Final Project

34241 - Digital video technology

Rasmus A. Lindholdt - s200745
Andreas Wendelboe - s215489

34 Department of Electrical and Photonics Engineering
May 16, 2025

Contents

1 - Introduction

UVision ApS has developed an underwater stereo-visual photogrammetry system capable of generating high-fidelity 3D models of subsea assets. Their technology employs dual cameras capturing synchronized image pairs, which are then processed through photogrammetry software to create detailed digital reconstructions.

The current pipeline faces several optimization challenges that limit its effectiveness:

- Limited frame rate (5 FPS) restricting data capture efficiency
- Substantial scan file sizes exceeding 10GB per hour of operation
- Processing bottlenecks within their cloud-based reconstruction workflow

To address these limitations, we have developed two key technological improvements:

1. Video-Based Photogrammetry Pipeline: Transitioning from individual JPEG storage to efficient video encoding while maintaining stereo synchronization
2. Intelligent Frame Selection Algorithm: Optimizing the input to photogrammetry software by eliminating redundant images

The current hardware consists of an NVIDIA Jetson Xavier NX, mounted on an Auvidea carrier board. The setup also includes 2 HD cameras connected to a DepthAI FFC board, which handles synchronizing the images and other image processing tasks. This is connected via USB to the Jetson.

1.1 - Work Distribution

Team Member	**Responsibilities**
Rasmus A. Lindholdt (s200745)	Intelligent Frame Selection Algorithm
Andreas Wendelboe (s215489)	Video-Based Photogrammetry Pipeline Generating Metashape models

Table 1: Project Work Distribution

2 - Implementation

The implementation section will be split into the two parts mentioned in the introduction - video-based photogrammetry pipeline & intelligent frame selection algorithm.

2.1 - Video-Based Photogrammetry Pipeline

Instead of modifying an existing UVision scanner, we chose to speed up development by creating a working prototype for dataset generation. We started with a standard Jetson setup based on the DepthAI Quickstart guide.

A - Video Format Selection

We had to decide which video format the DepthAI FFC board should output to the Jetson. After looking at different options like H.265, H.264, and MP4, we chose H.265 because it compresses files better than H.264 and the file isn't corrupted during unexpected shutdowns or crashes (unlike MP4). This way, we can keep the original files but still convert them to MP4 later if someone needs to watch the video directly.

B - Synchronization Implementation

Achieving synchronization between the dual cameras presented significant technical challenges. Our initial approach positioned the DepthAI sync node after the video encoders, with the bitstream as encoder output. Despite debugging efforts that indicated consistent frame sizes and frame numbers, the resulting files showed substantial size differences.

Through systematic experimentation, we discovered that positioning the encoder after the sync and demux nodes resolved the synchronization issues. This suggests that the Sync node was potentially disrupting the encoded stream by holding back frames during synchronization.

C - Video Encoder Configuration

We explored the VideoEncoder's output types (bitstream and out) which are mutually exclusive. Though documentation was limited, our testing determined the optimal configuration for our use case.

The current JPEG-based setup at UVision operates at approximately 8 FPS with each JPEG file around 500-700 KB, resulting in a bitrate of 30-45 Mbps. With a stereo camera pair, this means a one-hour-scan produces around 30 gigabytes of data.

We extensively tested various encoder settings to optimize performance:

Framerate Increased from 5 FPS to 10 FPS to capture more detailed data.

Keyframe Frequency Tested both configurations where every frame is an I-frame (frequency=1) and where keyframes occur once per second (frequency=FPS).

B-frames Maintained at zero to reduce temporal compression artifacts that could affect photogrammetry accuracy.

Rate Control Mode Shifted from Constant Bitrate (CBR) to Variable Bitrate (VBR) to prioritize consistent quality, which is preferable for photogrammetric reconstruction.

Quality Settings Empirical testing demonstrated that a quality setting of 50 with VBR yielded a bitrate of approximately 3.3 Mbps, reducing a one-hour-scan from 30GB to just 3GB while maintaining sufficient quality for photogrammetric processing.

D - Camera Settings Optimization

Beyond encoding parameters, we optimized camera settings for the underwater environment:

Exposure Control Limited exposure to 10ms to minimize motion blur, which was causing issues with the default auto-exposure algorithm. While this increased the bitrate by approximately 1.5x at the same quality setting (likely due to additional ISO noise creating more entropy in the video), the improvement in image clarity justified this tradeoff.

