS_23-02-2017 03:27:43.png
12.8945	27.725	-2.03602	PLANTS_23-02-2017 03:27:46.png
...	...	...	...

Table 10: *The table indicates and example of the structure how the data is recorded by the robot. The coordinates result from the coordinate reference with respect to the world of UWSim*

QGIS is capable to read the file as a comma separated file that can be interpreted by its Latitude, Longitude and an attribute which is the value of seagrass coverage in percent in this case. The value is presented as a dot and is only a value on a specific coordinate. The dots are represented as a „heat“ source for a heat-map. The higher the value, the higher the heat. This heat-map represents the spread of the seagrass (green objects) density and is interpreted for later calculations. Crabs, starfish, yellow and blue rocks are presented with symbols. A map is showing the results of what the robot saw during its mission. Plants occur as a symbol, its spread is indicated as lighter green color, rocks are represented as diamond squares in yellow and blue, crabs are red pentagon shapes and starfish are represented as purple stars. The higher the percentage value, the larger the icon is in size on the map. No value shows no icon and a small value just a small icon. The steps are fix to five different sizes so the occurrence can easily be interpreted on the maps.

3.6.2 Controlling the Robot Manually

The control of the robot is happening autonomously in every scene so the experiments are running under the same conditions every time. It is necessary to run the robot manually in the simulator from time to time to check the outcome of the code of OpenCV. To do so, a game pad by Logitech is used to control the robot. While the movement is actually done in the background, by a topic called "dataNavigator" which sets a velocity value to the robots movement in different directions, the controller is set between the control code of the simulator and the message. The dataNavigator receives the controller messages and sends the specific value of the Joystick to the simulator which will then interpret it to move the robot. The code, that is normally providing information to the robot movement is called a "teleoperator" but as the controller incurs this part, it is called a hybrid teleoperator (from the ROS package "joy_control").
For reasons of checking what the different topics said, the whole path and other messages can be stored in a so called "bagfile". This records the output from the controller, or better said, the incoming dataNavigator message as well as other messages that one can define in the bag file. This can also be played back after the recording finished or evaluated in external software.
The robot comes with thrusters (for realistic movement in a physics engine) which can be controlled ald its also possible to mor the robot by repositioning it in the simulators world. Those thrusters are needed when it comes to realistic physics. As the controlling of the thrusters is not necessary in this simulation due to the size of the area, I am just using the repositioning function. In terms of physics it would not make any sense to apply extra calculations onto the simulator because the main focus is set onto the detection instead of realistic physics. Nevertheless the robot is controlled either by a code that provides an automated behavior or by a Human interface device (HID) like a game pad. For the Simulation I am using a Logitech F310 which is showing a typical button configuration of a controller used for the Sony PlayStation. To make ROS recognize the controller, I am using a hybrid teleoperator that is configured to subscribe to the USB joystick interface and publish the signal to the dataNavigator of UWSim. It normally works for controllers that have a configuration file of an Xbox controller but works as well with the Logitech controller, since Xbox and PlayStation controllers are supported by the system.

Figure 53: *Logitech F310 USB Game pad for robot controlling in the simulator*

As every command in ROS, the velocity changing command can also be run by the „rosrun“ command. The velocity of the robot is manipulated by a value on a given Axis. It is structured by a command to the specific topic with a value in an array: [topic] [x] [y] [z] [r] [p] [y] and the units are m/s and rad/s, depending on the axis. To make the robot move into X direction, it is necessary to provide the command like this:

```
1 rosrun uwsim setVehicleVelocity /dataNavigator 0.2 0 0 0 0 0
```

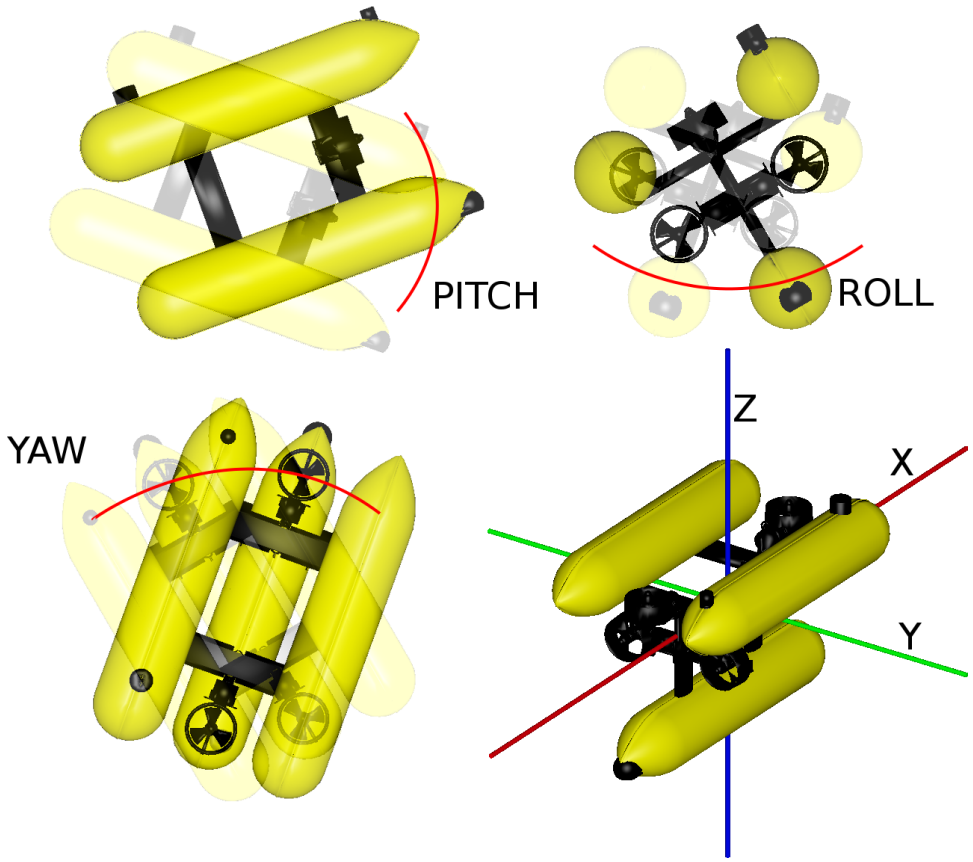

Figure 54: *[r]oll, [p]itch, [y]aw. The three rotation axis of an object in a 3D environment and [x] [y] [z] for left, right, up, down, forward and backwards*

The command makes the robot move in X-Axis (forward) direction on a constant velocity of 0.2m/s. For rotation, the robot needs a value on one of the [r] [p] [y] axis. After doing so, the robot will rotate. To move the robot more into depth it's the same command but just setting the [z] value. Instead of entering a negative value (due to downward movement) the value is positive, because of a velocity that pushes the robot into depth. Instead of velocity, the robot can also be placed into direct coordinates of the Cartesian world space. By using the command setVehiclePosition it is possible to not only move the robot but place it everywhere in the environment. by using the following command:

```
1 rosrun uwsim setVehiclePosition /dataNavigator 0.2 0.2 0 0 0 3.14
```

the robot will be set to the position X = 0.2, Y = 0.2, Z = 0 and will be rotated by 180°. While the program of setVehiclePosition is running, it is not possible to move the robot by setVehicleVelocity. There are different possibilities to move the robot and interact with it. The commands that can be used are for velocity, position, twist and pose. But to interact with the robot, it would be hard to just type in commands

and change them when its time to turn the robot or dive and come up again [25]. For this reason it is handy to use a program in the background, that is publishing a message constantly, which is changing the dataNavigator array. The way it is done in this work, is with the mentioned hybrid teleoperator in manual control and by code in autonomous control. It is started by the command of:

```
1 roslaunch joy_controls hybrid_teleoperator.launch
```

This is launching the code for using the joystick input from USB and uses the signals to create values for the dataNavigator. As the robot provides more systems to manipulate than the controller can provide buttons, there is a key button to change between the robot and the manipulator arm. While performing a mission, the data of the movement is stored into a .bag file. This Bagfile will provide the signal that is recorded from the teleoperator and can be replayed directly to the robot or the simulator. The Bagfile will normally be recorded by typing:

```
1 rosbag record -a
```

The only problem that comes with this command is that this would record everything that comes through the ros pipeline and would produce a humongous file size and also record the camera topic as well and the data output would always be the same. This why why I record only the topics that are necessary for controlling the robot.

```
1 rosbag record /dataNavogator /force /gps
```

This limitation writes the data that will always stay the same in all scenarios so the robot is expected to perform the same movement as recorded. To playback the pre-recorded movement, it is necessary to type the command

```
1 rosbag play mybagfile.bag
```

to be able to do that, one is recommended to start the command from the same directory as the .bag file is located in.

Every scenario is divided into two parts (West and East). It easier to re-record a mission if one part needs to be scanned again and to minimize failure.

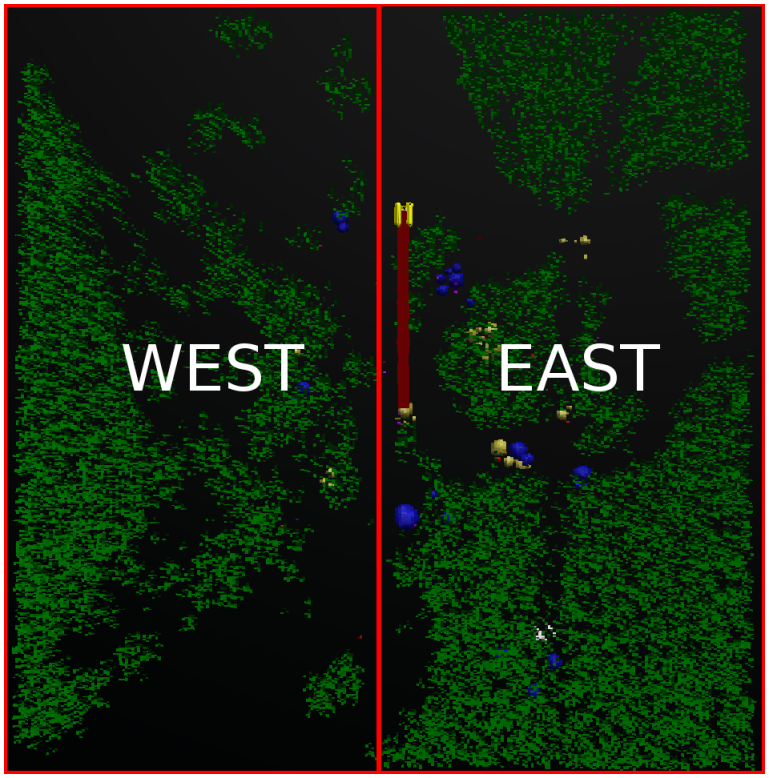

Figure 55: *Structure of the scanning of the working area which is in some parts only different in the eastern side. This makes it much easier to scan only one part if the western side is the same as in another scene*

The center point of the area is the initial point of the robot. While the task of the recording is running, ROS is writing the collected values in the .csv file in the background and it is ready to be read by QGIS. This can also be done live as the robot is performing the mission and by hitting f5 (refresh) within QGIS, the recorded value of the robots are presented to the user as they are recorded in that moment. This can be done either during the manual as well as during the autonomous control.

rosrun, roslaunch and rosbag

„rosrun“ allows the user to run executable code directly from the current directory. It is not necessary to go into a specific directory to execute the program. Every time one has changes something in the working directory, he has to „rosmake“ (recompile) his work space again. While doing so, ROS locates all executables to link the commands to the files. roslaunch therefore is used to read .launch files in XML format to do certain actions. The tool is normally used to launch the nodes locally and remotely via SSH and can be set as a parameter on a server [44]. In this work, it is used to

launch the autocontroller (autonomous movement of the robot) and the scenes at the same time. The rosbag command is used as a tool to record various operations in ROS. The rosbag files include recording of topics, playback of the topics and validating them. Therefore it is a handy tool for testing reasons. In this work, the rosrun command is used to start the detection as well as the hierarchical recorder Semrec. The roslaunch command is used when it comes to start a mission.

3.6.3 Robot Control Code (autonomous movement)

Controlling the robot with the controller brings possibilities to adapt a "learning by demo" algorithm but also contains the risk of failure for a system like I experienced in weeks of the working process. While the recording of the track and robot movement is continuously given, the playback is triggered by many influences so the controlling with a controller is good for testing purpose but not for simulation. As this system is a non-deterministic one, the outcome of ROS record is never the same as the input. The more easy way is to use an extra code to give the robot commands to rotate at a specific position during the mission execution. Here, a launch file is used that provides the movement of the robot by letting the dataNavigator topic move the robot forward by 0.5 m/s. As the robot reaches different coordinates in the world frame, it changes direction as it comes to that particular position. The world frame has a 50x50 meter dimension and the robot changes direction as it reaches 25 in positive or -25 in negative values on the Y-Axis. The robot goes 0.5 meters to the left or right, depending on the area, that needs to be scanned (western or eastern), and continues till it reaches the most left (or right) extend.
The code of the "autocontrol" package by Srinivasan [31] provides a launch file as well as a C++ file, that needs to be modified by

- setting the borders of the area
- removing other functions that have nothing to do with the movements
- setting the path to the scenes that need to be scanned

By setting the path, the new file needs to be saved as a new launch file so its easier to start a specific simulation:

Algorithm 2 *Code snippet of the launch file to start the simulation of scene 2.4. The rosparam sets the velocity of the robot forward at 0.5$^m/_s$*

```
<node
  pkg="uwsim"
  type="uwsim"
  name="autocontrol"
  respawn="false"
  args="--configfile scene24.xml" //scene24.xml equals "Scene 2.4"
  output="sceen"
/>
<rosparam
  param="Velocity"> [0.5,0.0,0.0,0.0,0.0,0.0]
</rosparam>
```

To change the scene, the XML file needs to be specified and to change the speed [m/s] the velocity parameters need to be changed. The value describes the movement in X direction and the others can be checked in Figure 54.
The controlling code is in the "tf_vehicle_control" where the code specifies the direction the robot has to move. The odometry is set to a specific rotation of the robot and is following with respect to the world and not with respect to the robot itself. To provide the command to turn western, it needs to rotate 90 degrees but if it had to move in southern direction it needs to rotate 180 degree instead or another 90 degree.

Direction	Degree
North	0
South	180
West	90
East	-90

Table 11: *The table describes the directions that need to be used within the code to let the robot go in a specific direction*

If the robot reaches the most upper boundary (+25) of the area it needs to rotate western to continue its scanning:

Algorithm 3 *This fragment of the code shows the command to move the robot. It rotates by 90 degrees if the coordinate of the "x" value is reaching 25 or higher*

```
if(first_time)
 {
  if(x_value >= 25)
   {
    odom = rotate_vehicle(0,0,90,odom,x_value,y_value,z_value,"UWVehicle_world");
    second_time = true; first_time = false;
   }
  else {
      odom = apply_movement(Vehicle_velocity,odom);
      movement_steps+=1;req_z=-10;moved_z=0;movement_steps=0;
      arm_retracted=false;
     }
 }
```

Each change of movement is set in the code so for each side, around 100 steps are needed due to change in direction in northern and southern part. This results in 200 steps for one whole scene which is present in the appendix drive as well as on

Github. The robot does not stop until it reaches the final step in the code and will then stay in position. One task for the future would be to let it drive back to the initial point and dive to a base station to recharge and send the gained data to a server.

3.6.4 Object Recognition and Detection (OpenCV)

OpenComputerVision (OpenCV) is an open source project to interact with images and videos in a way to separate it in different information and manipulate it. OpenCV can be used to interact with objects like identification or counting. In this project the OpenCV is used to threshold the incoming image from the camera of the robot and count the white pixels within the binary image.

short overview

The code is taking the Image as a sensor message (sensor_msgs) from the camera of the G500 through an imageCallback fuction and produces a Pointer of the type imgData. This is the point where the message will later become a format that OpenCV is able to handle. OpenCV can be called in C++ code with specific commands which are present in this code by cv::, followed by a type. cv::Mat is using the mathematical functions of OpenCV to handle images in this case. But before the mathematical threshold can be applied, the image needs to be set gray scale, otherwise the threshold wouldn't produce a binary image. As the cv_ptr is pointing on the image, represented by the variable imgMat of an OpenCV type, it is possible to handle the image in a mathematical way and return the result into a second variable called 'filtered_image'. This procedure is done by ctvColor and can be specified with CV_BGR2GRAY. There are more options that can apply filters to an image but at this point this isn't necessary

Detailed

For the system it is relatively easy to handle images but more complicated to handle ROS messages. As ROS is communicating with messages, it needs to be encoded into an image format that OpenCV is able to understand. Therefore the code needs to be like the following:

Algorithm 4 *defining a pointer with the image content*

```
1 cv::Mat imgMat = cv_ptr->image;
```

The cv::Mat function makes it possible to establish a communication between OpenCV and ROS. imgMat in this case is declared as a pointer that is pointing at the received image from ROS. This is possible by using the cv_bridge and initiate the pointer as a null pointer so it can be written with data. The lines:

Algorithm 5 *Starting to manage the image content if it comes as a color image*

```
1 if(sensor_msgs::image_encodings::isColor(imgData->encoding))
2   {
3   cv_ptr_green = cv_bridge::toCvCopy(imgData, sensor_msgs::image_encodings::BGR8)
        ;
4   }
```

are checking the incoming message if it has color and if it comes as a picture, it will be encoded so OpenCV understands how to handle the data. The cv_bridge uses the message and encodes it into a BGR8 image that OpenCV is capable to handle and manipulate. Actually the system works with the color range of HSV (Hue, Saturation, Value) and the BGR8 image from the camera source needs to be converted.

