Sidharth Shanmugam

Initial Project Report

Machine Vision-Based Anti-Backscatter Lighting System for Unmanned Underwater Vehicles

---

Submitted 7 March, 2024

4th Year Initial Project Report for Degree of
MEng in Electronic and Computer Engineering

School of Physics, Engineering and Technology,
University of York

Supervisors:

Prof. Paul D Mitchell, Prof. Andy M Tyrrell

Ethical Considerations

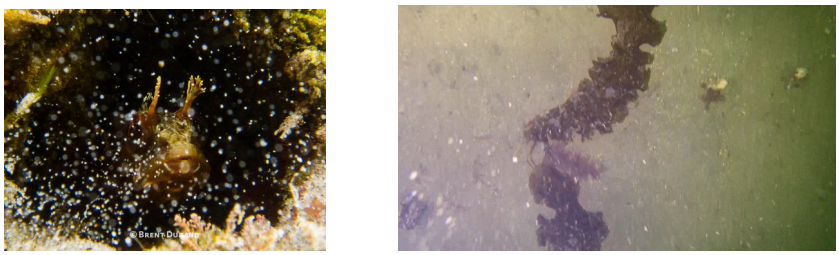
After consideration of the University of York's code of practice and principles for good ethical governance, I have identified no related issues in this project.

Contents

1 Introduction

Underwater imaging has long been crucial in various fields, such as marine research, environmental monitoring, underwater archaeology, and offshore industries. For example, scientists use underwater photography with quadrats to audit the abundance of coral over time at several reef locations [1]. Unmanned Underwater Vehicles (UUVs), a type of submersible vehicle, are pivotal in advancing underwater imaging capabilities. Often housing vast arrays of sophisticated sensors, these vehicles enable the end user to explore and analyse underwater environments with unprecedented accuracy and efficiency. Their autonomous or remote-controlled nature advocates the ideal platform for conducting missions of extended duration in hazardous conditions, impossible for direct human presence and intervention. With the benefits of UUV deployment, they have become common as a safer and cheaper alternative to manned vehicular operations and divers in the vast range of underwater imaging-related industries and applications, such as intelligence surveillance and reconnaissance in defence, inspection and identification of defects or foreign objects in maritime, and oceanography and hydrography in marine research [2].

The lack of light at greater sea depths is a critical challenge in underwater imaging, and to tackle this, an external high-power light source often accompanies a camera to ensure a well-lit scene. However, this produces an adverse side-effect: backscatter, shown in Figure 1, where suspended particles in water scatter light in an inhomogeneous manner, reflecting the light emitted by the light source back into the camera, creating exponentially bright spots and often saturating the image and degrading the quality. While there are a few simple and universal techniques to eliminate backscatter, such as reducing the separation between the subject and camera, fine-tuning the light source position such that only the edge of the light cone illuminates the subject, or achieving perfect buoyancy to minimise creating clouds of sand and debris [3], they lack viability for UUVs due to not only the erratic backscatter appearance from the continuous and arbitrary motion of the propellors and general vehicular movement but also the constant existence of backscatter-causing debris throughout water bodies.



(a) (b)

Figure 1: (a) Backscatter appears as the particles of sand drift between the aquatic animal and the camera, retrieved from [3]. (b) A captured frame in GoPro footage from a UUV of the seabed with backscatter increasing as the propellors disrupt the sand.

The ultimate goal of this project is to develop a novel light source system capable of aiding the generation of high-quality underwater images, of mainly the sea floor, from UUVs without backscatter interference. This project first aims to research systems and develop reliable

backscatter detection and elimination capabilities, with a specialised projector serving as a dynamic light source of tailored light patterns for selective scene illumination to mitigate backscatter effects. The second research aim studies architectures and methodologies to optimise the system for real-time to ensure predictability and stability by allowing for direct control over imaging performance, such as frame processing times to adapt the system for dynamic underwater environments. The final research aim investigates the engineering of a predictive system for anticipating the future positions of detected backscatter particles to alleviate computationally intensive machine vision processing for efficient and preemptive adjustments to the light projection patterns for optimal suppression.

This initial report forms the current project status. Section 2 summarises the information in related theoretical realms. The compiled background information will form the foundation of the subsequent sections: Section 3 discusses the specifications to achieve project goals with detail on hardware and software choices, Section 4 explains the intended project progression approach exemplified by a project schedule in Section 5.

2 Background Information

Underwater backscatter has a mixture of many compositions, from bubbles and sand to all sorts of marine debris. It is much easier to explore the image processing theory for one backscatter composition type and later expand to cover others. Many environmental analysis fields employ systems for underwater bubble quantification to study gas seepages from the sea floor with precise outline detection and movement tracking to calculate the volume, flux, and overall chemical composition. Gas escape measurement and monitoring systems require high precision, extensive range, strong anti-interference, and low cost under complex underwater conditions [4].

