Universiteit Leiden

Evaluating the Impact of Underwater Image Enhancement on Object Tracking for Autonomous Underwater Vehicles

Fiona Moerman (s3709108)

Supervisors:
Rita Pucci & Shima Javanmardi

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

July 2, 2026

Abstract

Underwater image degradation caused by light absorption and scattering poses significant challenges for computer vision tasks in autonomous underwater vehicles (AUVs), particularly object tracking. To mitigate these effects, a wide range of underwater image enhancement (UIE) methods have been proposed; however, their impact on downstream vision tasks remains unclear, as most evaluations rely on perceptual image quality metrics rather than task-specific performance. This thesis investigates whether underwater image enhancement improves object tracking performance. Three enhancement methods are evaluated: the U-Shape Transformer, DM-Underwater, and Spectroformer. Each method is applied to the WebUOT-1M dataset and assessed using the UCIQE and UIQM image quality metrics. DM-Underwater was subsequently excluded from tracking evaluation due to severe loss of structural detail, as indicated by strongly negative UIQM scores. The remaining two methods were evaluated using the deAOT tracking model on both enhanced and unenhanced video sequences, with performance measured using mean IoU, precision at an IoU threshold of 0.5, and area under the success curve (AUC). The results show that improvements in perceptual quality metrics do not consistently translate into improved object tracking performance. Furthermore, significant differences in computational cost are observed between the evaluated methods, with diffusion-based approaches being substantially more expensive than transformer-based alternatives. These findings suggest a disconnect between human-perception-based image quality metrics and downstream computer vision performance, highlighting the need for task-oriented evaluation frameworks for underwater image enhancement in AUV applications.

Contents

1 Introduction

1.1 The situation

Autonomous underwater vehicles (AUVs) have become an increasingly important tool in biodiversity monitoring [5], offering non-invasive access to remote and sensitive marine environments that would be difficult or disruptive for human divers to reach [21] [3]. A fundamental capability required for many AUV applications is reliable object detection and tracking, for example for monitoring marine species, mapping habitats, or detecting objects of interest. However, computer vision tasks like object tracking are difficult in underwater environments due to 'water noise'. The noise is caused by the absorption and scattering of light. Red light is absorbed by water while green and blue wavelengths are able to penetrate deeper, causing saturation to be decreased. The back-scattering of light can lead to fogginess, blurring and low contrast overall reducing the resolution of the footage [23] [2].

A proposed solution to this issue is image enhancement. Various non-physical models (e.g. Retinex [13]) and physical models (e.g. RGB and HSI based models [1]) have been created to account for the color difference in underwater images. Despite these models, it remained difficult to account for non-color based noise like transmission; the scattering of light. The physics behind transmission is non-linear and difficult to describe accurately. To overcome these limitations various deep-learning CNN networks (e.g. WaterNet [10]) have been developed to create mappings directly from data. By learning directly from large collections of underwater images, CNNs can implicitly capture the complex effects of transmission and scattering. Unfortunately these networks learn a mapping tied to their training data and are unable to generalize when the environment changes significantly. More recently deep generative approaches like Generative Adversarial Networks (GANs) (e.g. WaterGAN [11]) have been developed to better capture the variability of underwater environments. Despite these advances, it remains unclear whether any of these enhancement approaches actually benefit computer vision tasks.

1.2 Limitations of existing evaluation metrics

The question remains whether applying these enhancement methods is a valid solution to improving the accuracy of computer vision. These methods have been evaluated only on enhancement quality, not on their effect on the computer vision tasks they are intended to support. The metrics used aim to grade enhancement based on either structural values or on human perception. Two common metrics are the Peak Signal to Noise Ratio (PSNR) and Structural SIMilarity Index (SSIM) metrics. These metrics compare the reference image to the enhanced image on pixel level (PSNR) (i.e. one to one translation), and compare structural content (SSIM) (i.e. patterns) to measure how much of the embedded information was kept during enhancements. Studies on analogous enhancement tasks such as image dehazing have found no consistent positive relationship between high scores on these metrics and improved performance on computer vision tasks [16]. Although these measures are mathematically convenient, they do not truly represent perceived visual quality [26], as they treat all pixel-level errors equally regardless of their perceptual impact on a human observer. Consequently Image Quality Evaluation Metric (UCIQE) and Underwater Image Quality Measure (UIQM) are often used alongside PSNR and SSIM to get a more complete evaluation. As these two metrics are based on the human visual system [23] [15].

The fact that they are based on the human visual system is also an argument for the fact that these metrics are not consistent in pairwise comparisons because they are grounded in human perception and inherit its biases. Humans tend to be drawn to the center of images and bright colors and do not take the entire underwater environment into account [10]. Human visual attention differs from how a tracker processes a scene and thus is not representative of computer vision tasks. Whereas human visual attention is a selective and perception-driven process, computer vision algorithms process image information uniformly and algorithmically. Consequently, good scores on neither the structural metrics nor the metrics based on the human visual system ensure that enhancement models will improve performance of computer vision tasks for autonomous underwater vehicles in practice.

1.3 Research Question

This thesis therefore investigates whether underwater image enhancement influences object tracking performance for autonomous underwater vehicles. Additionally, this thesis assesses whether these enhancement models are capable of real-time processing, and considers the practical implications of their computational requirements for AUV deployment. To this end, three recent enhancement models are evaluated on the WebUOT-1M dataset: U-Shape Transformer [17], DM-Underwater [22] and Spectroformer[9]. These models differ considerably in their underlying approach: U-Shape Transformer introduces a GAN-based architecture; DM-Underwater is a diffusion-based model; and Spectroformer is a transformer-based model. This architectural diversity ensures a broad range of enhancement outputs, providing a varied basis for evaluating object tracking performance. Each model is applied to the WebUOT-1M dataset, after which the object tracker deAOT is applied on both the enhanced and unenhanced footage for each enhancement method, allowing a direct comparison of tracking performance. To the best of the author's knowledge, none of these enhancement models have previously been evaluated on the WebUOT-1M dataset. The performance of these enhancement models is compared to both each other and to the results in the original papers, using UCIQE and UIQM. Lastly, the tracking is evaluated using the Intersection over Union (IoU) metric and its variants.

1.4 Thesis overview

The remainder of the thesis is structured as follows: Section 2, Background, provides the necessary background on underwater optics and image enhancement approaches in order to understand rest of the thesis. In section 3, Methodology the enhancement models, the metrics and the tracking algorithm will be covered, including their design and reported performance as well as limitations noted in the literature. In section 4 the pipeline will be described containing information on how the data was processed together with the description of the implementation of the enhancement methods and tracking algorithm. Then Section 5, Results, will contain the results of the experiments and Section 6, Discussions, will contain the interpretation of the results and noteworthy remarks arising from the experiments. Finally in Section 7, Conclusions, the research question and sub-question will be answered and future works will be outlined.

2 Background

Underwater image enhancement (UIE) is the first stage of the processing pipeline used in this thesis. As mentioned in the introduction, raw underwater footage often suffers from 'water noise'due to the optical properties of the underwater environment. These degradations reduce image quality and can negatively affect the performance of downstream computer vision tasks such as object tracking. In order to understand the types of noise better and why they make enhancement difficult three different types of noise and their causes will be further explained. Then the next section reviews the main categories of UIE approaches before introducing and motivating the choice of the methods evaluated in this thesis.

2.1 Underwater optics

2.1.1 Scattering

Underwater imaging quality is primarily degraded by the absorption and scattering of light as it travels through water. These effects arise from multiple sources including: phytoplankton, colored dissolved organic matter (cDOM), and total suspended matter (TSM). All these elements are collectively referred to as Inherent Optical Parameters (IOP) [23].

Scattering manifests in two distinct ways. Forward scattering: light randomly deviated on its path from an object to the camera which causes blurring of image features. Backward scattering: light is reflected by water back towards the camera before reaching the scene which reduces contrast and produces a characteristic veil that obscures the image. Floating particles such as marine snow further amplify both effects.[23]

2.1.2 Color and Saturation

Absorption affects color channels selectively depending on wavelength. Red light is absorbed at approximately 3 meters depth, followed by orange at 5 meters, yellow at 10 meters, and green and purple at greater depths. Blue light, having the shortest wavelength, penetrates furthest, causing underwater imagery to be dominated by blue-green tones. Variation in light sources further contributes to non-uniform color cast across the underwater footage. [15]

2.1.3 Luminance

Artificial lighting, necessary to get footage at certain depths, can extend visibility range but introduces its own distortions, including localized bright spots and poorly illuminated surrounding areas. As a combined result of these phenomena, underwater images typically suffer from low contrast, non-uniform illumination, blurring, color distortion, and reduced visibility of scene content. [23]

2.2 Categories of Underwater Image Enhancement Methods

Existing UIE methods can be broadly divided into three categories: physical model-based methods, visual prior-based methods, and data-driven methods.

2.2.1 Physical Model-based methods

Physical model-based methods work by estimating the physical properties of the underwater environment, such as how much light is absorbed along the path to the camera (medium transmission) and the overall background illumination. Once these properties are estimated, they are used to mathematically reverse the degradation and recover a cleaner image.

Because the enhancement process is grounded in the actual physics of light propagation underwater, it is interpretable: the correction applied to the image has a direct physical justification. The limitation of this approach is that the quality of the result depends entirely on how accurately those physical properties can be estimated. In practice, underwater conditions vary considerably across locations, depths, and water types, and the physical assumptions these models rely on do not always hold. When the environment deviates from the assumed model, parameter estimation becomes unreliable and enhancement quality degrades accordingly.

2.2.2 Visual prior-based methods

Rather than modeling the physics of underwater degradation, visual prior-based methods improve image quality by directly manipulating pixel-level properties such as contrast, brightness, and saturation. The underlying assumption is that a degraded underwater image deviates from certain expected statistical or perceptual properties of natural images, and that correcting these deviations will produce a more visually appealing result. Common strategies include histogram equalization to redistribute pixel intensities, white balance correction to reduce color casts, and Retinex-based decomposition to separate illumination from reflectance.

Because these operations are relatively simple and do not require parameter estimation from a physical model, visual prior-based methods are generally computationally efficient and straightforward to apply. They can produce perceptually improved results in mildly degraded conditions. However, their fundamental limitation is that they treat enhancement as a statistical correction problem rather than a physical one. When degradation is severe, for instance in deep water where red wavelengths are almost entirely absent, adjusting pixel statistics alone cannot recover information that has been lost. The lack of any physical grounding means these methods also tend to produce inconsistent results across different underwater environments, as the same correction strategy may overcorrect in one scene and undercorrect in another.

2.2.3 Data-driven methods

Data-driven methods learn the enhancement process directly from data. With the release of large underwater enhancement datasets data-driven models have recently achieved state-of-the-art performance.
A large proportion of these methods are built on generative adversarial networks (GANs), in which a generator network learns to produce enhanced images while a discriminator network learns to distinguish them from real reference images. The adversarial dynamic between the two pushes the generator toward producing highly realistic outputs. However, this same dynamic introduces well-known training challenges: the balance between generator and discriminator is difficult to maintain, and training can become unstable or collapse entirely into a mode where the generator

produces a limited range of outputs regardless of the input [22].
In addition, many deep learning methods do not account for the way underwater degradation varies across both different colors and different parts of an image [17].