Underwater Configuration For field testing in underwater conditions, we:

- Set focus distance to 8cm to accommodate dome ports
- Maintained the 10ms maximum exposure setting
- Set white balance to 5000K to match dive lights
- Applied chroma denoise at level 4, consistent with UScanner settings
- Installed the Jetson in the UScanner underwater housing

2.2 - Intelligent Frame Selection Algorithm

The intelligent frame selection algorithm (IFSA) should detect when significant movement in the video feed has occurred, and note down the frame numbers where movement has happened.
It is not known, when *significant* movement is made, as it all depends on how the 3D model generation via Metashape (see section 3.3), reads the input. If the movement is too significant, the alignment will lose track and be unable to create the 3D model.

To determine the motion between frames in a video, the temporal information (TI) feature was the chosen metric. TI is based on the temporal perceptual information measurement, described in ITU-T standard, section 5.3.2 [4] and is a measure of movement between two frames. The more movement, the higher the TI. For reference, the average TI in the two sample videos (left & right channel) for this project, is around 37. Other sample videos (from an actual UVision customer) had an average TI of ≈ 4 - meaning much slower movement than the sample video used for this project.

Due to the stereo video feed, a common feature of all the algorithms below, is to find the lowest common denominator. E.g. if the frames from the left camera exceeds the TI threshold, the frames are saved in both the left and right channel - and vice versa.

A - Algorithm 1

Algorithm 1 is an algorithm that reads a video file and ends up exporting a list of .jpg files for Metashape. See figure 1 for the complete process. FFmpeg[1] is used to convert the .h265 video input to a .mp4, as well as for export of .jpg images from the mp4 file.

Figure 1: Algorithm 1 process of going from .h265 video to list of relevant .jpg images

To evaluate the TI values between each frame (step 3 in figure 1), we explored a repository called siti-tools by VQEG [2]. siti-tools will export the spatial information (SI) as well as the temporal information (TI) between each frame, through a python script. This is not possible through FFmpeg - FFmpeg will only print a summary of the entire input file. This is also the reason the video has to be converted to a .mp4 file, as the siti-tools library does not read .h265 files.

After computing all TI values via siti-tools, algorithm 1 for selecting frames is run. The TI between each frame is then compared to a threshold, as well as a threshold for the cumulative sum of TI values, illustrated in figure 2. This illustrates that if a single TI value exceeds the threshold, the frame is saved. Different thresholds are used for the single frame and the cumulative sum.

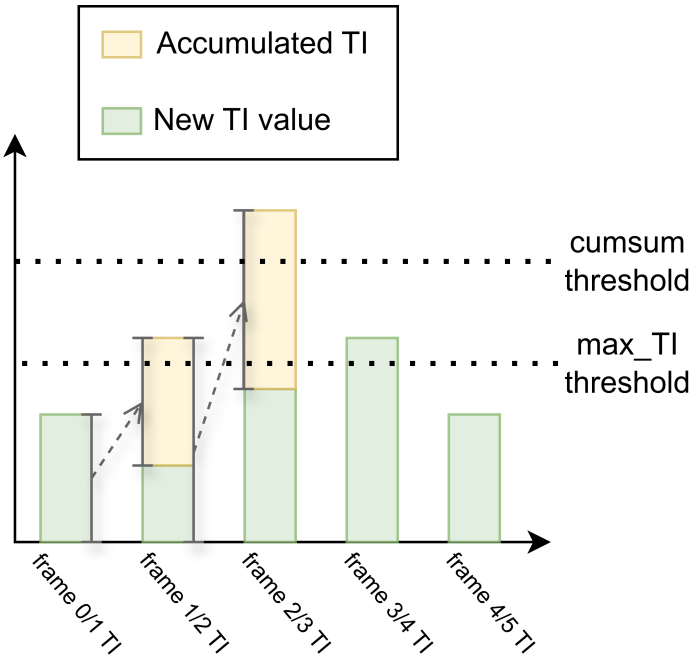

Figure 2: Illustration of algorithm 1, TI values exceeding threshold

In this case, frame 0, frame 2 as well as frame 3 is saved. Frame 2 due to the cumulative sum of TI values, and frame 3 as it exceeds the max_TI threshold. The cumulative sum is reset upon every saved frame.

A state diagram of algorithm 1 is found in figure 3. This will run for the entire length of the video - 1 frame due to the TI value being an inter-frame metric.