An accurate backscatter cancellation system must consider two main factors: (a) segmentation by isolating the bubbles from the image background and (b) determining their positions and sizes with bubble tracking in successive frames. Section 2.1 investigates environmental analysis-related literature to discuss these two factors. A real-time approach enables predictability by knowing the exact time it takes to process each task, ensuring stability when carrying out underwater imaging missions requiring image stitching, as it is imperative to match the system processing speed to the UUV motion. Section 2.2 begins discussing this topic with the real-time operating system (RTOS), analysing limitations and alternatives. A system to predict future positions concerning the tracked locations in a finite count of previous frames may help reduce machine-vision system complexity, considering two approaches: (a) an interpolative approach with an application of either a linear or polynomial interpolation algorithm or (b) an artificial intelligence (AI) approach, harnessing supervised machine learning (ML) methods with a potential to incorporate unsupervised methods for greater accuracy, Section 2.3 analyses possible strategies and limitations. Finally, Section 2.4 studies computer systems for powerful image processing, a specialised camera sensor for capturing fast-moving subjects with no distortion, a

specialised light source for projecting backscatter cancelling light patterns, and concludes with discussing programming languages and libraries for efficient development and run-time.

2.1 Machine Vision for Backscatter Detection and Tracking

Previous work on this project [5] outlines the backscatter segmentation system revolving around a simple blob detection algorithm [6]. A 'blob' in this context refers to a group of connected pixels in a binary image [7]. The algorithm first applies several thresholds to a source image, then extracts connected pixels from and calculates their centre positions. The algorithm groups the centre positions from the set of binary images and denotes the centre position, an adjustable distance apart, that forms a group as a blob, which the algorithm uses to estimate and return the final centre positions and radii for each blob. While simple blob detection is a viable method due to computational simplicity, as [5] mentions, for blob detection to work, the blobs must share at least one property, such as size or shape, to isolate these blobs. Underwater is a volatile environment where backscatter particles may be unique, so the system must account for varying backscatter properties.

The paper in [8] presents the design of a novel method for automated gas bubble imaging. In contrast to the work in [5], this paper employs the Canny edge detection algorithm [9] for greyscale images. The Canny filter compares the value of each pixel relative to its neighbours, so when the gradient between two adjacent pixels is higher than a certain threshold, the algorithm sets the bordering pixel to a value of binary '1', otherwise '0', resulting in the formation of edges around objects. The paper compares the Canny segmentation approach with simple thresholding (blob detection approach), an illustration in Figure 2, and deduces Canny as more accurate for segmentation, even in inhomogeneously illuminated areas. However, the disadvantages of Canny, as stated in [8], are the drastic increase in computing time and the requirement for implementing morphological techniques to account for the growing or shrinking of bubbles during segmentation. However, it has been 14 years since the publication of this paper, and the computational power of computers nowadays should handle the Canny workload easily. While this paper explores methodologies perfect for a starting point, it does focus more on the system construction and quantifying gas flux measurements, thus skipping some information on vital aspects such as how the bubbles were perfectly highlighted after Canny in Figure 2, one can assume they are using logic to fill closed loop edges.

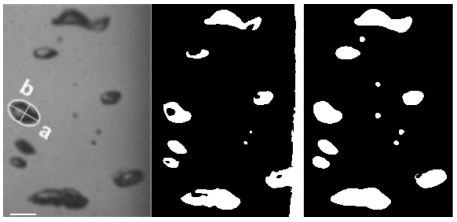
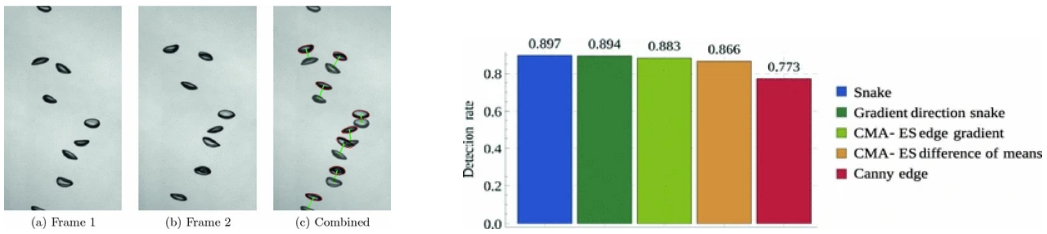


Figure 2: An illustration from the paper in [8] showing the original input in the left-most image, bubble segmentation using a simple thresholding method in the middle, and bubble segmentation using Canny in the right-most image.

The Canny-based system in [8] introduces a more accurate backscatter segmentation solution which, unlike the simple blob detection algorithm, does not require 'blobs' of similar shape and parameters due to the edge detector approach. The work in [10] expands on this Canny-based system with the motivation to derive more stable algorithms tackling the sporadical false detection characteristic of the Canny algorithm, introducing methods for precise bubble stream quantification and accurate bubble elliptical fitting with snake-based methods, where a snake is an energy-minimising spline guided by external constraint forces and influenced by image forces that pull it towards features such as lines and edges, and the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) optimisation algorithm [11]. Although false positive detections due to an unstable algorithm are not detrimental to the nature of this project, they will result in system inaccuracies, which may affect imaging quality. In an underwater environment where lighting conditions are highly variable, it is worth considering the Canny-based approach in [10] as a benchmark and expanding with a snake method implementation, as this method achieves the highest decision rate compared to the others, illustrated in Figure 3b, achieving an 89.7% decision rate, a 12.4% advantage over the standard Canny system.