3 Methodology

3.1 U-Shape Transformer

The U-Shape Transformer is an underwater image enhancement network proposed by Peng et al [17]. This model ia a well-motivated choice for this thesis as the network directly addresses two limitations of CNN-based UIE methods identified in section 2. On standard convolutional operations all spatial regions are processed using the same kernel weights, which results in an inability to account for the inconsistency in non-uniform degradation. Additionally convolutional networks apply uniform operations across all channels and are therefore unable to selectively correct the most degraded channels which leads to the inability to selectively correct the most attenuated color channels.

3.1.1 Architecture

To overcome these limitations, Peng et al. extend the Image-to-Image Translation Conditional Adversarial Network of Isola et al. [7] into a transformer-augmented encoder-decoder network. This network aimed to solve the generalization issue of CNN's by additionally learning a loss function such that it can be implemented on various problems without the need to changing the loss function.

The overall architecture of the U-Shape Transformer follows a U-Shape design, in which the encoder progressively downsamples the input image into compact feature representations, and the decoder reconstructs a clean image from those representations. Skip connections between encoder and decoder layers preserve fine-grained spatial detail that would otherwise be lost during downsampling.

Within the U-Shape designs two transformer modules are integrated one of each addressed limitation. The first transformer is the Spatial-wise Global Feature Modeling Transformer (SGFMT) and replaces the bottleneck layer between encoder and decoder. The network models global spatial dependencies to identify and prioritize spatially non-uniform degradation.

The second transformer is the Channel-wise Multi-Scale Feature Fusion Transformer (CMSFFT) and replaces the skip connections, without losing the retainment of spatial detail. This transformer-based fusion mechanism performs attention along the channel axis rather than the spatial axis, enabling the network to identify and prioritize the correction of the most severely attenuated color channel.

3.1.2 Loss function

The model is trained using a composite loss function consisting of four complementary terms. Each term targets a different aspect of image quality by operating in either the RGB, LAB, or LCH colour space. Together, these losses encourage accurate reconstruction, colour consistency,

perceptual quality, and realistic image generation.
The final training objective combines all four loss terms in a minimax formulation:

$$G^* = \arg\min_G \max_D \mathcal{L}_{GAN}(G, D) + \alpha \mathcal{L}^L_{LAB} + \beta \mathcal{L}_{LCH} + \gamma \mathcal{L}_{RGB} + \mu \mathcal{L}_{per}$$

where α, β, γ, μ are hyper parameters, which are set as 0.001, 1, 0.1, 100 respectively, determined empirically. The substantially higher weight assigned to the perceptual loss reflects its central role in producing visually coherent outputs. The individual loss functions are defined as the following:

RGB loss ($\mathcal{L}_{RGB}$) measures the mean squared error between the generated and reference images in RGB space, encouraging accurate pixel reconstruction:

$$\mathcal{L}_{RGB} = \|y - G(x)\|_2^2$$

The perceptual loss ($\mathcal{L}_{per}$) compares deep feature representations extracted from a pre-trained network rather than individual pixels [8]. This encourages the generated images to preserve textures, edges, and high-level structure:

$$\mathcal{L}_{per} = \|\phi(y) - \phi(G(x))\|_2^2$$

The LAB loss ($\mathcal{L}^L_{LAB}$) supervises the image in the perceptually uniform LAB colour space. Mean squared error is applied to the lightness channel, while cross-entropy is used for the quantized colour channels to better preserve colour consistency. The combined loss function is:

$$\mathcal{L}LAB = \mathbb{E}x, y \left[(L_y - L_{G(x)})^2 - \sum_{i=1}^{n} Q(A^i_y) \log(Q(A^i_{G(x)})) - \sum_{i=1}^{n} Q(B^i_y) \log(Q(B^i_{G(x)})) \right]$$

The LCH loss $\mathcal{L}_{LCH}$ supervises lightness, chroma, and hue. Cross-entropy is applied to the lightness channel, while mean squared error is used for the chroma and hue channels:

$$\mathcal{L}_{LCH} = \mathbb{E}_{x,y} \left[-\sum_{i=1}^{n} Q(L^i_y) \log(Q(L^i_{G(x)})) + (C_y - C_{G(x)})^2 + (H_y - H_{G(x)})^2 \right]$$

The adversarial loss ($\mathcal{L}_{GAN}(G, D)$) encourages the generator to produce images that are indistinguishable from real underwater images:

$$\mathcal{L}_{GAN}(G, D) = E_x \left[\log(D(y))\right] + E_y \left[\log(1 - D(G(x)))\right]$$

3.2 DM-Underwater

Tang et al. [22] propose DM-underwater, a conditional diffusion model (DM) for underwater image enhancement.Unlike GAN-based approaches, diffusion models avoid adversarial training, thereby eliminating common issues such as training instability and mode collapse. Instead, the model learns to progressively remove noise from an image, producing a clean enhancement through an iterative denoising process. To address the high computational cost typically associated with

diffusion models, DM-Underwater introduces a lightweight transformer-based denoising network together with an accelerated sampling strategy. This is particularly attractive for deployment on autonomous underwater vehicles (AUVs), where computational efficiency and lightweight models are important.

3.2.1 Architecture

The overall architecture follows the standard conditional diffusion framework, consisting of a forward diffusion process and a reverse diffusion process.During the forward process, Gaussian noise is progressively added to clean training images following the DDPM framework proposed by Ho et al. [6]. During the reverse process, a neural network learns to iteratively remove this noise while being conditioned on the degraded underwater image. Conditioning ensures that the recovered image corresponds to the input image rather than an arbitrary sample.

To improve the efficiency of the reverse process, the authors introduce two architectural modifications: First, the conventional denoising network is replaced with a lightweight transformer-based U-Net. UNet uses only eight transformer blocks for encoding and decoding, replacing heavier convolutional blocks to reduce the number of parameters and improve inference speed. During the reverse process the noisy image and the conditioning image are concatenated and processed using transformer blocks with channel-wise attention, reducing computational complexity while remaining well suited to correcting underwater color degradation.

Second, the standard diffusion sampling procedure is replaced by an accelerated sampling strategy based on DDIM [20]. By skipping selected denoising steps, inference time is significantly reduced. The authors further improve this approach by replacing skip sampling with piecewise sampling and an evolutionary search algorithm to determine more effective sampling schedules without increasing the number of iterations.
Two different non-uniform sampling strategies are proposed to improve the standar approach:

Piecewise sampling This type of sampling is motivated by the observation that earlier timesteps in the reverse process contribute more to image quality than later ones. The sequence is split into two intervals $[a, c]$ and $[c, b]$ with different strides d_1 d_2, allowing denser sampling where it matters most.

Evolutionary search sampling The second approach is an evolutionary search method that treats the sampling sequence as a gene sequence and applies mutation and crossover operations over multiple epochs to find the optimal sequence, evaluated by PSNR score. Both strategies are shown to improve enhancement performance over uniform sampling without increasing the total number of iterations.

3.2.2 Loss function

The model is trained to predict the noise added at each diffusion step using an L_1 loss:

$$L_s = \|\epsilon_t - \epsilon_\theta(x_t, c, t)\|$$

Given input $(x_t, c, t)$where x_t is the noisy image, c is the conditioning image, t is the diffusion timestep, ϵ denotes the true noise added during the forward process, and ϵ_θ is the noise predicted by the network.

3.3 Spectroformer

Khan et al [9] proposed Spectroformer, a transformer-based underwater image enhancement network. This model is a relevant choice for this thesis because it addresses a limitation of many transformer-based UIE methods identified in Section 2. Existing transformer approaches perform attention solely in the spatial domain, limiting their ability to exploit the frequency-domain information that is also affected by underwater degradation. Spectroformer instead learns features jointly in the spatial and frequency domains to improve color restoration, contrast enhancement, and detail reconstruction. This makes it a complementary approach to the U-Shape Transformer and DM-Underwater, which address underwater degradation through spatial attention and diffusion modeling, respectively.

3.3.1 Architecture

The overall architecture follows an encoder-decoder design. After extracting shallow image features, the encoder progressively learns higher-level feature representations before reconstructing the enhanced image in the decoder. Three dedicated modules distinguish Spectroformer from conventional transformer architectures: the Multi-Domain Query Cascaded Transformer (MQCT), the Spatio-Spectro Fusion-Based Attention Block (SSFAB), and the Hybrid Fourier-Spatial Upsampling Block.

Given a degraded underwater image, the network first extracts shallow features using a convolutional layer. These features are then processed by the Multi-Domain Query Cascaded Transformer (MQCT), which forms the bottleneck of the network. Unlike conventional transformer blocks that perform attention only in the spatial domain, the MQCT computes attention jointly in the spatial and Fourier domains. This enables the network to capture both local image structures and global frequency information, improving the correction of color distortions and contrast degradation.

The encoded features are subsequently passed to the Spatio-Spectro Fusion-Based Attention Block (SSFAB), which replaces the conventional skip connections between the encoder and decoder. Rather than directly forwarding encoder features, the SSFAB fuses spatial and spectral representations. The spectral branches combine these two methods using a channel attention mechanism with an adaptively determined kernel size based on the number of input channels. Because smooth kernel size results in loss of important information. This allows the decoder to receive richer feature representations during image reconstruction.
On the other hand the decoder employs a Hybrid Fourier-Spatial Upsampling Block before a final convolution produces the enhanced output image. Conventional upsampling methods operate only in the spatial domain and therefore primarily recover local image details. Spectroformer combines spatial upsampling with Fourier-domain upsampling to recover both local textures and global image structures, improving the reconstruction of fine details in enhanced underwater images.

3.3.2 Loss function

The network is trained using a weighted combination of four complementary loss functions:

$$L_T = \lambda_1 L_C + \lambda_2 L_G + \lambda_3 L_M + \lambda_4 L_P,$$

where L_C is the Charbonnier loss, L_G is the gradient loss, L_M is the multi-scale structural similarity (MS-SSIM) loss, and L_P is the perceptual loss. The weighting factors are empirically set to $\lambda_1 = 0.03$, $\lambda_2 = 0.02$, $\lambda_3 = 0.01$, and $\lambda_4 = 0.025$. Together, these losses encourage accurate pixel reconstruction, edge preservation, structural similarity, and perceptual image quality.

3.4 DeAOT

DeAOT (Decoupling Features in Hierarchical Propagation), proposed by Yang et al. [25], is a transformer-based state-of-the-art framework for video object segmentation (VOS) that builds upon the Associating Objects with Transformers (AOT) architecture [24].

The model is a suitable choice for this thesis because it addresses a fundamental limitation of AOT while maintaining efficient inference. In AOT, object-specific identity information and general visual features are propagated together within a single feature representation. As identity information is gradually added, some of the visual information required for reliable object matching is lost. This effect becomes more pronounced as the number of tracked objects increases, leading to reduced tracking performance.

DeAOT overcomes this limitation by separating object identity and visual information into different propagation pathways and introducing a hierarchical memory propagation mechanism. This preserves richer visual representations during long-term tracking, improving both segmentation accuracy and computational efficiency. This capability is particularly relevant for autonomous underwater vehicle applications, as the propagation strategy can associate and track multiple objects collaboratively throughout a video. In this thesis, multiple object tracking will not be implemented as the dataset used only contains one annotated object per video.

3.4.1 Architecture

Like AOT, deAOT follows a memory-based encoder-decoder architecture. Given an input video, a feature encoder first extracts visual features from each frame. Object information from previously segmented frames is stored in an external memory and propagated to subsequent frames, allowing the network to maintain temporal consistency throughout the video sequence.