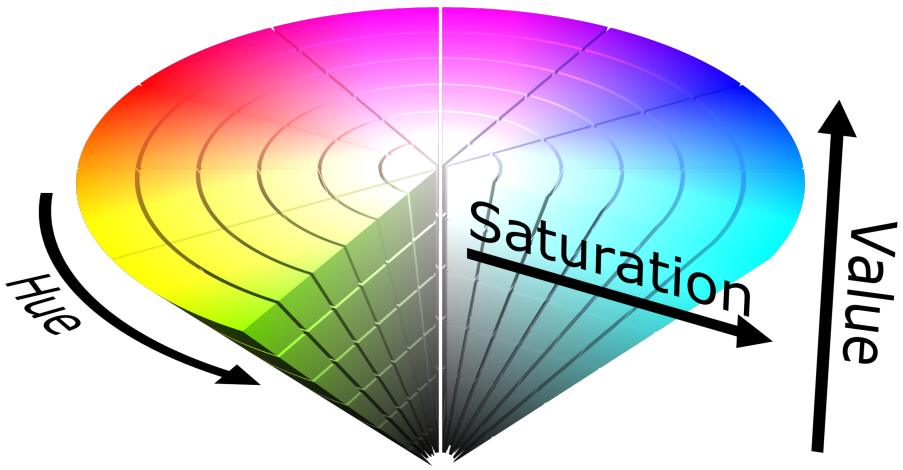

Figure 56: ***H****ue,* ***S****aturation,* ***V****alue. Instead of the wide known RGB, OpenCV uses the HSV color range*

Algorithm 6 *The image content is converted from BGR to HSV color range without losing information*

```
1 cv::Mat imgMat = cv_ptr_green->image;
2 cv::Mat hsv_image_green;
3 cv::cvtColor(imgMat_green,hsv_image_green,COLOR_BGR2HSV);
```

This gives the user the ability to set limits to the threshold. The next step is to define variables that contain the upper and lower limits of the threshold and perform the actual threshold:

Algorithm 7 *Defining the HSV range for colors that point to the objects*

```
1 cv::Mat lower_green_hue_range;
2 cv::Mat upper_green_hue_range;
3 cv::inRange(hsv_image_green, cv::Scalar(42.5, 50, 50), cv::Scalar(67.5, 255, 255)
      , lower_green_hue_range);
4 cv::inRange(hsv_image_green, cv::Scalar(42.5, 10, 35), cv::Scalar(67.5, 255, 255)
      , upper_green_hue_range);
```

This step gives the first possibility to manipulate the detection depth as the color range is given in table 12. inRange only delivers values from the image, that are within the set scalar of values occurring in the code snipped. The next step provides a deeper, secondary limitation as the user can define a weighting within the range of the upper and lower limit:

Algorithm 8 *Defining the weight for the hue values that trigger the detection*

```
1 cv::Mat green_hue_image;
2 cv::addWeighted(lower_green_hue_range, 1.0, upper_green_hue_range, 1.0, 0.0,
      green_hue_image);
```

As the binary image is not a color value, ROS has problems to handle it as message data. To make it more handy for the system, the threshold image is set to a color image again:

Algorithm 9 *ROS needs to use BGR image information to use it as a message*

```
1 cv::cvtColor(green_hue_image, cv_ptr_green->image, CV_GRAY2BGR);
```

The image is now a BGR color image (it actually comes only with black and white information) and needs to be converted into a ROS message again so the system can handle it again for other ROS services:

Algorithm 10 *The green percentage value is published as a message*

```
1 imgPub_perc_green.publish(*(cv_ptr_green->toImageMsg()));
```

This provides a message for the ROS environment and as the vision receives a certain value, it publishes it findings on a topic. As this will publish everything it will find due to green color, it would write the value into a file even if it finds just one pixel.

Due to image noise in real world conditions, this would produce a lot of misleading findings and the systems needs to react only at a specific limitation.

Algorithm 11 *Calculations of the percentage that the image is filled with a color (here: green)*

```
cv::Mat partROI_green;                //defining green color percentage
  float count_white_green = 0;      //set white count to zero
  float count_black_green = 0;     //set black count to zero
  count_white_green = countNonZero(green_hue_image);          //count white
      pixels
  count_black_green = green_hue_image.cols * green_hue_image.rows -
      count_white_green; //count all pixels - white pixels
  float perc_green =count_white_green/768;  //percentage calculation
```

This provides a calculation to get a percentage value but as the system needs the limit, another function can sort out the not needed values:

Algorithm 12 *The algorithm starts if the percentage is over 0.28%. This can be changed for noise reduction in camera sensors. The part in the clause defines how the values are written into the CSV file*

```
if (perc_green >= 0.28)
 {       cout << "I SEE GREEN PLANTS" << endl;
  [...]
myfile_green <<"Percentage, "<< perc_green << ", " << "latitude, " <<
    m_gpsPosition.latitude * 3.5 << ", longitude, " << m_gpsPosition.longitude *
    3.2 <<", " << filename_green << endl;
imgPub_perc_green_string.publish(*(cv_ptr_green->toImageMsg()));
 }
```

The function only writes the found values into the CSV if it is higher than 0.28%. Since the simulator has very low noise, I chose to set a very low limit. Depending on the used sensor, and different noise behavior, this value can be changed.
As binary pictures represent only two states with black (0) and white (1), the calculations are relatively easy by using the Indicator Function:

$$1_A : X \to \{0, 1\}$$

(Indicator Function)

defined as:

$$1_A(x) := \begin{cases} 1 \; if \; x \in A, \\ 0 \; if \; x \notin A. \end{cases}$$

```
white Pixels = countNonZero (imgMat)
black Pixels = (imgMat.cols x imgMat.rows) - white Pixels
```

Figure 57: *Theoretical construction of the calculations in the code, based on the indicator function*

Due to the resolution of the camera, the white pixels need to be divided by 768 (cf. Alg. 11)to get the percentage of the occurring color that is identified throughout the threshold.
The pixel count is published into a .CSV file together with the GPS position on the GPS node from ROS:

Algorithm 13 *„myfile“ creates and opens a new raw text file that receives a the counted percentage and GPS location (x,y) and divides them by a comma*

```
1  myfile_green <<"Percentage , "<< perc_green << ", " << "latitude , " <<
       m_gpsPosition.latitude * 3.5 << ", longitude , " << m_gpsPosition.longitude *
       3.2 <<", " << filename_green << endl;
```

Output:

```
Percentage, 8.66016, latitude, 29.656, longitude, -4.74584e-06, PLANTS_06-02-2017 10:35:39.png
Percentage, 8.05729, latitude, 29.6562, longitude, 1.88709e-05, PLANTS_06-02-2017 10:35:42.png
Percentage, 0.484375, latitude, 41.3502, longitude, 0.000226912, PLANTS_06-02-2017 10:35:46.png
Percentage, 1.92318, latitude, 41.9694, longitude, 1.60328, PLANTS_06-02-2017 10:36:39.png
Percentage, 11.9036, latitude, 36.1636, longitude, 1.60334, PLANTS_06-02-2017 10:36:42.png
Percentage, 10.0755, latitude, 30.1754, longitude, 1.60357, PLANTS_06-02-2017 10:36:46.png
Percentage, 8.14323, latitude, 24.3437, longitude, 1.60335, PLANTS_06-02-2017 10:36:49.png
Percentage, 7.60286, latitude, 24.3437, longitude, 1.60335, PLANTS_06-02-2017 10:36:49.png
Percentage, 5.86198, latitude, 18.4775, longitude, 1.60368, PLANTS_06-02-2017 10:36:53.png
```

Figure 58: *%, lat, lon, File name of photograph*

To discriminate each color and object, I've set the values in my code by the following schema (based on HSV color range):

Color	Hue min.	Hue max.
Red	0	10
Orange	12.5	17.5
Yellow	22.5	32.5
Green	42.5	67.5
Cyan	77.5	92.5
Blue	100	130
Purple	135	165
Red	170	179
White is based on Saturation and Value		

Table 12: *Table explaining the color range for the threshold based on [H]ue 0-180, [S]aturation 0-255, [V]alue 0-255*

Without defining the color range, the function defines the threshold itself. This leads to problems due to non specified object range. The example of the text above refers to the green color range of plant detection.

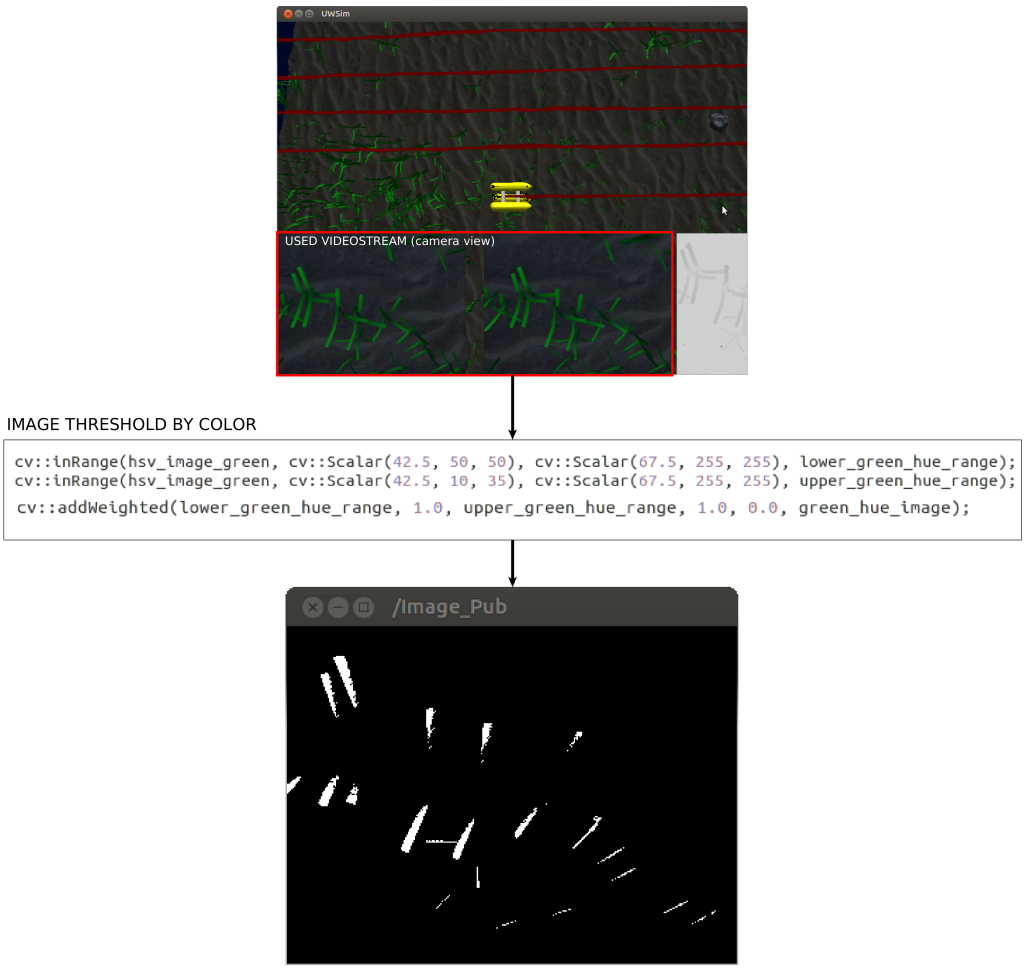

Figure 59: *The Robot's camera takes up the video stream and performs a threshold on green color which is presented as a binary image. The occurring white pixels are counted against black pixels and are used as percentage of plant appearance*

As the code runs, and publishes the found values on a topic as well as in a CSV file, it also publishes a message on a second topic that is defined the same way as the perception, just followed by a "_string" suffix.

Algorithm 14 *Beside the color detection, a trigger for later use is set*

```
1 ros::Publisher imgPub_perc_green_string = n_green.advertise<sensor_msgs::Image>("
      Image_Perception_green_string", 100);
```

This triggers SEMREC to write knowledge as the system detects the object.

3.7 Using Extracted Data (Robotics)

QGIS is able to read comma separated value files (.CSV format) and uses them to produce a point layer. The robotic system is producing such files which are used to represent the detection of the environments objects and lead to a map that is representing the detected objects. The file includes: 1^{st} the percentage of the occurring color/object, 2^{nd} Latitude or Y-Coordinate. 3^{rd} Longitude or X-Coordinate and the filename of the photograph from the camera with timestamp. To present the points correctly into the map, a new coordinate reference system is developed to produce the right coordinates. UWSim is using the map frame as a reference system to produce the coordinates so they must be calculated at two points:

- Inside the detection and publishing code by multiplicating the longitude by 3.2 and the latitude by 3.5. *(This is done to show a hand wheel on the robotics side)*
- Inside QGIS: The projects PseudoMercartor system is modified to fit the points onto the satellite images: Robot reference system for Australia (Green Island, Queensland) +proj=merc +a=6378145 +b=6378120 +lat_ts=0.0 +lon_0=145.96736 +x_0=0.3 +y_0=1892558 +k=1.0 +units=m +nadgrids=@null +wktext +no_def *(This is done to show a hand wheel on the GIS side - one of both steps is mandatory)*

As the coordinates as well as the occurrence density are recorded to the CSV file, QGIS is able to produce a heat-map from the data. Others are using this method to produce crime rate maps [30]. The heat-map is produced with fixed values so the changes of the growth are visible.

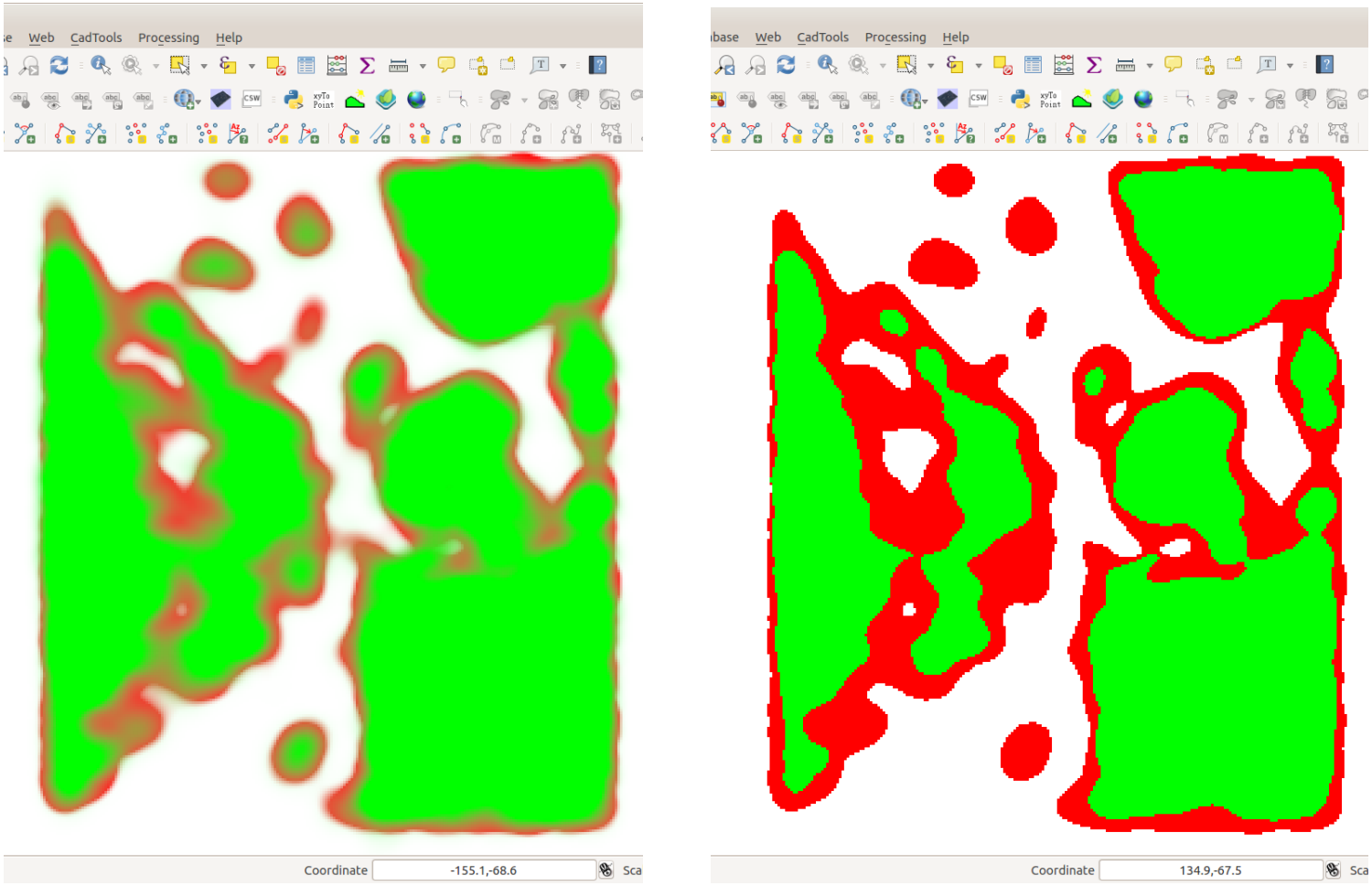

Figure 60: *The images are showing the heat-map of scene 1.2 (green) and 1.4 (red) on the left and the polygonized shapes on the right.*

To get the data into a semantic map, the heat-map is vectorized into a polygon and also reproduced as a mesh. Therefore the meshes are used to be viewed in Open-EASE to show differences while the data is also taken into a spread sheet to show statistical differences in time.

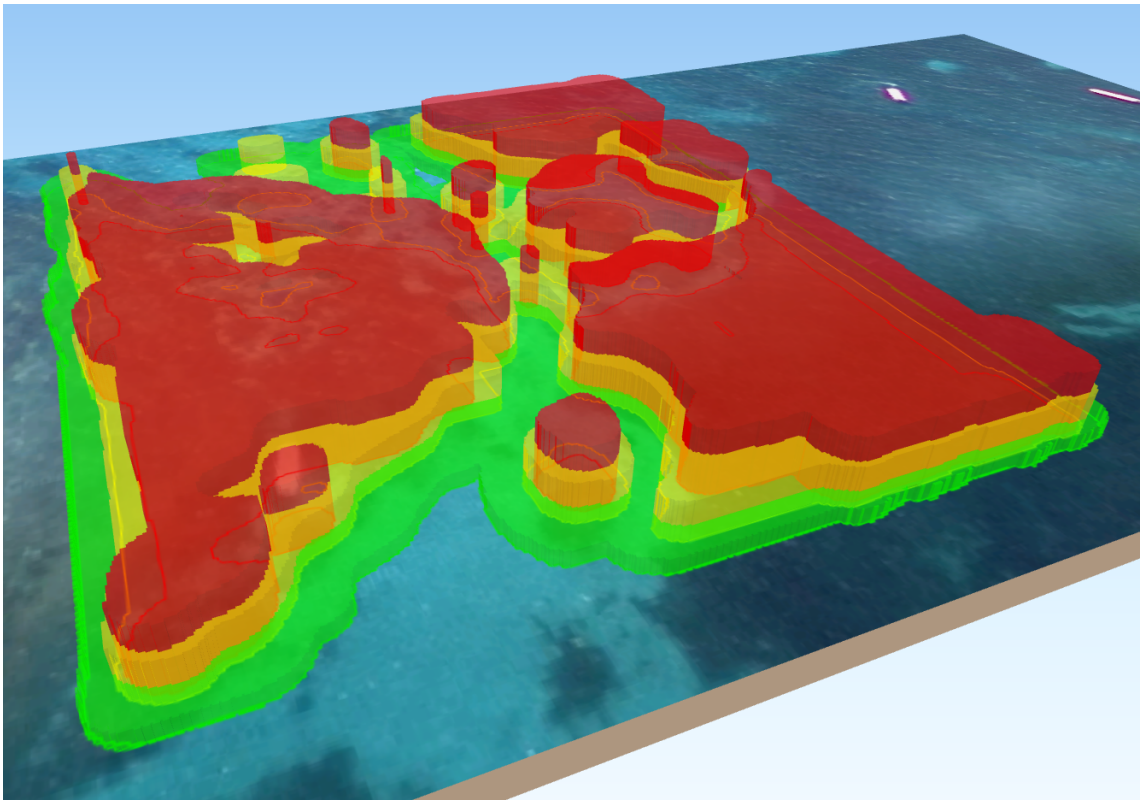

Figure 61: *The produced heat-maps of the different stages are polygonized and extruded to 3D meshes that are exported and given to Open-EASE to indicate differences from one scene to another.*

As the 3D mesh of every scenario is produced, they are loaded into Open-EASE to represent the area coverage of the Seagrass meadows. By putting the results of two scenarios into one Open-EASE scene, one is able to ask for the differences. The varying coverage and differences are also shown in QGIS and are also present on the resulting maps. The data of the scenarios that is produced by the robot is present in QGIS completely in different files so for every CSV file (kind of object) a mesh is produced so different meshes for different tasks can be provided to Open-EASE to work with later in the web interface. For time reasons, only seagrass meshes are used in this work. It's possible to produce meshes for every kind of object, detected by the robot but for this work, on topic is enough to demonstrate the possibilities.