(a) (b)

Figure 3: From [10]: (a) illustration of the movement of bubbles in between two frames at 100 fps. The combination shows frames 1 and 2 with the bubble detection in frame 2 in red and matching bubbles connected in green, (b) a graph comparing the detection rates for several selected methods on a high-quality GoPro image sequence with manual ground truth on 20 images.

The work in [8] also proposes a method for bubble tracking using the 'least distance rule', which is the realisation that the distance a bubble travels in two successive frames is smaller than the distance to its closest neighbour. This assumption, although not valid for overlapping bubbles, very high bubble concentrations, and cases where the travel distance exceeds the neighbouring bubble distances, enables a computationally simple method for calculating positions in successive frames and the bubble rise velocity from the sea floor. Reiterating the previously mentioned point regarding the increased computational power in computers nowadays, I don't think this assumption is a fair trade-off since backscatter, especially the type forming from bubbles, can be highly concentrated, overlapping, and fast-moving due to the UUV propellors. The paper in [10] proposes a new method for tracking bubbles using a Kalman filter [12] to predict the bubble position in a subsequent frame from detection, employing the Hungarian method [13] for the minimum weighted matching of the predicted positions and the new detection of bubble positions. Figure 3a illustrates the system in [10], notably observing the precise red border that follows the exact outline of the bubble in addition to the accurate tracking even in bubble-

dense regions. While it is unclear whether the Kalman filter-based method is resilient to the limitations of the 'least distance rule', the paper [10] does confirm a highly reliable tracking.

2.2 Real-time Systems

A real-time operating system (RTOS) enables the compliance of a 'hard' real-time system, where there is a guarantee of the maximum time a task requires for completion. An RTOS is typically used in low-power, embedded systems such as single-core STM32 microcontrollers due to their deterministic behaviour, low-latency interrupt handling, and low-complexity design. However, an STM32 microcontroller cannot handle the high computational requirements of machine vision tasks, thus invoking a preference towards a general-purpose computer system. While there are RTOS products for general-purpose computer systems, they involve many limitations requiring circumvention compared to a general-purpose OS such as Linux, drastically increasing development times and resulting in a much more complicated system due to the bare-metal knowledge requirement, thus not being the best option for this project.

Most multi-tasking and general-purpose OSes, such as Linux, are pre-emptive. In Linux, when a task in user space, which is a set of locations where normal user processes run (i.e., everything other than the kernel) [14], is interrupted, and if the interrupt handler can wake up another task, the scheduler will schedule this task as soon as the interrupt handler returns. However, many sections in kernel space, which is the location where the code and data of the kernel are stored and executed under [14], do not support pre-emption due to the presence of spinlocks, which are non-blockable and non-sleepable programmatical loops to protect critical sections of code, pre-emption logic falls apart when a user space task calls a kernel-specific function, thus accepting a promotion to kernel space as it receives an interrupt.

Figure 4, from the training material in [15], illustrates this issue of user space task promotions to kernel space: Critical sections are bound with spinlocks, ultimately preventing the immediate pre-emption of the scheduler by holding Task A to schedule Task B. When the interrupt handler wakes Task B, Task A resumes and will continue until the spinlock completes, resulting in an unpredictable jitter, denoted by the green question mark. Aside from the kernel space incompatibility, Linux already supports user space pre-emption with task priority-based real-time scheduling to allow for an RTOS. The PREEMPT-RT is a kernel patch that aims to make all kernel-space code preemptible and deterministic.

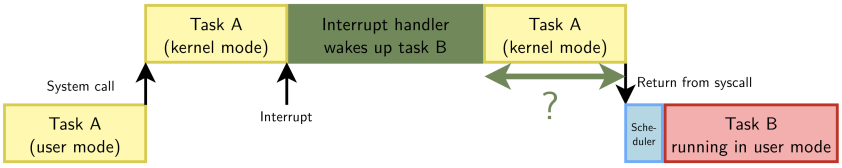
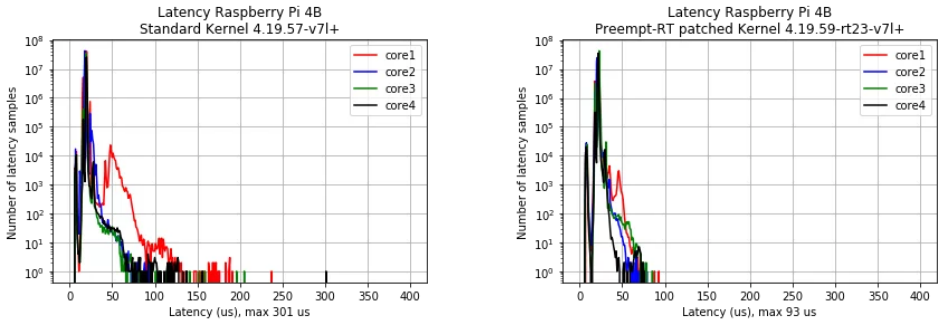


Figure 4: Illustration by [15] of the interrupt flow within two tasks, one in kernel space and another in user space.

