

M.Sc. Thesis

Master of Science in Engineering



Development and Application of a Seabed Mosaic System for ROVs

Zeina Antar

Kongens Lyngby 2020



DTU Aqua

Department of Aquatic Sciences and Technology

Technical University of Denmark

Kemitorvet

Building 202

2800 Kongens Lyngby, Denmark

Phone +45 35993300

aqua@aqua.dtu.dk

www.aqua.dtu.dk

Abstract

Underwater research is very important for several scientific and societal sectors. A globally important sector is the Blue Economy, which is based on all economic activities related to ocean and seas, has been expanding very rapidly during the past years. This expansion is normally happening along coastal lines where most human activities are being developed. Lack of information on the seabed and coastal areas leads to the need of seabed mapping in order to monitor aquatic environments, for resource exploration and exploitation and, how the climate change is impacting the underwater world. The introduction of remotely operated vehicles (ROVs) and autonomous operated vehicles (AUVs) increased the capability of inspecting and undertaking exploration tasks of data gathering from inaccessible areas. These factors have naturally led to the need for efficient and robust methods of data processing obtained by underwater robots. When it comes to processing vast amount of optical sensor data, image mosaicking is a useful tool for obtaining a high-resolution visual representation of the seabed. This thesis presents, a seabed mosaicking system that utilizes the extraction of invariant features on ROVs footage with the goal to create high resolution mosaic maps of the seabed. A color correction procedure and a bird-eye-view (top-down) conversion procedures, are also implemented. The Google Vision-machine learning API has also been integrated in order to detect objects and labels on the mosaics and show how new technological platforms can improve inspection of areas of interest.

Keywords: ROV, Seabed Mapping, Video Mosaicking, Google Vision

Preface

This thesis, worth of 35 ECTs, was prepared at the Department of DTU Aqua at the Technical University of Denmark (DTU) as part of fulfillment of the requirements for acquiring a Master's in Aquatic Sciences and Technology. It was prepared during autumn 2019 and winter 2020 and has been supervised by Professor Patrizio Mariani.

The thesis deals with the development and application of a mosaic system in Python 2.7 in combination with the computer vision library. The goal is to develop seabed mosaics from video input captured by a remotely operated vehicle (ROV) around coastal areas in Copenhagen, as well as recreational diving videos from Cyprus. It consists of an introduction to the topic including a theoretical background, a general discussion and a conclusion.



Kongens Lyngby 2020, March 9, 2020

Zeina Antar

Acknowledgements

I would like to thank my supervisor and Professor, Patrizio Mariani for giving me the opportunity to challenge myself with an idea that in the beginning seemed unreachable.

I would also like to thank deeply, Senior researcher. emeritus Bo Lundgren, for his guidance on every detail during this project by providing me his vast knowledge and assistance.

Special thanks to Fletcher Thomson as well, for teaching me how to be a proper mission planer and for his programming skills input.

For all the people who believe in me, Thank you!

“Nature’s great and wonderful power is more demonstrated in the sea than on the land”

Jason A. Shulman

Table of Contents

Abstract	3
Preface.....	4
Acknowledgements.....	5
List of Figures.....	8
List of Tables	9
1 Introduction.....	10
1.1 Thesis Goals.....	11
1.2 Thesis Outline.....	11
1.3 Software Access.....	11
2 Image Transformations	12
2.1 Pixel Coordinates.....	12
2.2 Homogeneous Coordinates	13
2.2 2D Transformations.....	13
2.4 Warping.....	15
2.5 Homography.....	16
2.4 Discussion.....	17
3 Feature Detection and Matching	18
3.1 Feature/Keypoint Detection.....	18
3.2 Feature/ Keypoint description	20
3.3 Feature/ Keypoint Matching.....	20
3.3.1 Discussion	20
3.4 Scale Invariant Feature transform (SIFT).....	21
3.4.1 SIFT in OpenCV and Python.....	23
3.4.2 Color correction.....	24
3.4.3 Matching Strategy	26
3.4.4 Feature match verification and densification – RANSAC.....	28
3.4.5 Finding Homography.....	29

3.4.6	Warp Perspective.....	29
4	Image Alignment.....	30
4.1	Caveats.....	31
4.2	Top- Down View/ Bird View	32
4.3	Discussion.....	33
5	Implementation	34
5.1	ROS & BlueROV2	35
5.2	Mosaic system execution on footage	36
5.2.1	Results	36
5.2.2	Google Vision	40
6	Conclusion.....	42
	Appendix A	43
	Bibliography.....	45

List of Figures

Figure 1: Pixel coordinates (x, y) of a 2D Image.	12
Figure 2: Basic 2D planar transformations on a Cartesian coordinate plane	14
Figure 3: Perspective distortion effect on an image.....	16
Figure 4: If OL is a point (x1,y1) in the first frame and If OC ,(x2,y2), are the coordinates of the same physical point in the second frame, then, the Homography matrix transforms them to the same image coordinate system. (Yin, 2015).....	17
Figure 5: Pipeline of a mosaic procedure on a video sequence.	18
Figure 6: Frame 0 and Frame 195 are prone to matching. How can a system detect certain features in order to establish alignment between them? (Frames were extracted by an ROV footage from Copenhagen harbor.).....	18
Figure 7: Image pairs accompanied with pixel-patches(below) depicting areas of interest. Some patches can be localized or matched better than others. Texture-less patches are nearly impossible to localize while patches with large contrast changes (gradients) are easier to localize.	19
Figure 8: Gaussian pyramids breaking down an image into successively smaller groups of pixels and down sampled by a factor of 2.	21
Figure 9: On the left; Each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images. Adjacent images are subtracted producing the difference of Gaussian images. On the right; maxima and minima of the difference of Gaussian images. There are detected by comparing a pixel(X) to its 26 neighbors in 3 x 3 regions marked with circles. (Lowe, 2004)	22
Figure 10: Computed keypoint descriptors by SIFT. On the left; image gradient magnitudes and orientations are sampled around the keypoint location. On the right; small arrows indicate the orientation of the keypoint descriptors whose coordinates are rotated relative to the keypoint orientation. (Lowe, 2004).....	23
Figure 11: A great example of SIFT features when computed in the Python 2.7 and OpenCV3 environment. The blue colored circles indicate the locations of features on the images. Notable, some of these features can be seen on the same locations of both images.....	24
Figure 12: Features detected by SIFT in Python 2.7 and OpenCV3 after color correcting underwater images with CLAHE. Notably, the number of features detected increases greatly because of the increase in contrast.	25
Figure 13: Too many matches found by the SIFT algorithm between the two frames.	26
Figure 14: SIFT matches when Lowe's distance ratio test is applied. (Brown & Lowe, 2007).....	27
Figure 15: An Illustration of RANSAC inliers (blue) and outliers (red) and which are a best fit to a linear model.	28

Figure 16: Algorithm I: Image Alignment algorithm of two images using SIFT, Bruforce matches and the Homography matrix.	30
Figure 17: Top Images; Frame 0 is warped in Frame's 195 perspective using the homography matrix that was computed from the SIFT features detected between them. Bottom Image; Final alignment result between Frame195 and Frame 0.	31
Figure 18: Top-down/ Bird Eye View transformation pipeline	32
Figure 19: Top-down or Bird Eye View transformation of frame 0	33
Figure 20: Algorithm II: Final algorithm of the mosaic system.	34
Figure 21: Illustration of the image alignment process followed by the mosaic system. First, the detection and matching of features takes place between Frame 0 and Frame 1. Then, possible image alignments are found by the Homography matrix. Finally, warp and align Frame 1 to the perspective of Frame 0. Continue to next Frame 2 and repeat the process by aligning the next frames to the previous mosaic.	35
Figure 22: BlueROV2, the remotely operated vehicle whose footage was used in this thesis.	36
Figure 23 Seagrass Seabed- mosaic generated with color enhancement and Top down view.	37
Figure 24 Seabed mosaic generated from BlueRoV2 footage acquired from Copenhagen harbor and consists of 108 frames.	37
Figure 25: Low resolution mosaics generated from ROV footage in Skovshovde Havn due to the green reflection from the waters.	38
Figure 26: Seabed mosaic generated by go-pro footage (512 frames) that was mounted on the ROV in Skovshovde Havn. The black arrows indicate the trajectory of the frame alignment that took place by the mosaic system.	39
Figure 27: Google Vision API applied on the Sea grass video frame which identified Confidence scores of objects on the mosaic as well as labels of the scenery such as: Nature, Green, Underwater, Organism, Marine Biology, Water, Algae, Plant, Seaweed, Chlorophyta.	40
Figure 28 Google Vision API applied on a generated underwater mosaic which identified labels such as: Mineral, Rock, Turquoise, Aqua, Geology, Organism, Coral reed, Marine biology.	41
Figure 29: Underwater mosaic developed by a go-pro diving video from Cyprus.	43
Figure 30: Underwater mosaic developed by a go-pro diving video from Cyprus.	44

List of Tables

Table 1: Matrix representation of 2D transformations stating their degrees of freedom (DOM) and the geometric primitives that they preserve.	15
---	----

1 Introduction

Human societies have always been astonished by the great power of oceans from impacting the climate, to providing millions of people with resources. Therefore, the oceans are from the most important areas for research. However, to understand the role of the ocean in the Earth System and the sustainable provision of its goods and services, more research and reliable data is needed.

Climate change is impacting the oceans at all scales and with unprecedented rate of change. This will also most likely affect ocean ecosystems as well as the ocean economy which many coastal areas are relying upon. In order to be one step ahead of these changes, it is required to have a better understanding of how these are impacting the ocean environment.

Submarine instruments, modern ships, satellite gliders and underwater and airborne robots, pushes the limits of underwater research to a degree never seen before. Recent systems, like underwater robots, can reach areas that are inaccessible to humans, gather visual data and get high-quality mappings of areas of interest.

In these contexts, underwater mosaics is an important possibility to provide better observations in the underwater environment. Image processing algorithms have been used to create suitable representations of the ocean floor and constitutes an important tool for seabed exploration, improving visualization and navigation of underwater medium. (Rzhanov, et al., 2000) (Elibol, et al., 2016) (Laranjeira, et al., 2016) (Nunes, et al., 2018) In the past few years a considerable research effort has been performed to increase the autonomy of underwater vehicles. (Gracias, et al., 2002). A completely automated mosaic system has been developed. (Leone, et al., 2006) Mosaic systems have been used in other scientific fields such as Astronomical research. A powerful image processing Mosaic system for the manipulation of digital astronomical image data. (Varosi & Gezari, 1992) Additionally, mosaic systems are also being used in order to make comparison of human impact on Earth's landscapes. (Joy, 1997) Underwater mosaicking is also used to generate archaeological mosaics with input of images from a remote operated vehicle. (Bellavia, et al., 2006)

These studies have been an inspiration for the present thesis work which is mostly based on the fundamental mosaic backbone theory. The major difference in this thesis is that the mosaics will be generated by video sequences acquired by a remotely operated vehicle and not by the acquisition of large amounts of images as the

traditional systems do. Additionally, the targeted quality of the seabed mosaics is that of a high resolution meaning that color correction will be applied. Most importantly, the present mosaic system is developed in a way that is independent; no other parameters are taken into consideration except of the visual input. The ROV videos are, of course, prioritized and the mosaic system's performance will be evaluated on those.

1.1 Thesis Goals

The purpose of this thesis is to develop a mosaic system which is based on extracting invariant features, and processes video sequences acquired by a remotely operated vehicle (ROV) with the aim on creating underwater seabed-mosaics.

1.2 Thesis Outline

Chapter 2 gives a brief theoretical background of 2D image transformations to prepare the reader for the theory used for image alignment, Chapter 3 introduces the concepts of feature detection, matching and preprocessing of the visual data before alignment. Chapter 4 gives an overview of how the process of creating mosaics was implemented on two images summing up the requirements for a mosaicking system. Capitalizing on the insights developed in Chapter 3 and 4, the implementation of the system is presented in Chapter 5. Chapter 6 gives a brief discussion of the results and presents how the Google Application Programming Interface (API) can be used on underwater mosaics in order to generate object recognition. Finally, Chapter 7 discuss the results and future research directions.

1.3 Software Access

The mosaic system that has been developed in this thesis can be found at <https://github.com/starfy7/UnderwaterMosaicSystem.git> accompanied with a short documentation of the requirements it needs and how the system should be operated.

2 Image Transformations

This chapter addresses the manipulations that are applied on the collected ROV videos in order to deliver geometrical frames that are suitable for mosaicking. Described is, the geometry of a scene when certain transformations are applied, in this case, 3D to 2D images. Transformations are an important concept when it comes to manipulating images. This term is called warping in the Computer Vision field and will be explained in Section 2.2.

2.1 Pixel Coordinates

When addressing images at a computerized level, points, lines, and planes are used to illustrate geometric transformations that project 3D coordinates into 2D. The form that a 2D images takes is; as a pixel's array (one-dimensional array), where each pixel's position in the array is denoted by the x value plus that y value multiplied by the width of the image. Hence, for any Cartesian point in two dimension, 2D points, can be represented by the parameters x and y coordinates, $\mathbf{x} = (x, y) \in R^2$, or alternatively,

Pixel Coordinates

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \quad (1)$$

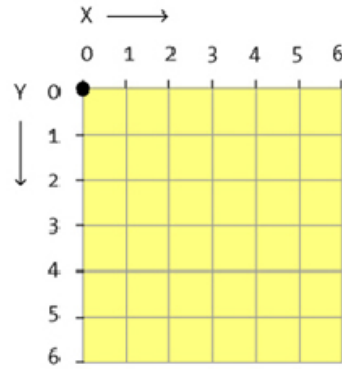


Figure 1: Pixel coordinates (x, y) of a 2D Image.