3.7.1 Logging Based on Belief State (semrec)

Semrec [46] is used to record the robots experience as it detects an object. Depending on the object, different belief states are triggered. As the robot receives the "_string" message from Algorithm 14, one of the different callback functions within the semrec client is triggered:

Algorithm 15 *Callback function for the logging of seagrass. This is done for all different object types within the scenes*

```
1  void stringCallback_green(const sensor_msgs::Image::ConstPtr& msg)
2   {
3    BeliefstateClient* bscl_green = new BeliefstateClient("bs_client_plants");
4     bscl_green->setMetaDataField("Survey", "UWSim_Object_Recognition");
5     Context* ctxMain_green = new Context(bscl_green, "Seagrass", "&uwsim;", "
          Seagrass", 0);
6     Object* objSeagrass = new Object("&uwsim;", "Seagrass");
7     Context* ctxSeagrassInView;
8     ctxSeagrassInView = ctxMain_green->startContext("SeagrassInView");
9     cout<<"Logging Seagrass"<<endl;
10    ctxSeagrassInView->addObject(objSeagrass, "Seagrass");
11    ctxSeagrassInView->end();
12    ctxMain_green->end();
13    bscl_green->exportFiles("UWSim_Object_Recognition");
14    cout<<"Writing Seagrass knowledge to file"<<endl;
15    delete objSeagrass;
16 }
```

The code above starts a "Context" which indicates the logging of seagrass when the plants object is in the view of the robot. It creates an object that is stored into a Mongo database (MongoDB) provided by the semrec server, running in the background. This data base is later used to connect the experienced logging (object recognition) with the knowledge from the ontology. As all steps in the system are logged with a specific time stamp, all information that are available for this specific time point can be viewed afterwards.

In the remainder of this section, the procedere of how to start and work with the logging system will be shown in detail. First the Server of Semrec needs to be started:

```
1 rosrun semrec semrec
```

and it needs to give the user the feedback that he can start to use the system:

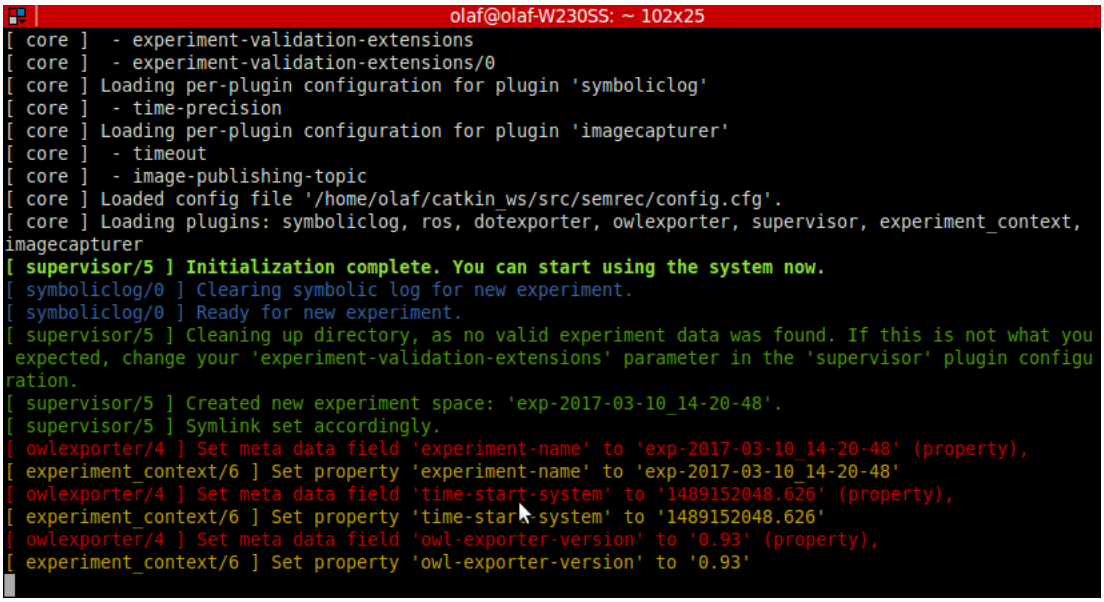

Figure 62: *After the semrec server started, the system reports a complete initialization, that it can be used*

The logging process is started by using the command of:

```
1 rosrun semrec_client uwsim_logger
```

As the detection of the objects is running, it is directly connected to the logger:

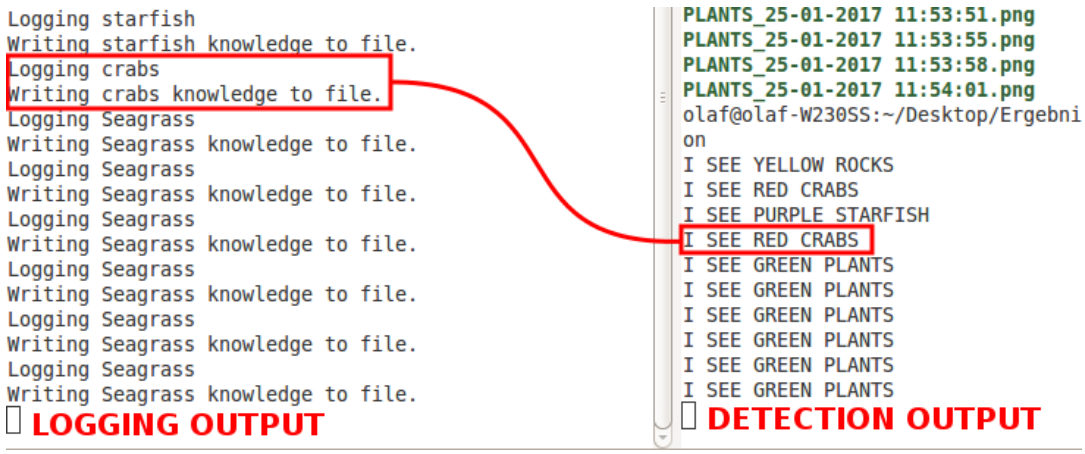

Figure 63: *The detection algorithm is triggering the logger to write knowledge into the data base (cf. Alg. 15 and 14)*

3.8 Producing and Transferring Knowledge (Open-EASE)

(Open Source Everyday Activity Science and Engineering) Open-Ease provides a web-based knowledge service to the robots experiences. The knowledge is stored semantically and represents an episodic memory of the robot. It is possible to reconstruct the complete interaction of the robot with the environment and store perception in a database that can be reached via internet or within the local network. If the robot is able to identify something on the seafloor, it will be stored within the database (MongoDB). This method combines cloud computing with big-data and can be used in the future to take a closer look on the seafloor by giving a robot the chance to store its knowledge within the database. It would give researchers the chance to have an automated researcher "right in the field" to obtain objects in the area. The robot is drawing experience in this field and other robots are able to learn from it as much as the robot learns itself. As robots are learning the same way as humans by investigating objects, they can store the gained information into cloud based databases. This gives the chance to other robots to look up these databases and profit from the knowledge. Open-EASE provides an interface for the user to ask queries to the system which is able to answer in terms of database entries and 3D visualization. A short example would be "where did you find rocks in the area?" And the robot would give an answer in ways like highlighting the spots where a rock are be found and can annotate them with speech bubbles. The User - Database interaction is done via browser as a so called "webrob". It is capable of action as a multi-user interface on different ports that will give each user a private place with transparency in knowledge and access to shared memories.
As the Robot is performing its mission, the experiences need to be recorded. This is done by Semrec, that automatically coverts files in the Ontology Web Language (OWL) format and can be loaded into Open-EASE. It contains the time stamp with the experience and stores it into the mentioned OWL file. While doing so, the transformation data also needs to be recorded. This is done by the “mongodb logger” which is able to record the transformation information from ROS topics so it produces an individual name for each time stamp and stores the transformation data of the robot and the objects. This gives the opportunity to visualize the robot model with the experiences that are logged with Semrec and reproduce the complete mission with the robots experiences. This in combination with the semantic database is a complete knowledge with 3D visualization of the robots experience.

3.8.1 Ontology and Semantics (Knowledge base)

The semantic matter can be found in almost every kind of map that includes objects and their information. But what is semantics?

"Semantics is primarily the linguistic, and also philosophical study of meaning - in language, programming languages, formal logics, and semiotics. It focuses on the relation between signifies - like words, phrases, signs and symbols - and what they stand for, their denotation" (Wikipedia: Semantics).

With this in mind, a map contains a lot of symbols, colors and annotations. This gives a semantic feedback of objects in the area the map describes. It is possible to create a digital map in a semantic way with an ontology in the background that allows to ask for objects and their relations in a 3 dimensional environment. This work is focused on creating a map with POI (points of interest) and AOI (areas of interest) that reflect a specific object like a rock, a crab or simply a part of a seagrass meadow. The map contains in the end an overview of the extension of seagrass and the appearance of animals and non living material. By using a semantic map it is possible to ask for the information that are recognized by the robot which is producing the maps contents.

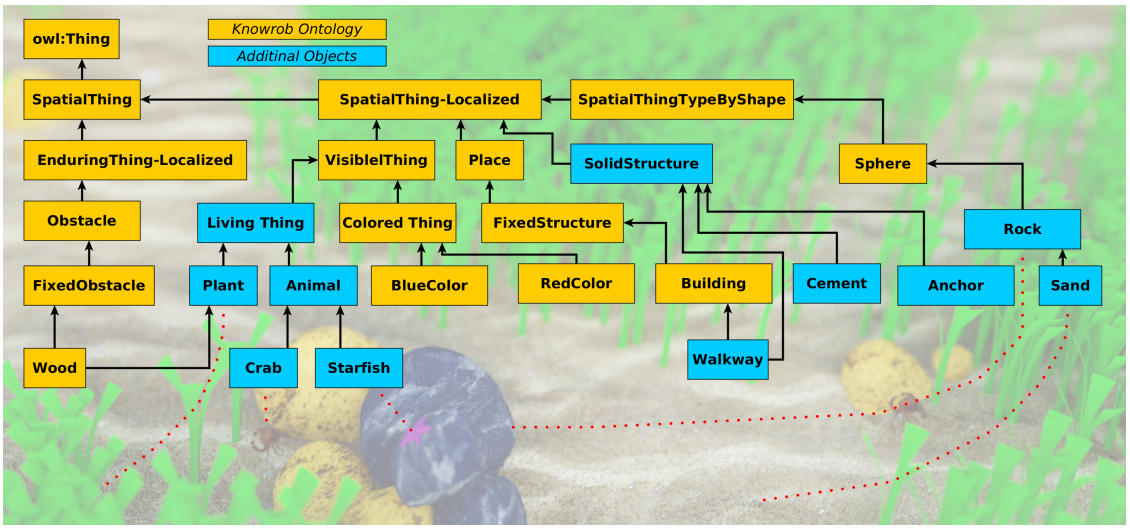

Figure 64: *This orthogonal structure of the ontology shows how the objects in the map are defined in the knowledge base. Yellow represents the original robotics ontology (KnowRob) and blue represents the added information for this work.*

Learning means that the information must be stored somewhere like in a written way or at least the information needs to be communicated by someone. The semantics in the background provide a knowledge base that the robot can work with and contain the needed information to identify and point to recognized objects. Approaches in meaning of geo spatial data can be seen in chapter 4. While performing the mission,

the robot is recognizing objects or at least colors that appear in the vision of the robot. As a specific color appears, the semantic meaning of it gets triggered and the robot records the object. This recording is stored in a database in the background which can later be accessed by Open-EASE to visualize it in the semantic map.

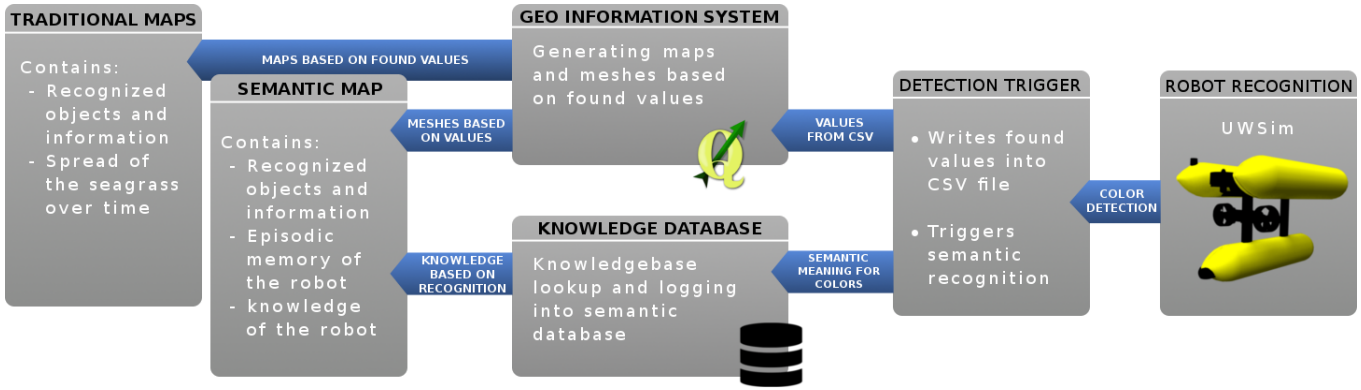

Figure 65: *As the robot recognizes an object, semrec gets triggered to write the knowledge into the data base as well as the algorithm is writing the values into the CSV file. From these data sets, traditional as well as semantic maps are generated. These can also be read by Open-EASE.*

3.8.2 Knowledge Based on Robots Experience

The knowledge of the robot is defined in an ontology that contains all objects and their specifications. In this case its the name of the object with its color. More detailed information can be added or linked to other ontologies that represent a much deeper degree of detail. Here, the knowledge is triggered by the incoming color on one of the ROS topics. Red represents crabs, green is indicating plants. The knowledge of the robot here is very limited as all green objects are plants for the robot as well as all purple objects are starfish. Reality also holds green starfish and purple plants but for these scenarios the knowledge is enough.
Semrec (the experience recorder mentioned in 3.7.1) uses the knowledge to produce a time stamp connection to objects that the user can ask for in Prolog language. As the robot is "seeing" red color, Semrec sets a time stamp with the annotation of "CrabInView" that directly is connected to the "Crab" individual in the ontology that holds the background knowledge. The result is a representation in Open-EASE that contain an experience of a seen object which is described in the background knowledge.
By using Open-EASE for the knowledge representation, it is possible to visualize the robots experiences and show a feedback of using the knowledge. The semantic background gives a response to identified objects and interaction within the environment. As the robot is exploring the surface and the objects, it is possible to give a representation in QGIS and export the objects for Open-EASE to represent the gained information. As QGIS is able to produce polygons from the heat-map (see chapter "Quantum GIS"), these are loaded into Open-EASE to present the experienced spread of the plants. By combining two of the polygons, a new polygon is produced which indicates the difference (growth/loss) between both states.

Figure 66: *Open-EASE is able to display the knowledge the robot experienced during the mission. For the user it is possible to ask for things like "where in the map are the living things". The response will highlight all living things in the map*

The knowledge recognition is triggered by the color detection from OpenCV so the recorder writes the "link" to the knowledge about the "red object" into the database with a time stamp so the semantic map can not only rebuild the environment with the specific objects but also with the position of the robot at the point it recognized the object.