The material in [15] also discusses the limitations of Linux and Linux-compatible hardware

with RTOS: Linux will never be a formally proven RTOS, and the hardware Linux typically runs on isn't designed with RT in mind. Although insufficient to convert Linux into a 'hard' RTOS, the PREEMPT-RT patch is a step in the correct direction to minimise task switching latencies. The PREEMPT-RT patch provides the best balance between low system complexity and maximum jitter prevention, making it the ideal choice to implement in this project. The blog post in [16], which details the installation steps for the PREEMPT-RT kernel patch in a Raspberry Pi 4 Linux system, quantifies my earlier statement. Using the RT-Tests, a test suite that contains programs to test various real-time Linux [17], measures the latency reduction with the kernel modification, the blog post [16] draws comparisons with the non-RT kernel. The Cyclictest, part of RT-Tests, accurately and repeatedly measures a thread's intended wake-up time and when it wakes up to provide statistics about the system's latencies, measuring latencies in real-time systems caused by the hardware, the firmware, and the operating system [18]. Figure 5 shows the maximum latency measurement with the standard kernel was 301 us, while the PREEMPT-RT kernel was 93 us, thus confirming a 3.23x latency reduction using the kernel patch.



(a) (b)

Figure 5: Illustration from [16] showing the latency measurements following a Cyclictest execution from the RT-Tests suite for (a) the standard Linux RPi kernel and (b) the PREEMPT-RT patch Linux RPi kernel.

2.3 Research into Predictive Systems

Following the bubble segmentation theory, using the Canny edge detection with a bubble elliptical fitting method, such as the snake, should result in a rough outline of the bubbles. From this outline, an elliptical best-fit technique can estimate the bubble shape and characteristics, such as orientation and axis, as proposed in [8] and illustrated in Figure 2 in the left-most image with the bubble bordered by a while ellipse with 'a' and 'b' denoting the dimensions. The blog post in [19] exemplifies this technique using the MATLAB 'regionprops' function [20], which automatically determines the properties of each contiguous white region that is 8-connected [21]. This technique allows for computing the exact centroid position of bubble detection. This function is not limited to MATLAB, as there are implementations for other programming languages, such as Python [22]. An assumption for this technique is the 8-connected region compliance, where from a given pixel, you can get to any other pixel in the region by a series

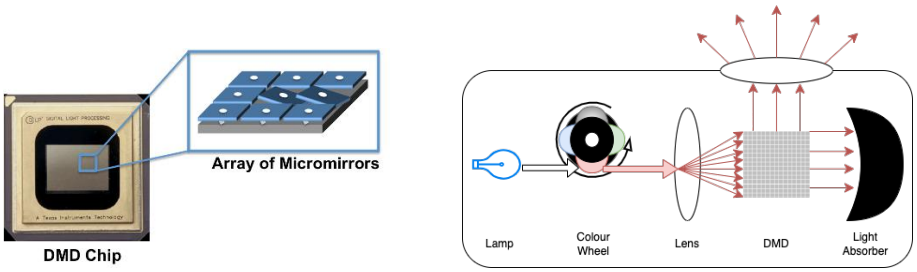
of 8-way moves (up, down, left, right, up-left, up-right, down-left, down-right) [23]. Therefore, edges that are not closed loops will not be compatible. However, given the highly optimised methods to improve the Canny edge detection technique, it is theoretically unlikely to have many open-looped detections.

By considering just the centroid of each bubble, we can reduce dimensionality, thus reducing complexity, in contrast to computing each pixel of a detected bubble border. This optimisation is crucial for efficiently storing and processing bubble positions across multiple frames. Predicting future bubble positions of the bubbles that travel in a straight line is straightforward with linear interpolation. Predicting non-straight moving bubble positions is possible by considering non-linear interpolation with spline curves. Dimensionality reduction is especially vital when considering machine learning approaches to aid in more accurate predictions and quick processing. A possible machine learning approach is with classifiers, which are algorithms that automatically order or categorise data into one or more of a set of "classes" [24]. The biggest challenge with an ML approach is generating ground-truth data for training the models since it is very labour-intensive to manually inspect each frame in example footage to select bubble locations. However, an elliptical best-fit technique can speed up this process. The project must research these methodologies in more detail.

2.4 System Building Blocks

Field Programmable Gate Arrays (FPGAs) are semiconductor devices based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects, enabling the hardware reprogramming to desired application or functionality requirements after manufacturing [25]. The flexibility of FPGA-based systems allows for very diverse and capable solutions, with their parallelism allowing for efficient computation and re-programmability for efficient use of hardware without the need to dedicate hardware for a task that may only last a few seconds [26]. Unlike FPGAs, CPUs compute sequentially, breaking algorithms into operations by sequence, thus limiting execution to one operation at a time [27]. Despite many highly efficient intellectual property (IP) cores for re-usable FPGA-based logic to accelerate FPGA development, the development and prototyping times will increase due to the low-level hardware intricacies. Furthermore, FPGAs are, on average, much more expensive than traditional CPU-based computers. The Raspberry Pi (RPi) company has been designing single-board and modular computers built on the Arm architecture and running the Linux operating system since 2012, with a mission to put high-performance, low-cost, general-purpose computing platforms in the hands of enthusiasts and engineers worldwide [28]. The regular RPi product series features single-board computers powered by Broadcom-built Arm Cortex-based multicore processors. Perfect for a non-FPGA, CPU-based route, the newer RPi models implement various features such as built-in RAM up to 8GB, a dedicated GPU and video decoder, HDMI output interfaces, dual-band WiFi with Bluetooth capabilities, a Gigabit Ethernet interface, USB 3.0 and 2.0 ports, MIPI-compatible serial interfaces for cameras and displays, and an array of 40 GPIO pins [29, 30].