Where, x and y coordinates should start from the top left of the images, see Figure 1. X is the column number y is the row number.

2.2 Homogeneous Coordinates

As mentioned before, pixels can be represented as (x, y) coordinates, but this form is not enough to simplify various calculations of graphics and transforms in projective. Therefore, August Ferdinand Möbius (Rogers, 1976) introduced the concept of homogeneous coordinates and are a fine extension of standard three-dimensional vectors and are fundamental when it comes to cameras and how the cameras are looking at the real world. (Bez, 2003)

These coordinates are a way of representing N -dimensional coordinates with $N+1$ numbers. In order to convert 2D coordinates to homogenous, an additional variable is added into existing coordinates, denoted as

$$\text{Homogeneous Coordinates: } (x, y, w) \quad (2)$$

where w is a non-zero real number representing the scaling transformation for the 2D coordinate (i.e. an extra dimension). When w increases, the coordinate expands (scale up); when it decreases the coordinates shrink down (scale down). The relation homogeneous coordinates on projective geometry (projective transformations) and this project, is explained at section 2.5.

2.2 2D Transformations

So far, it was mentioned how images can be translated into pixels and how an extra dimension can be added on these pixels so that image manipulations (transformations) are possible. In Figure 2, some of these planar transformations that can be performed on pixels in the 2D Cartesian coordinates are represented with a list of explanations below.

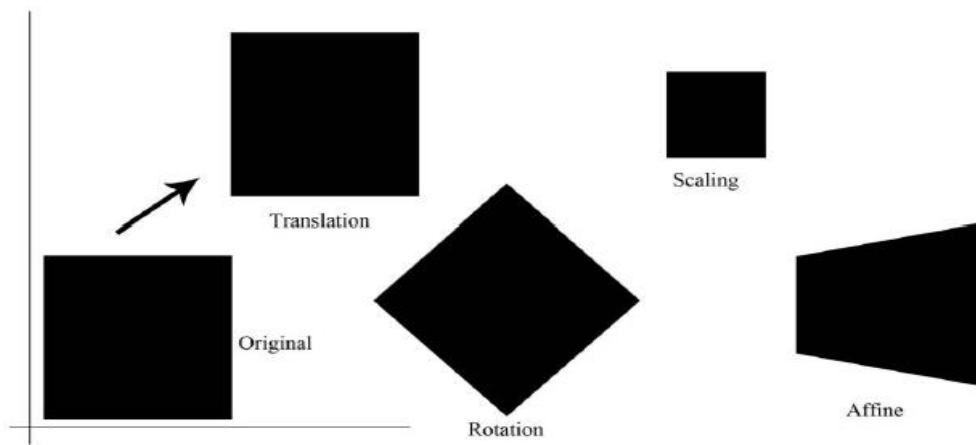


Figure 2: Basic 2D planar transformations on a Cartesian coordinate plane

Translation. Shifts any given input image by a vector to another point in the 2D Cartesian plane.






Rotation (Similarity). Rotates any given input image around the centermost point of the image.

Scaling (Euclidean). Affects size of any given input image by a constant factor for all image pixels.

Affine. When this transformation is applied on any given input image, it preserves the parallel lines and distance ratios between points on a straight line in a source image.

Projective. Also known as perspective transform, projection or, homography, operates on homogeneous coordinates, (x, y, w) , and preserves straight lines. An example is when a position in space is associated with a line from it to a fixed point (center of projection), this point is mapped by a plane by finding the point of intersection of that plane and the line.

Table 1: Matrix representation of 2D transformations stating their degrees of freedom (DOM) and the geometric primitives that they preserve.

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	

2.4 Warping

Combining the transformations listed at Table 1, gives rise to the concept called Warping in the Computer Vision field. Transformations are the manipulations that images can undergo geometrically. Since, images are represented as coordinates or arrays, mathematically, the transformations applied on them take the form of matrices. Each transformation preserves a different type of geometric primitive and have different degrees of freedom (DOF) i.e., the number of unknown numbers that they can solve. In order to apply these transformations on the images, matrix multiplication is used on the x and y coordinates of each pixel in an image.

In this project, it is assumed that consecutive frames in an underwater video footage will overlap. Certain transformations, while frames are changing, will be applied on the pixels. Considering the case of the ROV footage; the drone moves upwards/downwards with different view angle in each frame. Therefore, it is essential to have a “tool” that remaps the pixel coordinates from one image to another. All the mentioned transformations can be grouped in a matrix called Homography (H), also stated as projective transformation/perspective transformation. The homography matrix can essentially be used to map an image on a plane to another image with a slight distortion such as viewing a billboard image from an angle, see Figure 3.

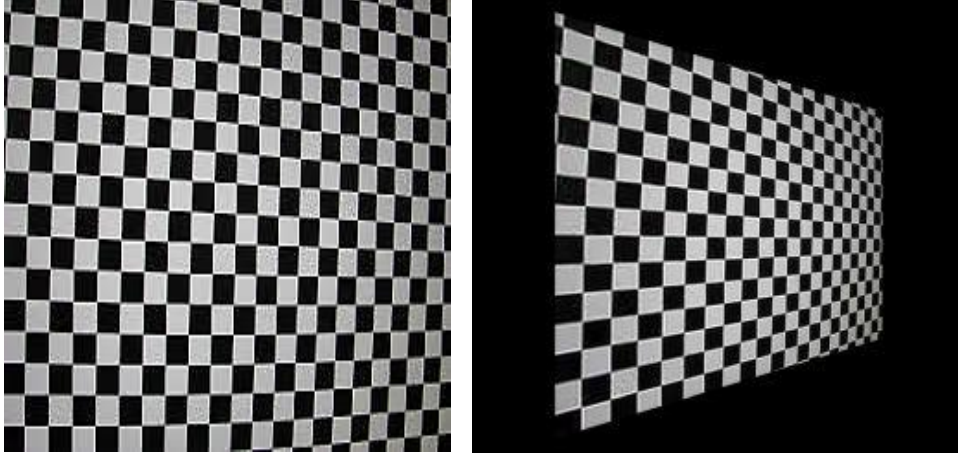


Figure 3: Distortion effect on an image.

2.5 Homography

The Homography 3x3 matrix, is the source of image alignment techniques shown below, and is used mostly for perspective transformations on images under some assumptions.

$$\mathbf{H} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \quad (3)$$

Two images of a scene are related by a Homography matrix if:

- 1 The two images are planar
- 2 The two images were acquired by rotating the camera about its optical axis.

If (x_1, y_1) is a point in the first image and (x_2, y_2) are the coordinates of the same physical point in the second images. Then, the Homography (\mathbf{H}), relates them in the following way:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{12} & h_{13} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \quad (4)$$

Therefore, if the Homography between two images is known, it can be applied to all the pixels of one image to obtain a warped image that is aligned with the second image, see Figure 4.

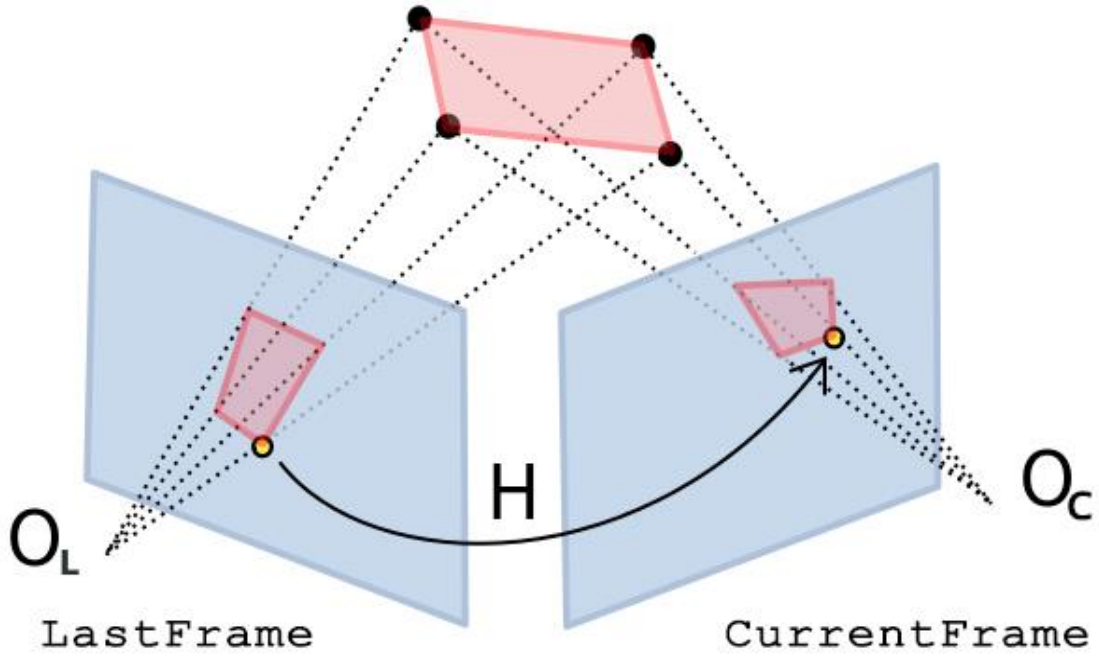


Figure 4: If O_L is a point (x_1, y_1) in the first frame and If O_C , (x_2, y_2) , are the coordinates of the same physical point in the second frame, then, the Homography matrix transforms them to the same image coordinate system. (Yin, 2015)

2.4 Discussion

Taking into consideration all these different transformation matrices, some assumptions can be made on how the video frames will be transformed and aligned in order to create a mosaic. One of the assumptions is, the seabed must be considered as a flat plane surface since the re-projections will happen between 3D to 2D images using standard video cameras. Since the system will deal with consecutive video frames, it is assumed that there will always be an overlapping region between the frames so the Homography matrix can be applied to align the transformations between them.

The following chapter describes an invariant feature-based strategy that leads to finding the Homography matrix, making image alignment possible.

3 Feature Detection and Matching

This chapter describes the theory behind feature extraction-detection and how these correspondences can be established across different frames and making their alignment possible by the mosaic system. It is important to note that most traditional approaches in mosaic development involve several images that can be ordered or unordered, finding features in each image, and then detect and match these images using an index, also called as Global Alignment (Gracias & Santos-Victor, n.d.). However, since this project focuses on the input of ROV videos, it would be storage consuming saving every video frame in order to follow these procedures. Therefore, a video processing pipeline, depicted in Figure 5, is used, whose stages are explained in this chapter.

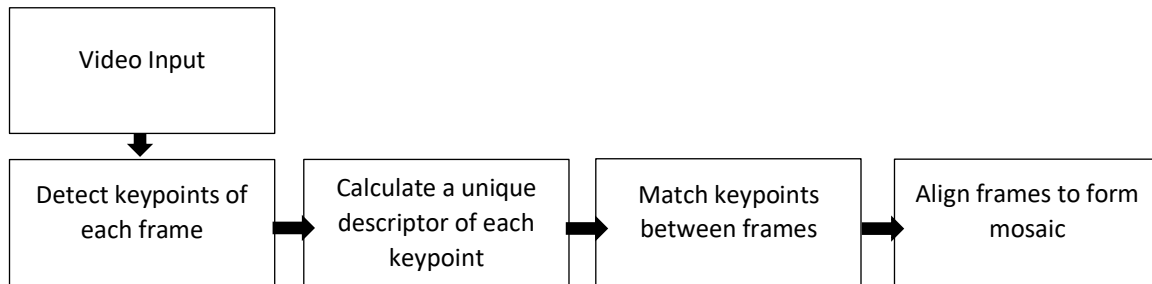


Figure 5: Pipeline of a mosaic procedure on a video sequence.

3.1 Feature/Keypoint Detection

The main question to be answered here is; considering Figure 6, how can a computerized system recognize features with the purpose of image alignment? (Brown & Lowe, 2003)

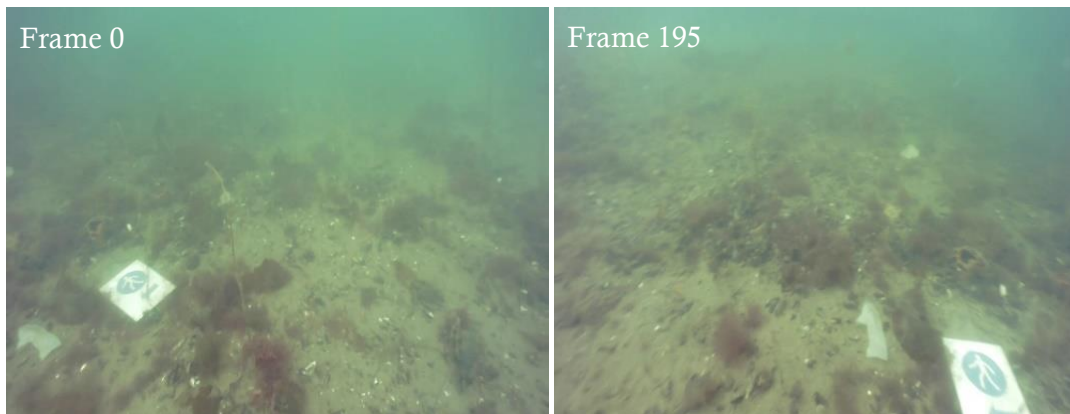


Figure 6: Frame 0 and Frame 195 are prone to matching. How can a system detect certain features in order to establish alignment between them? (Frames were extracted by an ROV footage from Copenhagen harbor.)