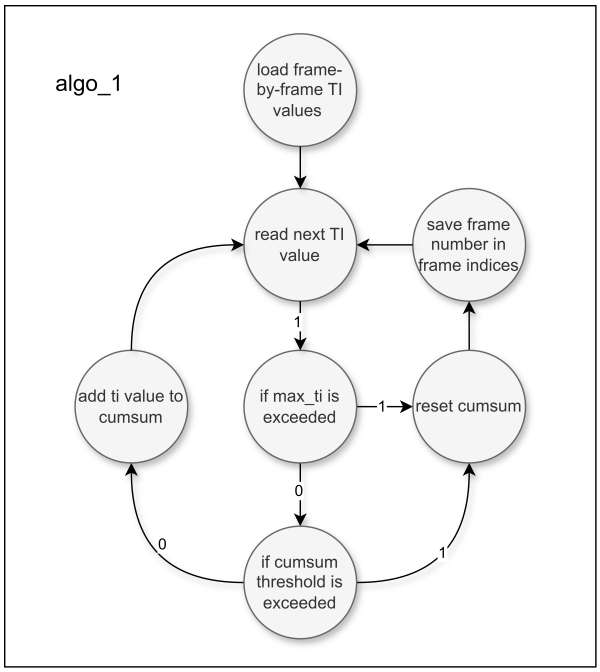

Figure 3: State diagram of algorithm 1

The export settings of the pictures are controlled through the `qscale` option, default ranging from 2-31, where 2 is the highest quality. For this project, the qscale is set to 2, resulting in an image size of roughly 300kB for each full-HD frame - the same quality will be used in the following algorithms.

B - Algorithm 2

While algorithm 1 worked, we realized that it may be more relevant to look at the TI value between the last saved frame, and the next potential frame to save. E.g. if the camera moves around slightly in the same spot for a while, the cumulative sum threshold might be triggered in algorithm 1, while the actual movement has been minimal. This is why we chose to develop algorithm 2 - an algorithm looking at relative TI between last selected frame and next potential frame.

This is illustrated in figure 4. Note the labels on the x-axis, comparing frame 0 to the next 3 frames, before continuing.

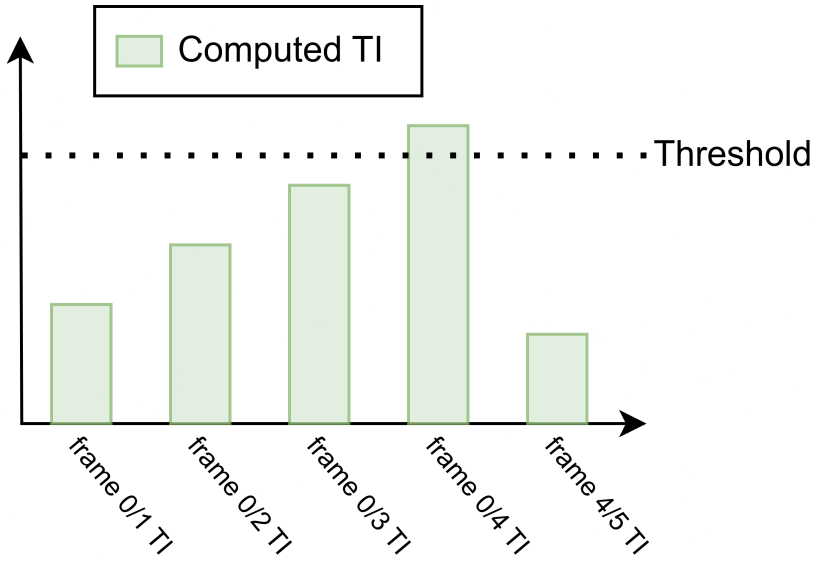

Figure 4: Illustration of algorithm 2

In this case, frame 0 and frame 4 are saved. The runtime of algorithm 2 will be the same as algorithm 1, as both algorithms make TI calculations of n number of frames. Runtime is $O(n)$.
In practice, it is significantly faster running algorithm 2, as this does not use the siti-tools library or has to compute the mp4 file. To generate the TI between two frames, FFmpeg takes two .jpg files and outputs the relative TI. See script `algo4.sh` on github repository [3].

As seen in figure 5, the process is simplified significantly in algorithm 2, compared to algorithm 1 and does not rely on external libraries like siti-tools. Only FFmpeg is used

in algorithm 2.

Figure 5: Algorithm 2 process of going from .h265 video to list of frames

2.3 - Algorithm 2.1

Finally, an algorithm was developed based on algorithm 2, but with a cumulative TI counter, like in algorithm 1. As section 3.3 shows, algorithm 2 caused some issues, when the max_TI was set too high. This is why we at last chose to reimplement the cumulative TI counter in algorithm 2.1. This acts as a safety-net, if the max_TI is set too high. Results of adding the cumulative counter follow in section 3.1.B. In this algorithm, the cumulative TI threshold is proportional to the average TI value for the entire video file. The cumulative threshold could e.g. be set to 3 times the average TI for the full video sample.

3 - Results

3.1 - Intelligent Frame Selection Algorithm

As mentioned in section 2, the frame selection algorithm relied on temporal information, to remove irrelevant frames. In this section, we will focus on algorithm 2 and 2.1, as we believe these are a the best versions. It has been difficult to have those beliefs verified, as Metashape has not been available around the clock - and especially not towards the end of the project. This is why we mostly rely on a subjective evaluation of the recreated videos.