The main architectural difference between deAOT and AOT is the introduction of a dual-branch propagation mechanism. Instead of jointly propagating object-specific and shared visual information through a single pathway, deAOT separates these into an object-independent branch and an object-specific branch. The object-independent branch captures global scene information shared by all objects, while the object-specific branch preserves the unique representation of each tracked object. Separating these representations reduces interference between objects and improves the

preservation of fine object details during long-term propagation.

To efficiently combine information from both branches, deAOT introduces the Gated Propagation Module (GPM). Rather than propagating all features equally, the GPM hierarchically fuses object-independent and object-specific representations using a gating mechanism that selectively transfers information between propagation stages. This enables the network to retain detailed object representations while limiting the computational overhead associated with maintaining multiple memory representations.

The propagated features are subsequently decoded into pixel-wise segmentation masks for each video frame. These predicted masks are then stored in memory and used to guide segmentation of future frames, enabling robust long-term object tracking even under occlusions and appearance changes.

3.4.2 Loss function

The network is trained using the same training objective as the original AOT framework. Supervision is applied to the predicted segmentation masks using a combination of pixel-wise segmentation losses, encouraging accurate object boundaries while maintaining temporal consistency throughout the video sequence. As deAOT primarily introduces architectural improvements to the propagation mechanism rather than a new optimization objective, the loss function remains unchanged from the baseline AOT implementation.

3.5 SAM-2

SAM 2 (Segment Anything Model 2), proposed by Ravi et al. [18], is a promptable visual segmentation foundation model that extends the original Segment Anything Model (SAM) from image segmentation to both image and video segmentation. Unlike conventional video object segmentation methods that require task-specific training, SAM 2 is designed to segment arbitrary objects from user prompts such as points, bounding boxes, or masks. The model combines strong zero-shot segmentation performance with efficient real-time inference, making it well suited for deployment in autonomous underwater vehicle (AUV) applications. Although SAM 2 supports both image and video segmentation through a unified architecture, it is used exclusively in its image segmentation mode in this thesis. In this configuration, the memory module remains inactive and the model processes each frame independently, producing the initial object mask that is subsequently passed to deAOT for tracking.

3.6 Metrics

This section introduces the two evaluation metrics used in this research: UCIQE and UIQM. Besides that the subsection will go into more depth on how these metrics are based on human vision, to strengthen the argument that good scores do not imply good tracking results.
Although PSNR and SSIM were mentioned in the introduction they severe no other purpose in this research as they require reference images to function, which the dataset that will be used in the experiments does not have.

3.6.1 UCIQE

The Underwater color Image Quality Evaluation (UCIQE) metric was proposed by Yang et al. [23] as a linear combination of three perceptual components in the CIELAB colorspace. The CIELAB colorspace represents color in terms of lightness and four opponent channels corresponding to red, green, blue, and yellow. [14]

$$\text{UCIQE} = c_1 \times \sigma_c + c_2 \times \text{con}_\text{L} + c_3 \times \mu_s$$

where σ_c denotes the standard deviation of chroma which reflects the color diversity, con_L denotes the contrast of luminance which reflects visibility of detail, and μ_s denotes the average saturation which reflects vividness. The weights c_1,c_2, c_3 were determined empirically and typically have the following value: c_1=0.4680, c_2=0.2745, c_3=0.2576.

UCIQE is a no-reference metric, meaning it does not require a clean reference image for comparison. This is particularly relevant for underwater imaging, where paired clean and degraded images are rarely available whether that be in real-time setting or in datasets. Another important aspect to note is that the metric is able to run in real-time, meaning it could be used during active deployment of an AUV when needing to quantify the quality of in real-time enhancement.

3.6.2 UIQM

Underwater Image Quality Metric (UIQM) was proposed by Panetta et al. [15] and is inspired by the human visual system (HVS), the metric is defined as the following:

$$\text{UIQM} = c_1 \times \text{UICM} + c_2 \times \text{UISM} + c_3 \times \text{UICONM}$$

Where UICM stands for the Underwater Image colorfulness Measure, UISM for the Underwater Image Sharpness Measure and UICONM for the Underwater Image Contrast Measure. These three parameters make up the aspect of the human visual system (HVS). The weights c_1,c_2, c_3 are application dependent, but typically have the following value: c_1=0.02082, c_2=0.2953, c_3=3.5753.

Because the metric is based on HVS it also uses the CIELAB color space. Besides that, just as UCIQE, UIQM also is a non-reference metric.

3.6.3 Relating metrics to the issue

Both UCIQE and UIQM are based on human perception and thus the human visual system (HVS). Both metrics work in the CIELAB colorspace; however, computer vision systems typically operate on raw RGB imagery, as this is the native output of digital camera sensors. CIELAB is not a standard representation in computer vision pipelines, and converting to it introduces an additional transformation that serves no functional purpose for tasks such as object tracking. What matters for tracking is not whether an image appears visually appealing to a human observer, but whether the spatial and structural features of objects are sufficiently distinct for a tracking algorithm to localise them accurately. This disconnect is further reinforced by the metric's validation methodology: Yang et al. [23] assessed UCIQE by having a human panel rate images based on perceived quality,

confirming that the metric is optimised for human perceptual judgment rather than machine vision performance. Consequently, a high UCIQE score indicates that an image looks better to a human observer, but does not imply that object tracking accuracy will improve.

4 Experimental setup

The experiments aim to investigate whether applying enhancement methods on underwater images influences the result of object tracking. Firstly, the work environment and the dataset will be discussed. This is followed by a discussion of data preparation for image enhancement and a detailed description for the implementation for all enhancement methods. Then data preparation will be discussed for object tracking including remapping annotations. Finally the tracking algorithm will be covered and the evaluation methods. This section overall aims for reproducibility.

4.1 Work Environment; ALICE

The experiments were conducted on ALICE (Advanced Leiden Interdisciplinary Research Environment), the high-performance computing cluster provided by the Leiden Institute of Advanced Computer Science (LIACS), which is freely accessible to students. ALICE operates as a Linux-based cluster environment, in which jobs are submitted and managed through the Slurm workload manager. Slurm allocates computational resources and schedules jobs across the cluster, allowing scripts to be executed non-interactively via batch job submission scripts. The cluster provides access to both CPU and GPU nodes, where GPU nodes are equipped with NVIDIA GPUs that support CUDA. In total there are three publicly available CPU partitions and five publicly available GPU partitions, all consisting of different amounts of computational nodes, hardware and available runtime. More information on the partitions can be found on the HPCwiki [19].

4.2 WebUOT-1m

The dataset used for the experiments will be WebUOT-1M which was proposed by Li et al. in 2020 [10]. The dataset is a collection of Youtube and BiliBili videos all containing underwater footage of submerged entities. In total there are 1500 video clips spread over a test and train set, making for a total of 1.1 million frames. It contains a wide variety of footage as the dataset consists of 480 tag categories to make sure any models trained on this dataset are able to track submerged entities in variety of circumstances; e.g. sea, pool and aquarium. Besides that there are in total of 23 tracking attributes in which the circumstances of the footage differ; e.g. low resolution and fast motion. The dataset contains manually annotated bounding boxes, which are necessary to apply tracking methods like deAOT. The bounding boxes are defined as `(x,y,w,h)` in which `(x,y)` are the coordinates of the left upper corner and `w,h` are width and height respectively.

4.3 Data Preparation For Image Enhancement

4.3.1 Video deconstruction

Since the enhancement methods operate on individual images rather than video sequences, all videos in the dataset were first decomposed into frames using the extraction script `Videos2Frames.py`

provided in the WebUOT-1M GitHub repository. The script generates a directory `/imgs` containing the extracted frames for each video (e.g. `WebUOT-1M_Test_000001/imgs/00000001.jpg`). The Test set was converted to frames and was used for inference on the enhancement models and for the tracking model deAOT. The data was stored in a separate directory, with the same data structure as the original `WebUOT-1M` dataset.

4.3.2 Dimensionality reduction

To reduce the impact of the resizing operations applied by the enhancement methods, a center crop was applied to extract a `512` $\times$ `512` pixel region from each frame. The crop boundaries were computed by determining the center of each frame and offsetting by half the target size in each direction:

$$\texttt{left} = \lfloor (W - 512)/2 \rfloor, \quad \texttt{right} = \texttt{left} + 512$$
$$\texttt{top} = \lfloor (H - 512)/2 \rfloor, \quad \texttt{bottom} = \texttt{top} + 512$$

The crop was then applied using the `crop()` method from the Python Imaging Library (PIL). This center crop was applied exclusively to the test set, as only the test frames are subject to the enhancement pipeline. The center crop was executed for every frame; no dimension check was implemented beforehand to verify that frames met the minimum resolution requirement. To store the output, a separate directory was created to store the cut frames, i.e. `/WebUOT-1M_cut`.

4.3.3 Runtime

Resource	Video-to-frames	Dimensionality reduction
CPUs	1	1
Memory	500 MB	500 MB
Nodes	1	1
Billing units	1	1
CPU time	$\approx$ 2 h	$\approx$ 6 h

Table 1: SLURM resource allocation for dataset preprocessing jobs.

4.4 U-Shape Transformer

The U-Shape Transformer inference script was executed in a virtual environment with dependencies installed from the provided `requirements.txt`. The enhancement is applied zero-shot, meaning the pretrained model is used directly for inference without any additional training or fine-tuning on the WebUOT-1M dataset. The only available pretrained model, `generator_795.pth`, was downloaded separately from the repository linked in the `README` of the GitHub page, which also specifies the expected directory structure for model placement. Before enhancement, the U-Shape Transformer transforms all input to a size `256` $\times$ `256`. The original script computed PSNR as an image quality metric; since WebUOT-1M does not provide reference images, this metric is not applicable and was removed.

4.4.1 Inference

The original inference script, called `test.ipynb`, was provided as a Jupyter notebook. Since Jupyter requires a browser-based interface and the experiments were run on the ALICE high-performance computing cluster, the notebook was converted to a standard Python script to enable execution via the Slurm job scheduler.

The following additional modifications were made to the original script. First, all hard-coded paths were updated to point to the correct input and output locations on the file-system. Second, the original script assumed a flat directory of images as input, which is incompatible with the folder structure of the WebUOT-1M dataset. A nested loop was therefore introduced to iterate over each video sub-folder and subsequently over each individual frame within it. Third, a checkpoint mechanism was added to skip frames that had already been processed. Since Slurm may terminate jobs before completion (e.g. Time-out, in case an incorrect time estimate was made), this prevents redundant reprocessing of already enhanced frames and reduces overall computation time upon resubmission.

The produced frames were saved in a separate directory where all enhanced frames were collected, i.e. `WebUOT-1M_enhanced/u-shape_enhanced`. Further into the directory a distinction was made between cut and original sized frames.

4.4.2 Runtime

`test.py` requires a GPU for model inference and was executed on the `gpu-l4-24g` partition, which provides access to an NVIDIA L4 GPU. The low CPU and memory allocation reflects that the computational load is handled entirely by the GPU. Enhancement was run twice in total: once for the original input data and once for the center-cropped input data. A note is, both jobs have had too little time allocated, resulting in a timed out job. For both the original sized data and the cropped data the runtime for all jobs have been added up. As the same bash file was used, all the other allocated resources remained the same.