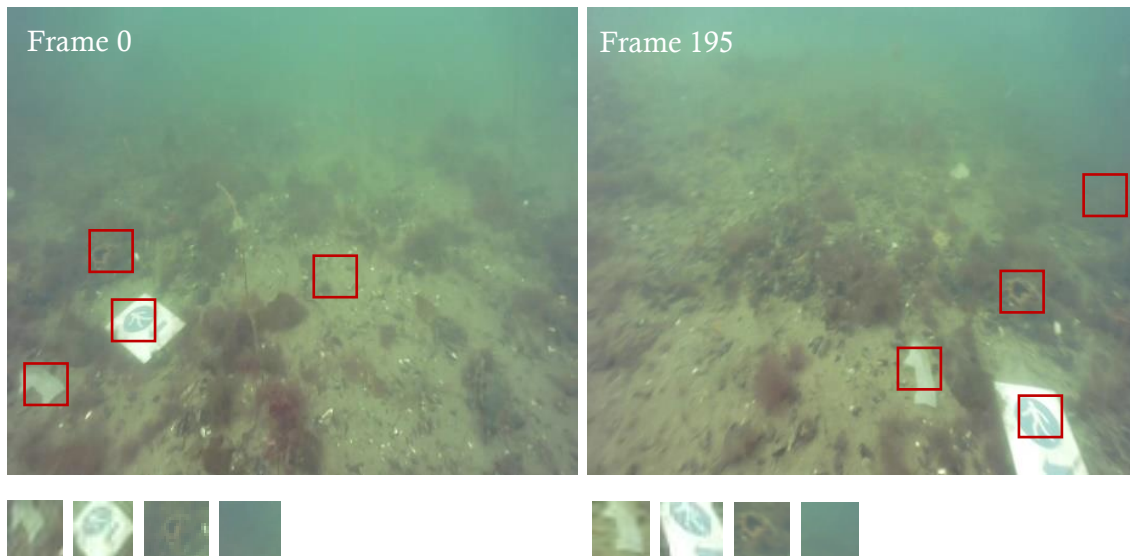


Figure 7: Image pairs accompanied with pixel-patches(below) depicting areas of interest. Some patches can be localized or matched better than others. Texture-less patches are nearly impossible to localize while patches with large contrast changes (gradients) are easier to localize.

Features are specific locations on images which can be recognized. In the underwater environment (seabed), these features can be in the form of coral peaks, objects with corners, for example, plants and rocks, see Figure 7. These localized features are called keypoint features (interest points) and are often described by the appearance of pixel patches surrounding the point locations. If a decent set of corresponding locations are found in overlapping images, it makes alignment between them possible. Additionally, such features are used to achieve object and category recognition. (Brown & Lowe, 2007) (Lowe, 1999)

Human cognitive skills can easily detect such similarities but when it comes down to bits and bytes, a system must independently detect features in all the images under consideration and then *match* features based on their local appearance.

During the feature detection (keypoint extraction) stage, every consecutive frame is searched for locations that will match the next frame. An important factor which affects the keypoint detection is, the larger the contrast (gradient) the easier to localize whereas, in locations like water or sky, features are likely to be blended in the background and therefore harder to detect.

3.2 Feature/ Keypoint description

During the feature description stage, each region around a detected keypoint's location is computed into a more compact and stable (invariant) descriptor in vector form.

Once features and their descriptors have been detected and computed from the images, the next step is to establish some preliminary feature matching between them. However, the matching part has to have an appropriate selection method that determines which corresponding features are retained for the next stage for further processing. Additionally, efficient data structures and algorithms are needed to perform this matching as quickly as possible.

3.3 Feature/ Keypoint Matching

When it comes to image alignment, the term matching means that the two sets of detected keypoints and their descriptors (for each image respectively) need to be matched. It is the process that efficiently searches for feature candidate that are in similar corresponding locations among images.

In most cases, as mentioned in section 3.1, the local appearance of features will change in orientation and scale, affected by motion blur, and even undergo affine deformations from frame to frame. Since the current system deals with ROV video footage which will most likely be affected by motion blur, as well as other factors like water turbidity, color changes due to light noise, quality of the waters etc., the capability of feature detection can be affected greatly.

3.3.1 Discussion

The fact that the local appearance of frames patches will vary, leads into considering frame preprocessing before the image alignment part. Firstly, the video frames should undergo color filtering and correction for the system to detect more features. Secondly, extracting a local scale, orientation, or affine frame estimate and then using this to resample the patch before forming the feature descriptor would be preferable.

Lastly, even after compensating for these changes, an appropriate feature detector algorithm should be chosen so as the descriptors computed will be more invariant to changes and still preserve the possibility to discriminate between different (non-corresponding) patches.

3.4 Scale Invariant Feature transform (SIFT)

In 2004, David Lowe, (Lowe, 2004), published a paper which described the development of his algorithm called Scale Invariant Feature Transform (SIFT). According to Lowe's paper, SIFT can extract highly distinctive features that are invariant to rotation, scaling and partially to change in illumination. There are other algorithms that can detect and compute features like, SURF, ORB, BRISK and some others, but for the reasons stated in the discussion above, SIFT seemed the appropriate candidate for the development of this mosaic system.

Features extracted by SIFT, are formed by computing the gradient at each pixel in a 16×16 window around the detected key point using the appropriate level of Gaussian pyramid at which the key point was detect (see Figure 8 for Gaussian pyramid).

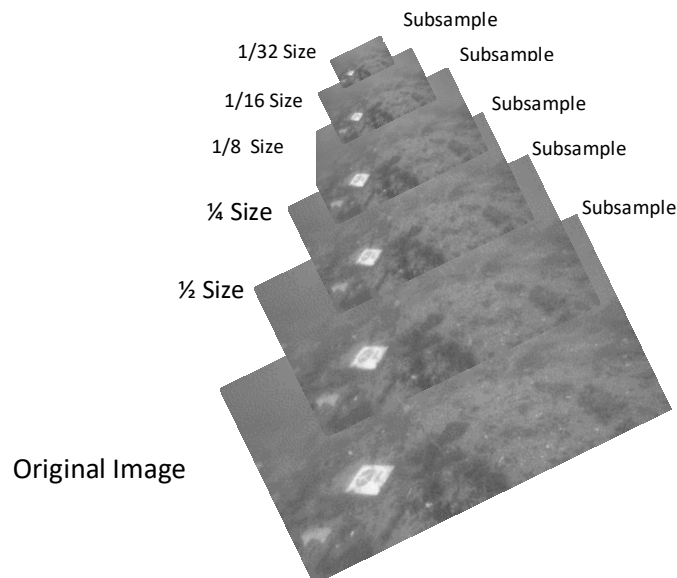


Figure 8: Gaussian pyramids breaking down an image into successively smaller groups of pixels and by down-sampling by a factor of 2.

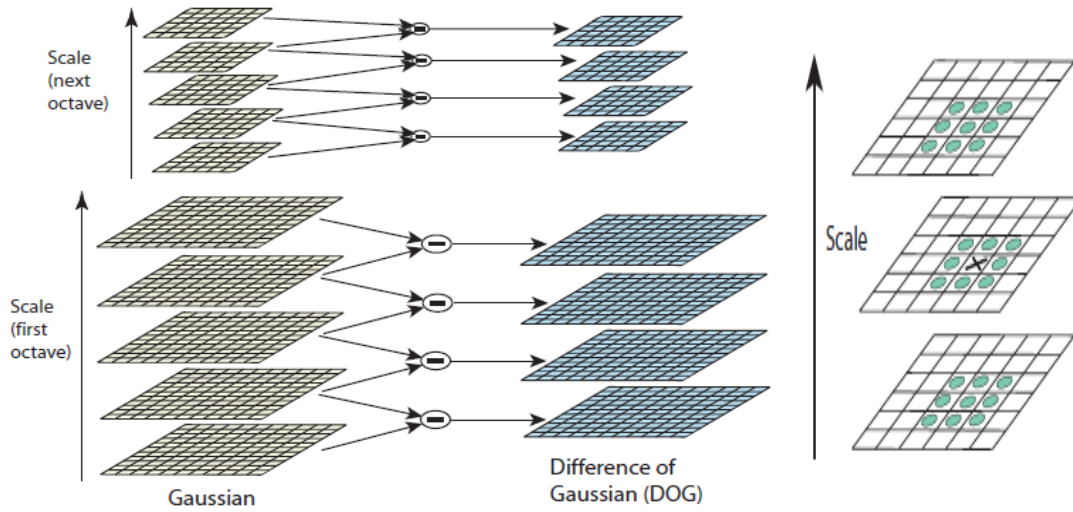


Figure 9: On the left; Each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images. Adjacent images are subtracted producing the difference of Gaussian images. On the right; maxima and minima of the difference of Gaussian images. There are detected by comparing a pixel(X) to its 26 neighbors in 3 x 3 regions marked with circles. (Lowe, 2004)

In each 4 x 4 quadrant, a gradient orientation histogram is formed by adding the weighted gradient value to one of the eight orientation histogram bins. To reduce the effects of location and dominant orientation misestimation, each of the original 256 weighted gradient magnitudes is softly added to 2 x 2 x 2 histogram bins using trilinear interpolation.

The resulting 128 non-negative values form a raw version of the SIFT descriptor vector. To reduce the effects of contrast or gain, the 128-D vector is normalized to unit length. SIFT features are located at scale space maxima/minima, (Figure 9, on the right) (Lowe, 2004), of a difference of the Gaussian pyramid. At each feature location, a characteristic scale and orientation is established. This gives similarity-invariant frame in which to make measurements. Although, simply sampling intensity values in this frame would be similarity invariant, the invariant descriptor is computed by accumulating local gradients in orientation histograms. This allows edges to shift slightly without altering the descriptor vector, giving some robustness to affine change. (Lowe, 2004) (Brown & Lowe, 2003)

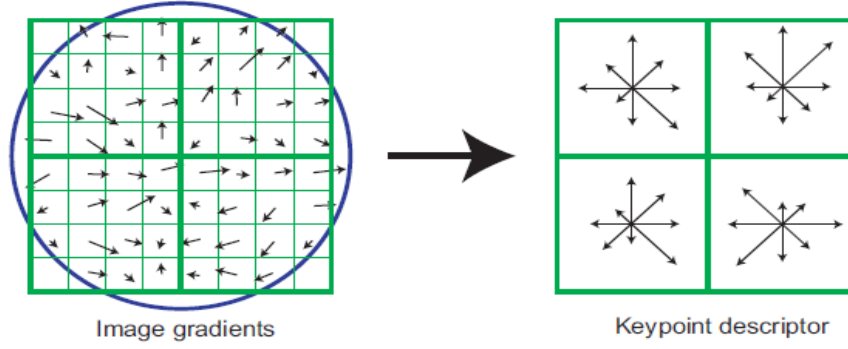


Figure 10: Computed keypoint descriptors by SIFT. On the left; image gradient magnitudes and orientations are sampled around the keypoint location. On the right; small arrows indicate the orientation of the keypoint descriptors whose coordinates are rotated relative to the keypoint orientation. (Lowe, 2004)

The method of SIFT computing the key point descriptors is depicted at Figure 10. Firstly, the image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to establish orientation invariance, the coordinates of the descriptors are rotated relative to the keypoint orientation. These are illustrated with small arrows at each sample location on the left of Figure 10. Thus, the reason why SIFT features are invariant to rotation and scale changes. (Lowe, 2004)

3.4.1 SIFT in OpenCV and Python

As mentioned in the start of this thesis, the system is developed in python 2.7 which has access to the Open Source Computer Vision (OpenCV) library (The OpenCV Reference Manual, 2019), initially created with C++. The choice of this python version is because it is compatible with the OpenCV that has access to the SIFT algorithm. The version of the OpenCV used is 3.4.12 since in the newer versions (4.0.0 +) the SIFT and SURF algorithms are patented and there's a licensing fee if one wishes to use them in a real-world application.

Figure 11 illustrates of how SIFT represents feature detection in the python environment. One can define the number of features that want to be detected on an image or between a lot of images.