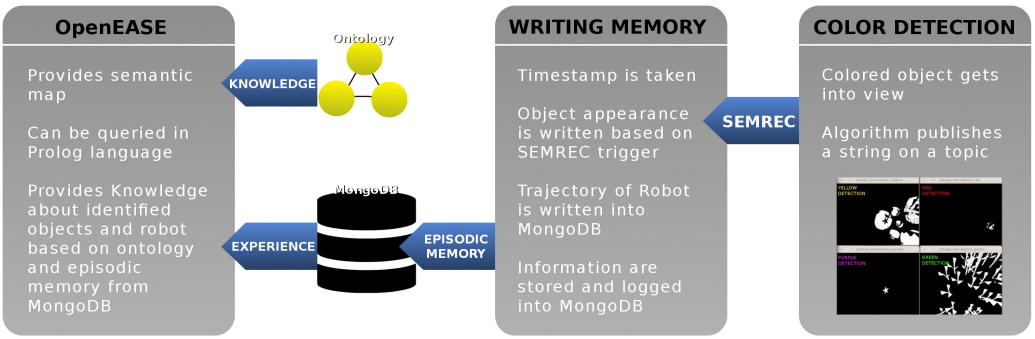

Figure 67: *Schematic diagram of how the episodic memory is written and the experience gets linked to the knowledge while its displayed in Open-EASE. The Knowledge and experience can be queried for in Prolog language*

4 Results

4.1 Resulting Maps from gained Data

The results come with different maps and data sets. Here the maps are represented as two maps on each side:

$$\text{Scene 1: Unaffected} \begin{cases} 1.1 & 1.2 \\ 1.3 & 1.4 \end{cases}$$

$$\text{Scene 2: Affected by Anchoring} \begin{cases} 2.1 & 2.2 \\ 2.3 & 2.4 \end{cases}$$

$$\text{Scene 3: Affected by Shadow Casting} \begin{cases} 3.1 & 3.2 \\ 3.3 & 3.4 \end{cases}$$

$$\text{Scene 4: Affected by Animal Grazing} \begin{cases} 4.1 & 4.2 \\ 4.3 & 4.4 \end{cases}$$