A - Image generation

It is previously mentioned we chose the highest jpg quality setting when exporting the chosen frames from the video input; 2 out of 31. Figure 6 shows how the quality changes from 2 to 15 to 31. Even though the images below are cropped to roughly 1/4 of the original frame, it is not easy to distinguish the quality between the three quality settings. Quality setting 31 does show a bit of pixelation, but nothing major. Future work would include testing the 3D model with different quality settings.

(a) Quality setting 2 (b) Quality setting 15 (c) Quality setting 31

Figure 6: Comparison of different quality settings cropped significantly

Table 2 shows how the quality setting impacts the file size.

Quality setting	**Total size (1616 frames)**	**Average frame size**	**Relative size (vs. quality 2)**
2	478MB	301.36 kB	1:1
15	145MB	89.79 kB	3.35:1
31	98MB	60.83 kB	4.95:1

Table 2: Comparison of file sizes for varying quality settings

From table 2 it is clear how the quality setting has a big impact on the file-size, while the quality difference between 2 and 15 is nearly indistinguishable to the human eye (even on a big screen)[1]. Going from quality 15 to 31 shows pixelation with only a 30% relative reduction in file size. As UVision expects the quality to be remarkable, quality setting 31 simply is not good enough.
We did not have time to test how the quality impacted the 3D model, but we expect the runtime of Metashape to be mostly linear with the total filesize.

B - Frame selection

The original sample video consists of 1616 frames, recorded at 10 fps. Section 3.3 will dive deeper into which models were actually tested in via Metashape. This section will be a subjective evaluation of recreated videos from the frame generation.

Below is a list of created videos including the number of frames, plus some subjective comments on the recreated video. The most interresting model to upload to Metashape, would be IDs #6 through #8, starting from the bottom of the list. If neither of them are successful, another measure for measuring movement might have to be introduced in combination with the TI measure. Common for all models is that the reduction in total number of frames is significantly more than 50%. The only verified 3D model is ID#1 and has a reduction of 56%. We believe this could be improved significantly, probably down to 4-500 frames with an optimized algorithm. This would be up to a 4x reduction in frames.

ID	Algo	max_TI	Max cumsum	Frames selected	Subjective evaluation	3D verified
#1	1	25	60	707	*Small jumps, watchable*	Yes
#2	2	20	∞	915	*Medium jumps*	No
#3	2	22	∞	623	*Medium jumps*	Failed
#4	2	24	∞	370	*Medium jumps*	Failed
#5	2.1	25.1	5· max_TI	412	*Medium jumps*	No
#6	2.1	25.1	3· max_TI	499	*Fast movement, watchable*	No
#7	2.1	22.5	5· max_TI	621	*Small jumps, fast movement*	No
#8	2.1	22.5	3· max_TI	687	*Some fast movement, watchable*	No

Table 3: Overview of algorithm and frames

[1]Larger images are found in appendix A

When looking at the frame selection of ID 6, 262 of the 499 frames were chosen from the max_TI threshold, while the rest were chosen from the cumulative TI threshold. This, as well as TI triggers for other configurations of algorithm 2.1 is found in figure 7.