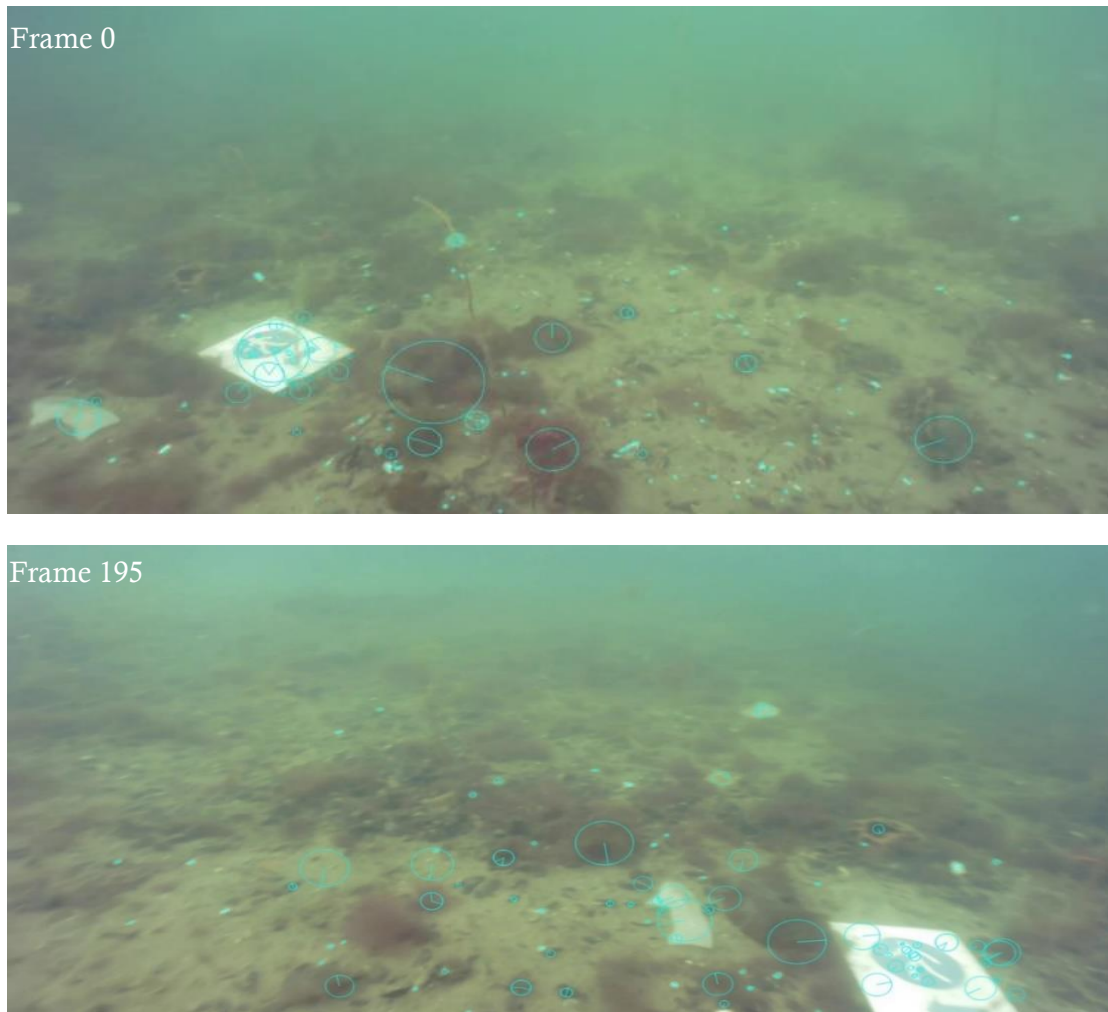


Figure 11: A great example of SIFT features when computed in the Python 2.7 and OpenCV3 environment. The blue colored circles indicate the locations of features on the images. Notable, some of these features can be seen on the same locations of both images.

3.4.2 Color correction

Underwater images are difficult to process due to illumination, changing image contrast and lack of well-defined features. In order to improve the color conditions a filter technique is applied.

The filtering method used is mostly based on a method called CLAHE adaptive enhancement. As proposed by (Leone, et al., 2006) (Garg, et al., 2018), CLAHE is an extension of histogram equalization performed on each 8 x 8 squared region in an image, setting the desired histogram shape for the tiles so the underwater images look

more natural. The results of such color enhancement can be seen at Figure 12. Notably, the SIFT algorithm detected more features on color-manipulated images than the images presented at Figure 11. Color correction can impact in a great way the computation of features and respectively the homography matrix. Therefore, it is beneficial to color process the frames before feature detection and alignment in this mosaic system and thus, increase the probability of an improved mosaic result.

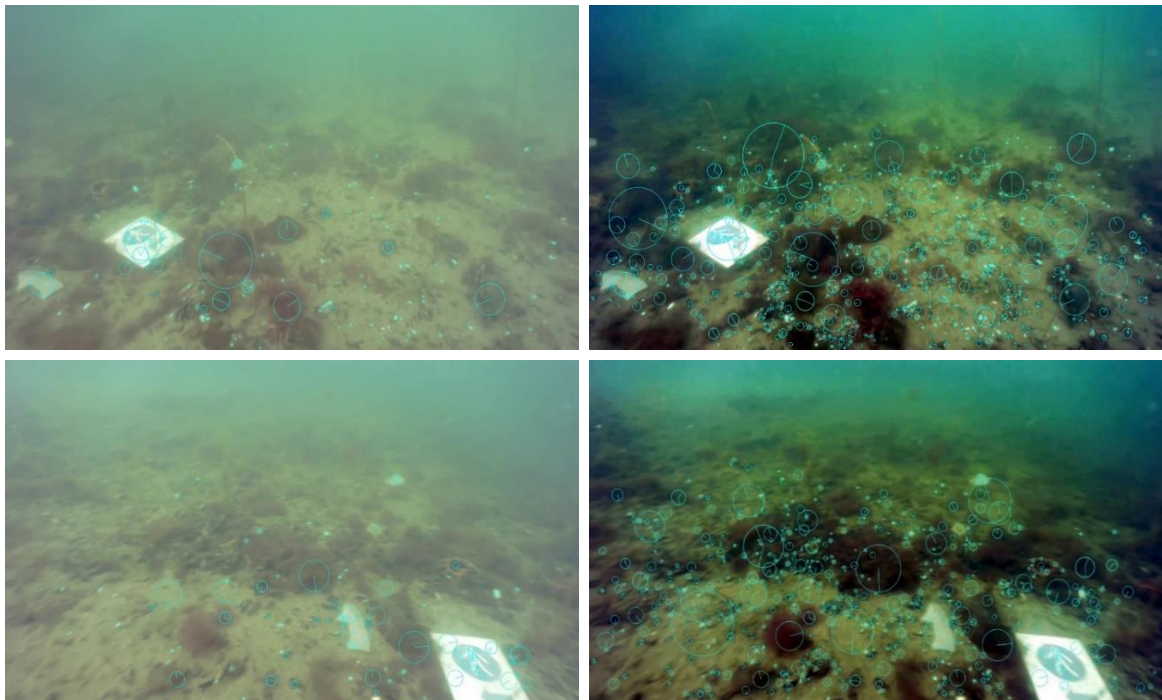


Figure 12: Features detected by SIFT in Python 2.7 and OpenCV3 after color correcting underwater images with CLAHE. Notably, the number of features detected increases greatly because of the increase in contrast.

3.4.3 Matching Strategy

After the process of SIFT detecting and computing features, a matching strategy needs to be established. Even if the ROV video frames overlap, the same features may not be detected in consecutive frames because they are blocked, or their appearance has changed too much due to motion blurring or affine deformations. Therefore, it is important to know which feature matches are reasonable for further processing.

As described in section 3.4, the feature descriptors are designed in a way that the Euclidean (vector magnitude) distance in feature space can be used for ranking matches. This can be done by setting a threshold (maximum distance), that all matches from other images must be within this distance.

In terms of the python environment and OpenCV3, this method is called Brute-Force Matcher (Jakubovic & Velagic, 2018), and it works well with SIFT. The Brute-Force matcher takes a descriptor from the first image, finds all other descriptors in the second image, and based on the Euclidean distance, by trying each one tries to find the closest ones. The descriptors with the smallest distance between them are considered to be pairs. Ranking the descriptors only by their distance has some issues. By setting the threshold too high, many false positives (incorrect matches) are being returned, and by setting the threshold too low will return too many false negative (many correct matches being missed), see Figure 13.

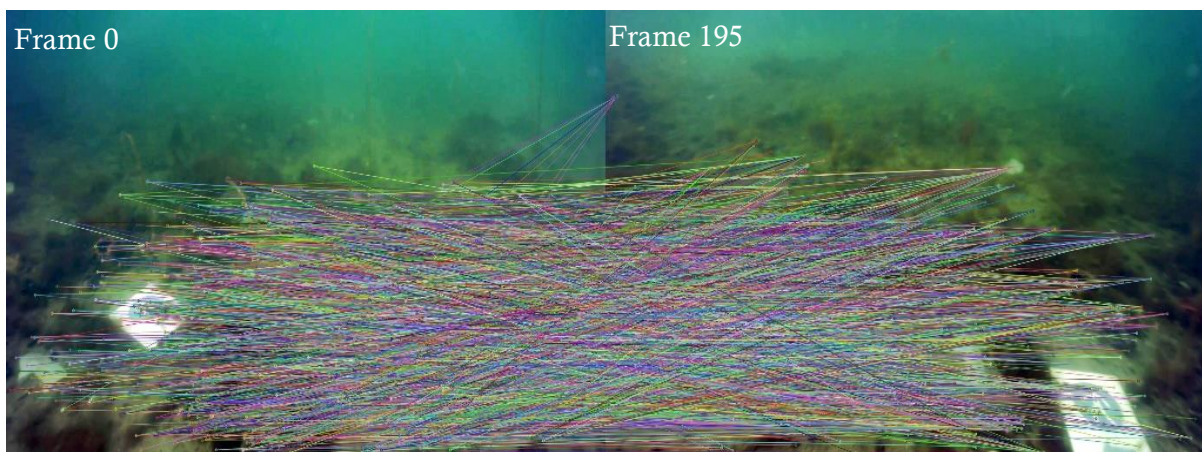


Figure 13: Too many matches found by the Brute-Force matcher between the two frames.

However, the Brute-Force matcher has an alternative function when matching descriptors. This function is indicated as `knnMatch` which returns the k best matches for each descriptor from an image where k is specified as 2 best matches in this project.

To complement the Brute-Force matching, Lowe's distance ratio test (Brown & Lowe, 2007) is applied. David Lowe proposed a simple method for filtering keypoint matches by eliminating matches when the second-best match is almost as good. First, the keypoints in image 1 are matched with two keypoints in image 2. Considering the assumption that a keypoint in image 1 cannot have more than one equivalent in image 2, it is concluded that those two matches cannot both be right. Following Lowe's reasoning, the match with the smallest distance is the "good" match, and the match with the second-smallest distance is the equivalent of random noise. If the "good" match cannot be distinguished from noise, then the "good" match should be rejected because it is not new information. Lowe's principle is that a "good" match is valid only if the ratio between the distance of the best "good" match and the second-best "good" match is smaller than a given number, that is:

If

$$\text{distance1} < \text{distance2} * \text{constant number (e.g. 0.77)}$$

 Then
 store "good" matches

where `distance1` is the distance between the keypoint and its best match while `distance2` is the distance between the keypoint and its second-best match. Figure 14 represents all the "good" matches after applying the distance ratio test; notably, the number of lines decreased significantly compared to the ones in Figure 13.

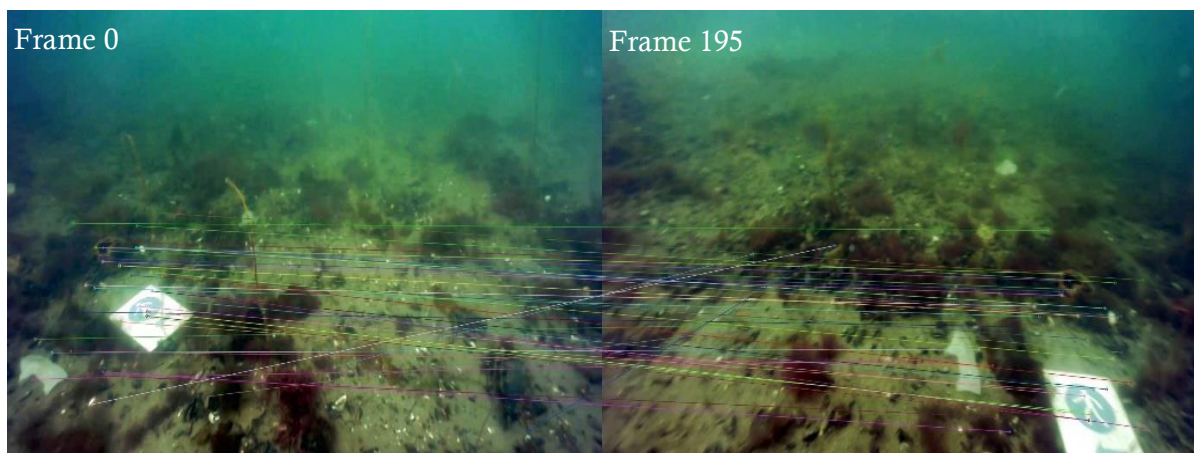


Figure 14: SIFT matches when Lowe's distance ratio test is applied. (Brown & Lowe, 2007)

3.4.4 Feature match verification and densification – RANSAC

Once some hypothetical good matches are found, and in order to find the best Homography matrix for the image alignment part, a geometric alignment can be used to verify which matches are inliers and which ones are outliers. For this, a method exists called RANSAC.