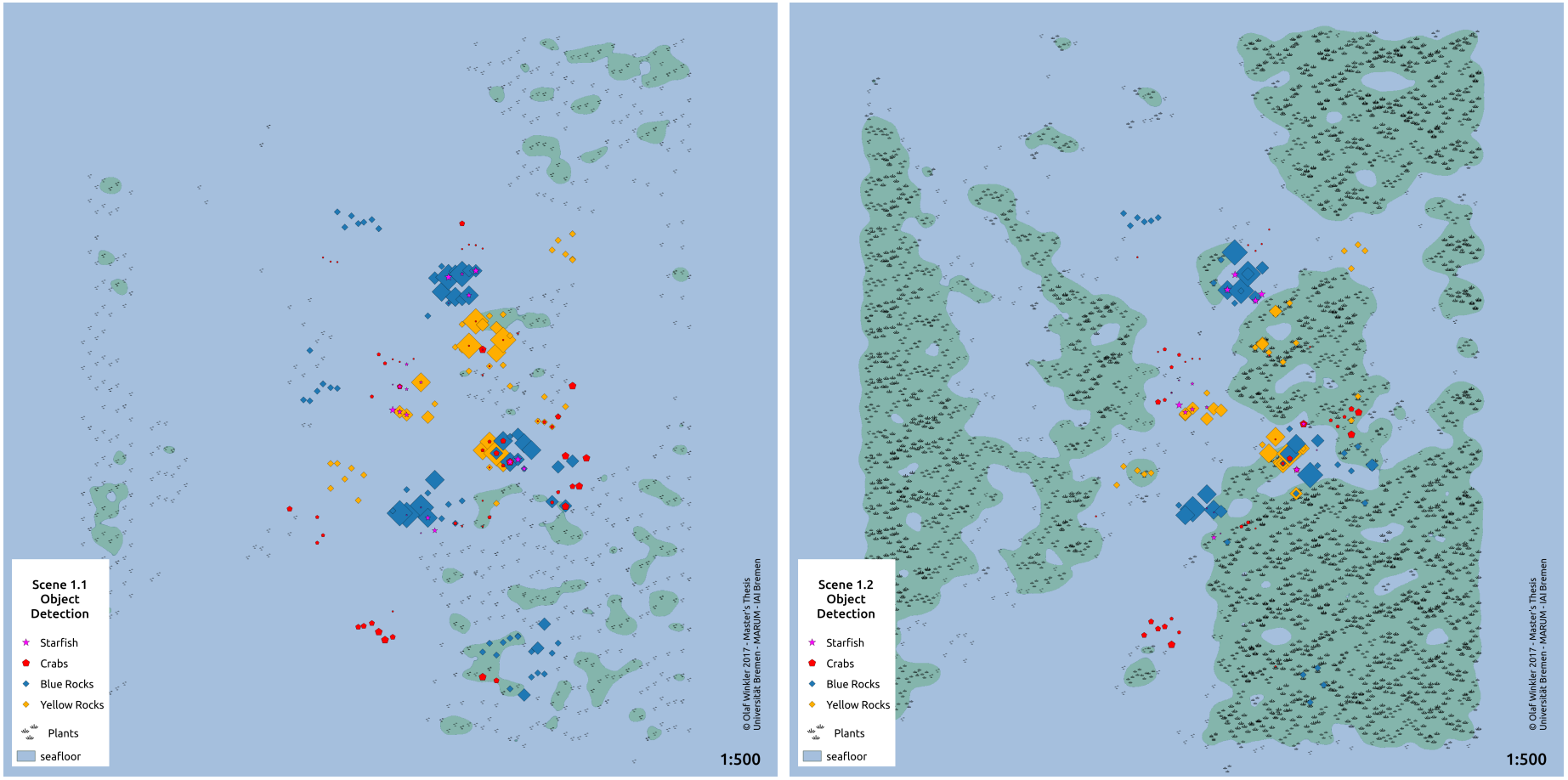

Figure 68: *Resulting maps of sequences 1.1 and 1.2*

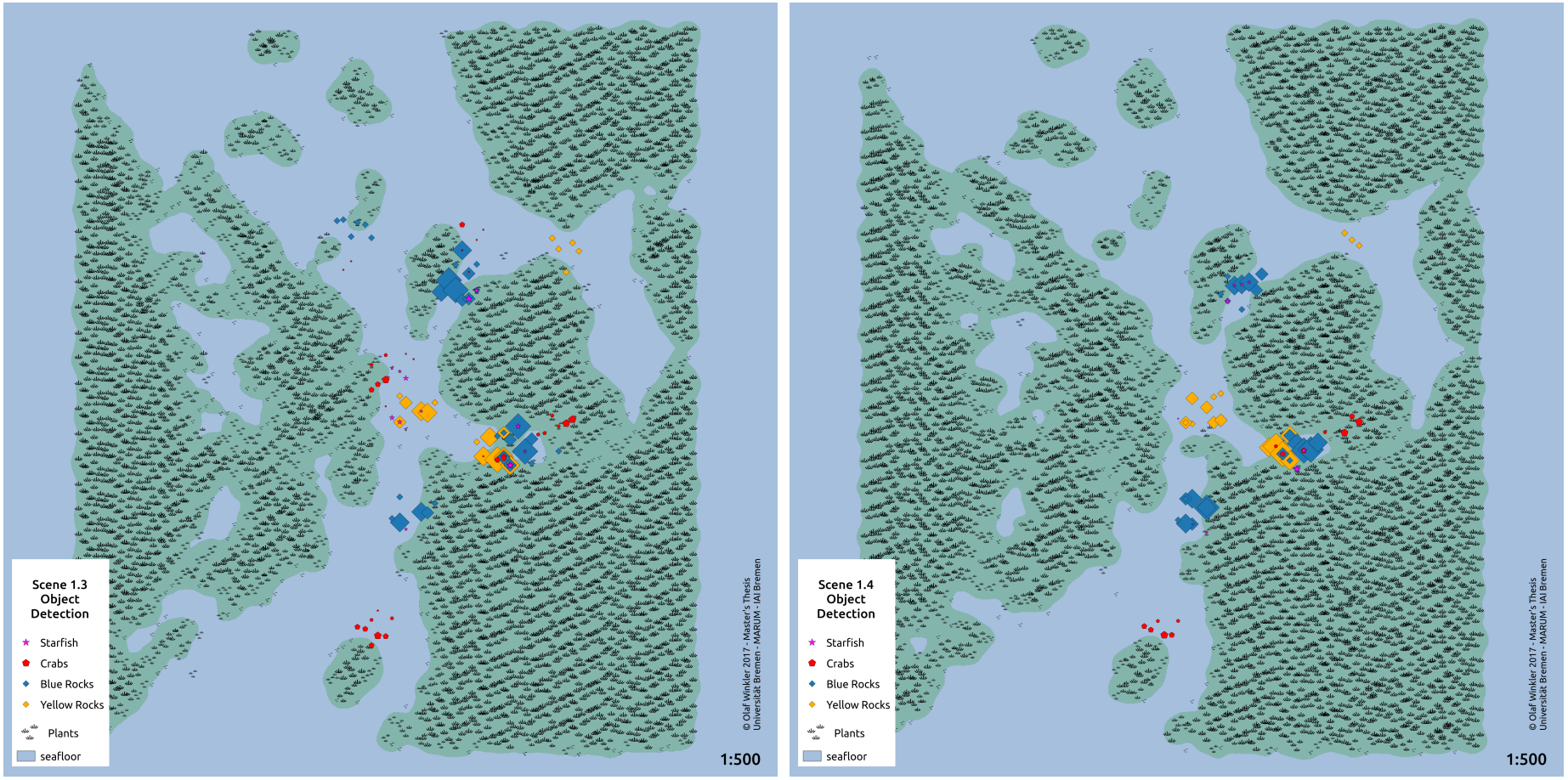

Figure 69: *Resulting maps of sequences 1.3 and 1.4*

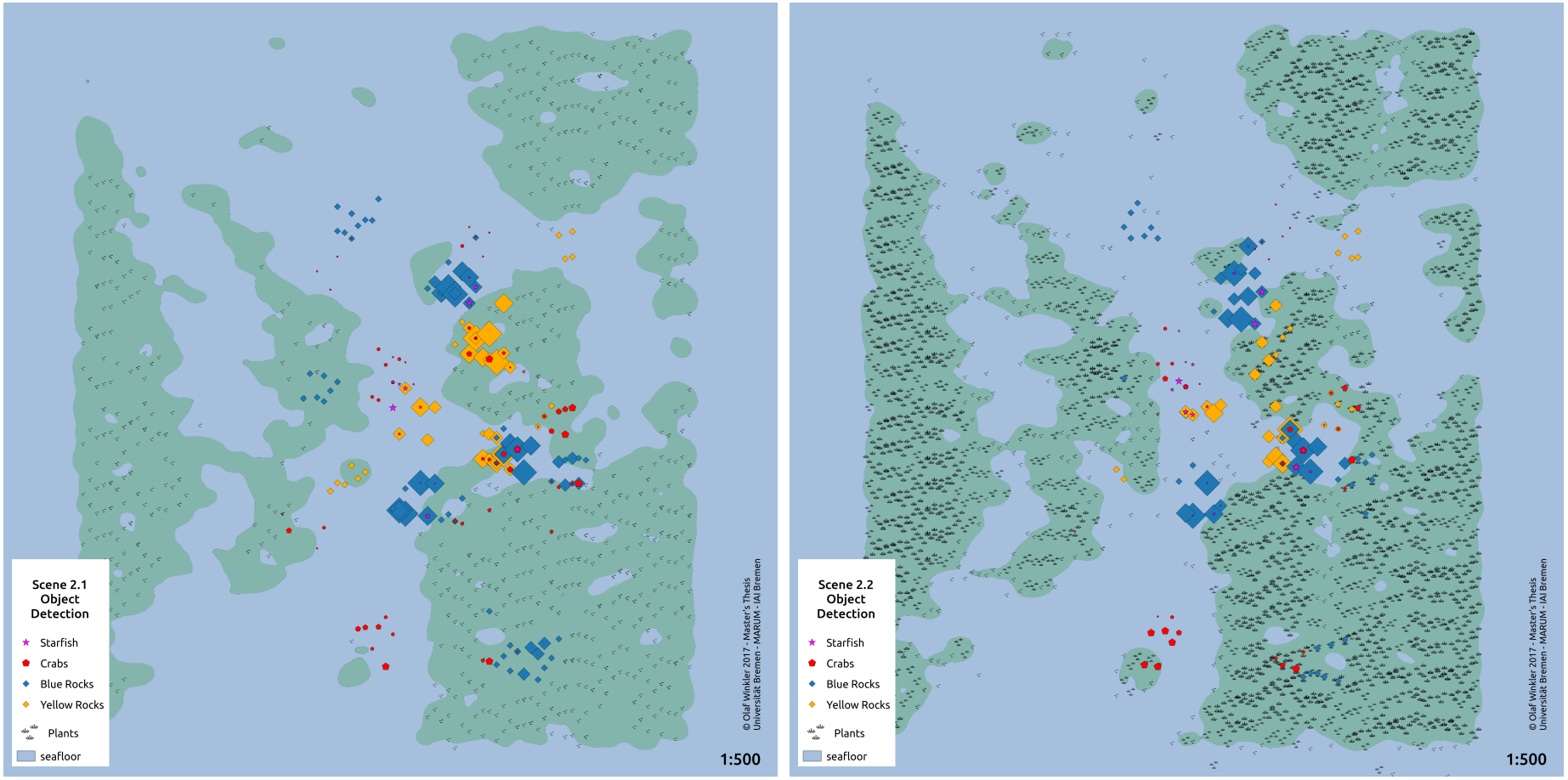

Figure 70: *Resulting maps of sequences 2.2 and 2.3*

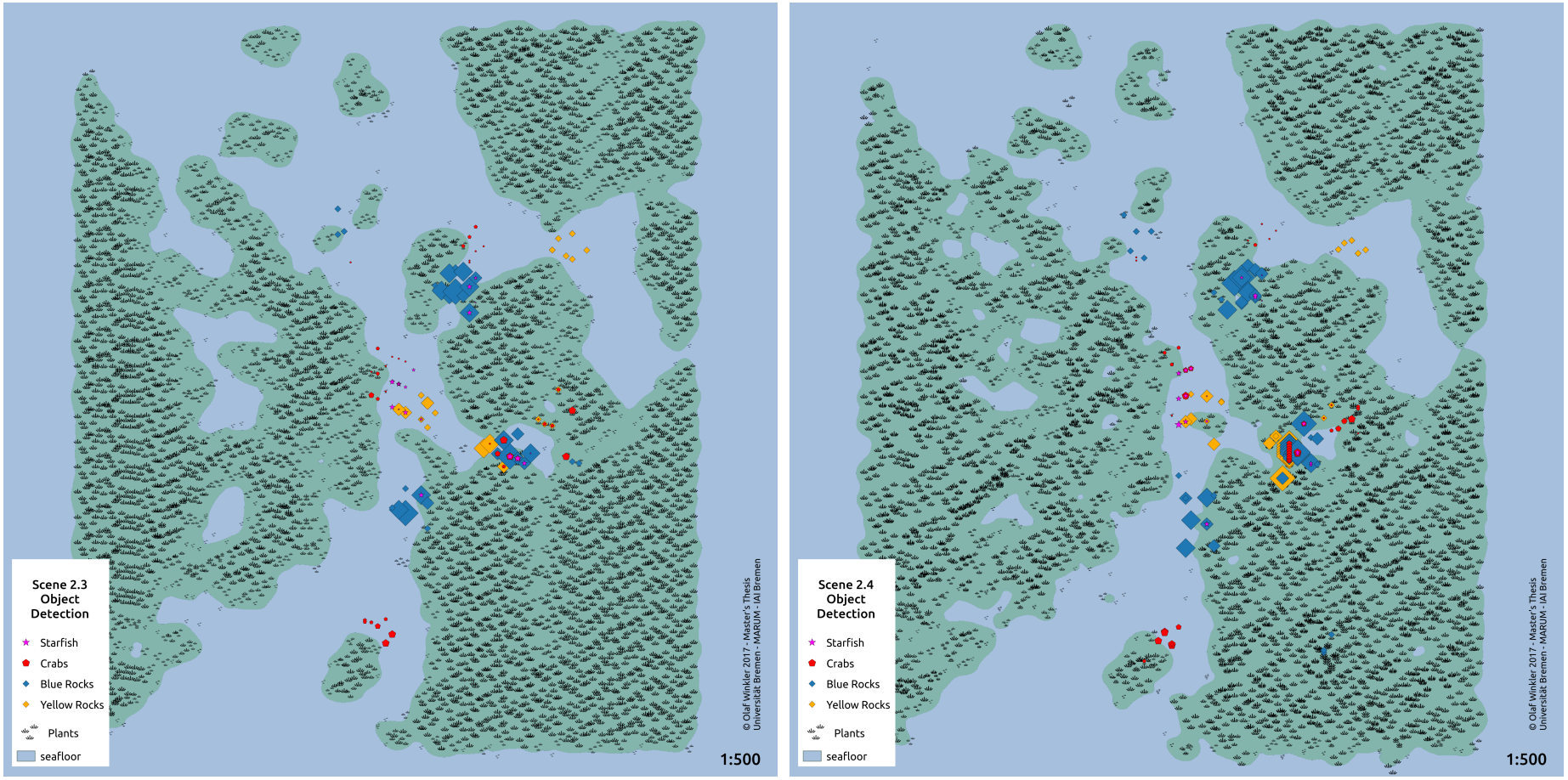

Figure 71: *Resulting maps of sequences 2.3 and 2.4*

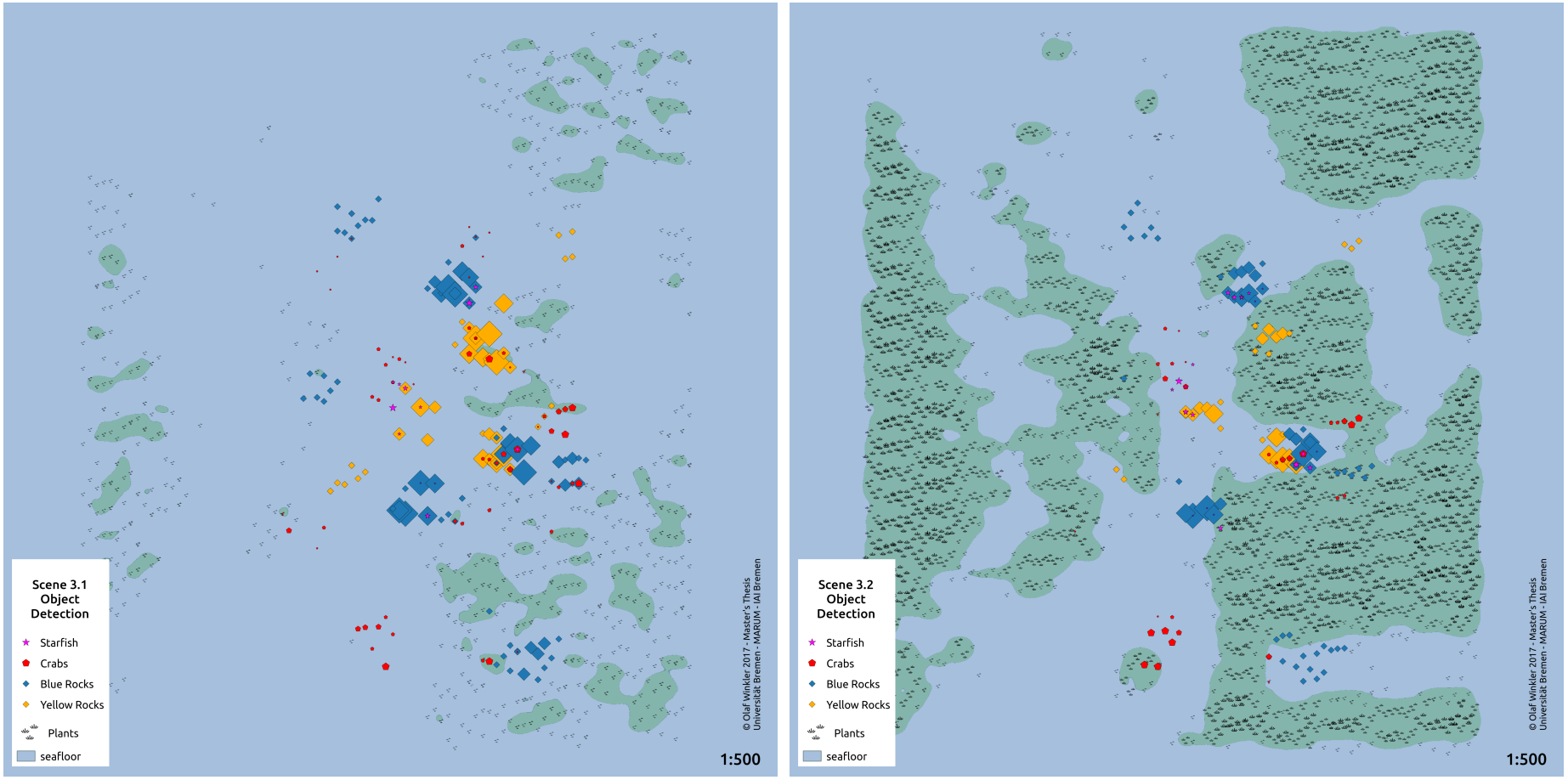

Figure 72: *Resulting maps of sequences 3.1 and 3.2*

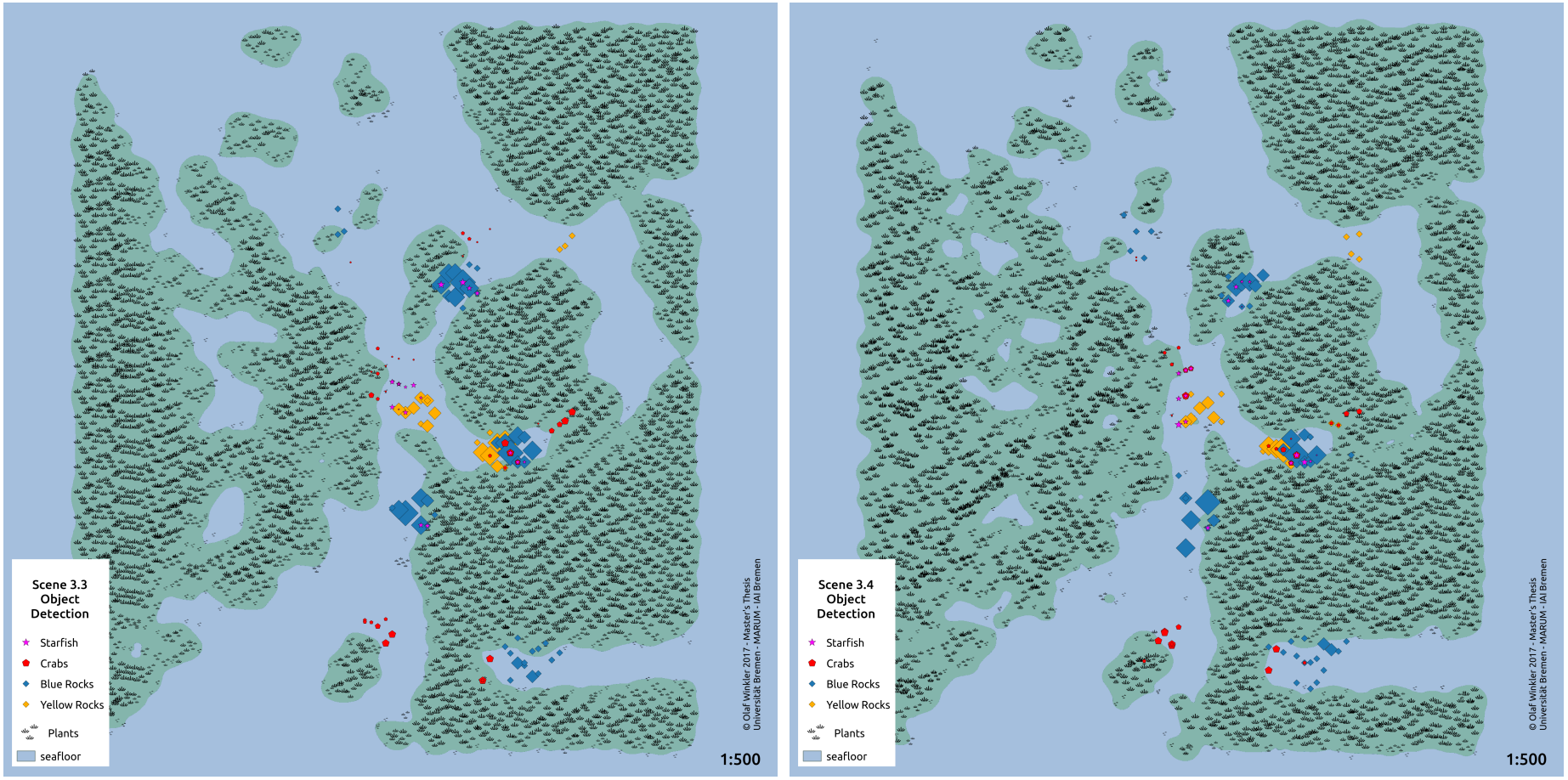

Figure 73: *Resulting maps of sequences 3.3 and 3.4*

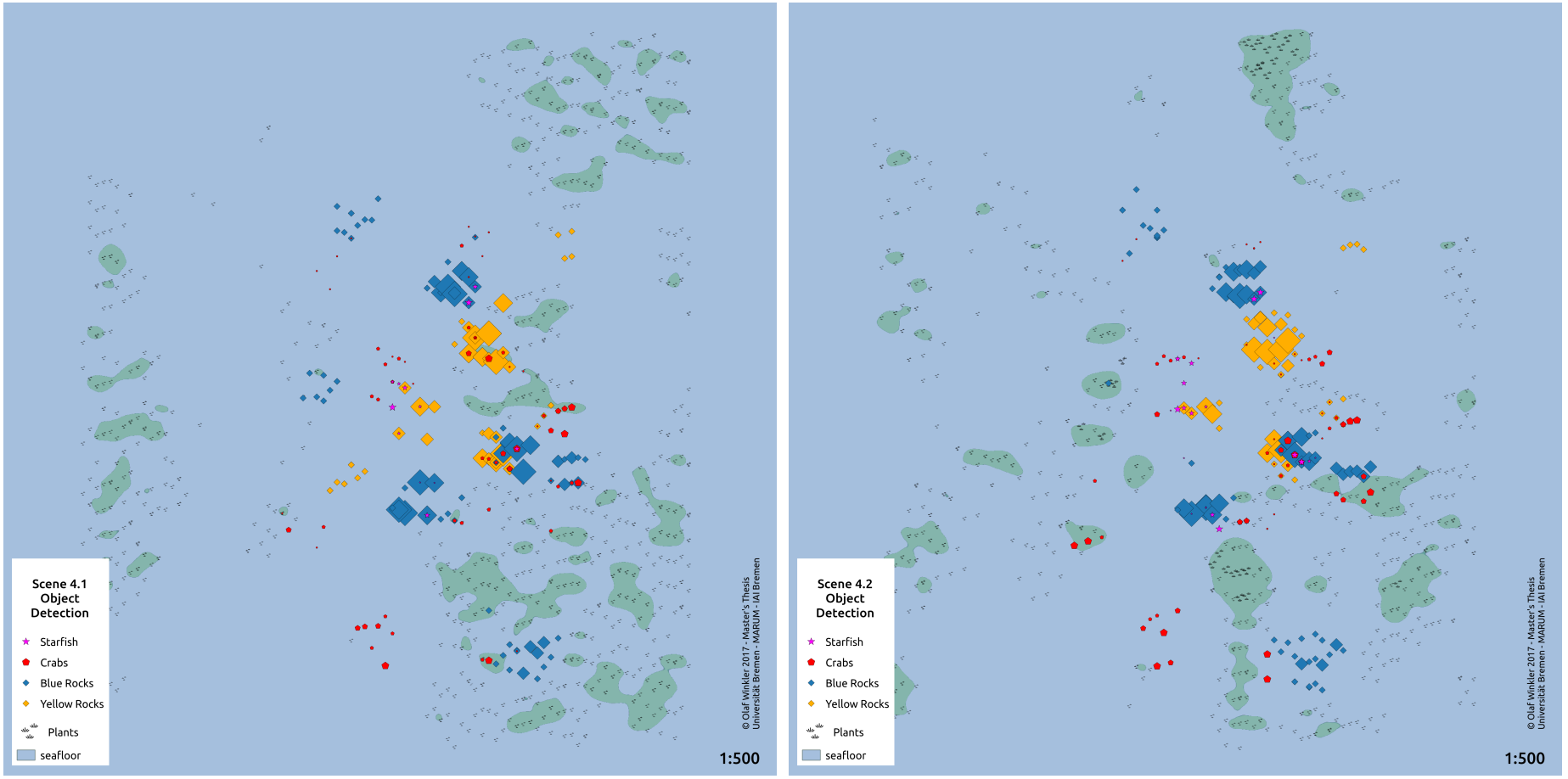

Figure 74: *Resulting maps of sequences 4.1 and 4.2*

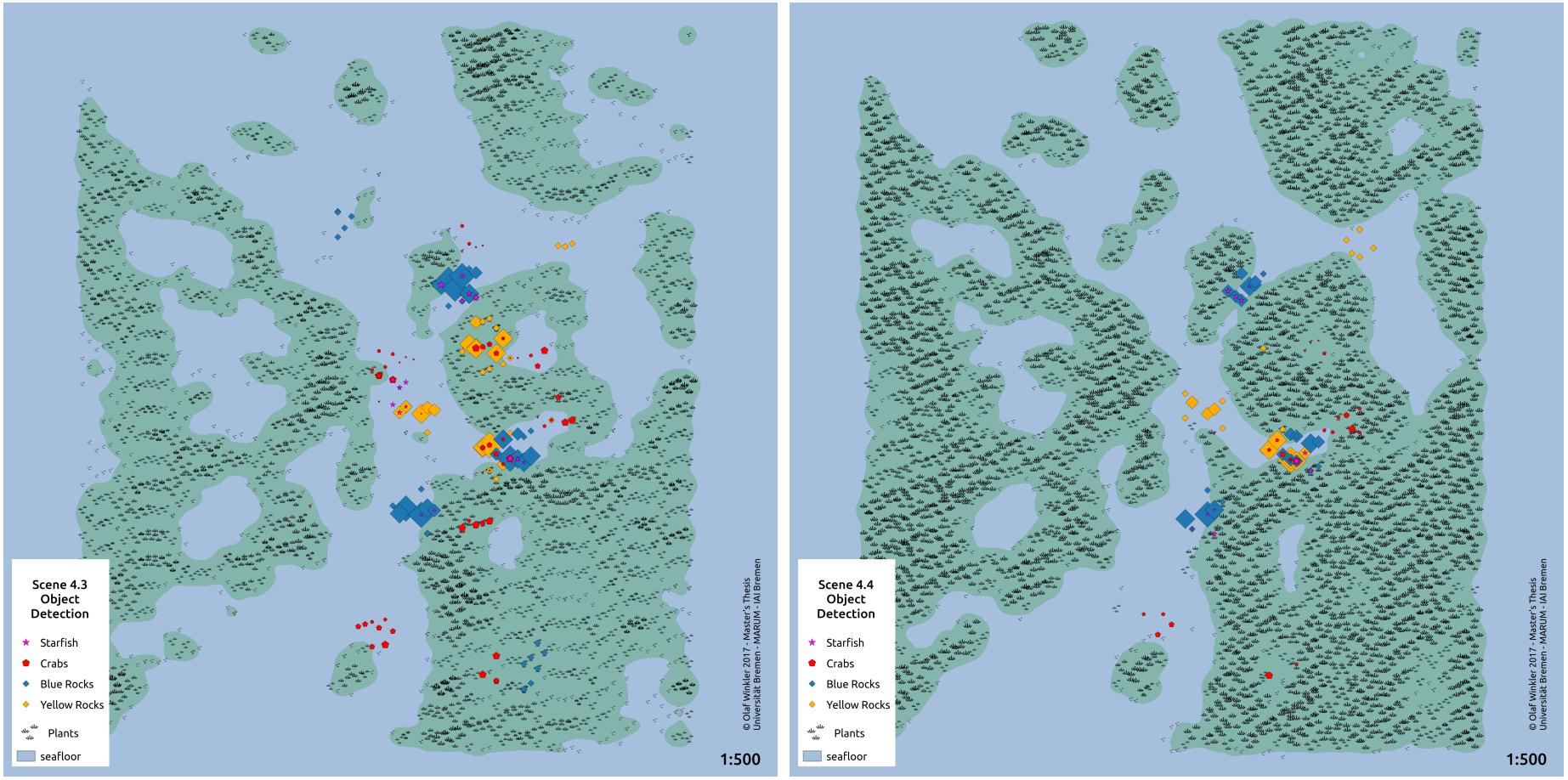

Figure 75: *Resulting maps of sequences 4.3 and 4.4*

4.2 Resulting Data

The Robot scans the area and receives a color RGB image from the camera topic which is separated into the color channels of green, purple, red, yellow and blue. These channels create with a threshold a binary image, which is divided into white and black pixels (similar to 0 and 1). The appearing white pixels are calculated as percentage of the camera image resolution and stored into a CSV file. By receiving data from the GPS, the latitude and longitude are also stored into the CSV file. With the created reference system (s. chapter "Using Extracted Data"), the points containing the percentage, are displayed onto the map and indicate appearance density.

From the results, different maps are generated with the gained data. Every scene contains four sequences that contain starfish, crabs, blue and yellow rocks and plants. Starfish are represented as purple stars, crabs as red pentagon, blue and yellow rocks as blue and yellow, rotated squares and plants are represented in two ways: i) as a plant symbol and ii) as a polygon that indicates the spread.

The maps background is kept blue since the area is in a marine environment. They are build up by values from the robots created CSV files which contain the percentage or density of the appearance of the objects. These values provide the indicator to the size of the different objects icon.

As the objects are detected by their color, the image from the video stream is stored for every object with a definition to the object and a time stamp. Also the geoposition to the image is given. Therefore, also color images are stored as .PNG files so the objects can be checked. The images have the same resolution as the camera sensor from the simulated robot so all of them are present in 320x230 pixels.

Every time an object is identified, a picture is taken so the count of images is the same as the count of in-view identifications. This can be seen in table 13, 15, 17 and 19. The count of objects in the scene are also given as percentage of the whole scene detection. This gives the chance to calculate trends which are also given.

Scenario 1 - Unaffected

Scene 1 - Sequences	**1.1**		**1.2**		**1.3**		**1.4**	
	Count	[%]	Count	[%]	Count	[%]	Count	[%]
Starfish	71	2.60	59	2.12	56	1.80	27	0.89
Crabs	111	4.07	79	2.84	59	1.89	20	0.66
Yellow Rocks	158	5.79	144	5.18	116	3.73	100	3.30
Blue Rocks	189	6.93	176	6.34	135	4.34	82	2.70
Plants	2200	80.62	2320	83.51	2748	88.25	2804	92.45

Table 13: *Detected objects of each kind in scene 1 - Undisturbed sequence*

Scene	**Plants**	**Blue Rocks**	**Yellow Rocks**	**Crabs**	**Starfish**
1.1 - 1.2	5.45%	-6.88%	-8.86%	-28.83%	-16.90%
1.2 - 1.3	18.45%	-23.30%	-19.44%	-25.32%	-5.08%
1.3 - 1.4	2.04%	-39.26%	-13.79%	-66.10%	-51.79%
1.1 - 1.4	*27.45%*	*-56.61%*	*-36.71%*	*-81.98%*	*-61.97%*

Table 14: *Differences of detection between the scenes*

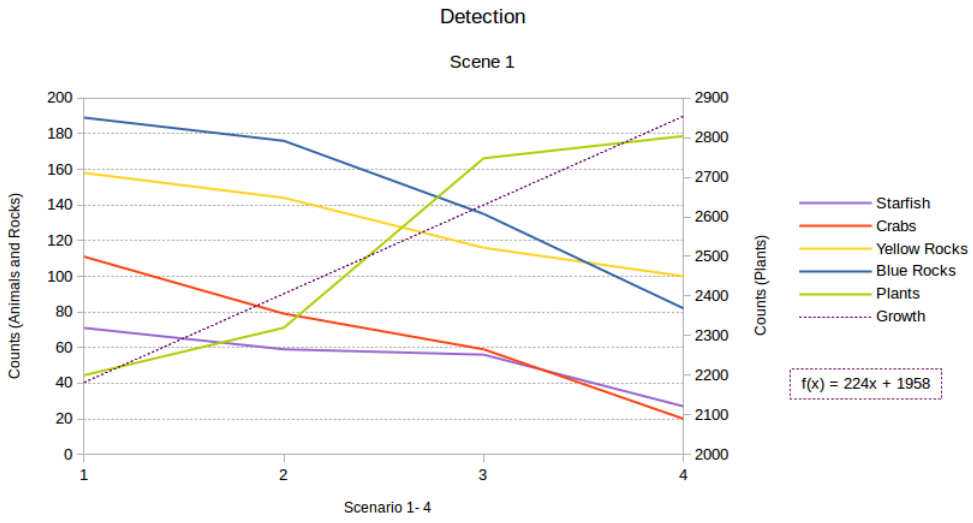

Figure 76: *Overall detection of objects in Scene 1. Recognition of plants increase while other objects are less detected. This means that the growth of the plants encrusted other objects*

The undisturbed scenario is showing an expected trend. As the growth is set as increased in 1.2 and 1.3, the trend line is higher within this area. While the intensified growth of the plants is given and the other objects are not modified, the detection of them need to decrease over time. Scenario 1 is showing this exact trend with all of the objects but with a more steep decline of blue rocks. While the detection differences of plants between scenario 1.1 and 1.2 show an difference of around 120 counts, the blue rocks are showing 13 counts difference. The plants representing an increase of ~5.45% while the blue rocks are showing a decrease of 6.88%. From 1.2 to 1.3 the increase of the plants are even larger by 18.45% and the detection of blue rocks is declining by 23.3%. From 1.3 to 1.4, the seagrass increases by only 2.04% but here, the crab detection drops by 66.1%. Overall the Crabs detection decreases by 81.98% from scene 1.1 to scene 1.4 so the crabs are losing the most overall detection of all objects. While starfish and blue rocks are losing a detection of 61.97% and 56.61%, yellow rocks are losing 36.71% in detection. The plants detection is increasing by 27.45% overall.