Resource	**original sized data**	**cropped data**
Partition	`gpu-l4-24g`	`gpu-l4-24g`
CPUs	1	1
GPU	NVIDIA L4 ($\times$1)	NVIDIA L4 ($\times$1)
Memory	500 MB	500 MB
Nodes	1	1
Billing units	3	3
Runtime	$\approx$ 5 h + $\approx$ 9 h $\approx$ 14 hours	$\approx$ 9,5 h + $\approx$ 5,5 h $\approx$ 15 h

Table 2: SLURM resource allocation for U-Shape Transformer inference.

4.5 DM-underwater

4.5.1 Configuration

The base parameters for running DM-Underwater are defined in the `underwater.json` configuration file. This file specifies the data paths and contains settings related to both training and inference. To use a pretrained model, the corresponding model path must be uncommented. The pretrained DM-Underwater model is made publicly available by the authors.

4.5.2 Data Preparation

The inference algorithm requires input data in a specific format. Therefore, the `prepare_data` script must be executed for every dataset before training or inference. This preprocessing step generates three versions of each input image. First, an `.sr` file is created, which serves as the image to which noise will be applied during the diffusion process. Second, an `.hr` file is generated, representing the high-resolution reference image that the model aims to reconstruct from the noisy input. The images are resized to `256` $\times$ `256` pixels to match the input resolution expected by the pretrained DM-Underwater model. Finally, a low-resolution (`.lr`) version of the image is produced with a resolution of (16×16) pixels.

Several modifications were made to the `prepare_data` script. The original implementation generated output files with filenames that were unsuitable for processing video datasets. Since the code was designed for individual images rather than videos, all processed frames were stored in a single directory and identified only by their frame index. This approach becomes impractical when handling multiple videos. To address this issue, the video directory name was extracted from the input path and used to create a separate output directory for each video. The corresponding preprocessed frames were then stored within these directories.

Additionally, a verification step was implemented to determine whether a frame had already been preprocessed. This prevents redundant computations and reduces processing time when jobs must be restarted, for example after interruption by the Slurm scheduler.

4.5.3 Inference

The `infer.py` script was modified to accommodate the workflow used in this study. During inference, the preprocessed `.sr` images are provided as input to the diffusion model, which generates enhanced images intended to approximate the noise in the corresponding `.hr` references. The enhanced output of the inference file are the generated `.sr` images, which are saved in the `imgs` directory within each video directory, as this location is also used by the other enhancement architectures. The remaining output files, including `.hr`, `.sr_process`, and `.inf`, are stored within the corresponding video-specific directory. They will not be needed further in the pipeline. The `.sr_process` files contain intermediate diffusion outputs, while `.inf` files store metadata generated during inference.

4.5.4 Runtime

`prepare_data.py` does not require CUDA and was therefore executed on a CPU-only partition. Given the expected runtime, a partition supporting long-running jobs was needed. As visible in table 3, the `cpu-zen4` partition was selected as it supports a maximum runtime of 7 days. Preprocessing the test set took approximately 16 hours and was executed twice: once for the original input data and once for the dimensionality-reduced input data. `infer.py` requires a GPU for model inference and was executed on the `gpu-l4-24g` partition, which provides access to an NVIDIA L4 GPU, and the `gpu-mig-40g` partition as an expected queue time of 10 hours forces a different choice of gpu. Enhancement of the full test set took approximately 34 hours and was similarly executed twice, once per input condition. The relatively low CPU and memory allocation reflects that the computational load is handled entirely by the GPU.

Resource	**Original data**	**Cropped data**
Partition	`cpu-zen4`	`cpu-zen4`
CPUs	8	8
Memory	4 GB	4 GB
Nodes	1	1
Billing units	9	9
CPU time	$\approx$ 27 h	$\approx$ 16,5 h
Real elapsed time	$\approx$ 3,5 h	$\approx$ 2 h

Table 3: SLURM resource allocation for `prepare_data.py` (DM-Underwater preprocessing).

Resource	**Original data**	**Cropped data**
Partition	`gpu-mig-40g`	`gpu-l4-24g`
CPUs	1	1
GPU	NVIDIA A100 40 GB (MIG 4g.40gb, $\times$1)	NVIDIA L4 ($\times$1)
Memory	1 GB	1 GB
Nodes	1	1
Billing units	5	3
CPU time	$\approx$ 24 h	$\approx$ 34 h

Table 4: SLURM resource allocation for `infer.py` (DM-Underwater inference).

4.6 Spectroformer

4.6.1 Configuration

The author has made the code for Spectroformer publically available through GitHub. The checkpoints for inference are already embedded in the `/checkpoints` directory and called in the inference code `test.py`. To execute the code, various libraries are required, of which no overview is given in a separate text file. Manually each python file was opened to write down the necessary import statement before installing each library.

4.6.2 Data Handeling

For data input, Spectroformer expects a singular directory with all frames, this is not the datastructure of WebUOT-1M. To avoid adapting the code as much as possible, the `test.py` file is called through a bash file, in which the bash file loops through the dataset and re-executes the code for each video. The output of `test.py` has similar shape of the input, which would simply be a directory at a given path containing all frames. Considering all other functions are built on the `Test/video/imgs/frames` datastructure adaptions are made in `test.py` such that for every video a base directory and a directory `/imgs` is made, in which `/imgs` contains all the frames (i.e. `WebUOT-1M_Test_000100/imgs/000001.jpeg`)

4.6.3 Modifications

Additionally, two modifications to `test.py` were necessary. First, a PyTorch 2.6 API change required explicitly passing `weights_only=False` to `torch.load`, as the checkpoint was saved as a full model object rather than a state dictionary. Second, the output filename construction was corrected using `os.path.basename()` to prevent the parent directory name from being prepended to the saved frame filenames.

4.6.4 Runtime

The allocated resources for Spectroformer inference can be seen in table 5. Spectroformer is executed twice, once for images on which center crop was applied and once for full size images. In order to compare the two, the same resources were allocated to illustrate the difference between computational cost.

Resource	**cropped data**	**original sized data**
Partition	`gpu-l4-24g`	`gpu-l4-24g`
CPUs	1	1
GPU	NVIDIA L4 (×1)	NVIDIA L4 (×1)
Memory	1 GB	1 GB
Nodes	1	1
Billing units	3	3
Runtime	≈25,5 h	≈30 h

Table 5: SLURM resource allocation for Spectroformer inference.

4.7 Evaluation Metrics

Image quality was assessed using the UCIQE and UIQM metrics, computed using the implementation provided by `xueleichen` [4]. During evaluation, the UCIQE scores produced by this implementation returned values outside the expected range (0-1). Upon inspection, the implementation was found to deviate from the original formulation as described in the paper. The UCIQE computation was therefore replaced with the implementation from the CE-VAE underwater image enhancement repository [12], which more closely follows the original formula. The UIQM implementation remained unchanged.

4.7.1 Implementation

Metrics were computed per frame using the `nmetrics` function, which returns UIQM and UCIQE scores for a single image. For each video, scores were accumulated across all valid frames and averaged to produce a per-video mean. Two exception cases were handled: frames with near-zero contrast (standard deviation below 1.0) were skipped to avoid degenerate inputs, and frames triggering a `ZeroDivisionError` in the metric computation were excluded. The dataset-level mean and standard deviation were then computed across all per-video means.
Since enhanced frames from all methods were saved to the `/imgs` directory, the same code could be used for evaluation across all enhancement methods without adaptations.

4.7.2 ZeroDivisionError

Considering the ZeroDivisionError occurred more often than anticipated, a separate python file was made to quantify the occurrence of the error. The script parses the log file produced during evaluation and reports the total number of videos processed, the total number of frames found, the total number of frames successfully processed, and the number of frames skipped due to degenerate inputs or errors, expressed as a percentage.

4.7.3 Runtime

Resource	**U-Shape (orig.)**	**Spectroformer (orig.)**	**DM-UW (orig.)**
CPUs	1	8	1
Memory	500 MB	4 GB	500 MB
Nodes	1	1	1
Billing units	1	9	1
CPU time	$\approx$ 7,5 h	$\approx$ 1 d, 18 h	$\approx$ 10 h
Real elapsed time	$\approx$ 7,5 h	$\approx$ 5 h	$\approx$ 10 h

Table 6: SLURM resource allocation for UCIQE and UIQM metric evaluation on original data.

Resource	**U-Shape (cropped)**	**Spectroformer (cropped)**	**DM-UW (cropped)**
CPUs	1	8	1
Memory	500 MB	4 GB	500 MB
Nodes	1	1	1
Billing units	1	9	1
CPU time	$\approx$ 7 h	$\approx$ 1 d, 11 h	$\approx$ 10 h
Real elapsed time	$\approx$ 7 h	$\approx$ 4,5 h	$\approx$ 10 h

Table 7: SLURM resource allocation for UCIQE and UIQM metric evaluation on cropped data.

4.8 Data Processing For Object Tracking

4.8.1 Preprocessing data

In order to properly compare the results of object tracking between enhanced and unenhanced footage, both videos need to have the same dimensions. To enable direct comparison with U-Shape Transformer output, the unenhanced frames are downscaled by a factor of 0.5 to match the 256×256 output resolution.

4.8.2 Preprocessing annotations

Remapping bounding boxes Since the test data was resized both manually and by the enhancement methods, the original bounding box annotations are no longer valid for the transformed image format and must be remapped accordingly. Equal dimensions for all videos could not be assumed.

Given an original frame of resolution $W_{orig} \times H_{orig}$ pixels, and a center-crop to a fixed resolution of $W_{crop} \times H_{crop}$ pixels, the pixel offset introduced by the crop is:

$$\Delta x = \frac{W_{orig} - W_{crop}}{2} \quad \Delta y = \frac{H_{orig} - H_{crop}}{2}$$

To remap a bounding box annotation (x,y,w,h) from the original frame to the cropped frame, the following transformations are applied:

$$x' = x - \Delta x$$
$$y' = y - \Delta y$$
$$w' = w$$
$$h' = h$$

The width and height of the bounding box itself remain unchanged, as the crop does not scale the image content, it only shifts the coordinate origin. Only the top-left corner coordinates (x,y) need to be adjusted to reflect the new spatial offset introduced by the center crop. Since $W_{crop} < W_{orig}$ and $H_{crop} < H_{orig}$, both Δx and Δy are positive, meaning the top-left corner of the original frame lies outside the crop region and its coordinates must be reduced accordingly. Δx and Δy are subtracted from x, y, which shifts the coordinate origin inward toward the crop region.

U-Shape Transformer reduced the dimensions of the enhanced frames to a size of 256×256. A factor of 0.5 is needed to rescale all parameters accordingly:

$$x'' = 0.5 \times x', \quad y'' = 0.5 \times y', \quad w'' = 0.5 \times w', \quad h'' = 0.5 \times h'$$

Converting bounding boxes to masks using SAM-2 As a VOS algorithm (video object segmentation), deAOT expects a segmentation mask for the first frame of each video, while WebUOT-1M annotations are provided in SOT (single object tracking) format as bounding boxes. The method chosen to transform bounding boxes to mask is SAM-2 (Segment Anything Model 2). The decision was made to use the SAM-2 image predictor as it only processed the first

frame, and deAOT only requires the first frame for inference. In total four different checkpoint models are available, `sam2.1_hiera_large` is used. The large variant was selected to maximize segmentation quality for the bounding box prompt task and with ALICE resources the model remained computationally acceptable. SAM-2 was prompted using the remapped bounding box annotations. The resulting masks were saved as `.png` files in a directory parallel to the `/imgs` directory, named `/masks`.