Random sample consensus, RANSAC, is a non-deterministic algorithm first published by Fischler and Bolles in 1981 (Fischler & Bolles, 1981). RANSAC estimates that the data (keypoints) consists of “inliers” (data whose distribution is explained by some model parameters. But, if there’s noise in the data (as mentioned before) then these are subjected as “outliers” and do not fit the model, see Figure 15. Thus, by robust estimation, RANSAC uses a minimal set of randomly sampled correspondences to estimate image transformation parameters. Then, it finds a solution that has the best consensus with the data. When it comes to image alignment, RANSAC is the sampling approach of estimating the most robust Homography matrix. (Hartley & Zisserman, 2002)

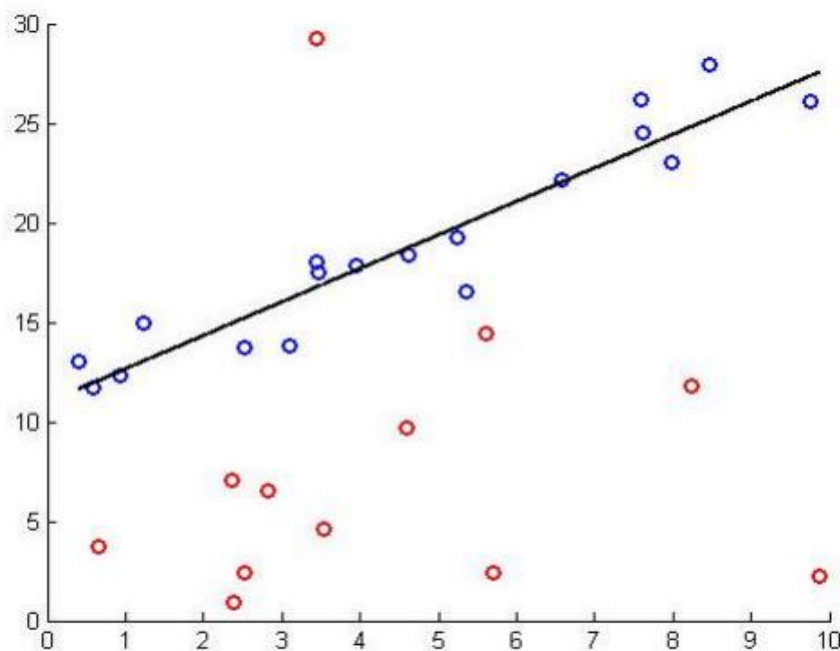


Figure 15: An Illustration of RANSAC inliers (blue) and outliers (red) and which are a best fit to a linear model.

3.4.5 Finding Homography

In OpenCV, (The OpenCV Reference Manual, 2019), finding the homography matrix is quite simple. It comes with a function called `findHomography` which finds a perspective transformation between two planes/images. It takes as input the coordinates of the points in the original plane (either points on an image or keypoint sets) and coordinates of the points in the target plane. RANSAC is also implemented to detect outliers. The perspective transformation, H , returned is in the form of a 3×3 matrix which can be used in order to align images. It is important to note that, if less than four keypoints are found by the matcher and the distance ratio test, the Homography matrix cannot be computed and no further alignment is possible. If this occurs, it might be due to the reason that not enough keypoints were detected initially, and/or, there are no significant matches between the images due to transformation issues between them. This can be improved with adjusting a Min Max Count number (4-10), so that the number of good matches has to be larger than that number in order to proceed with finding the Homography matrix. In the case that this method is not sufficient, correction becomes very crucial when trying to align underwater frames of a video sequence because of the lack of keypoints. If the video quality is poor, that means that the Homography computed is not sufficiently accurate enough for alignment. Some examples are, when the ROV is descending towards the bottom of the seabed; all the frames from top-down are unnecessary for the alignment part and will not make sense in the general perspective transformation between frames. Therefore, in this thesis, each video was manually cropped in order to get the significant parts of the videos that would be perfect candidates for mosaic creation.

3.4.6 Warp Perspective

Additional to the `findHomography` function, OpenCV (The OpenCV Reference Manual, 2019), also comes with a `warpPerspective` function which allows an image to be warped and transformed using the Homography matrix computed from the keypoints. It takes an input image (the one that should be warped/transformed), the 3×3 transformation matrix, Homography, and the size of the output image. An example is presented in the following chapter.

4 Image Alignment

This chapter shows how using only the Homography matrix it is possible to align an image at the perspective of another image. The final piece needed to operate the mosaic system, is to align the video frames in such a way that they resemble a “panorama”/mosaic. In order to do so, the first part was to see how all the theory mentioned this far could operate on two images only. Figure 16 represents the most standard algorithm used to align a pair of images using SIFT keypoints.

Figure 16: Algorithm I: Image Alignment algorithm of two images using SIFT, Brute-force matches and the Homography matrix.

```

Initiate SIFT

Turn image1 and image2 to grey scale

Detect and Compute key points and descriptors from both images (kp1, kp2, des1,
des2)

Initiate Brute-force matcher

Match descriptors with Brute-force(knnMatch), where k=2

Apply Lowes ratio test

For each m, n in matches
    If the distance(m) < (Lowes ratio threshold = 0.77) *distance(n)
        Save the good m matches in a list called good
Then
    Apply min max count threshold on the matches
    If the length of the good list is < MINMAXCOUNT = 10
        Find the homography for kp1, kp2 applying RANSAC
Then
    Warp the perspective of image2 on image 1
    Make size adjustments to fit image1 on image 2

End

```


The algorithm is applied on the two images presented this far, in the python environment using OpenCV 3, see Figure 12. Notably, the first image changed its perspective when applying the Homography matrix to be aligned on the same points on image 2. Image 1 is the one noted as Frame 0 in Figure 12 and image 2 is noted as Frame 195 that were extracted from an ROV footage. This were done on frames with such a gap between them in order to establish that homography works if there are similar feature locations within both images. A perspective distortion is applied in order to make sense of the matching between the images, see Figure 17.

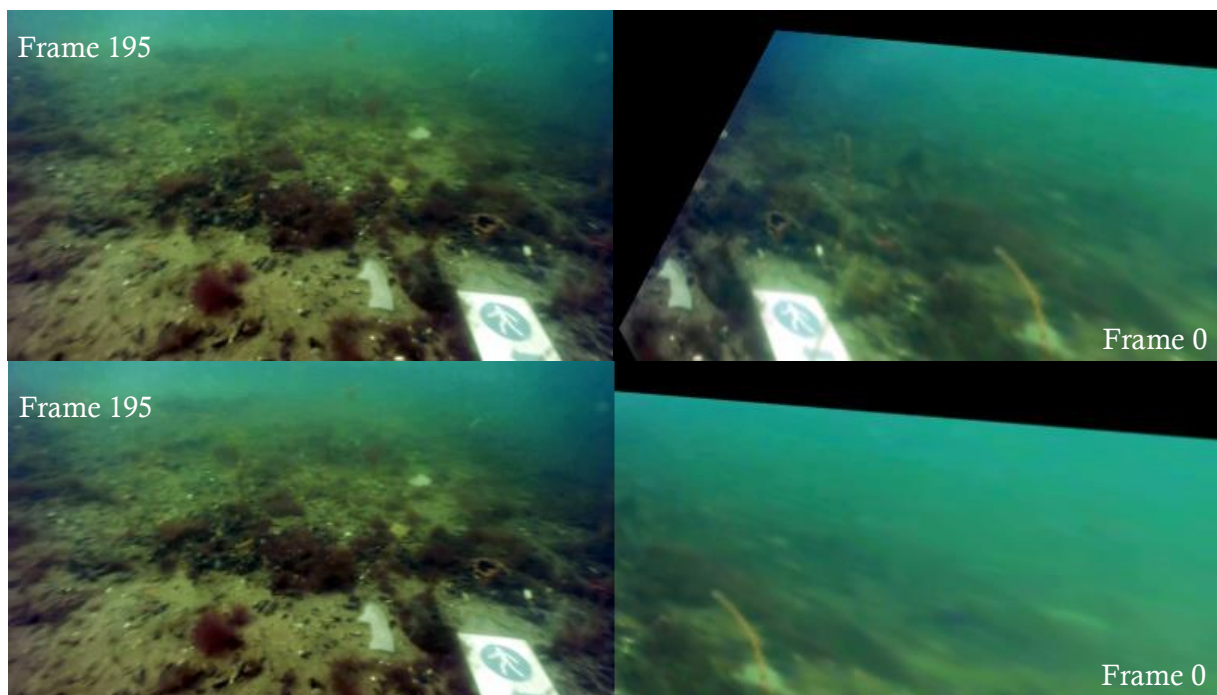


Figure 17: Top Images; Frame 0 is warped in Frame's 195 perspective using the homography matrix that was computed from the SIFT features detected between them. Bottom Image; Final alignment/stitched result between Frame195 and Frame 0.

4.1 Caveats

It is known that two images can be aligned if SIFT is used to detected and compute keypoints and the Bruteforce matcher with the KNN option for matching them and filtering them with Lowe's distance ratio test and RANSAC. However, there are some caveats to take in consideration. The Homography matrix needs at least four points in order to be computed, as previously mentioned, otherwise the alignment of the images cannot proceed.

There is a perspective distortion in frames if the video camera is not pointing perpendicular towards the bottom. This does not work well when processing video sequences under water. The frames must be aligned in an affine matter in order to create an appropriate seabed-map.

The homography matrix alone is not enough to compute a constant scale alignment of the images. Therefore, additional factors such as corner alignment and the translation matrix should be considered.

Since, the major goal of this mosaic system is to map the seabed, a geometric manipulation should be done on the frames so that a top-down view transformation is also applied in order to get frames that do align with the seabed scenery.

4.2 Top- Down View/ Bird View

According to the authors Abbas and Zisserman (2019), it is possible to get a Bird Eyed View perspective just by applying some geometrical manipulations on the images. (Abbas & Zisserman, 2019). Bird eye view transformation technique used to generate a top view perspective of an image (Venkatesh & Vijayakumar, 2012).

This technique is divided into three steps; first representation of the image in a shifted coordinate system, then perform rotation of image and then project the image on a two dimensional plane. The pipeline of the bird View transformation used in the current mosaic system is given in Figure 18 and the Bird Eye View transformed Frame 0 is shown at Figure 19.

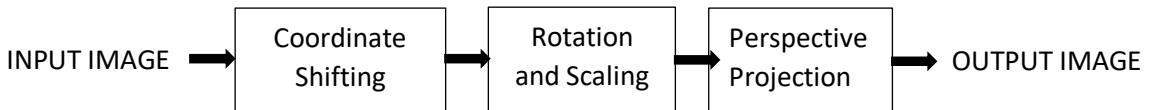


Figure 18: Top-down/ Bird Eye View transformation pipeline



Figure 19: Top-down or Bird Eye View transformation of frame 0

As mentioned in Chapter 2, an image is represented as a 2D plane with (x, y) coordinates. Then, the shifted coordinates are given as (X, Y) :

$$X = x - \left(\frac{\text{IMAGE WIDTH}}{2} \right) \quad (5)$$

$$Y = y - \left(\frac{\text{IMAGE HEIGHT}}{2} \right)$$

The rotation can be performed by matrix multiplication. If we introduce a third coordinate Z (to represent the image in 3D), then the location (X, Y, Z) of the pixel in the input image will be transformed as (p, q, r) in the rotated image:

$$\begin{bmatrix} p \\ q \\ r \\ 1 \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (6)$$

After rotation is performed on a set of pixels, the image's height decreases hence it needs to be stretched resulting in twice than the original height.

4.3 Discussion

Since the ROV's camera is mounted in the front of the robot looking from the 3D frame plane and has a tilting mechanism that allows the camera to look perpendicularly towards the seabed requiring no image modifications. This is not always possible when the robot is moving. This causes fewer feature detection as well as not uniform image alignment. Therefore, the Bird's Eye method was introduced so by the modifications of applying different angles in the mosaic system, it is possible to get the top view of the given image.

5 Implementation

This Chapter will first introduce the reader of the ROV used in this project as well as, present how the theory mentioned this far has been implemented in the mosaic system. Then, the chapter is split into two parts; The first part is implementing the mosaic system on ROV footage and go-pro videos without taking into consideration the orientation data from the ROV. The algorithm used for this part is presented at Figure 20 Algorithm II and the alignment process is illustrated at Figure 21.