Scenario 2 - Anchorage

Scene 2 - Sequences	**2.1**		**2.2**		**2.3**		**2.4**	
	Count	[%]	Count	[%]	Count	[%]	Count	[%]
Starfish	62	2.09	71	2.34	59	1.86%	74	2.04
Crabs	117	3.95	78	2.57	64	2.02%	101	2.78
Yellow Rocks	168	5.67	155	5.11	127	4.00%	171	4.71
Blue Rocks	194	6.54	183	6.04	142	4.48%	181	4.99
Plants	2424	81.75	2544	83.93	2781	87.65%	3102	85.48

Table 15: *Detected objects of each kind in scene 2 - Disturbed by anchor*

Scene	**Plants**	**Blue Rocks**	**Yellow Rocks**	**Crabs**	**Starfish**
2.1 - 2.2	4.95%	-5.67%	-7.74%	-33.33%	14.52%
2.2 - 2.3	9.32%	-22.40%	-18.06%	-17.95%	-16.90%
2.3 - 2.4	11.54%	27.46%	34.65%	57.81%	25.42%
2.1 - 2.4	*27.97%*	*-6.70%*	*1.79%*	*-13.68%*	*19.35%*

Table 16: *Differences of detection between the scenes*

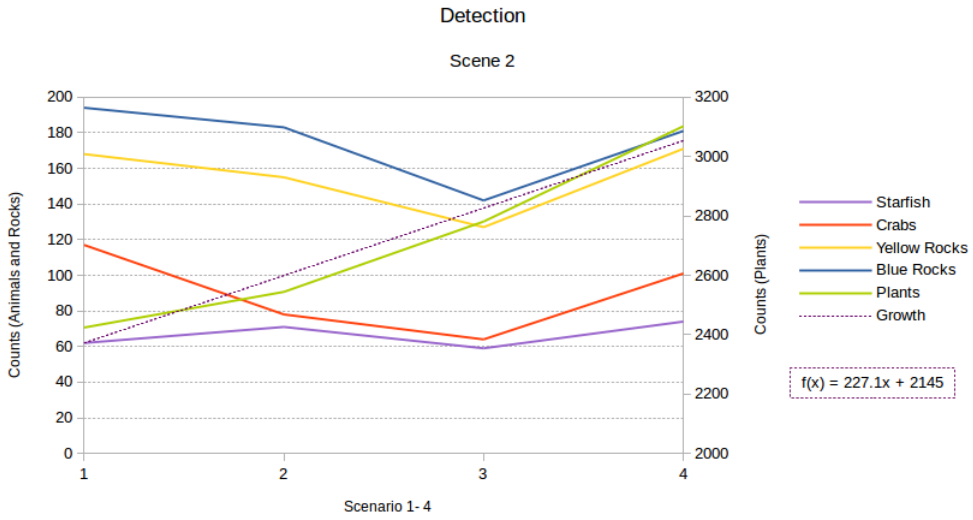

Figure 77: *Overall detection of objects in Scene 2. Recognition of plants increase while other objects are less detected except in sequence 4 where the overall detection increases*

The second scenario is disturbed by an anchor that is rearing a trench into the eastern section of the scene. This shows a first difference in plants detection and from scene 2.1 to 2.2 the detection is increased by 4.95%. This makes a difference of 0.5% compared to the first, undisturbed scenario. As the plants are growing, the detection is showing an increase of 4.37%, a difference of 8.63% compared to an undisturbed scenario. Blue and yellow rocks and starfish show a smaller detection decrease compared to the undisturbed scenario from 2.1 to 2.2. Blue rocks are 1.21%, yellow rocks are 1.12% more detected compared to the undisturbed area and starfish even show a positive detection count of 14.52% despite the disturbance. Crabs detection decreased by 33.33% between these scenes and shows the highest detection lost in this scene but nevertheless by decreasing additional 17.95% from 2.2 to 2.3, the crab detection is showing a declination peak at 64 counts compared to initial 117 counts. While scene 1.1 to 1.2 shows a decline of 16.9% in starfish detection, scene 2.1 to 2.2 is showing an increase of 14.52%. 2.2 to 2.3 is showing an increase of 9.32% of seagrass plants but in this case all other objects are less detected. The highest detection lost from 2.2 to 2.3 is in blue rocks. A lost of 22.4% also shows a peak of 142 counts. This is a relatively small lost of 0.9% compared to the undisturbed scene. As starfish experienced an increased detection from 2.1 to 2.2, the detection decreased from 2.2 to 2.3 by 16.9%. This is a decrease by 4.84% comparing 2.3 with 2.1. As the first scenario is showing an increase of plants by only 2.04% from 1.3 to 1.4, the scenes 2.3 to 2.4 are showing an increase by 11.54% which is a very high value compared to the first scenario. As the undisturbed scene is supposed to show a schematic like “small increase, high increase, small increase”, this scenario is showing “small increase, intermediate increase, high increase”. Overall the seagrass is showing a slightly bent line of detection. In opposite to the perceptions in this scene, scene 2.3 to 2.4 showing a positive trend in detection. As the plants are getting detected much more than in the undisturbed scene, also more objects are getting detected. The highest value can be found in crab detection by 57.81% of more detection comparing the overall detection of 2.3 to 2.4. Comparing 2.1 with 2.4, the plants detection increases by 27.97%. This is a similar value like in the unaffected scenario which 27.45% so the affected scenario 2 has an overall increase of 0.52% compared to the undisturbed. Blue rocks and crabs show a decreasing detection of 6.7% and 13.68% from the first to the final stage of this scene. Beside the manner of decreasing, yellow rocks and starfish show an increasing detection from first to final scenario. Yellow rocks are getting detected more by 1.79% and starfish are even 19.35% detected more.

Scenario 3 - Shadow casting

Scene 3 - Sequences	**3.1**		**3.2**		**3.3**		**3.4**	
	Count	[%]	Count	[%]	Count	[%]	Count	[%]
Starfish	62	2.09	57	1.87	56	1.78	53	1.69
Crabs	117	3.95	67	2.20	69	2.19	69	2.20
Yellow Rocks	168	5.67	151	4.95	122	3.88	108	3.44
Blue Rocks	194	6.54	189	6.19	156	4.96	132	4.20
Plants	2424	81.75	2588	84.80	2744	87.19	2778	88.47

Table 17: *Detected objects of each kind in scene 3 - Disturbed by a shadow casting walkway*

Scene	**Plants**	**Blue Rocks**	**Yellow Rocks**	**Crabs**	**Starfish**
3.1 - 3.2	6.77%	-2.58%	-10.12%	-42.74%	-8.06%
3.2 - 3.3	6.03%	-17.46%	-19.21%	2.99%	-1.75%
3.3 - 3.4	1.24%	-15.38%	-11.48%	0.00%	-5.36%
3.1 - 3.4	*14.60%*	*-31.96%*	*-35.71%*	*-41.03%*	*-14.52%*

Table 18: *Differences of detection between the scenes*

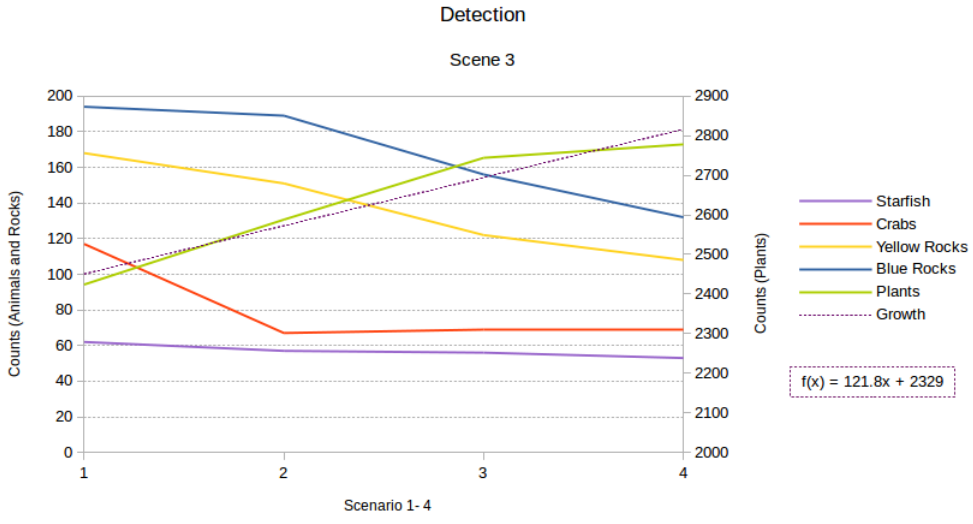

Figure 78: *Overall detection of objects in Scene 3. Recognition of plants increase and lower in the later scenarios while other objects are less detected with a drop between the firs and the second sequence*

The third scenario shows in scenes 3.1 to 3.2 in increasing plants detection of 6.77% and is slightly higher than in the undisturbed scenario. All other objects are showing a decreasing detection and blue rocks are getting 4.3% more detected than in the undisturbed scenario (1.1 to 1.2). Also starfish are getting less detected (lost of 8.06%) but are getting 8.84% more detected than in the undisturbed scene of 1.1 to 1.2. Yellow rocks and crabs are getting less detected between scene 3.1 and 3.2. Yellow rocks are detected 10.12% less (1.26% less than in undisturbed scene) and crabs show a decreased detection by 42.74% (13.91% less than in undisturbed scene). Comparing the plants growing trend, the plant detection is relatively similar of 6.03% from scene 3.2 to 3.3 compared to 6.77% in scene 3.1 to 3.2 and can be called almost linear. The plant detection increases from 3.3 to 3.4 but with a smaller gradient by only 1.24%. in 3.2 to 3.3, the yellow rocks show a very high detection lost by 19.21%. Also blue rocks are showing a detection lost of a high value of 17.46% and starfish showing a relatively small decrease by 1.75% but crabs are getting 2.99% more detected. From 3.3 to 3.4, blue rocks are showing also a high detection lost of 15.38% which is not as high as in 3.2 to 3.3 but still a lot of detection lost. Yellow rocks are detected 11.48% less in 3.3 to 3.4 and starfish detection lost shows a value of 5.36%. Crabs are detected as before with 69 counts and don't change in this stage. Overall, scene 3 is showing a similar trend as the unaffected scene of increasing plant detection and decreasing other objects detection but with less detection in plants and more detection of objects compared to the undisturbed scene. With overall 14.6% from 3.1 to 3.4 the plants are 12.85% less detected than in 1.1 to 1.4. Blue rocks show with overall 31.96% a higher detection rate of 24.92% more detection in 3.1 to 3.4 compared to the undisturbed scenery. Yellow rocks are almost similar with 35.71% and show a slight difference of 1% to the undisturbed scene. Crabs are detected 41.03% less from 3.1 to 3.4 in show, compared to the first scenario, a difference of 40.95%. Starfish detection decreases by 14.52% and is compared with the first scenario and a difference of 47,45% a very low detection lost.

Scenario 4 - Animal grazing

Scene 3 - Sequences	**4.1**		**4.2**		**4.3**		**4.4**	
	Count	[%]	Count	[%]	Count	[%]	Count	[%]
Starfish	62	2.09	70	2.38	59	1.90	30	1.05
Crabs	117	3.95	126	4.29	92	2.96	39	1.37
Yellow Rocks	168	5.67	157	5.34	157	5.06	95	3.34
Blue Rocks	194	6.54	191	6.50	176	5.67	97	3.41
Plants	2424	81.75	2305	81.49	2619	84.40	2584	90.83

Table 19: *Detected objects of each kind in scene 4 - Disturbed by Manatee grazing*

Scene	**Plants**	**Blue Rocks**	**Yellow Rocks**	**Crabs**	**Starfish**
4.1 - 4.2	-1.20%	-1.55%	-6.55%	7.69%	12.90%
4.2 - 4.3	9.35%	-7.85%	0.00%	-26.98%	-15.71%
4.3 - 4.4	-1.34%	-44.89%	-39.49%	-57.61%	-49.15%
4.1 - 4.4	*6.60%*	*-50.00%*	*-43.45%*	*-66.67%*	*-51.61%*

Table 20: *Differences of detection between the scenes*

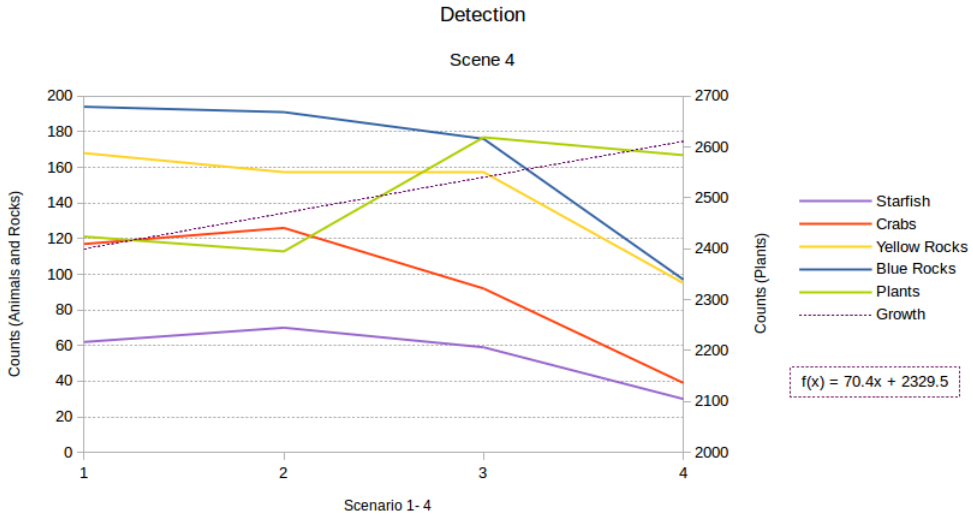

Figure 79: *Overall detection of objects in Scene 4. Recognition of plants increase and decreases in the end scenario. Other objects are less detected except crabs and starfish which are increasing from 4.1 to 4.2. After the second scenario a drop ob objects can be seen*

In comparison to all the other scenarios, the plants detection decreasing from 4.1 to 4.2 by 1.2%. The detection shows an increase from 4.2 to 4.3 by 9.35% but loses in the scenes 4.3 to 4.4 by 1.34%. This makes an overall increase of plant detection of 6.6% but this value is very small compared to the other scenarios and makes a difference of 20.85% compared to the undisturbed scene. In the beginning, by comparing 4.1 with 4.2, blue rocks are showing a decreased detection of 1.55% which is 5.33% less detection than in the undisturbed scenario and yellow rocks are with 6.55% detection lost 2.31% more detected than in the unaffected scene. Crabs otherwise, are getting more detected by 7.69% from scene 4.1 to 4.2. Also starfish are producing a 12.9% positive detection by comparing these two sequences. From 4.2 to 4.3, as mentioned before, plants detection increases but also blue rocks decrease even more by 7.85%. Now crabs are getting 26.98% less detected which is a slightly similar value as known from the undisturbed scenario (26.98% in 4.2 to 4.3 vs. 25.32% in 1.2 to 1.3 which makes a difference of 1.66%). Also starfish are getting less detected by 15.71% in 4.2 to 4.3 but are getting much less detected than in the undisturbed scene (10.63% less detection). Only yellow rocks are showing a 0% detection difference (both 157 counts) between 4.2 and 4.3. The final stages of the scenery show a very hard drop in object detection. While plants are detected 1.34% less from 4.3 to 4.4, crabs are getting 57.61%, starfish 49,15%, blue rocks 44.89% and yellow rocks 39.49% less detected. These numbers are covering the undisturbed scene in a similar way but are showing a very hard drop of detection. Overall the plants detection increases from 4.1 to 4.4 by 6.6% but all other objects are getting less detected. Blue rocks show a detection drop of 50% while starfish are at values of 51.61% and crabs are even at 66.67% detection lost. Yellow rocks are with a value of 43.45% the least losing detection objects in comparing 4.1 with 4.4.

5 Discussion and Conclusion

As the statistics already show, the first scenario has an undisturbed development with an expected growth of the seagrass and therefore a detection that delivers values of guidance. Map 1 of Scene 1 shows a fair to low appearance of seagrass but Map 2 is indicating a higher appearance. It rises by more than 5% and the map indicates the difference. Map 2 can truly indicate the outer borders of the meadows and the appearance is even higher in map 3. By rising of 18.45% compared to map 2, map 3 shows clearly the appearance of the seagrass. Map 4 is by a rise of 2.04% just a final step and results as fully developed meadows. The maps of Scene 1 are references to the other maps so a statement can be made due to lower or higher detection of objects in the scenes. As other objects than the seagrass plants are fix on the seafloor and don't move, they can also be used as reference to the growing meadows. As the meadows grow, they will outgrow the objects so more plants are detected, while other objects are less detected. This can clearly be seen in scene 1. While the plants are detected more and more, the other objects like rocks, getting less detected. The only object that is experiencing a lower detection loss is the starfish between scene 1.2 and 1.3. While there are still individuals that are not detected, it is a trend against the growth of the seagrass. The loss of starfish detection is expected much higher in these scenes since the growing rate between 1.2 and 1.3 is much higher than on the other scenes. Rocks and Crabs are showing a direct drop and starfish are running against this trend. On the one hand this can occur since the starfish are more located on the rocks that are not covered by the plants therefore more get detected. On the other hand, the results show a significant drop in the scene from 1.3 to 1.4 and can also be seen on the maps 3 and 4. Less starfish getting located on the eastern side as well as rocks. The area between the southeastern meadow and the skull shaped meadow in the middle of the eastern part show that the distribution of blue rocks are decreasing as well as the starfish. Crabs seem not as affected as other objects in that area.