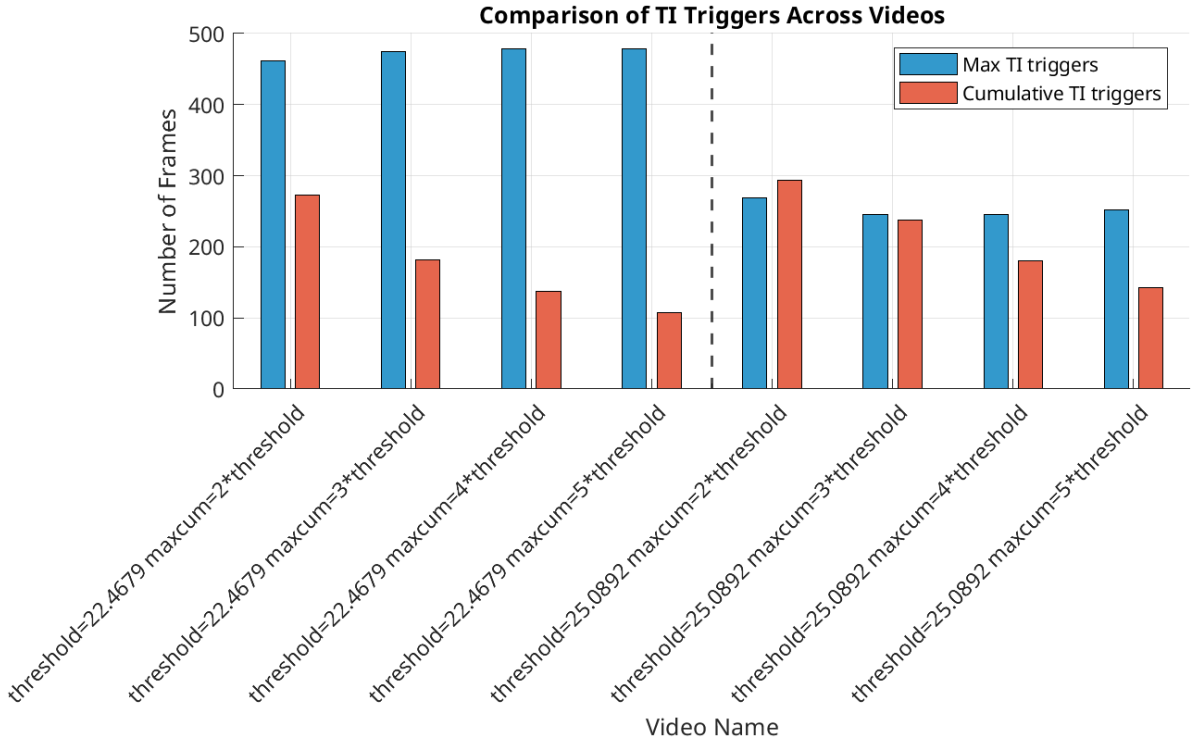

Figure 7: Comparison of TI triggers of algorithm 2.1

Figure 7 shows how the frame selection is split between the max_TI threshold and cumulative threshold and is split into a left and right section. To the left of the line, the max_TI threshold is ≈ 22.5 , while the threshold on the right hand side is ≈ 25.1 . These are based on the average TI for the full sample video, the left and right side being 0.6 and 0.67 of the average TI, respectively. This multiplier is one of the most important parameters to adjust, for future adjustment. Furthermore, if a scenechange is detected, the average TI should be recomputed.

3.2 - Video-Based Photogrammetry Pipeline

A - Encoding Performance

Our implementation demonstrated substantial improvements over the existing JPEG-based system:

Configuration	FPS	Rate Control	Quality	Key Freq.	Bitrate (Mbps)
JPEG (Current)	8	-	-	-	30-45
H.265 CBR	10	CBR	-	10	4.72
H.265 VBR Q100	10	VBR	100	10	22.74
H.265 VBR Q80	10	VBR	80	10	22.30
H.265 VBR Q50	10	VBR	50	10	3.31
H.265 VBR Q0	10	VBR	0	10	0.06

Table 4: Comparison of encoding configurations and resulting bitrates

The optimal configuration (VBR with quality 50) achieved a compression ratio of approximately 10:1 compared to the original JPEG approach while increasing the frame rate from 8 to 10 FPS. CPU utilization dropped from 30% with the JPEG method to just 1.6% with our H.265 encoding approach, despite the higher frame rate.

B - Field Testing

We conducted field tests in the canals at Gammel Strand, employing both horizontal scans along the canal and a lawnmower pattern (moving up and down) similar to standard UScanner operation. The system performed as designed with no technical issues, validating our implementation in a real-world environment.

3.3 - Model Generation

To test if our video-based approach for photogrammetry works well, we ran the footage through Metashape using different frame selection methods. Metashape attempts to track the camera's movement during recording, which is called alignment. Sometimes Metashape loses track and can't align some images. When we tested these downsampling algorithms, we found that images typically fail to align when we downsample too aggressively. The tests are described below:

Processing All Frames With 1,616 total frames, the alignment and model building process took over 4 hours, with all frames being successfully aligned. Texture generation required an additional 25 minutes.

Figure 8: Model generated using all frames

Fixed downsampling This approach kept every 10th image, resulting in a 1 FPS output. Only 21/160 frames ended up being aligned, and the model generation failed. This was too coarse. Due to time constraints, different versions of this algorithm with less aggresive downsampling was not tested.

Sequential Cumsum Processing 1,277 frames with the lawnmower pattern resulted in 1,173 successfully aligned frames over approximately 2.5 hours of processing.