Figure 20: Algorithm II: Final algorithm of the mosaic system.

```

Initiate SIFT (n Features)
Initiate Video object
For each consecutive frame:
    Pre-process
        Rescale (50%, keep interpolation)
        Apply colour correction (fix light, contrast and colour)
        Apply top-down view transformation (Bird Eye View)
    Align frames
        Turn frame1 and frame2 to grey scale
        Detect and Compute key points and descriptors
        Initiate Bruteforce matcher
        Initiate Brute-force matcher
        Match descriptors with Brute-force(knnMatch), where k=2

        Apply Lowes ratio test
        For each m, n in matches
            If the distance(m) < 0.77 *distance(n)
                Save the good m matches in a list called good
        If the length of the good list is < MINMAXCOUNT= 10
            Find the Homography (H) for kp1, kp2 applying outlier
            filter =RANSAC 6.0
        State the corners of both frames (corners1, corners2)
        Perspective Transform corners 2 with H
        State the translation matrix
        Warp the perspective of frame 1 using the translation matrix
        Warp the perspective of frame 2 using the dot product of H and
        the translation matrix

```

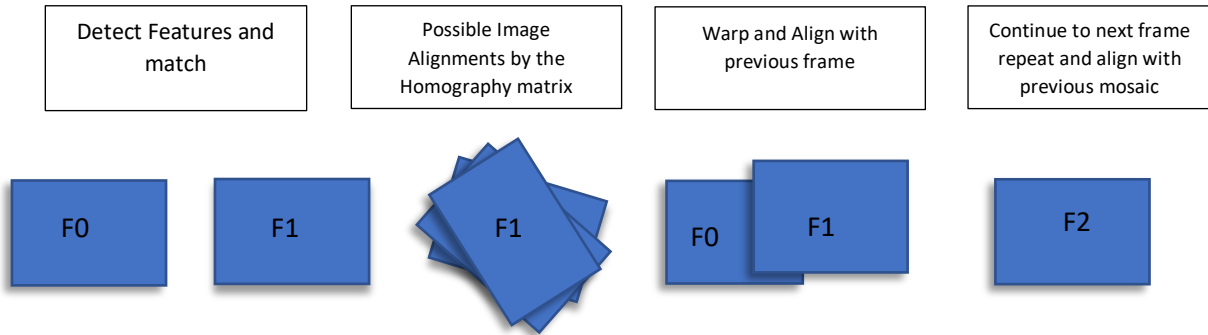


Figure 21: Illustration of the image alignment process followed by the mosaic system. First, the detection and matching of features takes place between Frame 0 and Frame 1. Then, possible image alignments are found by the Homography matrix. Finally, warp and align Frame 1 to the perspective of Frame 0. Continue to next Frame 2 and repeat the process by aligning the next frames to the previous mosaic.

5.1 ROS & BlueROV2

The Robot Operating System (ROS) is a collection of tools, libraries and protocols that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. An underlying operating system, in this case a Raspberry Pie on the BueROV2, “communicates” with the ROS interface in carrying out its tasks.

The underwater robotic system used in this project is a BlueROV2 which consist of features such as, a maneuverable vectored configuration with six thrusters, a USB camera mounted on a server motor in order to configure the inclination as needed, a cable reaching up to fifty meters, a system that runs on Raspberry Pie and sends messages through the ROS interface. These messages can be information received from the sensors (videos, ROV position, velocity, depth etc.) send through a fathom tether cable and it is stored as compact data in a ROS bag (data logging file) as well as inputs to the BlueROV2 (control signals to thrusters, camera tilting etc.).

Additionally, it is a stable ROV, i.e., it can hold its depth, which helps to have a better performance while video taking. The BlueRov2 has a Low Light HD USB camera mounted in the nose of the robot. The camera has a sampling time of 15 fps and a tilting system. A calibration of the camera was done previously underwater, therefore lower distortion and higher accuracy was a considered.

The videos acquired by the ROV stored in a ROS bag are extracted additionally to orientation and timestamp data of each frame. This can be used in order to create a more automated mosaic program further on.



Figure 22: BlueROV2, the remotely operated vehicle whose footage was used in this project.

5.2 Mosaic system execution on footage

This section will describe of how the mosaic system developed this far performed on footage taken by the BlueROV2 as well as videos from a high definition go-pro camera. Different footage taken by the BlueRov2 is used from coastal areas in Copenhagen.

Nothing else except the visual data was taken into consideration while developing the mosaics. This is due to the fact to see if the mosaic system can be used with the input of only visual data and the background theory presented this far in order to see if the mosaicking result is beneficial enough for seabed investigations. Each video was temporally cropped manually in order to get the area of interest to be subjected to mosaicking.

5.2.1 Results

To start with, the mosaic system is used on a sea grass video, see Figure 23. Due to multiple colors and the video quality it is interesting to see how the color correction enhanced the quality of the frames as well as the alignment. The result might not look striking at first glance, but the causes are discussed further on.

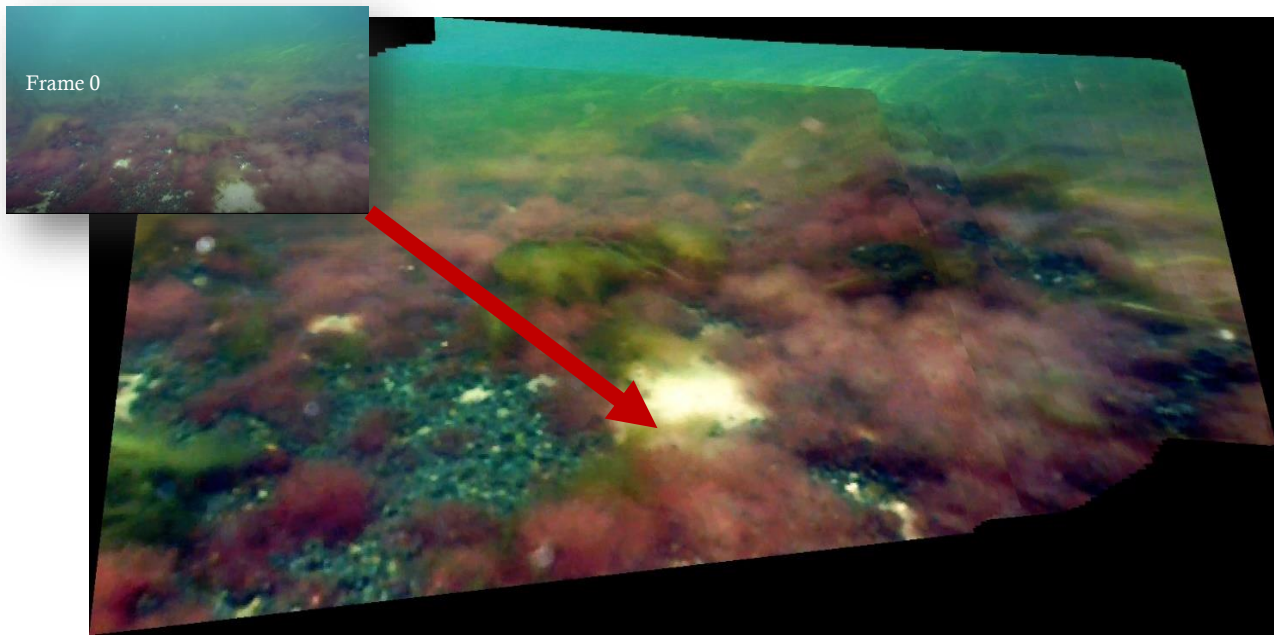


Figure 23 Seagrass Seabed- mosaic generated with color enhancement and Top down view.

The following mosaic, Figure 24, was captured in Copenhagen harbor and is the reference footage of Frame 0 and Frame 195 used in the theoretical part of this project. The present mosaic is a combination of 108 frames. The Top Down View transformation is applied as well as color enhancement. Notably, the sign, it seems like the ROV's reference camera has taken the footage from the top.

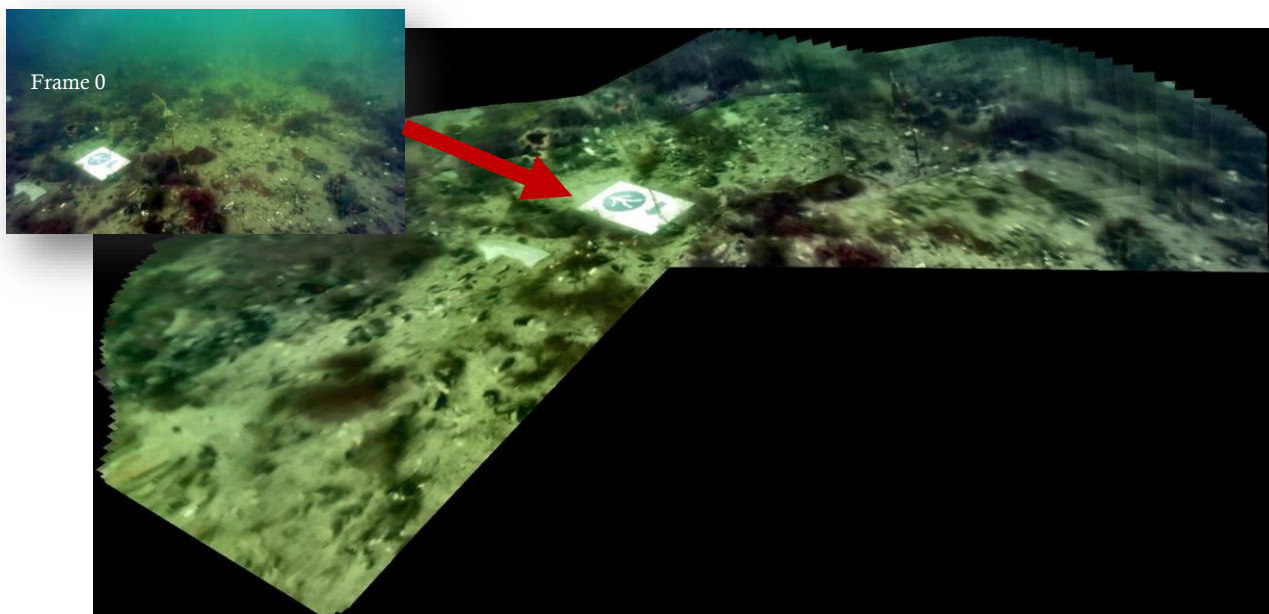


Figure 24 Seabed mosaic generated from BlueRoV2 footage acquired from Copenhagen harbor and consists of 108 frames.

Another step that is taken in consideration is an experimental day that took place in Skovshoved Havn in order to acquire different ROV footage. The experimental part took place by testing the ROV in different trajectories in order to see afterwards how the mosaic system would be able to align frames taken in different trajectories:

- 1 Straight Line (constant depth)
- 2 Lawn- Mower direction (constant depth)
- 3 360 degrees fixed position (constant depth)
- 4 Free- Roaming (constant depth)

A High Definition go-pro camera was also mounted on the ROV with the camera being tilted downwards in order to be able to make comparisons between the different footage.

Unfortunately, most of the ROV footage acquired on that specific day was not ideal enough for the generation of high-resolution mosaics, see Figure 24. The footage is too green, and the color enhancement method could not improve the quality. However, the mosaic system is still capable of aligning a small part of the footages. This is extraordinary since it can find features even in such resolution videos, but the overall product is not pictured as the best. Surprisingly, when the mosaic system was tested on the go-pro footage, it showed significantly better results, see Figure 26.

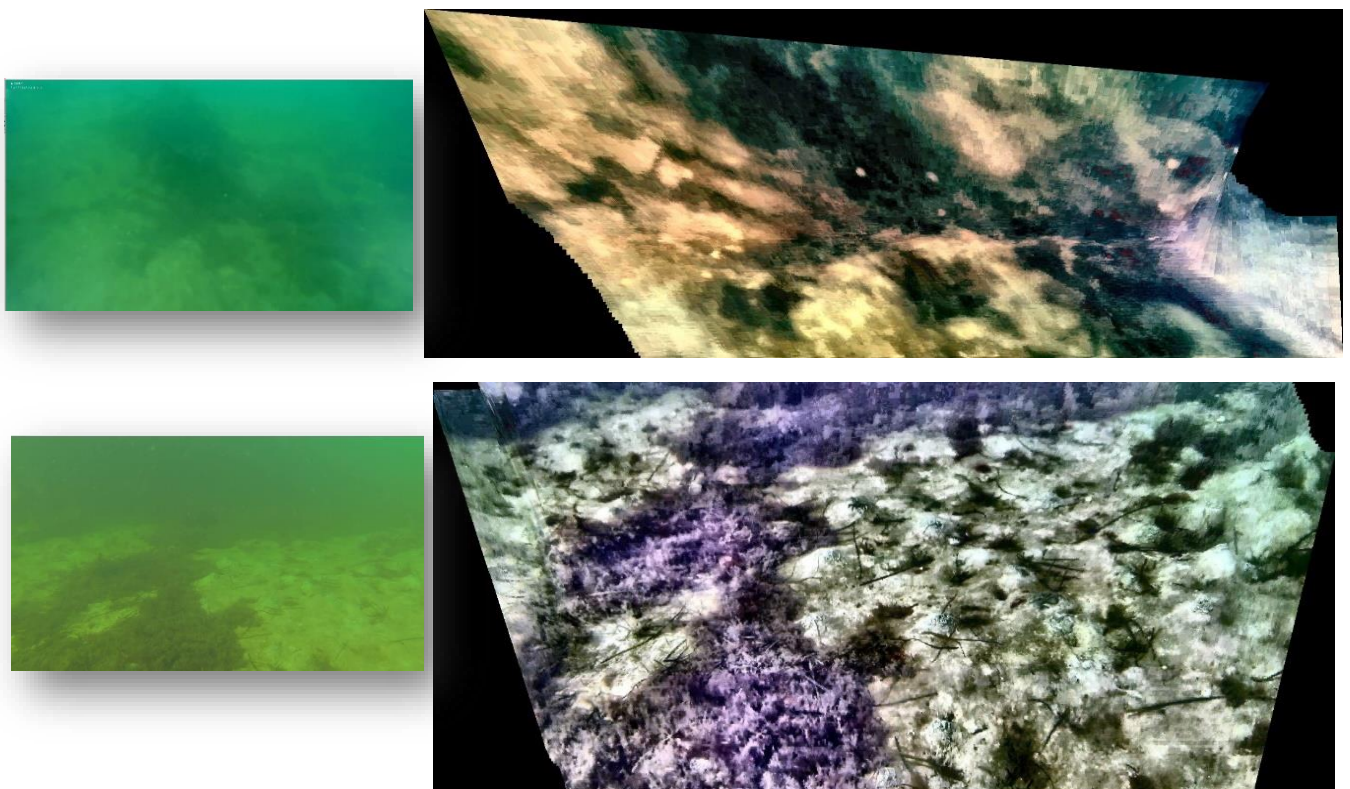


Figure 25: Low resolution mosaics generated from ROV footage in Skovshoved Havn due to the green reflection from the waters.