4.8.3 Runtime

Resizing was performed on the `cpu-short` partition, as the expected runtime was well within its maximum wall time. As shown in Table 8, a single CPU and 500 MB of memory were sufficient, reflecting the lightweight nature of the operation. Resizing the original data took approximately 2 hours. For the Spectroformer output, a resume run was required, bringing the total runtime to approximately 5 hours and 16 minutes.

Resource	**Original data**	**Spectroformer**
Partition	`cpu-short`	`cpu-short`
CPUs	1	1
Memory	500 MB	500 MB
Nodes	1	1
Billing units	1	1
CPU time	$\approx$ 1 h, 58 min	$\approx$ 5 h, 16 min

Table 8: SLURM resource allocation for resizing frames to 256×256.

Ground truth bounding boxes were remapped once, as all pipeline variants share the same rescaling factor. Each video was processed at a 512×512 resolution, and since U-Shape Transformer natively outputs 256×256 frames and all other enhancement outputs were manually resized to the same resolution, a uniform rescaling factor applies across all conditions. The remapping job, shown in Table 9, completed in under 40 seconds on the `cpu-short` partition.

Resource	**ground truth remapping**
Partition	`cpu-short`
CPUs	1
Memory	500 MB
Nodes	1
Billing units	1
CPU time	$\approx$ 37 seconds

Table 9: SLURM resource allocation for remapping the ground truth bounding boxes.

Since the ground truth bounding boxes are identical across pipeline variants, the first-frame masks derived from them are likewise shared. Mask generation was therefore executed once using SAM 2,

as shown in Table 10. The job ran on the `gpu-l4-24g` partition and completed in under 6 minutes, adding minimal overhead to the overall pipeline.

Resource	Box-to-mask (SAM 2)
Partition	`gpu-l4-24g`
CPUs	1
GPU	NVIDIA L4 (×1)
Memory	1 GB
Nodes	1
Billing units	3
CPU time	≈ 5 min, 49 sec

Table 10: SLURM resource allocation for first-frame mask generation using SAM 2.

4.9 DeAOT

4.9.1 Dataset preparation

The tracking algorithm deAOT contains a dataset Demo that can be used for datasets other than the ones proposed by the authors. The decision was made to copy the test data of WebUOT-1M to Demo to avoid any conflicts. deAOT expects a structure of `Demo/images/video_dir/frames` and `Demo/masks/video_dir/mask`. In order to have this structure, the masks were moved from the images directory to the mask directory and the `/imgs` directory containing the frames was flattened. As the code loading the dataset does not filter files, any files unrelated to the frames were removed. The files are still located elsewhere on ALICE. Once the code ran successfully, the directories were renamed and new `Demo/images` and `Demo/masks` directories were created such that the next experiment on different data could be executed.

4.9.2 Zero-shot evaluation

Checkpoint model In total 6 different checkpoint models can be used. For each model, two checkpoint stages are available: pre-training on static images (`PRE`) and main-training on YouTube-VOS and DAVIS (`PRE_YTB_DAV`).
For zero-shot evaluation the `PRE_YTB_DAV` checkpoints model are the correct choice.
Out of all models `R50-DeAOTL` and `SwinB-DeAOTL` perform the highest. The decision was made to use `R50-DeAOTL` over `SwinB-DeAOTL`. `SwinB` scores overall higher but has a low fps of 11.9 while `R50` has good scores as well and an fps of 22.4 [25]. All videos of the dataset were 30 fps, meaning a processing speed of 22.4 would still be too low for in real-time processing, but significantly better than 11.9 fps.

Inference For inference the bash file `inference.sh` can be used. The models are initiated as aot, which had to be changed to deaot. The output of the inference code is a directory with predicted masks, in which the predicted mask for every frame was given, and a `.AVI` video was made of the frames with 15 fps.

4.9.3 Runtime

Considering a deep learning model has to be loaded, 10GB of memory was allocated together with 4 CPUs to speed up the process. The SLURM job was configured with the following resources:

Resource	Original data	U-Shape Transformer	Spectroformer
CPUs	4	4	4
GPU	NVIDIA L4 (×1)	NVIDIA L4 (×1)	NVIDIA L4 (×1)
Memory	10 GB	10 GB	10 GB
Nodes	1	1	1
Billing units	8	8	8
CPU time	≈ 12 h, 19 min	≈ 12 h, 20 min	≈ 13 h, 18 min
Real Elapsed time	≈ 3 h	≈ 3 h	≈ 3,5 h

Table 11: SLURM resource allocation for deAOT inference.

With these allocated resources, a single video took approximately 5 minutes to process. The average processing time for the entire test dataset of 480 videos was 3 hours and 9 min.

4.10 Evaluating tracking results

Evaluation of tracking performance was implemented as a two-stage pipeline.
In the first stage, predicted segmentation masks produced by deAOT were compared against the WebUOT-1M ground truth annotations on a per-frame basis, yielding three summary metrics per sequence: mean IoU, precision, and AUC.
In the second stage, the per-frame and per-sequence results were aggregated across all sequences and visualized to enable qualitative comparison between the original and enhanced pipeline conditions.

4.10.1 Mask to bounding box conversion

Tracking performance was evaluated by comparing deAOT's predicted segmentation masks against the ground truth bounding boxes provided by WebUOT-1M. The first step was to transform the masks predicted by deAOT back to bounding boxes. For each predicted mask, a tight bounding box was derived by identifying the minimal enclosing rectangle over all non-zero pixels. Formally, given a binary mask M, the predicted box is defined as $(\min_x, \min_y, \max_x - \min_x, \max_y - \min_y)$, matching the (x, y, w, h) format of the WebUOT-1M ground truth annotations.

4.10.2 IoU calculation

Per-frame Intersection over Union (IoU) was computed as:

$$\text{IoU} = \frac{|B_{\text{pred}} \cap B_{\text{gt}}|}{|B_{\text{pred}} \cup B_{\text{gt}}|} \tag{1}$$

where B_{pred} and B_{gt} denote the predicted and ground truth bounding boxes respectively. If no foreground pixels were present in the predicted mask, no bounding box could be constructed and the frame was assigned an IoU of zero.

Because deAOT uses the first frame as the initialization frame and does not predict a mask for it, the first predicted mask was compared with the second ground-truth annotation, i.e., prediction i was aligned with ground-truth frame $i + 1$.

Three metrics were computed per sequence. Mean IoU was taken as the arithmetic mean of all per-frame IoU values. Precision was defined as the fraction of frames with IoU ≥ 0.5. The Area Under the Success Curve (AUC) was computed by evaluating the success rate at 101 evenly spaced thresholds $t \in [0, 1]$, where the success rate at threshold t is the fraction of frames with IoU $\geq t$, and integrating the resulting curve via the trapezoidal rule:

$$\text{AUC} = \int_0^1 S(t)\, dt \approx \sum_{i=0}^{99} \frac{S(t_i) + S(t_{i+1})}{2} \cdot \Delta t \tag{2}$$

where $S(t)$ denotes the success rate at threshold t and $\Delta t = 0.01$.

4.10.3 Visualization

Per-sequence results (sequence name, number of evaluated frames, mean IoU, precision at IoU = 0.5, and AUC) for each pipeline condition were written to CSV. Additionally six visualizations were generated to support qualitative comparison across pipeline conditions:

- **Success curve**: Plots the success rate $S(t)$ as a function of the IoU threshold for each pipeline condition. The area under the curve (AUC) is reported in the legend as an overall performance measure.
- **Per-frame IoU distribution**: Shows the distribution of frame-level IoU values for each condition using a normalized histogram, with the mean IoU indicated by a vertical line.
- **Precision-at-threshold**: Reports the precision at nine IoU thresholds, $t \in \{0.1, 0.2, \ldots, 0.9\}$, allowing performance to be compared under increasingly strict matching criteria.
- **Per-sequence mean IoU**: Displays the average IoU achieved on each video sequence, highlighting sequences for which image enhancement consistently improves or degrades tracking performance.
- **Mean IoU scatter plots**: Compare the per-sequence mean IoU of each enhanced condition against the original pipeline. Points above the diagonal indicate improved tracking performance.
- **IoU-over-time**: Visualizes the IoU for every frame in a sequence, revealing whether performance degrades gradually or drops due to specific events such as occlusions or rapid object motion.

4.10.4 Runtime

Evaluation was parallelized across sequences using Python's `multiprocessing` pool, with each worker independently loading masks and ground truth for a single sequence. Multiprocessing significantly reduced the real elapsed time as can be seen in table 12.

Resource	Tracking evaluation
Partition	`cpu-short`
CPUs	8
Memory	8 GB
Nodes	1
Billing units	10
CPU time	$\approx$ 2,5 h
Real elapsed time	18 min

Table 12: SLURM resource allocation for tracking evaluation.

5 Results

5.1 Image Enhancement

For visual comparison between cut and uncut input for the enhancement methods frame `000058` was chosen from video `WebUOT-1M_Test_000146`.

5.2 Metric Results

Neither the original UCIQE nor UIQM publication specifies a theoretical value range for the metric. Furthermore, no explicit range was found in the reviewed literature. Based on the values reported across multiple studies, the following empirical ranges were observed: 0-1 for UCIQE and 0-3 for UIQM. In table 13 can the results be seen of applying the metrics on the enhanced results after inputting full sized input. In table 14 can be seen the results of applying the metrics on preprocessed center cropped data.

Metric	U-Shape (Full)	DM_Underwater (Full)	Spectroformer (Full)
Mean UCIQE	0.5518	0.6050	0.5895
Mean UIQM	0.5364	-0.1375	0.7060
Std UCIQE	0.0509	0.0467	0.0518
Std UIQM	0.4412	0.1888	0.4458

Table 13: Performance comparison on the 480 test videos of the full sized WebUOT-1M dataset.

5.2.1 Zero-Division Error

During the calculation of the DM-Underwater evaluation metrics, a division-by-zero error occurred for a subset of the processed frames. This occurs when the contrast measure used in the metric computation becomes undefined due to insufficient variation between image regions.
Visual inspection of the affected frames did not reveal any obvious abnormalities that could explain the occurrence of the error.