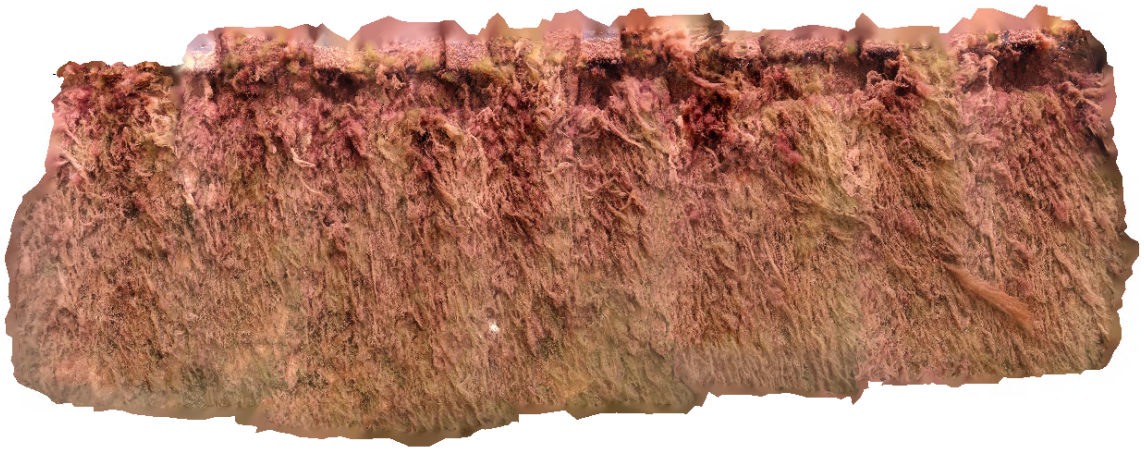

Figure 9: Model generated using sequential cummulative sum algorithm

Optimized Sequential Cumsum Using TI=25 and Threshold=60 parameters, we processed 707 frames over approximately 1.75 hours with no alignment problems and produced a high-quality model. This is ID #1 in table 3.

Adaptive Reference Frame Tests with this approach at various thresholds (22-24) without cumsum were less successful, with only a small percentage of frames aligning

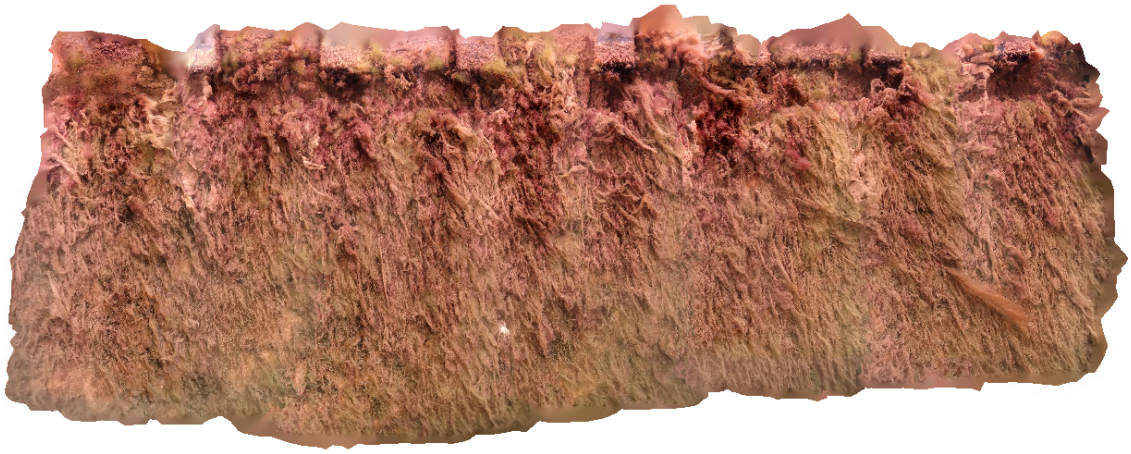

Figure 10: Model generated using optimized sequential cummulative sum algorithm

properly. This is IDs #3 and # 4 in table 3. No model was generated.

Looking at the 3D models we created, when the photos align properly, the quality is consistently high. These models meet UVision's quality standards very well. The video-based approach has clearly succeeded based on these results. We can adjust the downsampling to reduce the number of images needed for a good model, and we can optimize this process even more in the future, so they still manage to align.

4 - Improvements / Future work

4.1 - Integrating video pipeline with current software

Currently, the video pipeline was set up on an empty NVIDIA Jetson with non of UVision's other software. This software handles tasks such as generating a live point cloud and position track, starting and stopping recordings from a customer GUI, transferring videos from scanner to laptop, etc. In the future, the current video pipeline will have to be integrated with all this.

4.2 - Tuning the IFSA

The tuning of the IFSA has been quite limited due to the limited runtime on Metashape. While the algorithm works as expected, and has output a successfull model, it is difficult to determine how rough the algorithm can be and how the parameters should be set. Some of the interresting tests for the future would be:

- Testing model with varying image quality settings.
- Testing varying thresholds, both max TI and cumulative threshold
- Testing with other movement measures, like PSNR, SSIM, raw pixel difference or motion vectors.

5 - References

[1] FFmpeg developers. *FFmpeg*. hhttps://ffmpeg.org/. Accessed: 2025-05-16. 2025.