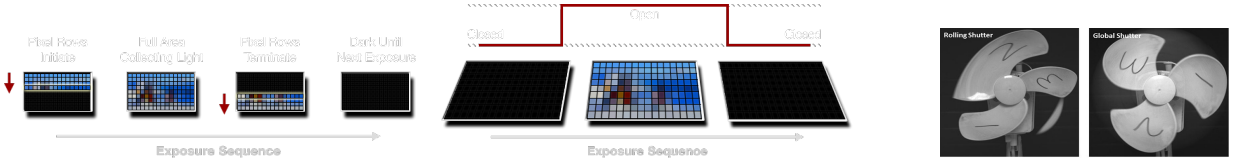
The Digital Light Processing (DLP) chipset, created by Larry Hornbeck of Texas Instruments in 1987, consists of a Digital Micromirror Device (DMD) that houses millions of reflective aluminium mirrors, usually a few microns wide [31, 32]. The DMD is a Micro-Opto-Electro-Mechanical System (MOEMS) Spatial Light Modulator (SLM) that uses digital signals to control the angle of each mirror, enabling the modulation and attenuation of an incident light beam [33]. As [34] explains the intricacies and Figure 6 illustrates, a DLP projector implements this specialised technology: The beam of a high-intensity white lamp directs into a spinning colour-tinted lens wheel consisting of red, green, and blue, and a clear lens for a white lamp pass-through. An internal lens diffracts the coloured beam incident into the DMD, with the angle and time spent in each microscopic mirror controlling the individual pixel colour and intensity. The DMD directs the desired beams of each pixel into the diffraction lens to exit the projector. The DMD will not actuate for undesired pixel beams, thus sending the beam into a light-absorbent region to prevent leakage and image distortion.



(a) (b)

Figure 6: (a) An illustration of the microscopic mirror array within a DMD chip [32]. (b) A cross-section illustrating the internals of an example projector utilising DLP technology.

The cameras in most traditional and consumer-grade electronic devices, such as phones, all employ a CMOS sensor with a rolling shutter. While these sensors are smaller and much cheaper, they cause distortion effects when capturing fast-moving subjects due to their line-by-line scan image-capturing characteristic. The article in [35] explains: Instead of having pixels from the top of the sensor switch on and work its way down like a scan, the global shutter sensor takes a snap of the scene using all of the pixels all at once. Figures 7a and 7b illustrate the differences in image capture, and Figure 7c illustrates the drastic differences in each shutter type. The Raspberry Pi company produces a global shutter camera featuring a 1.6MP Sony IMX296 sensor, with plug-and-play compatibility with RPi computers [36].



(a) (b) (c)

Figure 7: (a) An image capture timeline of a rolling shutter sensor, (b) An image capture timeline of a global shutter sensor [37], (c) The distortion in the image capture of a spinning fan due to a rolling shutter, compared to a non-distorted capture from a global shutter [38].

OpenCV is an open-source computer vision and machine learning software library with over 2500 optimised algorithms, including a comprehensive set of classic and state-of-the-art computer vision and machine learning algorithms [39]. Although native to C++, OpenCV features interfaces for Python, Java and MATLAB, ensuring compatibility with Windows, Linux, and macOS. OpenCV targets real-time vision applications by taking advantage of CPU accelerators, such as MMX and SSE instructions for parallel data processing, and GPU accelerators, with ongoing development of a full-featured CUDA and OpenGL interface. OpenCV's vast platform support, both for development and deployment, and simple and high-level wrappers for advanced computer vision algorithms and tools will drastically reduce development and prototyping duration with little performance trade-off, which for this project isn't vital. The same applies to the Python programming language, with easy-to-read and write syntax and great debugging possibilities. Python is not as efficient as C++, but as mentioned above, the level of performance that C++ provides is not a crucial objective in this project.

3 Project Objectives

3.1 Reliable and Accurate Backscatter Cancellation

The system must be able to accurately and reliably pinpoint the location of backscatter particles as well as map out the shape of the outline, enabling the perfect segmentation of each particle for elimination. Although previous work on this project [5] outlines the backscatter detection and elimination system revolving around a simple blob detection algorithm, this project will focus on a Canny-based approach with the snake method optimisation. Harnessing the simplicity of a CPU-based machine, a high-level programming language such as Python, and heavily optimised libraries from OpenCV for rapid machine vision application development, an RPi 4 or 5 with an RPi Global Shutter camera will drive a DLP projector, utilising the particle centre location and outline mapping to project holes in the light beam in addition to the implementation of position calibration, which is crucial to minimise the parallax effect, the displacement of the target from the perceived position in the video capture, ultimately ensuring accurate backscatter elimination.