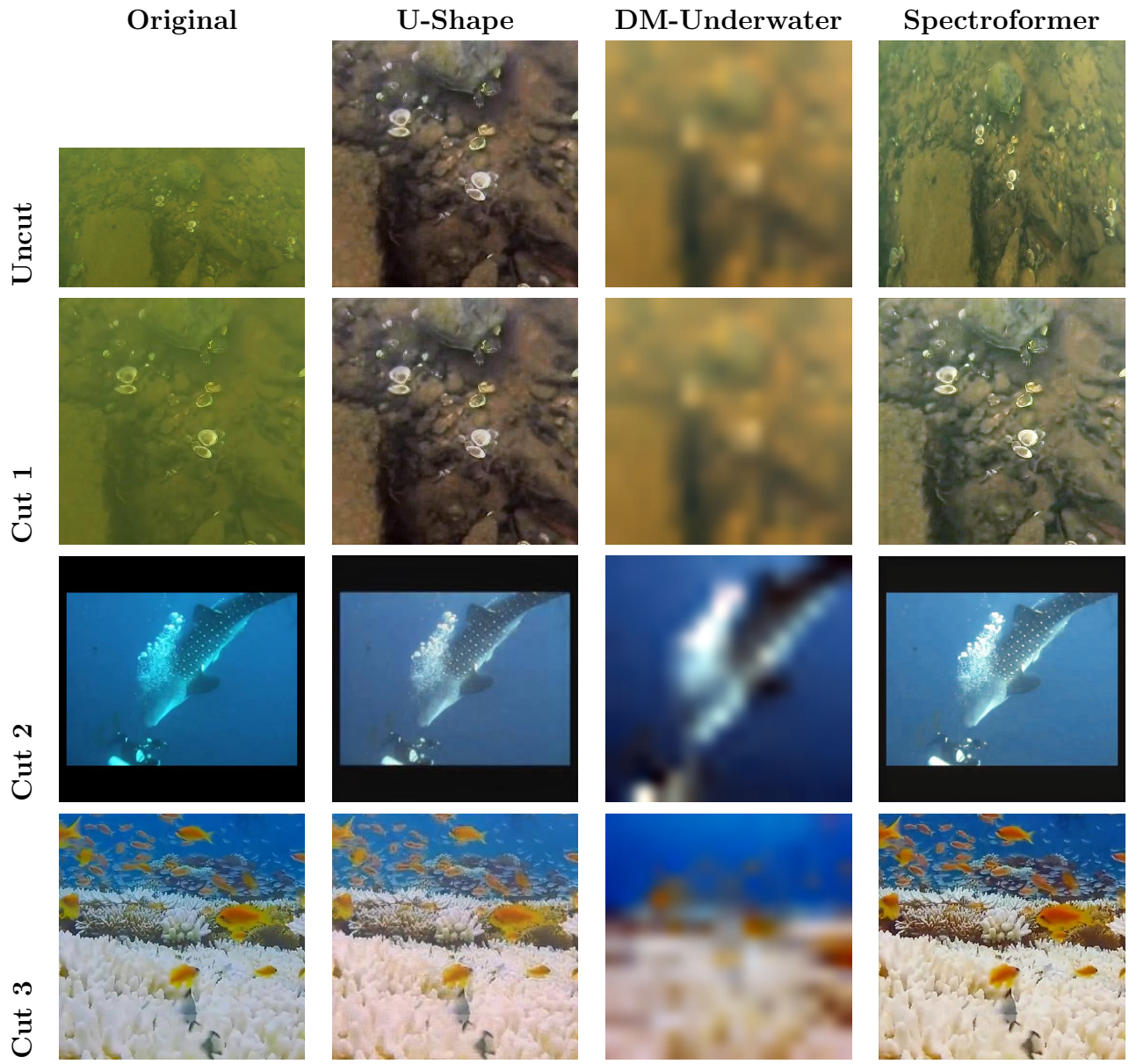


Figure 1: Qualitative comparison of underwater image enhancement methods. The first row shows an uncut WebUOT-1M frame, while the remaining rows correspond to three different cropped (cut) input scenarios. Columns show the original input and the outputs of U-Shape, DM-Underwater, and Spectroformer.

Metric	U-Shape (Cropped)	DM_Underwater (Cropped)	Spectroformer (Cropped)
Mean UCIQE	0.5497	0.6048	0.5753
Mean UIQM	0.4480	-0.0467	0.6497
Std UCIQE	0.0511	0.0467	0.0504
Std UIQM	0.4290	0.1897	0.4152

Table 14: Performance comparison on the 480 test videos of the center cropped WebUOT-1M dataset.

In total, the division-by-zero error occurred for 6,156 frames across all videos, representing approximately 1.5% of the entire test dataset. Given the relatively small proportion of affected frames and the absence of observable image quality issues, the decision was made to retain the metric results and proceed with the analysis.

5.3 Object tracking results

5.3.1 Numerical results

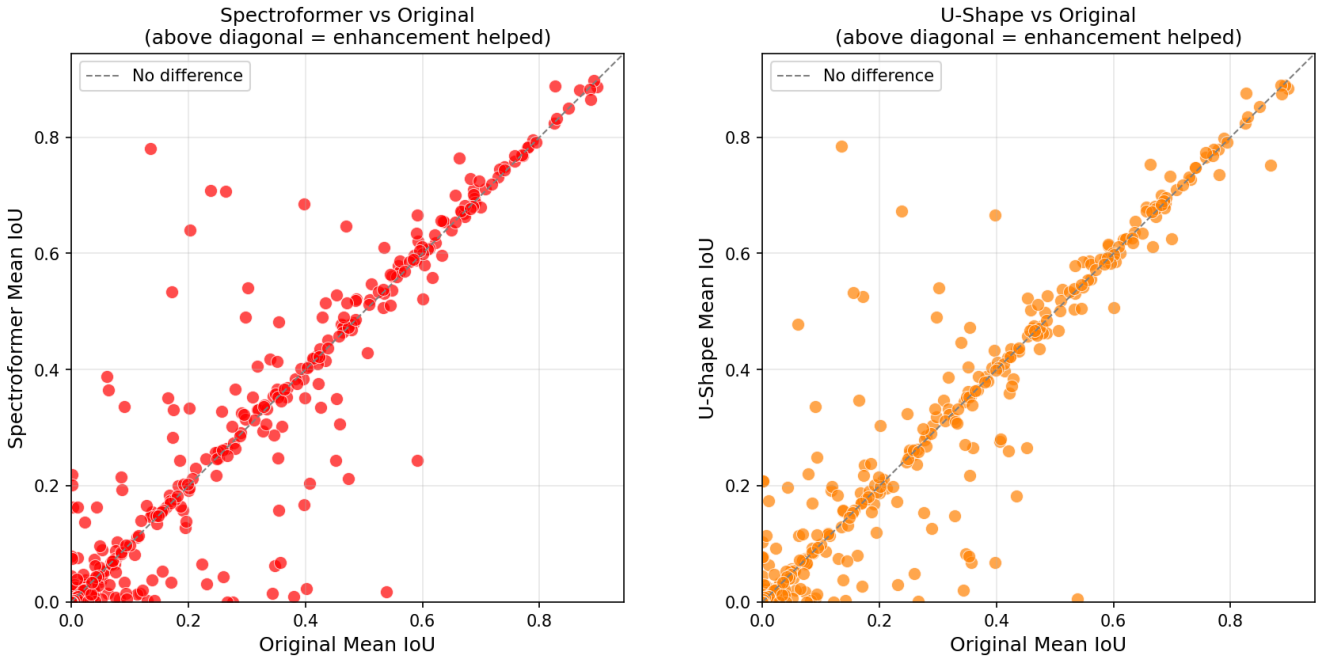
To support the visual results and graphs, numerical results for all videos were calculated. The first 10 entries can be seen in table 15.

		Original			**Spectroformer**			**U-Shape Transformer**		
Sequence	**Frames**	**mIoU**	**Prec@0.5**	**AUC**	**mIoU**	**Prec@0.5**	**AUC**	**mIoU**	**Prec@0.5**	**AUC**
WebUOT-1M_Test_000001	7146	0.6867	0.8508	0.6872	0.7094	0.8669	0.7097	0.6867	0.8508	0.6872
WebUOT-1M_Test_000002	600	0.4516	0.5567	0.4523	0.2440	0.2283	0.2457	0.2657	0.2217	0.2671
WebUOT-1M_Test_000003	360	0.3539	0.4194	0.3553	0.1584	0.0750	0.1607	0.2179	0.2194	0.2202
WebUOT-1M_Test_000004	870	0.3976	0.4207	0.3993	0.1682	0.1816	0.1716	0.0682	0.0598	0.0724
WebUOT-1M_Test_000005	642	0.4822	0.6012	0.4840	0.4814	0.5732	0.4825	0.4979	0.5981	0.4988
WebUOT-1M_Test_000006	1470	0.4134	0.4476	0.4153	0.4179	0.4653	0.4198	0.3981	0.4361	0.4001
WebUOT-1M_Test_000007	831	0.2897	0.3634	0.2929	0.2916	0.3646	0.2948	0.1267	0.1348	0.1310
WebUOT-1M_Test_000008	200	0.4056	0.5450	0.4075	0.4065	0.5450	0.4090	0.2763	0.1950	0.2783
WebUOT-1M_Test_000009	250	0.0000	0.0000	0.0050	0.0011	0.0000	0.0060	0.0014	0.0000	0.0060
WebUOT-1M_Test_000010	475	0.0386	0.0211	0.0427	0.0733	0.0211	0.0770	0.0175	0.0021	0.0220

Table 15: Per-sequence tracking results of mIoU: mean Intersection over Union, Prec@0.5: precision at IoU threshold 0.5 and AUC: area under the success curve for the first 10 entries of the WebUOT-1M test set.

5.3.2 Visual results

A scatterplot was created to show the differences between the enhanced footage vs unenhanced footage. In figure 2a the mean IoU for the original footage is plotted on the x-axis and the mean IoU for Spectroformer on the y-axis. The line in the middle indicates the threshold 0.5. Above the threshold indicates success. A similar graph was created for U-Shape Transformer and can be seen in 2b.

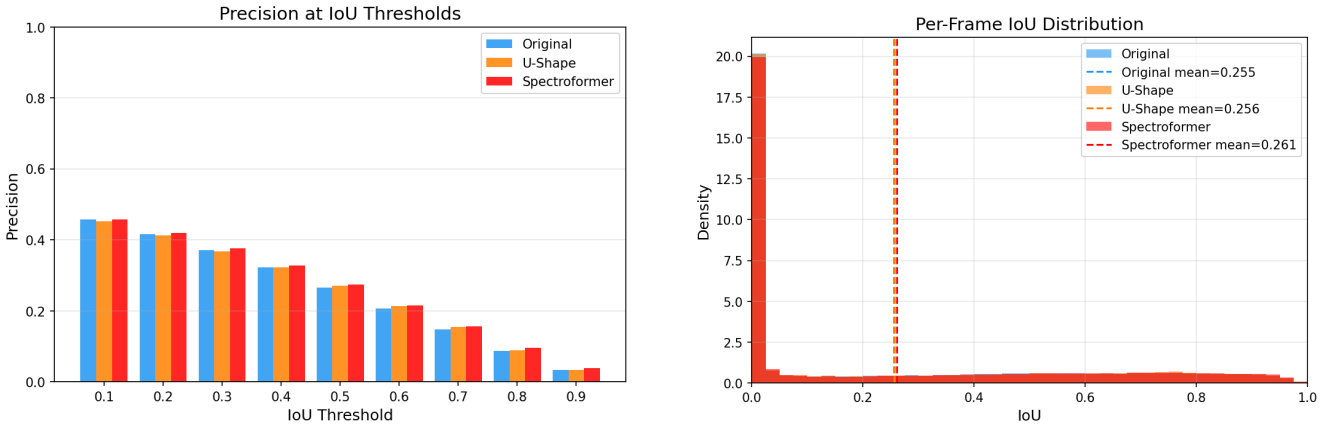


(a) Spectroformer

(b) U-Shape Transformer

Figure 2: Scatterplots comparing the mean IoU per video for the original dataset and the enhanced datasets produced by Spectroformer and U-Shape Transformer.

In Figure 3a, the precision at various IoU thresholds is shown. Together with the IoU distribution across all videos, it becomes clear that the mean tracking performance of the original data is 0.255, compared to 0.256 for the U-shape method and 0.261 for the Spectroformer method.



(a) Precision at various IoU thresholds.

(b) IoU distribution for all enhancement methods.

Figure 3: Comparison of the evaluation metrics. The left figure shows the precision at different IoU thresholds, while the right figure shows the distribution of IoU scores across all enhancement methods and videos.