Map 1 of Scene 2 indicates that the robot has seen more seagrass but as the initial sequence is always the same, the distribution must come from a different influence to the simulation. Since the simulator underlies a deterministic chaos (like everything in nature), this can come from external influences like background calculations of other software that nor necessarily needed to run and caused difficulties during simulation or differences in the Geo Information System during Heat map calculations. Other results don't seem to have that problem since they are appearing as expected. In Map 2 of scene 2 an anchor tears a trench in the ground in the eastern area from north to south. By comparing Map 2.2 and 1.2, this can be indicated by the line

of lower appearance percentage of the seagrass. Map 2.3 indicates this even more. The trench is expected to disappear in map 2.4 since the growth of the seagrass is as high as the trench should be covered for the vision of the robot but by looking at map 2.4, the trench is still slightly visible. This shows that the robot is capable to see this even after a renaturation process. By looking at map 2.3 and comparing it with 2.4, one fact appears to be strange since the blue rocks of the lower middle part are appearing to be moved or some kind of spread over the ground. This can be explained by the coincidence of appearing in the vision field of the robot. The vision is set to take a picture at a frequency of 0.3 Hz. This makes it likely that if one object appears half in the upper part of the vision, it will appear also half but in the lower part of the vision. Therefore one bigger object will be recognized as two smaller objects. By looking more close to the produced data, this can be seen in "BLUE_ROCKS_22-02-2017 10_53_22.png" and "BLUE_ROCKS_22-02-2017 10_53_25.png" in the results of Scene 2.3:

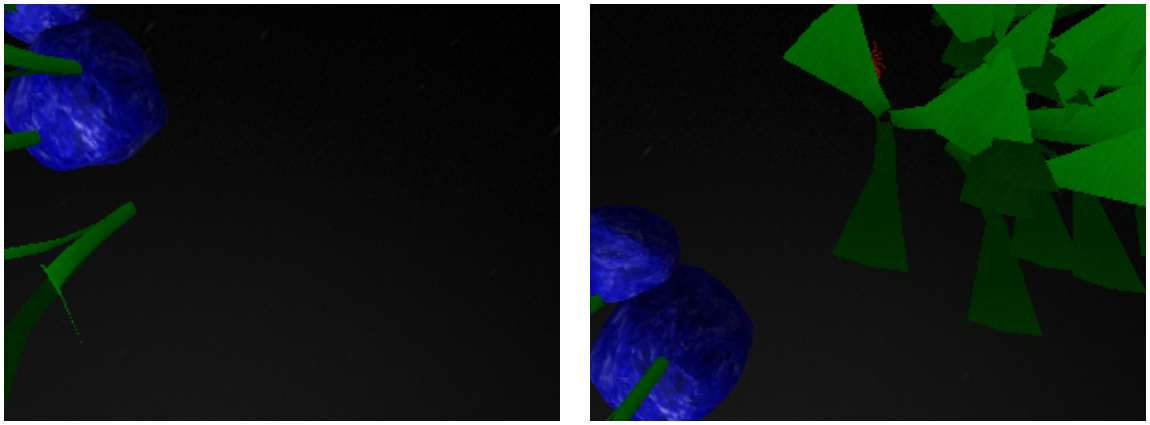

Figure 80: *Both pictures are taken from the robots camera stream while performing the scanning mission. The rocks appear twice in the picture but some parts of them are cut away. In some cases this can lead in the results as two smaller objects instead of one big*

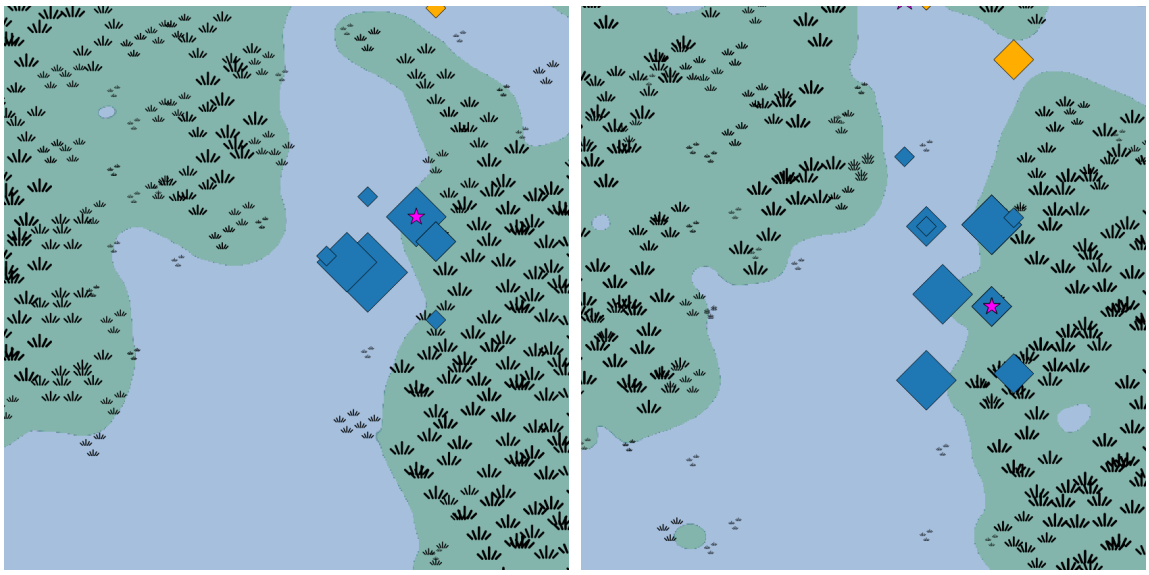

Figure 81: *The images are from the detection area of scene 2.3 and 2.4 where the double detection effect happened during performing the mission. The Squares are the same size because the threshold of icon separation doesn't go as deep as the original distinction in the data set*

This can lead to confusion but only on a small scanning scale. By looking at a bigger picture, this would truly not interfere with the end result. Also as the starfish is identified to be located on the more southern, smaller rock, it can be interpreted that it is located on a larger rock at the southern end. This effect can also be seen in scene 3.

Scene 3 is equipped with a walkway that is casting shadow over the plants. As the plants are dependent on sunlight and shadows are a threat for them, they start to shrivel and disappear. The walkway is located in the south eastern part of the scene 3 and can be located even just by looking at the map. While the coverage of seagrass plants seem to close the gap in map 3.3, the next map shows that the gap not only stays there but becomes more wide. The weight map of the area is prepared to become less weighted over time and the surrounding plants seem to cover the outer parts of the gap in 3.3 therefore as the plants become larger and more dense, they are expected to cover the gap even more in 3.4 but it is as large as in 2.2 and larger as in 3.3. In map 3 the same effect applies to the environment as in map 2, that the robot is able to see and reveal more than one expects and can show influences even if they are not remain anymore.

Maps from scene 4 indicate a grazing process of a manatee. These animals can be found in seagrass meadows as they graze the plants for their food source.

Figure 82: *Manatee (or Dugong) grazing a seagrass meadow which represents its food source*

It is very natural that an animal like this is grazing a seagrass meadow but in times of climate change, habitats as well as food sources shift into warmer areas. Not only manatees are grazing the meadows. As the seagrass environment is hosting a very wide range of animals, they also represent a food source for many animals like birds or larger fish. By hunting for smaller animals, others can damage the plants and leave scars in the remaining meadow or even destroy the meadows if the grazing is happening more often or by many individuals at the same time. The manatee grazing is happening in scene 4.2 and the meadows are getting affected. Afterwards the remaining fields are just growing but if 4.3 and 4.4 are compared to the undisturbed meadows, the remaining holes are humongous. Keeping in mind, that the simulation in this work is exaggerated in terms of growth and this is showing a one time affection, this is indicating that a habitat shift can affect a meadow even harder. In this work, the meadows seem to regenerate very fast but in reality these plant colonies need much longer (many years) to regenerate to a stage like this and as manatees may search for food several times a year, the impact might be even magnitudes higher.

5.1 Robotics Knowledge

As the robot is producing episodic memory, it can be used external with other software like RoboSherlock to identify the objects again and write knowledge in terms of detection. The system comes with specific knowledge about color as indicator for different objects - this is definitely expandable and can truly be a good start for future work. In fact the robot detects objects by color and different objects can come in the same color like red or blue plants (even under water) but in terms of chemical analysis live in the water column with a chemical signature, this approach can directly be used by changing the topic that the system subscribes to. Instead of optics, a chemical sensor can be used to identify the chemical compound. As mentioned in Chapter 4 (alternative scenarios), this could be anything, like salinity, CH_4 seepage, etc. The robot is able to identify objects or changes in the environment as "objects" so the semantic recognition is triggered due to the change.

Semantics started to come up in geo spatial data since the late 90's and early 2000's and develop ever since. The systems, handling these semantics and knowledge bases as flexible enough to ensure that a new knowledge base can be added to an existing system so everyone who attempts to work with a system like this, can be sure that the knowledge contribution can be used if it is made based on a few rules. One attempt can be to use the robot to produce data that can be added to projects like OpenStreetMap, which comes with an hierarchical ontology, or at least for projects like BioDB or Pangaea, a data publishing platform for earth- and environmental Data.

5.2 Data Usage

Since the data from this work is based on a very low level, it is enough to produce semantic maps with the possibility to query for the environment and even point out changes. The Plants in this work show a growing process that the automated robot is able to record and even small differences can be seen in the data presentation (s. different maps in results). Some changes that are not explicit in the maps can be found in the statistics as well as in the semantic maps. Some maps, like 1.4 and 2.4 look very similar but changes become clear after a second look or if one is looking into the statistics. Since the seagrass is clearly damaged in 2.4 and it is expected to have less perceptions, since more of the plants are destroyed, it has on the one hand ~300 perceptions more but on the other hand, the overall detection of plants is 92.45% in 1.4 and only 85.84% in 2.4. Therefore the detection of seagrass went down and comes with a loss of overall biomass.

5.3 Carbon fixation

As the plants are storing carbon in the sediment, it can be calculated for the different stages. In the scenes, I took only high percentage values with always the same frame conditions. As mentioned in the goals, the value is given by $83\frac{gC}{m^2\,yr^{-1}}$. This would give the following overview:

Scene	Coverage	Coverage $[m^2]$	Carbon fixation $[\frac{gC}{yr}]$
1.1	0.1%	0.05	4.15
1.2	3.1%	1.55	128.65
1.3	9.4%	4.7	390.1
1.4	11.3%	5.65	468.95

Table 21: *Carbon uptake from scene 1*

Scene	Coverage	Coverage $[m^2]$	Carbon fixation $[\frac{gC}{yr}]$
2.1	0.1%	0.05	4.15
2.2	3.2%	1.6	132.8
2.3	9.3%	4.65	385.95
2.4	12.3%	6.15	510.45

Table 22: *Carbon uptake from scene 2*

Scene	Coverage	Coverage $[m^2]$	Carbon fixation $[\frac{gC}{yr}]$
3.1	0.1%	0.05	4.15
3.2	2.4%	1.2	99.6
3.3	8.5%	4.25	352.75
3.4	10.4%	5.2	431.6

Table 23: *Carbon uptake from scene 3*

Scene	Coverage	Coverage $[m^2]$	Carbon fixation $[\frac{gC}{yr}]$
4.1	0.2%	0.1	8.3
4.2	0.1%	0.05	4.15
4.3	2.6%	1.3	107.9
4.4	8.1%	4.05	336.15

Table 24: *Carbon uptake from scene 4*

This results in the following chart:

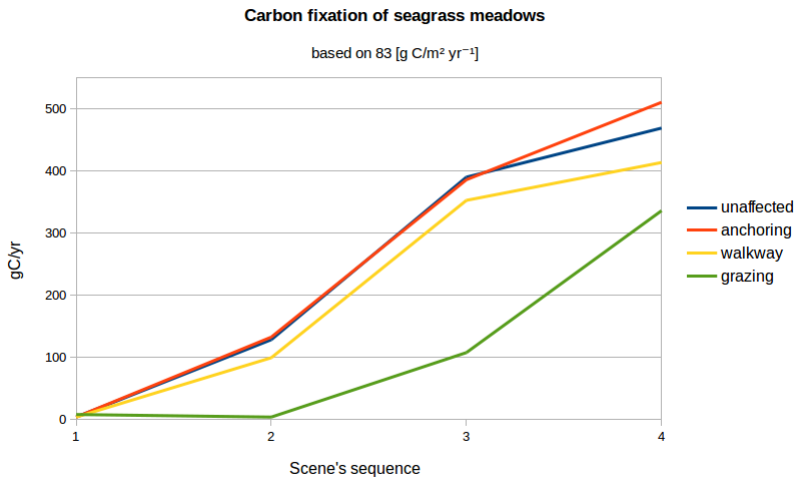

Figure 83: *Carbon fixation of the scenes, based on the presumption of* $83 \frac{g\,C}{m^2\,yr^{-1}}$

The chart is showing an increasing carbon storage and this is clearly because of the increasing plants and their growth. The only counterpoint is that the detection found more potential storage in 2.4. This can result from either errors in detection (which is unlikely) or minor aberrations in forming the heat map in QGIS. On the other Hand, 2.4 has 298 perceptions more than 1.4 so either the settings in Blender's particle map are different to or the robot might have driven over a certain area twice. This kind of evaluation might be a topic of further work. Anyway the carbon fixation is showing an increasing trend which is put on a par with the growth of the meadows in the sequences, except scene 4 which is showing a decrease in 4.2 which is based on the big impact of the grazing event. If this would be a real world chart, the habitat healing would be enormously fast since the grazing event has destroyed a huge amount of the meadows area and the chart is indicating that the healing would normalize the carbon fixation ability in a bit more than a year. The shadow casting walkway is a fixed construction in the scene that isn't disappearing like the other affecting objects. As the object stays in the scene, it influences the overall meadow in a sustainable way. It keeps the fixation ability lower than a healing meadow or the unaffected one. This makes clear that a one-time event or a big impact is less destructive to the fixation ability than an object that is influencing the plants over a long time.

As the shadow is reducing (and even killing) a part of the meadow, this could also be adapted to a chemical inflow or something else that could reduce the seagrass. Normally an inflow is not only a one-time event but happens over a time period. This leads to changing chemical conditions in the environment so the plants get affected

by either algae or shrivel due to unhealthy conditions for them. Even if only one part of the are is affected, the overall process of carbon fixation would be affected in a similar way as the walkway affects the area.

5.4 Conclusion

The system is running very well with the optical sensor. It is detecting the objects by their color and the gained data is collected in GIS readable format as well as hierarchically. This opens a connection between the artificial intelligence and geo sciences. As the algorithm implements parts of robotics, it can be concluded that geosciences and artifical intelligence are finding a connection on the domain of robotics.
The main geoscientific concern so far was to collect data and evaluate it afterwards. The main concern of artificial intelligence was to produce an environmental framework to understand robotics behavior. From the end point of this thesis, both can be done: Data colltection is possible with this approach and the collected data can be used in Open-EASE to evaluate the robots episodic memory. Data representation is done in both disciplines and provides a new kind of interdisciplinary mission review and analysis. As the data is collected in several ways, the exploration can be rebuild by both in their specific methods - semantic and traditional geoscientific maps. The traditional maps provide an insight to the observed scenery as all objects are printed on a map. These objects can be interacted with in the GIS and a user is able to produce statistical calculations from it to produce predictions. The AI scientists is able to use the gained data to evaluate the robots behavior and recognition states. As the robotic under water systems need to become smarter in every way, this represents a cognition level for these systems with a geoscintific domain.

6 Future Outlook

As the artificial knowledge and episodic memory of under water robotics is very small compared to other fields, a lot of work is still left. The AI is very much forward in terms of daily tasks but also needs more focus on non-daily business like exploration, monitoring and non IT-science in general to accomplish to become a new method in these fields. This work can be used as a basis to future work simulating a task that represents a rough idea of an area that needs to be examined. A learning algorithm can be used and the improvements might be tested in further work.
The idea is not only to identify objects within a simulator but to use one to train the algorithm. This can also be done by showing images of objects to the system. If such a task is done, the system can be used to learn on its own and scan the seafloor for other objects.
By equipping the robot with chemical sensors the task would result in a chemical profile with values of salinity, methane concentration, temperature or other information that the sensors are sensible to. If the optical sensor is used, a monitoring mission can be applied.
A sampler would improve the functionality a lot by taking additional samples to the pictures of a certain area of interest. This would support scientists to sample a large area instead of only one spot and as the robot would drive on its own, the tasks would be much more cost effective than driving with an open sea research vessel.

Object recognition within the simulator

As the diffusing fog and scattering objects are simulating dirt and light reducing situation, reality is always different and much more difficult to handle. In this work, the camera is only struggling with the simulated fog but real world missions struggle with image scattering that is influencing the camera signal significantly.

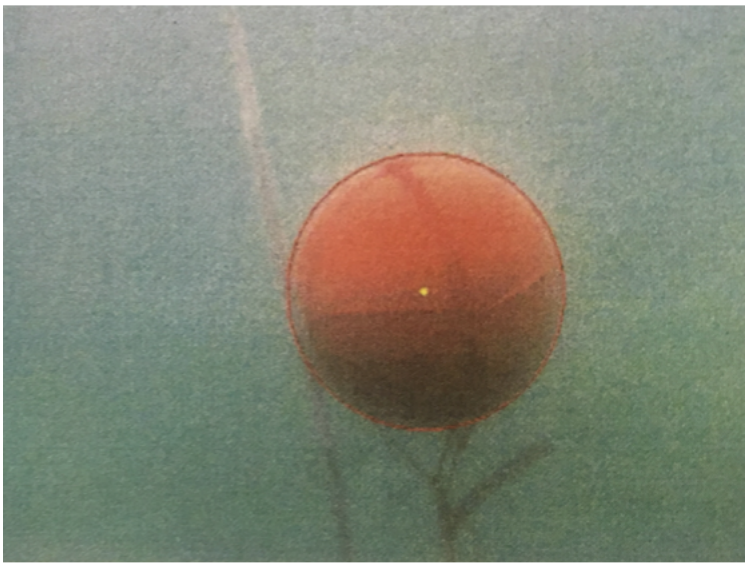

Figure 84: *Image from the RGB Camera. Its scattered because of the bad situation of the water column that surrounds the Buoy but it is fair enough to make the algorithm work properly (source: [32])*

Süto et al. [32] are using the Tesseract OCR library in OpenCV to process video footage in RGB format. The video is provided by the frontal camera and the frames are split into segments. While doing so, they use the hue and saturation values to identify a Buoy to mask it within a binary image. This step separates the foreground from the background and makes identification of objects easier. The Canny edge detection algorithm helps to outline the edges of the object. Further research allows to not only identify buoys but also other objects. The detection from projects like RoboSherlock already provides a framework for object detection and identification that could be implemented and tested in underwater situations. As Süto et al. are using a very simple shape detection from an OCR library (circle, square, triangle), RoboSherlock is using cognitive perception methods to identify i.e. kitchen tools and objects. A future approach would be to implement these techniques in an AUV system to provide additional knowledge to the recognition system and knowledge base of the semantics.

Figure 85: *The kitchen environment of the PR2 robot contains many objects that the robot is able to identify and annotate due to RoboSherlock running in the background (source: [2])*

Based on object detection and identification, the search for specific objects can be improved by using external knowledge. As one robot can profit from knowledge, gained by other missions, a semantic network of underwater knowledge would grow fast and become an important source of information in many fields of research. Exploration, monitoring and automated mapping can become a much easier, efficient and cost effective task than it is nowadays. Salvage work can be improved by training the robot for objects of interest like in fields of searching black boxes, shipwrecks and resources. OpenCV already provides libraries of object detection that is specified on learning which is applied in some testing fields of traffic counting and number plate detection. This is not often used but provides, in combination with semantic knowledge, a free and open source approach to future work of underwater exploration.