3.2 Architectures and Methodologies for Real-Time

The work in [5] discusses the hindrance to the successful operation of the blob detection-based backscatter elimination software due to the effects of the Linux operating system (OS): "Linux's ability to run tasks in the background while the program is running interfered with the execution of the program, making the camera image buffer and freeze" [5]. With the analysis of RTOS in Section 2.2, I have decided that the project will be RT-driven by implementing the PREEMPT-RT kernel patch in Linux to minimise task switching latencies to mitigate the jitter issues as much as possible. Sacrificing performance for rapid development, Python enables real-time development for less stringent requirements.

3.3 Optimisations Using Predictive Systems

Although reducing the resolution of high-quality image frame captures is viable to reduce the computational overheads with machine vision applications, it often trades off object detection and segmentation accuracy. A predictive system approach can mitigate the computationally intensive requirement of machine vision application to every frame of an input video feed. Accurate backscatter particle movement and future location prediction eliminate the requirement to apply machine vision-based technologies to every frame in the video feed to identify and segment the particles. Initially, the project will implement a non-ML approach using interpolation with splines, which I must research in more detail as the project progresses. With interpolation in place, I must research ML-based classifier approaches with thought into generating the training data as the project progresses.

4 Approach Description

The three sequential project objectives power the overarching software development process by providing major version milestones. Following the software version milestones, each development stint will consist of subtasks with a scope for ongoing research. The prior completion of many project aspects, such as crucial background research, finalising and ordering parts, and some software development, makes room for much experimental work.

4.1 Milestone 1: Basic Canny-Based Backscatter Segmentation (Software V1)

With the technical background research in place, I have developed a software prototype that uses the Canny edge detection algorithm, utilising GoPro footage of the sea floor. Due to the rolling-shutter sensor on board the GoPro and intense video compression, this footage has many artefacts and image distortion. The first step forward is to utilise the underwater testing facilities at the Institute for Safe Autonomy (ISA) to generate uncompressed footage of a stream of backscatter using the RPi global shutter camera system. So far, I have developed a Python script intended for the RPi computers with an installed RPi global shutter camera to record a video feed. Before recording, I must allocate sufficient time to update this script to record in raw format, as it currently uses H.264 encoding, and to set up the recording rig to use at the testing facilities. In parallel, I must research metrics for quantifying the real-time performance of the software to prepare for the V1 software requirements: (a) Canny-based backscatter segmentation, optimised with the test footage recording, and (b) real-time metrics tracking logic. Although much time spent researching hardware compatibility across the components, such as the RPi computer, camera, and projector, there may be unfound issue risks as it's the first time I'm putting this together. I must develop modular software to ensure future feature implementations will be seamless and will not require re-writing large code sections, thus resulting in longer development time, with the risk of backlogs.

4.2 Milestone 2: Advanced Canny-Based Backscatter Segmentation With Tracking (Software V2)

Following the first milestone, the RPi test systems, consisting of the Pi 4 and the Pi 5, must contain the PREEMPT-RT kernel patch. The implementation must house simple logic for rapid switching between the PREEMPT-RT kernel and the standard kernel, aiding for quick parameterisation of performance. On the software side, the first requirement for V2 is the snake method implementation for a more accurate backscatter segmentation. In parallel, the second requirement is the research and development of the least distance rule and Kalman filter methodologies for backscatter tracking. In a fast-paced environment, such as underwater from a UUV, where the rapidly moving propellors disrupt backscatter particles, it may be worth considering the computational performance and development time trade-offs with the Kalman filter-based tracking approach when the least distance approach may suffice. The final requirement of this software version is the mitigation of parallax effects, in addition to implementing logic to drive the DLP projector, after which the real-time performance of backscatter cancellation between the standard and PREEMPT-RT kernel is ready for evaluation. Due to the nature of software development, there is a risk of long programming times since these tracking methods are very complex.

4.3 Deliverable 1: Demonstration of Software V2

The completion of V2 from the previous milestone showcases a fully-fledged Canny edge detection-based backscatter cancellation system. A department-wide project demonstration to industry professionals on the 29th of April, 2024, forms the ideal platform to exhibit the project and to welcome feedback for improvement.

4.4 Milestone 3: Predictive Backscatter Segmentation Approaches (Software V3)

In parallel with the demonstration day preparation for milestone 2, there must be a focus on the third and final software iteration. Consisting of researching, implementing, and evaluating non-ML approaches, such as interpolative methodologies, with ML-based approaches, fulfils the requirements of the final software version. Research into layering the predictive logic on machine-vision technologies to enhance system efficiency by assessing real-time metrics will complete the third objective of this project. As with all software-related tasks, there is a risk of slow development progress due to previous backlogs or challenging implementations. It is vital to consider the time it may take to train an ML model between each trial run for fine-tuning the model hyperparameters. With the completion of V3, underwater testing can occur with scope for real-life application with system deployment into a lake, if time permits.

4.5 Deliverables 2 & 3: The Final Report, Project Presentation, and Viva Voce

Only when the final testing runs and the acquisition of project evaluation material, such as real-time metrics, is complete can writing the final project report proceed. Due to the underestimated difficulty and time consumption of articulating metrics, results, and summaries, the plan must consider the risk of slowdowns when writing the report and the risk of illness. The same applies to the final presentation and project viva.