In the following results various tracking results are visualized by portraying the bounding boxes

for each tracking result next to the ground truth bounding boxes:

Enhanced footage performs better For this example WebUOT-1M_Test_000154 is chosen. In figure 4 can be seen that tracking on U-Shape transformer and Spectroformer enhancement outperforms tracking on the original data.
Two timestamps were chosen for visualization based on the figure: frame 380 - frame 400 can be seen in 5. In this timespan the original data outperforms the enhanced footage. From frame 770 until frame 790 6 the original data performs the worst.

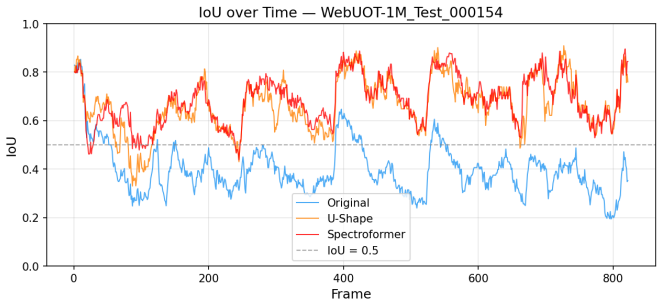


Figure 4: IoU over time for WebUOT-1M_Test_000154

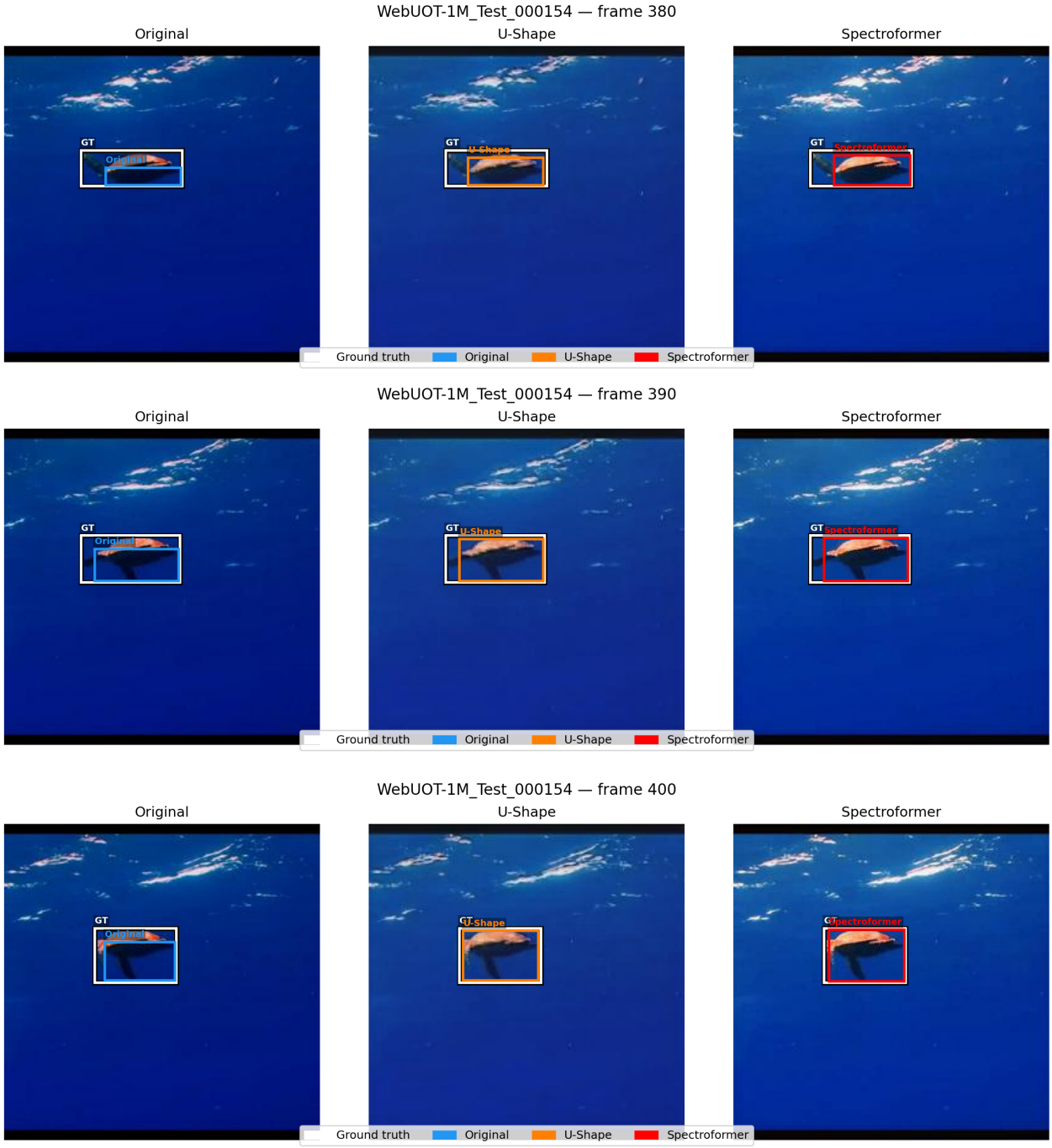


Figure 5: Bounding box comparison for all tracking results at frame 380, 390 and 400

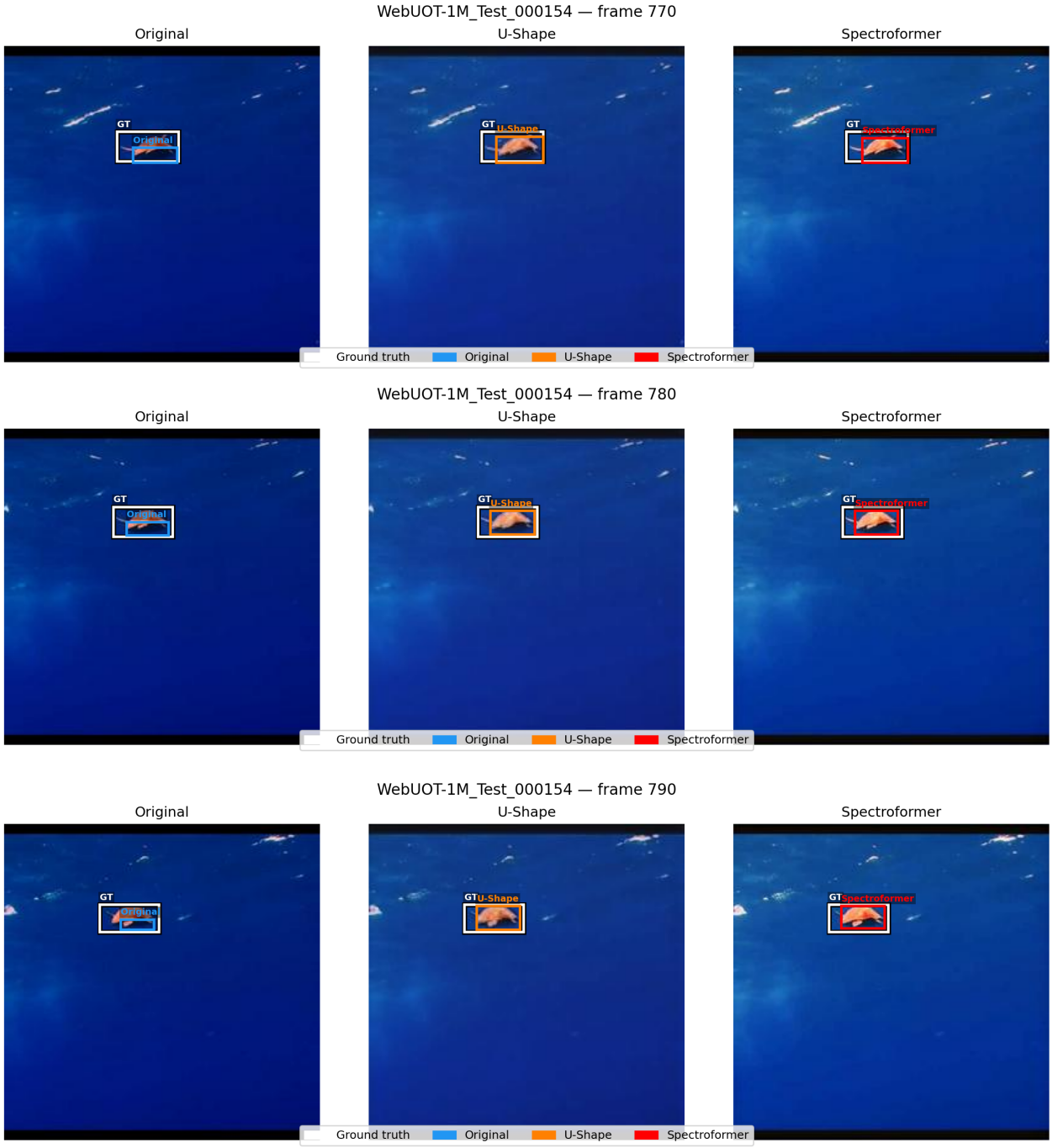


Figure 6: Bounding box comparison for all tracking results at frame 770, 780 and 790

Unenhanced data performed better than enhanced data In some cases the unenhanced data performed better than the enhanced data, WebUOT-1M_Test_474 is an example of that and can be seen in figure 7. Figure 8 shows an example at frame 1350 where the tracking algorithm was able to perform a near perfect bounding box for original data. Notice the bounding box of Spectroformer and U-Shape transformer in the left corner.

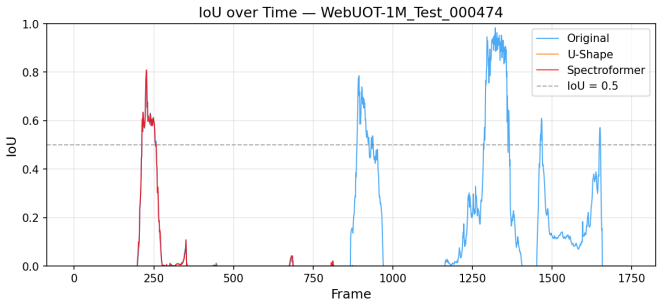


Figure 7: IoU for WebUOT-1M_Test_474

Figure 8: Original data performs better than enhanced data

6 Discussion

6.1 Enhancement results

Figure 1 serves the purpose to show the difference between center cropped input and original sized input for the enhancement methods. Although visually no difference can be seen between U-shape Transformer and DM_underwater, it did matter for Spectroformer. Whereas in the enhancement based on the original size the angle is shifted by water distortion, this is removed by center cropping. Together with the fact that runtime is significantly reduced when data dimensions are reduced, it can be concluded that smaller dimensions are beneficial to object tracking for AUVs.

When comparing cut 1 in figure 1 to figure cut 2 and 3 it becomes clear which model addresses which type of water noise. The original image in figure cut 1 suffers from reduced contrast and a dominant green color cast due to underwater light attenuation and scattering. The U-Shape Transformer method substantially improves local contrast and edge sharpness, enhancing the visibility of rock boundaries and shell structures, although it introduces increased image noise and darker overall illumination. In figure 1, cut 2, the original image exhibits less severe degradation and thus the enhancement primarily increases saturation.

In all cases, to the human eye, Spectroformer achieves the best balance between contrast enhancement, color preservation, and structural detail, producing a more natural appearance while improving the visibility of fine underwater features. Whereas in U-Shape Transformers the highlights are more saturated, they are pure white in the Spectroformer enhancement, this is especially visible in cut 3.