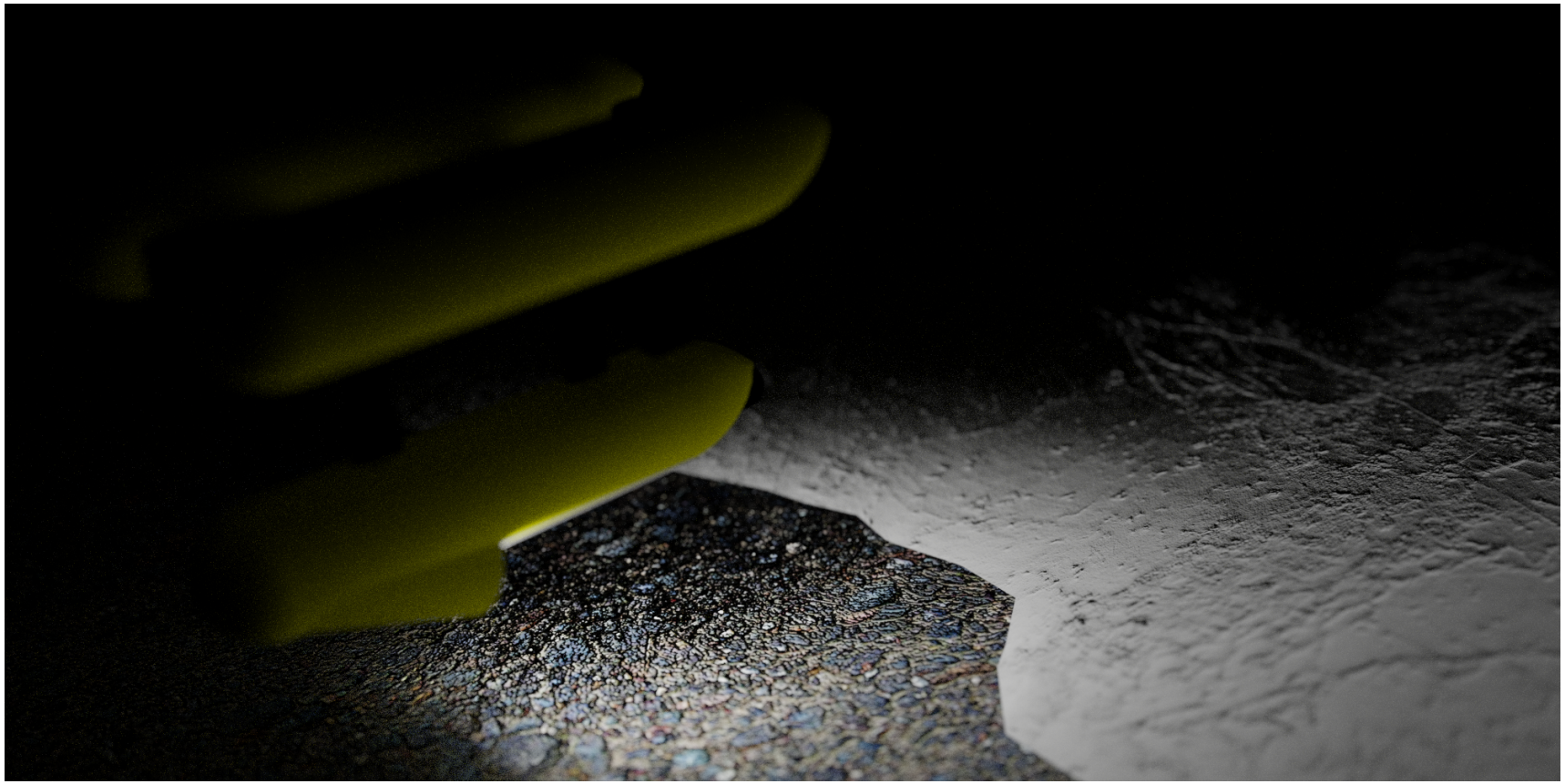

Figure 86: *Possible future field of work: Exploration for underwater resources*

Figure 87: *Another possible field of work would be the search for back boxes of crashed planes*

7 Acknowledgments

I want to thank my family and all my friends who helped me during the time of this work. Especially I want to thank my loving wife Anna who whitstand with me during this time. Thank you for being such a great support and for being the love and light of my life. Also I want to thank my Parents who always gave me feedback whenever I needed it and helped in every possible way. I really appreciate all the hopes that also my parents in law set in me and I am thankful for all their wishes.
A very big thanks goes to my mentor Fereshta Yazdani, who was always able to take care of the many problems I had during this time and the many questions I came up with. ROS, Open-EASE, programming - No matter how big the problem was, Fereshta was always able to help me. Thank you Fereshta for beeing such a great mentor. Another thanks goes to my brother Jan Winkler, who helped me with the understanding of OpenCV and semantic recording as well as to get an access to ROS. Also a big help for me was the staff of the Institute for Artificial Intelligence of Daniel Beßler and Asil Kaan Bozcuoglu, who helped me whenever I had questions regarding Open-EASE and ROS. Thank you Jan-Hendrik Worch, for being a big help in the OpenCV Part. I also appreciate the help of Venkatachalam Srinivasan with the automated control of the robot and for becoming a good friend in this time. Speaking of friends, I also want to thank Sebastian Koralewski and Mark Niehaus for always providing thought-provoking impulses. Also very important to mention are Axel Welter, Marco Tack, Julia Gallas, Avrinder Babu Sandhu and Awais Saeed, who are always a very good company for me and always come up with many brilliant ideas.
I appreciate the support of all of you and can not thank you enough for being such a great support. Thank you all for making this possible.

References

[1] Albiez, J., Joyeux, S., & Hildebrandt, M. (2010, September). Adaptive AUV mission management in under-informed situations. In OCEANS 2010 (pp. 1-10). IEEE.

[2] Bálint-Benczédi, F., Mania, P., & Beetz, M. (2016, May). Scaling perception towards autonomous object manipulation—in knowledge lies the power. In Robotics and Automation (ICRA), 2016 IEEE International Conference on (pp. 5774-5781). IEEE.

[3] Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. Scientific american, 284(5), 28-37.

[4] Blumenthal, S., Brieber, B., Huebel, N., Yazdani, F., Beetz, M., & Bruyninckx, H. (2016, October). A Case Study for Integrating Heterogeneous Knowledge Bases for Outdoor Environments. In Integrating Multiple Knowledge Representation and Reasoning Techniques in Robotics (MIRROR-16).

[5] Caress et al., 2012 D.W. Caress, D.A. Clague, J.B. Paduan, J.F. Martin, B.M. Dreyer, W.W. Chadwick Jr., A. Denny, D.S. Kelley. Repeat bathymetric surveys at 1-metre resolution of lava flows erupted at Axial Seamount in April 2011

[6] Congdon, R. A., & McComb, A. J. (1979). Productivity of Ruppia: seasonal changes and dependence on light in an Australian estuary. Aquatic Botany, 6, 121-132.

[7] Costanza, R., d'Arge, R., De Groot, R., Farber, S., Grasso, M., Hannon, B., ... & Raskin, R. G. (2016). The Value of the World's Ecosystem Services and Natural Capital (1997). The Globalization and Environment Reader, 117.

[8] Creel, L. (2003). Ripple effects: population and coastal regions (pp. 1-7). Washington, DC: Population Reference Bureau.

[9] Dupré, S., Buffet, G., Mascle, J. et al. Mar Geophys Res (2008) 29: 275. doi:10.1007/s11001-009-9063-3

[10] Foucher, J.-P., G.K. Westbrook, A. Boetius, S. Ceramicola, S. Dupré, J. Mascle, J. Mienert, O. Pfannkuche, C. Pierre, and D. Praeg. 2009. Structure and drivers of cold seep ecosystems. Oceanography 22(1):92–109, http://dx.doi.org/10.5670/oceanog.2009.11.

[11] Fox, S. E., Teichberg, M., Olsen, Y. S., Heffner, L., & Valiela, I. (2009). Restructuring of benthic communities in eutrophic estuaries: lower abundance of prey leads to trophic shifts from omnivory to grazing. Marine Ecology Progress Series, 380, 43-57.

[12] Galerne, E., 1983. Epaulard ROV used in NOAA polymetallic sul fi de research. Sea Technol- ogy 24, 40 – 42.

[13] Griffiths, G., Birch, K. G., Millard, N. W., McPhail, S. D., Stevenson, P., Pebody, M., ... & Harris, A. (2000, January). Oceanographic surveys with a 50 hour endurance autonomous underwater vehicle. In Offshore Technology Conference. Offshore Technology Conference.

[14] Heck Jnr, K. L., Hays, G., & Orth, R. J. (2003). Critical evaluation of the nursery role hypothesis for seagrass meadows. Marine Ecology Progress Series, 253, 123-136.

[15] Hu, J., Liu, S., & Zhang, R. (2016). A New Exploitation Tool of Seafloor Massive Sulfide. Thalassas: An International Journal of Marine Sciences, 32(2), 101-104.

[16] Jaynes, C., Riseman, E., & Hanson, A. (2003). Recognition and reconstruction of buildings from multiple aerial images. Computer Vision and Image Understanding, 90(1), 68-98.

[17] Kennish, M.J., Haag, S.M., Sakowicz, G.P., Tidd, R.A., 2004. Sidescan sonar imaging of subtidal benthic habitats in the Mullica River – Great Bay Estuarine System. Journal of Coastal Research 45, 227 – 240

[18] Y. Marcon, H. Sahling, C. Borowski, C. dos Santos Ferreira, J. Thal, G. Bohrmann, Megafaunal distribution and assessment of total methane and sulfide consumption by mussel beds at Menez Gwen hydrothermal vent, based on geo-referenced photomosaics, Deep Sea Research Part I: Oceanographic Research Papers, Volume 75, May 2013, Pages 93-109, ISSN 0967-0637, http://dx.doi.org/10.1016/j.dsr.2013.01.008.

[19] McKenzie, L. J. (2003). Guidelines for the rapid assessment and mapping of tropical seagrass habitats. Cairns: QFS, NFC.

[20] Elizabeth Mcleod, Gail L Chmura, Steven Bouillon, Rodney Salm, Mats Björk, Carlos M Duarte, Catherine E Lovelock, William H Schlesinger, and Brian R Silliman (2011). A blueprint for blue carbon: toward an improved understanding of the role of vegetated coastal habitats in sequestering CO_2, Front Ecol Environ 2011; 9(10): 552–560, doi:10.1890/110004 (published online 20 Jun 2011)

[21] McRoy, C. P. (1974). Seagrass productivity: carbon uptake experiments in eelgrass, Zostera marina. Aquaculture, 4, 131-137.

[22] Nogueras-Iso, J., Zarazaga-Soria, F. J., & Muro-Medrano, P. R. (2005). Geographic information metadata for spatial data infrastructures. Resources, Interoperability and Information Retrieval.

[23] Orth, R. J., Carruthers, T. J., Dennison, W. C., Duarte, C. M., Fourqurean, J. W., Heck, K. L., ... & Short, F. T. (2006). A global crisis for seagrass ecosystems. Bioscience, 56(12), 987-996.

[24] Pangercic, D., Pitzer, B., Tenorth, M., & Beetz, M. (2012, October). Semantic object maps for robotic housework-representation, acquisition and use. In Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on (pp. 4644-4651). IEEE.

[25] Prats, M.; Perez, J.; Fernandez, J.J.; Sanz, P.J., "An open source tool for simulation and supervision of underwater intervention missions", 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2577-2582, 7-12 Oct. 2012

[26] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

[27] Raskin, R. G., & Pan, M. J. (2005). Knowledge representation in the semantic web for Earth and environmental terminology (SWEET). Computers & geosciences, 31(9), 1119-1125.

[28] Roelfsema, C. M., Phinn, S. R., Udy, N., & Maxwell, P. (2009). An integrated field and remote sensing approach for mapping seagrass cover, Moreton Bay, Australia. Journal of Spatial Science, 54(1), 45-62.

[29] Ruhl et al., 2013 H.A. Ruhl, et al. RRS Discovery Cruise 377 & 378. Autonomous ecological surveying of the abyss: understanding mesoscale spatial heterogeneity at the Porcupine Abyssal Plain. National Oceanography Centre Cruise Report No. 23, Southampton, UK (2013) (73 pp.)

[30] Sandig, J. D. E., Somoba, R. M., Concepcion, M. B., & Gerardo, B. D. (2013). Mining Online GIS for Crime Rate and Models based on Frequent Pattern Analysis. In Proceedings of the World Congress on Engineering and Computer Science (Vol. 2, pp. 23-27).

[31] Srinivasan, V. (2016, November). Robot Control For Underwater Robotic Systems, Master's Thesis, Universität Bremen

[32] Sütő, B., Dóczi, R., Kalló, J., Takács, B., Várkonyi, T. A., Haidegger, T., & Kozlovszky, M. (2015, November). HSV color space based buoy detection module for autonomous underwater vehicles. In Computational Intelligence and Informatics (CINTI), 2015 16th IEEE International Symposium on (pp. 329-332). IEEE.

[33] Tenorth, M., & Beetz, M. (2013). KnowRob: A knowledge processing infrastructure for cognition-enabled robots. The International Journal of Robotics Research, 32(5), 566-590.

[34] Tenorth, M., & Beetz, M. (2009, October). KnowRob—knowledge processing for autonomous personal robots. In Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on (pp. 4261-4266). IEEE.

[35] Jamie K.S. Wagner, Molly H. McEntee, Laura L. Brothers, Christopher R. German, Carl L. Kaiser, Dana R. Yoerger, Cindy Lee Van Dover, Cold-seep habitat mapping: High-resolution spatial characterization of the Blake Ridge Diapir seep field, Deep Sea Research Part II: Topical Studies in Oceanography, Volume 92, August 2013, Pages 183-188, ISSN 0967-0645, http://dx.doi.org/10.1016/j.dsr2.2013.02.008.

[36] West, J. A., Calumpong, H. P., Martin, G., & van Gaever, S. (2016). . Kelp Forests and Seagrass Meadows.

[37] Wheeler et al., 2005 A.J. Wheeler, B.J. Bett, D.G. Masson, D. Mayor. The impact of demersal trawling on North East Atlantic deepwater coral habitats: the case of the Darwin Mounds, United Kingdom, in: P.W. Barnes, J.P. Thomas (Eds.), Benthic habitats and the effects of fishing, American Fisheries Society Symposium, 41 (2005) Bethesda, (890 pp.)

[38] Williams, D.P. Intel Serv Robotics (2012) 5: 33. doi:10.1007/s11370-011-0102-y

[39] Winters, G., Edelist, D., Shem-Tov, R., Beer, S., & Rilov, G. (2016). A low cost field-survey method for mapping seagrasses and their potential threats: an example from the northern Gulf of Aqaba, Red Sea. Aquatic Conservation: Marine and Freshwater Ecosystems.

[40] Wynn, R.B., Bett, B.J., Evans, A.J., Grif fi ths, G., Huvenne, V.A.I., Jones, A.R., Palmer, M.R., Dove, D., Howe, J.A., Boyd, T.J., MAREMAP partners,

2012. Investigating the feasibility of utilizing AUV and Glider technology for mapping and monitoring of the UK MPA network. Final Report for Defra Project MB0118. National Oceanography Centre, Southampton (244 pp.).

[41] Wynn, R. B., Huvenne, V. A., Le Bas, T. P., Murton, B. J., Connelly, D. P., Bett, B. J., ... & Sumner, E. J. (2014). Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience. Marine Geology, 352, 451-468.

[42] Yazdani, F., Brieber, B., & Beetz, M. (2016). Cognition-enabled robot control for mixed human-robot rescue teams. In Intelligent Autonomous Systems 13 (pp. 1357-1369). Springer International Publishing.

[43] Yoerger et al., 2007 D.R. Yoerger, M. Jakuba, A.M. Bradley. Techniques for deep sea near-bottom survey using an autonomous underwater vehicle. International Journal of Robotics Research, 26 (2007), pp. 41–54

[44] Documentation to rosrun and rosbash package (http://wiki.ros.org/rosbash#rosrun (last check: 03/14/17 3:38pm))

[45] Sedimentary Structures (http://dawnssedstrat.blogspot.de/2013/01/sedimentary-structures-part-2.html, (last check: 01/31/17 3:24pm))

[46] Semrec source for hierarchical cognition recording (https://github.com/code-iai/semrec (last check: 03/10/17 3:05pm))

[47] Smithsonian National Museum of Natural History (http://ocean.si.edu/seagrass-and-seagrass-beds (Last check: 03/10/17 12:30pm))

List of Tables

List of Figures

Image Sources

Figure 2: *http://www.irs.uji.es/uwsim/wiki/index.php?title=File:PipeFollowing.png (03/11/17 12:55pm)*
Figure 4: *Source Mesh: http://www.blendswap.com/blends/view/83277 (03/14/17 9:37pm)*
Figure 7: *http://eatlas.org.au/ts/seagrass (03/11/17 1:17pm)*
Figure 5: *http://www.geolsoc.org.uk/Geology-Career-Pathways/University/During-your-degr Fieldwork/Undergraduate-Mapping-Projects (03/11/17 1:30pm)*
Figure 9: *http://www.dep.state.fl.us/coastal/habitats/seagrass/management/mapping.htm (03/11/17 1:20pm)*
Figure 17: *http://www.seagrasswatch.org/seagrass.htmlandhttp://www.npr.org/2014/07/15/330440072/underwater-meadows-might-serve-as-antacid-for-acid-seas (03/11/17 1:38pm)*
Figure 18: *https://samwagura.wordpress.com/2015/02/02/friends-foe/ and http://jalopnik.com/you-need-a-spark-plug-to-smash-a-car-window-1527157089 (03/12/17 3:36pm)*
Figure 43: *Mesh source: http://www.cadnav.com/3d-models/model-29325.html (03/12/17 3:38pm)*
Figure 44: *Left image: https://www.lincolnshire.gov.uk/coastalcountrypark/about/history/settling-down/flooding-and-defences/flood-defences-after-1953/121281.article (03/12/17 3:40pm)*
Figure 56: *https://upload.wikimedia.org/wikipedia/commons/0/00/HSV_color_solid_cone_chroma_gray.png (03/12/17 3:45pm)*
Figure 82: *http://ian.umces.edu/blog/2013/04/05/great-barrier-reef-literacy/ (03/11/17 12:29pm)*

Used Software

QGIS https://www.qgis.org/de/site/

Blender 2.77a https://www.blender.org/

ROS Indigo http://wiki.ros.org/indigo

UWSim http://www.irs.uji.es/uwsim/

Semrec https://github.com/code-iai/semrec

Knowrob http://www.knowrob.org/

CrazyBump http://www.crazybump.com/

GIMP https://www.gimp.org/

Protégé http://protege.stanford.edu/

yEd http://www.yworks.com/products/yed

Appendix

The appendix can be found in the containing Double Layer DVD. It contains this thesis as PDF as well as LyX/LaTeX Code, all images in full resolution and the complete source code of the developed methods and algorithms. Also the produced results, maps and statistics can be found on the DVD.
The complete content is also available on GitHub and can be accessed via:
https://github.com/owinklerhb/MasterThesisContent.git