Compared to the ROV mosaics, the go-pro example mosaic was generated by 512 frames and it is better in quality, the color enhancement improved the contrast of the features in the footage and one can see all the interesting areas where the ROV roamed in this footage. The reasons of these extreme differences of mosaic development will be discussed thoroughly in chapter 6.

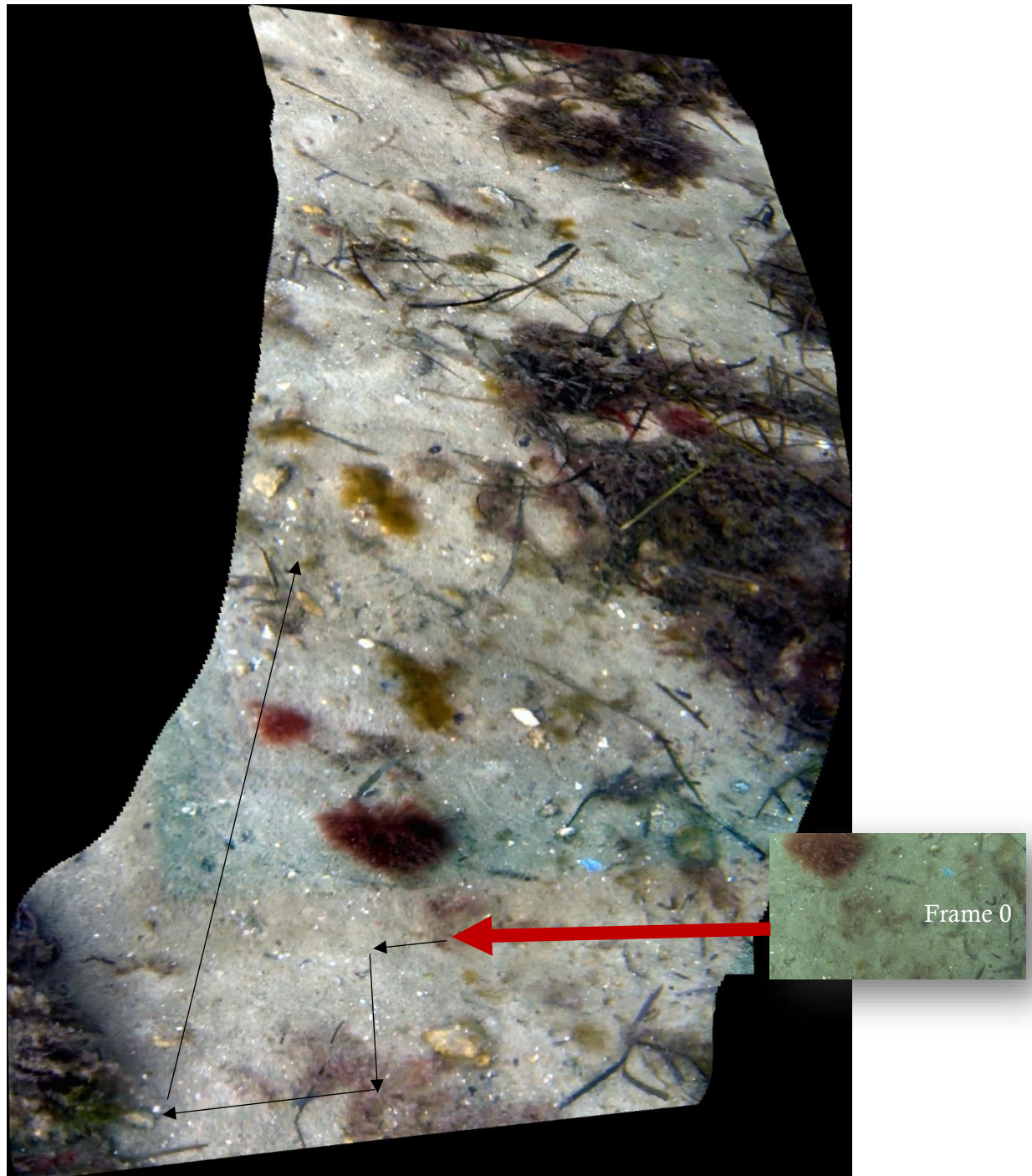


Figure 26: Seabed mosaic generated by go-pro footage (512 frames) that was mounted on the ROV in Skovshovde Havn. The black arrows indicate the trajectory of the frame alignment that took place by the mosaic system.

5.2.2 Google Vision

An additional part that is used on the mosaic system is the implementation of the Google Vision Application Programming Interface (API) (Cloud Google Vision, 2020) which is an open machine learning library which can recognize objects and give them confidence scores and labeling the scenery, see Figure 27. One can access the Google Vision API on Google's Cloud website and apply for a one-year free trial worth of 300 USD. The users can use the tokens received for implementing different services in their projects provided by Google Cloud.

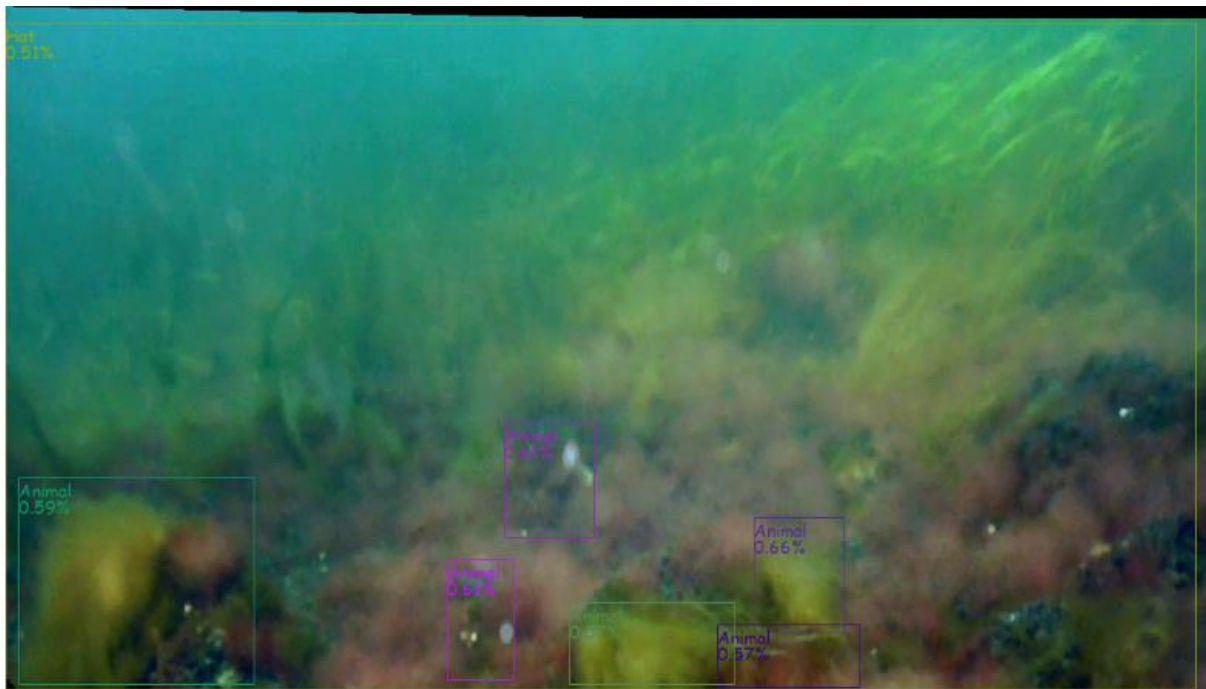


Figure 27: Google Vision API applied on the Sea grass video frame which identified Confidence scores of objects on the mosaic as well as labels of the scenery such as: Nature, Green, Underwater, Organism, Marine Biology, Water, Algae, Plant, Seaweed, Chlorophyta.

New technological services such as Google or Microsoft Azure are astonishing in the way that they give researchers an efficient way of large data analysis without time limitations. Another example of the Google Vision API implemented on a complete mosaic is shown at Figure 28.

The underwater mosaic illustrated in Figure 28, was developed by aligning 1500 frames from a 1080 HD go-pro diving video taken in Cyprus. The color correction method worked better on this video rather than the ROV videos. This might be due to the quality of the resolution acquired by the ROV camera.

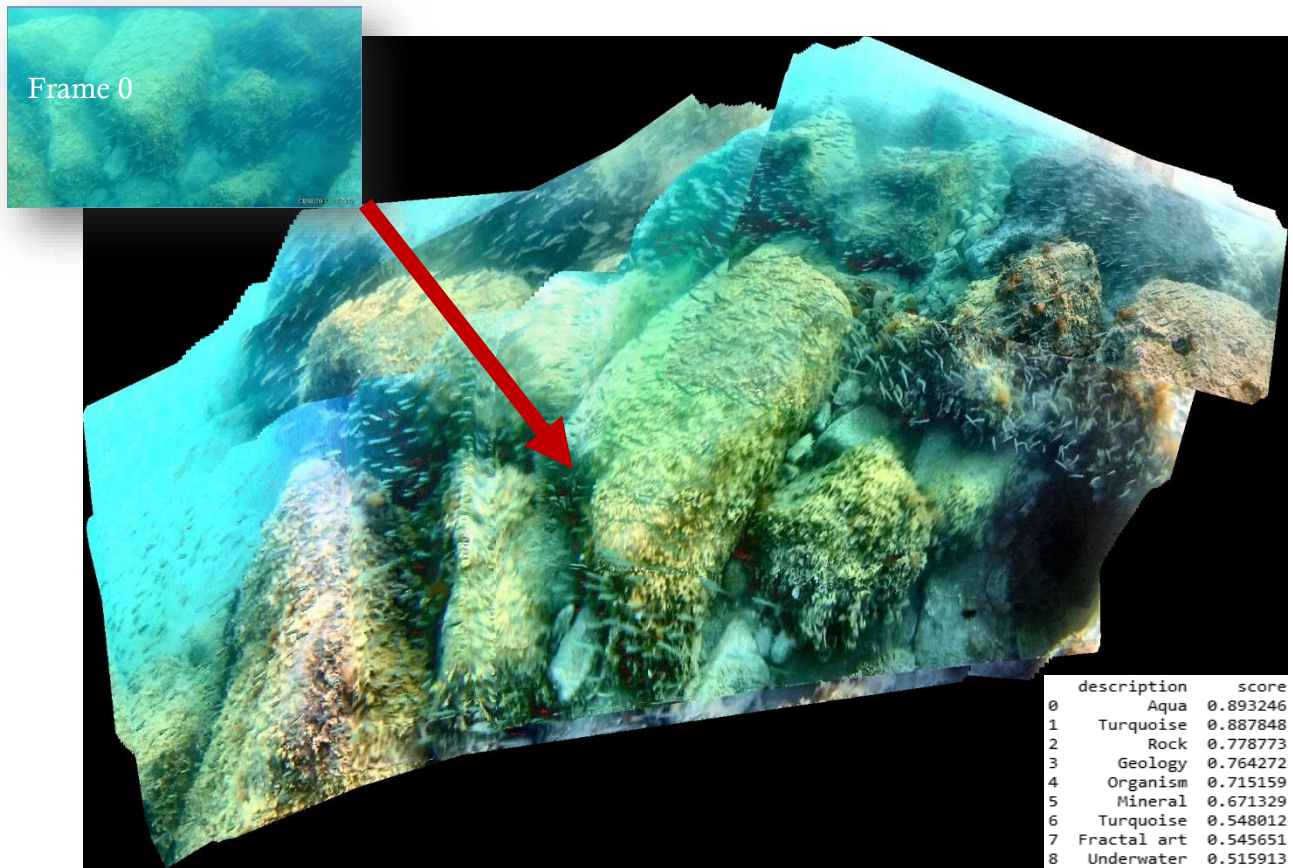


Figure 28 Google Vision API applied on a generated underwater mosaic which identified labels such as: Mineral, Rock, Turquoise, Aqua, Geology, Organism, Coral reef, Marine biology.

6 Conclusion

A mosaic system has been presented in this thesis which is capable of processing video sequences acquired by an ROV and other video means with the aim of developing underwater mosaics. The following objectives have been fulfilled; detecting and computing features among consecutive frames; aligning frames in order to get a visualization of the seabed; application of the Google API which is capable of object recognition on the seabed-map.

However, a few problems arise from the acquisition of images by a remote operated vehicle. Radial distortion, poor luminosity, cloud water is part of the issues that can occur. Due to poor or high luminosity, ghosting effects can be seen at the frames' borders, for instance in Appendix A (Figures 29 and 30), are presented two examples of mosaics which the ghosting issue occurred. The color correction method could solve this in some videos while in others it did not succeed as much. A blending or filtering technique could implement in the future so issues like that can be avoided.

The flight data of the ROV was not taken into consideration, however, there is a possibility integrating the transformations between frames, by integrating the positional data of the ROV, and thus, get a more aligned mosaic. Doing so, would give the possibility of transforming the mosaic system as a fully automated system that could develop seabed mosaics in real time.