5 Project Timetable

Figure 8 portrays a Gantt chart outlining the approach from the section above. At the end of each week on Friday, in parallel with the weekly email updates, I will evaluate this schedule to update the chart with any progress changes. I intend to approach this schedule as a 'live' document with constant change by preserving a complete version history.

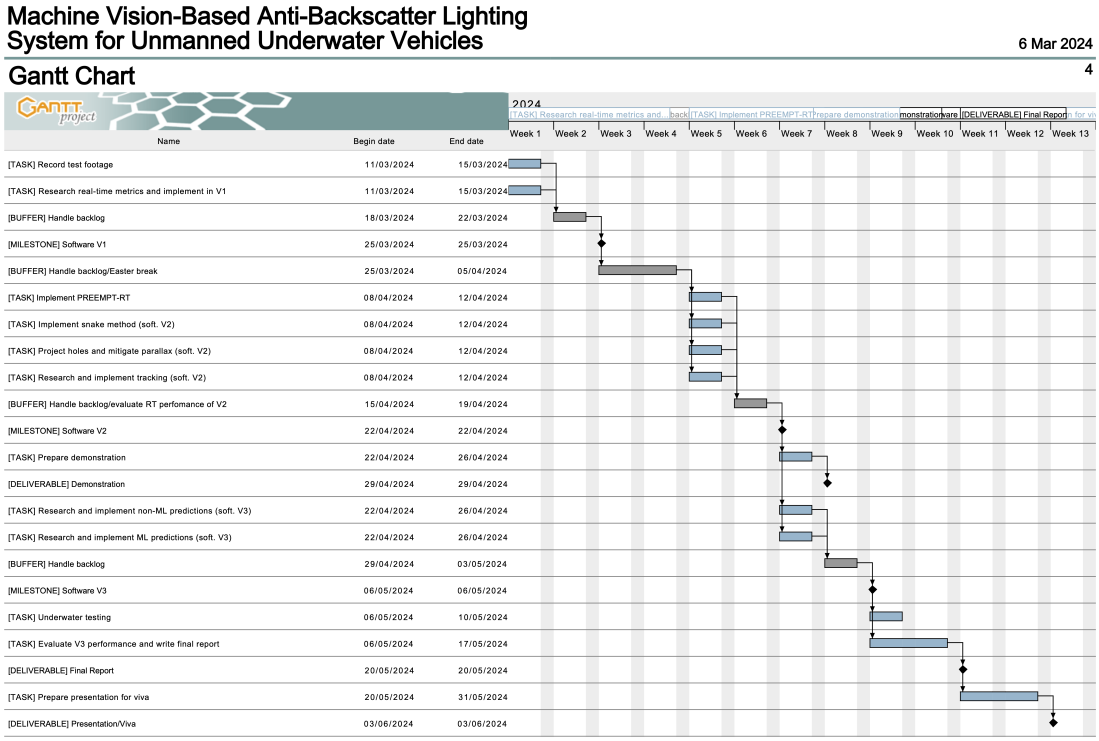


Figure 8: A Gantt chart illustrating the overall project schedule plan.

6 Conclusion

This report has described the current progress status, outlining the ultimate goal, the aims, and an investigation of background information which has decided some project choices. The report has also discussed the plan for the remaining tasks, illustrated with a detailed and active Gantt chart schedule.

References

[1] University of Hawai'i, "Practices of Science: Underwater Photography and Videography," https://manoa.hawaii.edu/exploringourfluidearth/physical/ocean-depths/light-ocean/practices-science-underwater-photography-and-videography, [Accessed February 13, 2024].

[2] Yannick Allard and Elisa Shahbazian, *Unmanned Underwater Vehicle (UUV) Information Study.* Defence Research & Development Canada, Atlantic Research Centre, Nov. 2014, [Accessed February 14, 2024].

[3] Brent Durand, "Easy Ways to Eliminate Backscatter in your Photos," https://www.uwphotographyguide.com/eliminate-backscatter-underwater, Oct. 2013, [Accessed February 14, 2024].

[4] Y. Zhang, Y. Yu, X. Rui, Z. Feng, J. Zhang, Y. Chen, L. Qi, X. Chen, and X. Zhou, "Underwater bubble escape volume measurement based on passive acoustic under noise factors: Simulation and experimental research," *Measurement*, vol. 207, p. 112400, Feb. 2023, [Accessed February 24, 2024].

[5] Katie Shepherd, "Machine Vision Based Underwater Anti-Backscatter Lighting System," MEng Project Report, University of York, York, UK, 2023, [Accessed September 1, 2023].

[6] OpenCV, "OpenCV: Cv::SimpleBlobDetector Class Reference," https://docs.opencv.org/3.4/d0/d7a/classcv_1_1SimpleBlobDetector.html, [Accessed March 5, 2024].

[7] theodore, "What exactly is a Blob in OpenCV ? - OpenCV Q&A Forum," https://answers.opencv.org/question/50025/what-exactly-is-a-blob-in-opencv/, [Accessed March 5, 2024].