[2] Video Quality Experts Group. *siti-tools: SI TI calculation tools*. https://github.com/VQEG/siti-tools. Accessed: 2025-05-15. 2023.

[3] Rasmus A. Lindholdt - s200745. *Github repo for final project files*. https://github.com/McQueen24/34241_final-project. Accessed: 2025-05-16. 2025.

[4] *Subjective video quality assessment methods for multimedia applications*. Recommendation P.910. ITU-T, 2021.

Appendix A

Full size images of jpg quality assessment

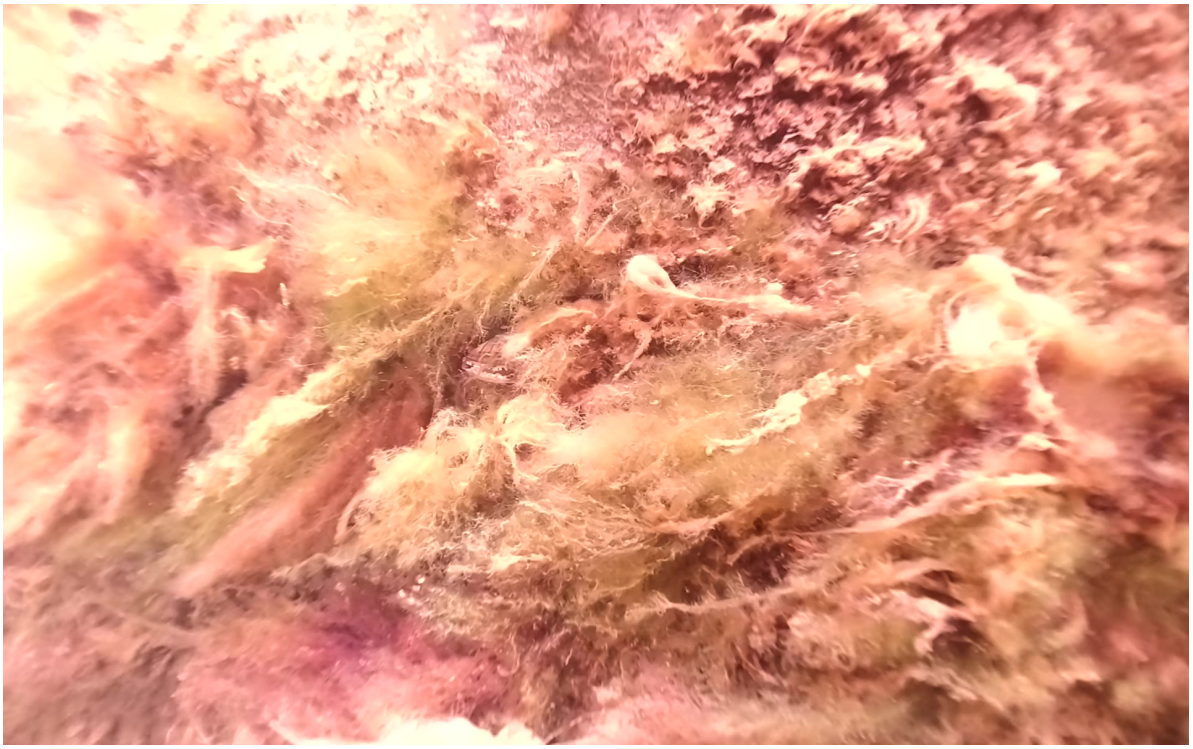

Figure A.1: Quality setting 2

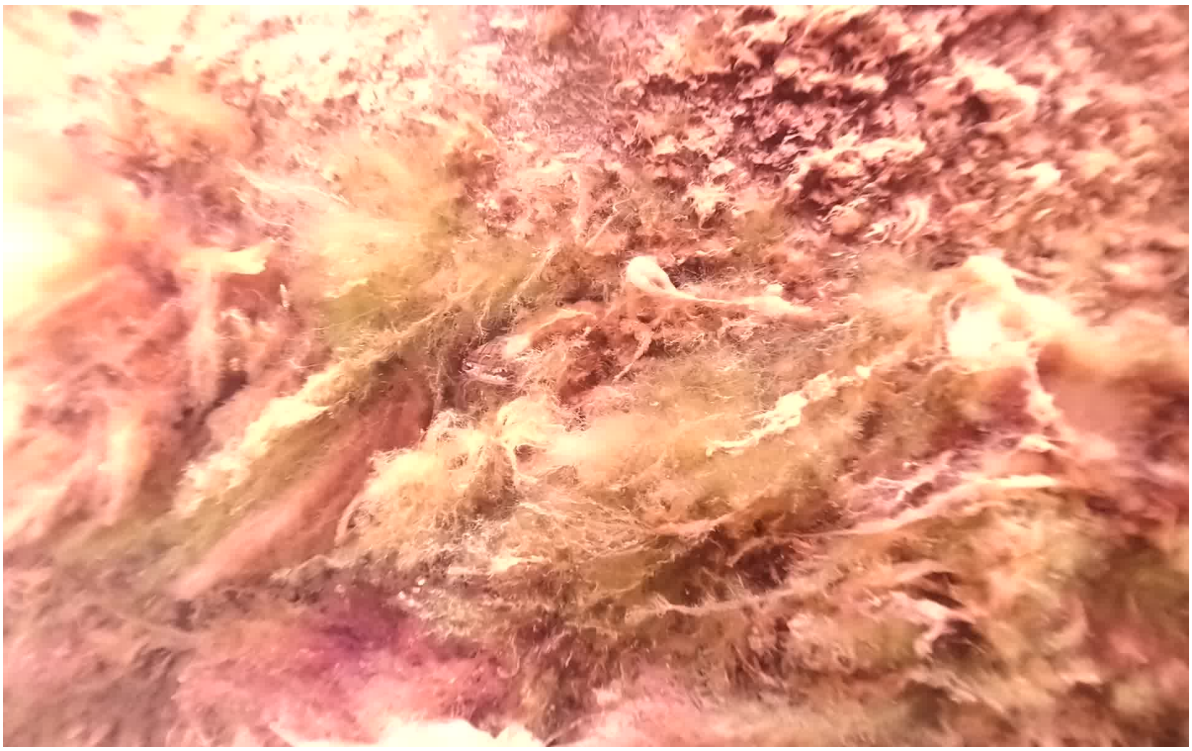

Figure A.2: Quality setting 15

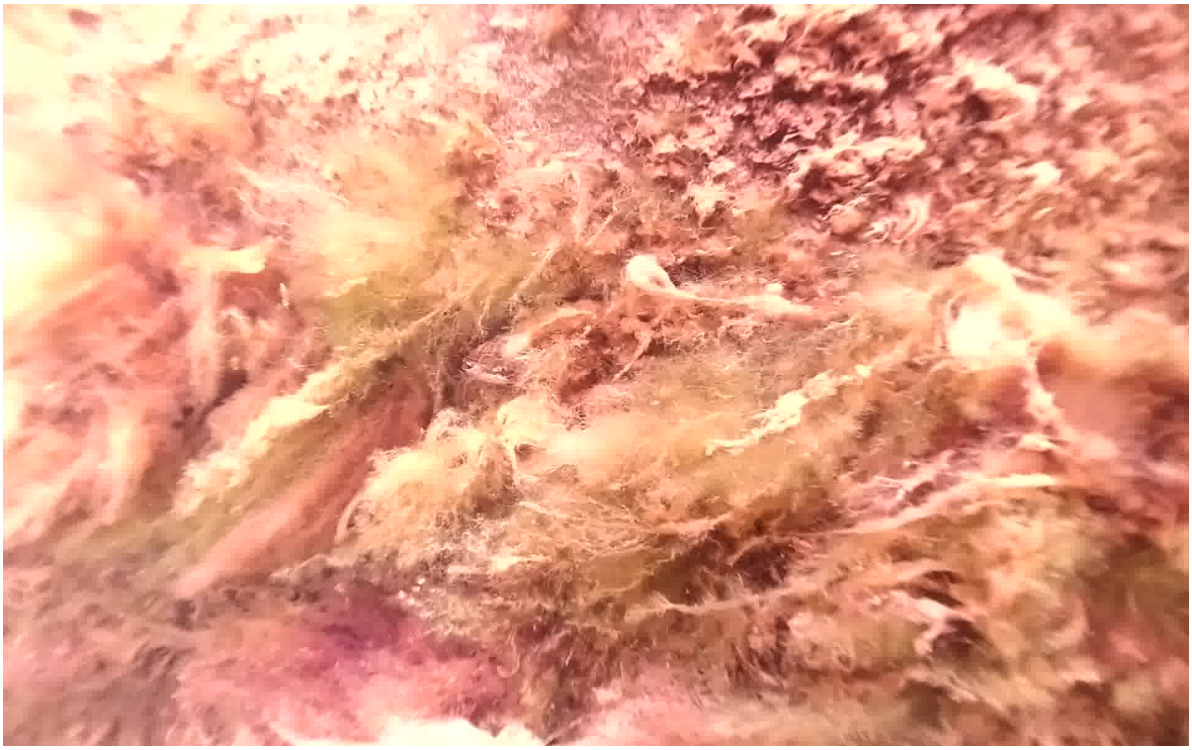

Figure A.3: Quality setting 31