A strong issue that occurred was that since the ROV footage was in low resolution it could not be processed further. Compared to other videos of higher resolution, the mosaic system showed better result since it is impossible to compute a homography if no features can be detected. Thus, it can be concluded that the quality of videos plays a key role when it comes to mosaics development.

Additionally, from all the different angles of the mosaics showed, it is concluded that for seabed mosaic the camera needs to look downwards and the vehicle needs to stay at a constant depth in order to avoid perspective distortion.

The feature finding depends as well on the quality and resolution of the videos. Otherwise the same feature-related problems persist. Another example of a finished mosaic acquired by a go-pro camera can be seen in Appendix A.

Regardless the issues present, the future of systems as such can provide new means in underwater exploration, increase the ability of getting a different perspective of the oceans and improve autonomous missions underwater.

Appendix A

Figure 29 and 30 are underwater mosaics developed with the current mosaic system. The luminosity effects of the sun can be seen among the borders of each frame. This is called as ghosting effect or seams.

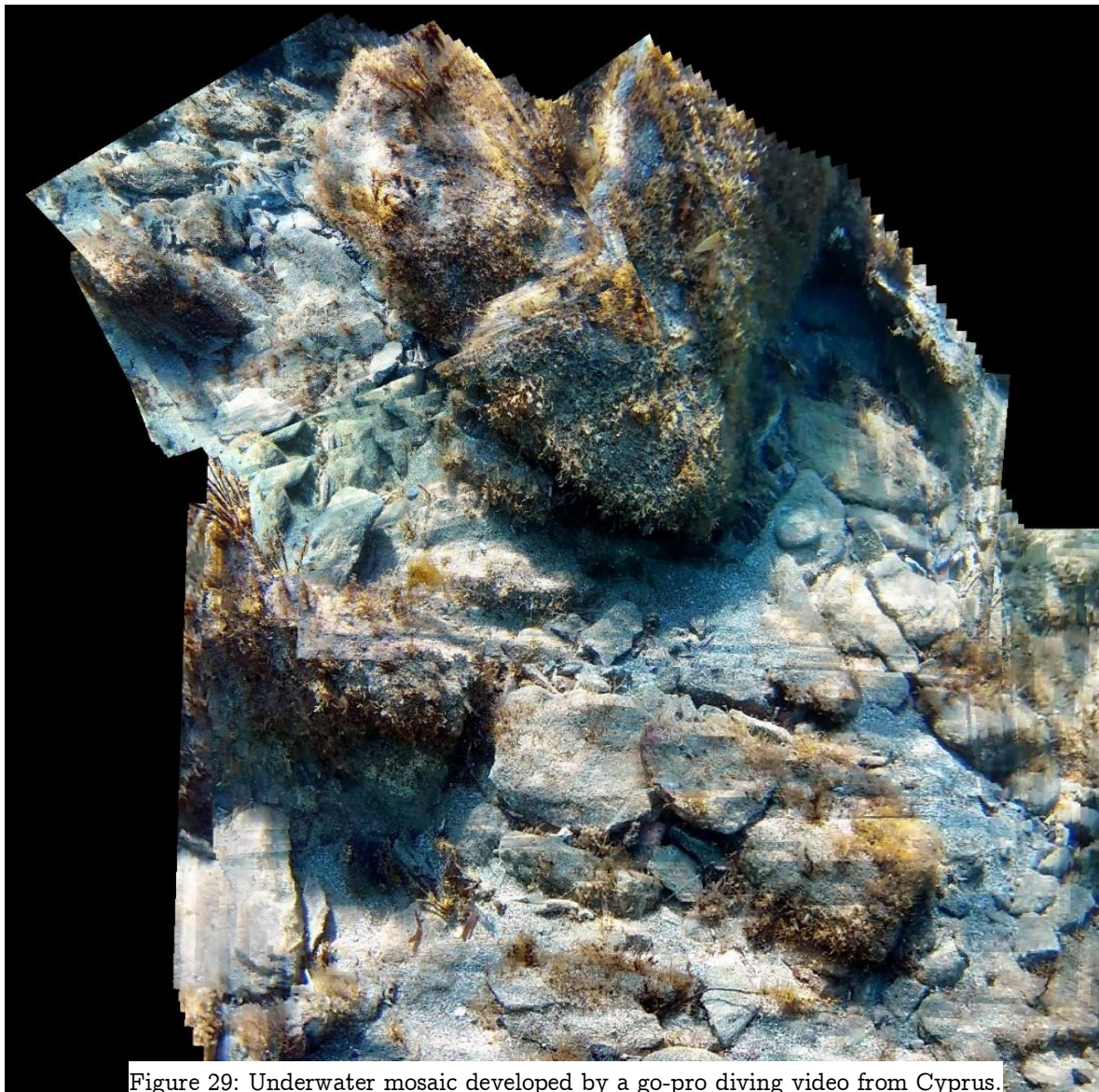


Figure 29: Underwater mosaic developed by a go-pro diving video from Cyprus.

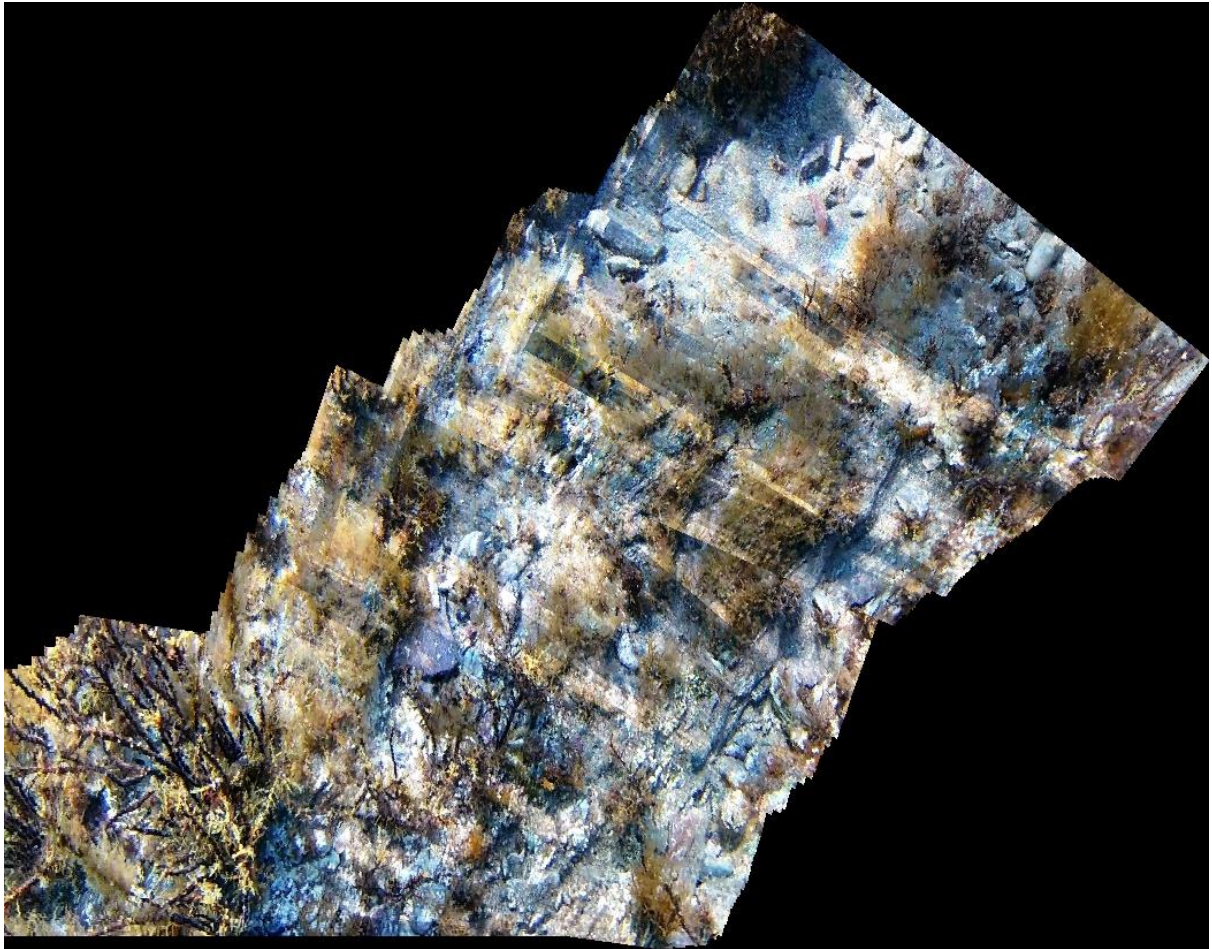


Figure 30: Underwater mosaic developed by a go-pro diving video from Cyprus.

Bibliography

- Abbas, A. & Zisserman, A., 2019. A Geometric Approach to Obtain a Bird's Eye View from an Image.
- Adel, E., Elmogy, M. & Elbakry, H., 2014. *International Journal of Computer Applications*, 99(6).
- Bellavia, F., Gagliano, G., Tegolo, D. & Valenti, C., 2006. *Underwater archaeological mosaicing*. Vouliagmeni, Athens, Greece, Proceedings of the 10th WSEAS International Conference on Computers, pp. 1141-1146.
- Bez, H. E., 2003. *Homogeneous coordinates for computer graphics*, Department of Computer Studies, Loughborough University of Technology, Uk: s.n.
- Brown, M. & Lowe, D. G., 2003. *Recognising Panoramas*. Vancouver, Department of Computer Science, University of British Columbia.
- Brown, M. & Lowe, D. G., 2007. *Automatic Panoramic Image Stitching using Invariant Features*. Vancouver, Department of Computer Science, University of British Columbia,.
- Chen, M. et al., 2015. *Underwater Image Stitching based on SIFT and Wavelet Fusion*, s.l.: s.n.
- Chu, E. & Sandy Yu, E. H., 2007. *Image-Guided Tours: Fast Approximated SIFT with U-SURF Features*, s.l.: s.n.
- Cloud Google Vision, 2020. *Cloud Google Vision*. [Online]
Available at: <https://cloud.google.com/vision/docs/drag-and-drop>
- Elibol, A., Kim, J., Gracias, N. & Garcia, R., 2016. *Fast Underwater Image Mosaicing through Submapping*, s.l.: Springer Science+Business Dordrecht .
- Fischler, M. A. & Bolles, R. C., 1981. *Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography.*, s.l.: Communications of the ACM, 24(6):381-395.
- Garg, D., Garg, N. K. & Kumar, M., 2018. *Underwater image enhancement using blending of CLAHE and percentile methodologies*, s.l.: Multimed Tools Appl.
- Gracias, N. & Santos-Victor, J., n.d. *Underwater Mosaicing and Trajectory Reconstruction using Global Alignment*, Lisboa Codex, Portugal: Av. Rovisco Pais.
- Gracias, N., Sjoerd van der Zwaan, Bernardino, A. & Santos-Vector, J., 2002. *Results on Underwater Mosaic-based Navigation*, Lisboa Codex, Portugal: Av. Rovisco Pais.
- Hartley, R. & Zisserman, A., 2002. *Multiple View Geometry in Computer Vision*, s.l.: Cambridge University press.
- Jakubovic, A. & Velagic, J., 2018 . Image Feature Matching and Object Detection Using Brute-Force Matchers. pp. 83-86.
- Joy, M., 1997. Landscape Mosaics. *Ecology*. 78. 642..

Laranjeira, M., Jaulin, L. & Tauvry, S., 2016. *Bulding Underwater Mosaics Using Navigation Data and Feature Extractiom*, s.l.: s.n.

Le Bars, F. et al., 2018. *Estimating the Trajectory of Low-Cost Autonomous Robots Using Interval Analysis: Application to the euRathlon Compeition*, s.l.: Springer International Publishing AG.

Leone, A., Distante, C., Mastrolia, A. & Indiveri, G., 2006. *A fully automated approach for mosaicking*, Lecce: s.n.

Lowe, D. G., 1999. *Object Recognition from Local Scale-Invariant Features*. Vancouver, Computer Science Department, University of British Columbia .

Lowe, D. G., 2004. Distinctive Image Features. *International Journal of Computer Vision* .

Nunes, A. P., Gaspar, A. R. S., Pinto, A. M. & Matos, A. C., 2018. *A mosaicking technique for object identification in underwater environments*, Porto, Portugal: Emerald Publishing Limited.

Rogers, D. F., 1976. Mathematical elements for computer graphics..

Rzhanov, Y., Linnet, L. M. & Forbes, R., 2000. *Underwater video mosaicing for seabed mapping*. s.l., s.n.

The OpenCV Reference Manual, 2019. *opencv.org*. [Online]
Available at: <https://docs.opencv.org/2.4/opencv2refman.pdf>
[Accessed 2020].

Varosi, F. & Gezari, D. Y., 1992. Mosaic: Software for creating mosaics from collections of images..

Venkatesh, M. & Vijayakumar, P., 2012. Transformation Technique. *International Journal of Scientific & Engineering Research*, 3(5).

Yin, C. & Y. S. & Y. X. & W. Z. & W. Y. & Z. B. & T. Y., 2015. *Removing Dynamic 3D Objects from Point Clouds of a Moving RGB-D Camera*, s.l.: s.n.