[8] K. Thomanek, O. Zielinski, H. Sahling, and G. Bohrmann, "Automated gas bubble imaging at sea floor - a new method of in situ gas flux quantification," *Ocean Science*, vol. 6, no. 2, pp. 549–562, Jun. 2010, [Accessed February 24, 2024].

[9] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986, [Accessed February 24, 2024].

[10] C. Zelenka, "Gas Bubble Shape Measurement and Analysis," in *Pattern Recognition*, X. Jiang, J. Hornegger, and R. Koch, Eds. Cham: Springer International Publishing, 2014, vol. 8753, pp. 743–749, [Accessed February 26, 2024].

[11] N. Hansen and S. Kern, "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. Lecture Notes in Computer Science, X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A.

Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, Eds. Berlin, Heidelberg: Springer, 2004, pp. 282–291.

[12] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, [Accessed February 28, 2024].

[13] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, Mar. 1955, [Accessed February 28, 2024].

[14] NlightNFotis, "Answer to "What is difference between User space and Kernel space?"," Aug. 2013, [Accessed March 6, 2024].

[15] Bootlin, "Understanding Linux real-time with PREEMPT_RT training," Jan. 2024, [Accessed January 31, 2024].

[16] Mauro Riva, "Raspberry Pi 4B: Real-Time System using Preempt-RT (kernel 4.19.y)," Sep. 2019, [Accessed March 1, 2024].

[17] Costa Shul, "RT-Tests," https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/rt-tests, Sep. 2023, [Accessed March 1, 2024].

[18] Costa Shul, "Cyclictest," https://wiki.linuxfoundation.org/realtime/documentation/howto/tools/cycl
Aug. 2023, [Accessed March 1, 2024].

[19] Steve Eddins, "Ellipse visualization and regionprops," Aug. 2015, [Accessed March 6, 2024].

[20] "Measure properties of image regions - MATLAB regionprops - MathWorks United Kingdom," https://uk.mathworks.com/help/images/ref/regionprops.html, [Accessed March 6, 2024].

[21] rayryeng, "Answer to "Explanation of Matlab's bwlabel,regionprops & centroid functions"," Sep. 2014, [Accessed March 6, 2024].

[22] "Measure region properties — skimage 0.22.0 documentation," https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_regionprops.html, [Accessed March 6, 2024].

[23] "Course Note of Computer Graphics Chapter 5," https://redirect.cs.umbc.edu/~ebert/435/notes/435_
[Accessed March 6, 2024].

[24] Tobias Geisler Mesevage, "Machine Learning Classifiers - The Algorithms & How They Work," https://monkeylearn.com/blog/what-is-a-classifier/, Dec. 2020, [Accessed March 6, 2024].

[25] "What is an FPGA? Field Programmable Gate Array," https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html, [Accessed February 23, 2024].

[26] C. Villalpando and R. Some, "Reconfigurable machine vision systems using FPGAs," in *2010 NASA/ESA Conference on Adaptive Hardware and Systems*, Jun. 2010, pp. 31–35, [Accessed February 23, 2024].

[27] Alex Liang, "Boosting Machine Vision with Built-in FPGA Image Preprocessing," Measurement & Automation Product Segment, ADLINK Technology, White Paper, May 2016, [Accessed February 23, 2024].

[28] Raspberry Pi, "Raspberry Pi - About us," [Accessed February 23, 2024].

[29] Raspberry Pi Ltd, "Buy a Raspberry Pi 4 Model B," https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, [Accessed February 23, 2024].

[30] Raspberry Pi Ltd, "Buy a Raspberry Pi 5," https://www.raspberrypi.com/products/raspberry-pi-5/, [Accessed February 23, 2024].

[31] "Digital Light Processing," *Wikipedia*, Feb. 2024, [Accessed February 24, 2024].

[32] "How does a DLP projector work?" https://www.projectorscreen.com/blog/How-does-a-DLP-projector-work, [Accessed February 24, 2024].

[33] "DLP4500 Digital Micromirror Device - TI | Mouser," https://www.mouser.co.uk/new/texas-instruments/ti-dlp4500-micromirror-device/, [Accessed February 24, 2024].

[34] "How DLP projector technology works - YouTube," https://www.youtube.com/, [Accessed February 24, 2024].

[35] U. D. last updated, "What is a global shutter – and why is it so important?" https://www.digitalcameraworld.com/news/what-is-a-global-shutter-and-why-is-it-so-important, Feb. 2021, [Accessed February 24, 2024].

[36] Raspberry Pi Ltd, "Buy a Raspberry Pi Global Shutter Camera," https://www.raspberrypi.com/products/raspberry-pi-global-shutter-camera/, [Accessed February 24, 2024].

[37] RED Digital Cinema, "Global & Rolling Shutters," https://www.red.com/red-101/global-rolling-shutter, [Accessed February 24, 2024].

[38] "Rolling Shutter vs Global Shutter sCMOS Camera Mode," https://andor.oxinst.com/learning/view/article/rolling-and-global-shutter, [Accessed February 24, 2024].

[39] OpenCV, "About," https://opencv.org/about/, [Accessed March 1, 2024].