In contrast to U-Shape transformer and Spectroformer, DM_underwater output exhibits significant smoothing, resulting in the loss of high-frequency texture and edge information. Although image noise is reduced, important structural details necessary for underwater scene interpretation are no longer preserved.

6.2 Metric Results

For all enhanced versions, the UIQM values remain relatively low, with the highest observed score being 0.7060, whereas the theoretical upper bound of the UIQM scale is approximately 3.0. This indicates that none of the evaluated enhancement methods fully restore underwater image quality to an ideal level according to this metric.

A particularly notable result is observed for DM-Underwater, which yields a negative UIQM score. Since UIQM is designed such that higher values correspond to improved perceptual image quality, negative values fall outside the expected range and typically indicate severe degradation in one or more contributing components of the metric. In practice, this most commonly occurs when there is extreme loss of sharpness or structural detail. This interpretation is supported by the qualitative results shown in Figure 1, where DM-Underwater exhibits significant blurring and loss of fine structural information.

Despite its poor performance in UIQM, DM-Underwater achieves the highest UCIQE scores for both full-sized and cropped inputs. This suggests that while the method is effective in improving global color balance, contrast distribution, and overall colorfulness, it does so at the expense of image sharpness and structural fidelity. As also observed in Figure 1, this method substantially alters luminance and saturation characteristics, resulting in visually more "balanced" but less detailed imagery.

In contrast, Spectroformer produces the most visually convincing results across all enhancement methods. This observation is supported quantitatively by its consistently highest UIQM scores, indicating superior performance in terms of perceptual quality factors such as sharpness, contrast, and overall natural appearance. The alignment between visual inspection and UIQM scoring suggests that UIQM better reflects human perception of underwater image quality in this context than UCIQE.

Based on these findings, it can be argued that UIQM is a more relevant metric than UCIQE for evaluating downstream object tracking performance. Tracking algorithms rely heavily on structural consistency, edge definition, and object integrity; when these properties are degraded to the extent observed in some enhanced outputs (e.g., DM-Underwater), reliable tracking becomes infeasible as objects lose distinguishable shape and continuity.

Consequently, given the strong correlation between UIQM and perceptual/structural quality in the observed results, and the observed failure cases under low UIQM conditions, the decision has been made not to proceed with object tracking using the deAOT framework on DM_underwater enhanced outputs.

U-Shape Transformer consistently occupies an intermediate position between the two other methods. It achieves moderate UIQM scores (0.5364 for full images and 0.4480 for cropped inputs), outperforming DM-Underwater but remaining significantly below Spectroformer. Similarly, its UCIQE performance is the lowest among all evaluated methods, indicating that it does not strongly enhance color balance or contrast.

Visually, U-Shape Transformer tends to preserve structural information more reliably than DM-Underwater, avoiding the extreme blur and degradation observed in that method. However, this comes at the cost of limited enhancement effects, resulting in outputs that are closer to the original degraded input rather than significantly improved imagery.

Overall, U-Shape Transformer can be interpreted as a conservative enhancement approach that maintains stability without introducing major artifacts, but also without achieving strong perceptual or color improvements.

6.2.1 Tracking Results

The tracking results do not support the hypothesis that underwater image enhancement consistently improves object tracking performance. Across the 480 test sequences, enhancement with either Spectroformer or U-Shape Transformer produced mixed outcomes relative to tracking on the original unenhanced footage. In some sequences the enhanced input yielded higher mean IoU and AUC values, while in others the original footage outperformed both enhanced conditions. No enhancement method produced a systematic improvement across the dataset.

The scatter plots in Figures 2a and 2b illustrate this mixed pattern clearly. In both plots, data points are distributed on both sides of the identity diagonal, indicating that neither method consistently improved nor consistently degraded tracking performance relative to the baseline. Aggregate mean IoU values across all 480 sequences were approximately 0.255 for the original footage, 0.261 for Spectroformer, and 0.256 for U-Shape Transformer, with similarly small differences observed in AUC and precision at 0.5. The absence of a meaningful aggregate difference, combined with the high variance in per-sequence outcomes, indicates that the effect of enhancement on tracking performance is sequence-dependent rather than systematic.

This is further illustrated by the contrasting examples presented in Section 5. For video `WebUOT-1M_Test_000154` Spectroformer and U-Shape Transformer enhancement improved tracking performance over the baseline at specific temporal intervals, with particularly clear gains visible between frames 770 and 790. The enhanced footage provided better-defined object boundaries in these frames, which aided the propagation mechanism of deAOT. Conversely, for video `WebUOT-1M_Test_000474`, original unenhanced footage supported near-perfect bounding box prediction at frame 1350, while both

enhanced conditions displaced the predicted box to an incorrect region. In this case, the contrast manipulations introduced by enhancement appear to have altered local appearance features in a way that misled the tracker's matching process.

Taken together, these results suggest that image enhancement can be beneficial in specific situations, such as sequences with severe color cast or low contrast, but can also be detrimental in sequences where the tracker is already performing reliably on the original footage. The changes to local texture and luminance introduced by enhancement affect different tracking scenarios in different ways, and no single enhancement method provides a uniformly beneficial input transformation.

6.3 Disconnect between perceptual quality and tracking performance

The results of this thesis support the argument made in section 1 Introduction that perceptual image quality metrics do not reliably predict performance on downstream computer vision tasks. DM-Underwater, which achieved the highest UCIQE scores, produced the worst tracking conditions due to the loss of structural detail. Spectroformer, which produced the best perceptual quality scores on UIQM, did not translate this advantage into consistent tracking improvements over the original data. U-Shape Transformer, despite achieving moderate metric scores, performed comparably to Spectroformer on tracking across the full test set.

This disconnect has a straightforward explanation. Both UCIQE and UIQM are designed to assess image quality from the perspective of the human visual system. Their component measures, namely chroma distribution, luminance contrast, saturation, sharpness, and colorfulness, reflect properties that correlate with perceived visual quality for a human observer. A tracking algorithm, by contrast, relies on the consistency and distinctiveness of local feature representations across frames. What matters for tracking is not whether an image looks more natural or colourful to a human, but whether the spatial structure of the target object remains stable and discriminable after enhancement. These requirements are orthogonal, and it is entirely possible to improve a perceptual quality metric while simultaneously disrupting the feature representations on which a tracker depends.

This finding is analogous to observations from related domains. Studies on image dehazing have likewise reported that improvements in PSNR and SSIM do not translate reliably into gains on object detection or scene understanding tasks [16]. The present results extend this observation to the underwater image enhancement setting and to the specific task of video object tracking.

6.4 Computational cost and real-time feasibility

A secondary finding of this thesis concerns the substantial differences in computational cost between the evaluated methods. DM-Underwater required approximately 34 hours to process the 480-video test set on an NVIDIA A100 40 GB MIG instance and an NVIDIA L4 GPU, in addition to approximately 16 hours of CPU preprocessing. U-Shape Transformer completed inference in approximately 14–15 hours on the NVIDIA L4, and Spectroformer in approximately 25–30 hours. All three methods process a single 256×256 frame in well under one second in isolation; however, the

overhead from data loading, preprocessing, and sequential job scheduling accumulates significantly over a large dataset.

The processing speeds of deAOT are similarly relevant. The `R50-DeAOTL` model achieves approximately 22.4 frames per second, which falls below the 30 fps frame rate of the WebUOT-1M sequences. This means that even the tracking stage alone cannot be executed in real time on the hardware used in this study. When the cost of enhancement preprocessing is added, real-time combined enhancement and tracking is currently infeasible with any of the evaluated methods on standard GPU hardware. Diffusion-based models such as DM-Underwater are the least feasible: iterative denoising requires multiple forward passes per frame, making them orders of magnitude more expensive than transformer-based approaches. Even with the accelerated DDIM sampling strategy introduced in DM-Underwater, inference time per frame is far too high for real-time AUV deployment.

Transformer-based methods such as U-Shape Transformer and Spectroformer are more feasible candidates for optimization toward real-time deployment, but would still require hardware acceleration, model quantization, or architectural simplification to meet real-time constraints in an embedded AUV system.

6.5 Limitations

Several limitations of this study should be noted. First, all enhancement models were applied zero-shot, without fine-tuning on the WebUOT-1M dataset. The models were trained on different underwater image datasets with different distribution characteristics, and their performance on WebUOT-1M may not reflect their full potential. Fine-tuning on a representative subset of the dataset could yield different metric and tracking results. Second, the bounding box to mask conversion using SAM 2 introduces an additional source of error. The quality of the initial segmentation mask constrains the maximum achievable tracking performance, and any inaccuracies in the first-frame mask will propagate through the deAOT memory mechanism. Third, DM-Underwater tracking results were not obtained due to its severely negative UIQM scores and qualitatively observed loss of structural detail. While this decision was justified, it means that a direct tracking comparison with DM-Underwater cannot be made, and it remains possible that the blurred outputs could still support tracking under specific conditions. Fourth, the evaluation relies on a single tracking model (deAOT) and a single dataset. Different trackers with different feature extraction and matching strategies may respond differently to image enhancement, and the findings may not generalize beyond deAOT.

7 Conclusions and Further Research

7.1 Conclusions

This thesis investigated whether underwater image enhancement improves object tracking performance for autonomous underwater vehicles, using the WebUOT-1M dataset as the evaluation benchmark. Three enhancement methods were evaluated: U-Shape Transformer, DM-Underwater,

and Spectroformer. Each was applied to the 480 test sequences of WebUOT-1M, and tracking performance was subsequently assessed using deAOT with mean IoU, precision at 0.5, and AUC as evaluation metrics.

The results do not support the hypothesis that underwater image enhancement consistently improves object tracking performance. Neither Spectroformer nor U-Shape Transformer produced a systematic improvement in tracking metrics over unenhanced footage across the test set. Instead, outcomes were highly sequence-dependent, with individual sequences showing both improvements and degradations relative to the baseline. This suggests that the benefit of enhancement is contingent on the specific degradation characteristics of each sequence rather than representing a general effect.

DM-Underwater produced severely negative UIQM scores and qualitatively exhibited significant blurring of structural detail, rendering it unsuitable as a preprocessing step for object tracking. Despite achieving the highest UCIQE scores of all evaluated methods, DM-Underwater's enhancement output degraded the structural properties of the images in ways that would be detrimental to feature-based tracking algorithms. This result highlights a key limitation of UCIQE as an evaluation criterion: its sensitivity to global colour and contrast properties does not reflect the structural fidelity required for reliable object tracking.

Taken together, these findings confirm the existence of a disconnect between perceptual image quality metrics and downstream computer vision performance in the underwater domain. High UCIQE or UIQM scores do not guarantee improved tracking performance, and in some cases the transformations introduced by enhancement actively degraded tracking results. This supports the broader argument that UIE methods should not be evaluated solely on perceptual quality metrics when the intended application is a computer vision task.

Regarding the sub-question of real-time feasibility, none of the evaluated enhancement methods is currently suitable for real-time deployment on an AUV. DM-Underwater is the least feasible due to its iterative diffusion process, which requires multiple forward passes per frame and cannot approach real-time processing speeds. Transformer-based methods are comparatively more efficient, but still require significant optimization before meeting the frame rate requirements of AUV applications. Furthermore, deAOT itself operates below the 30 fps frame rate of the test data, meaning that even the tracking stage in isolation cannot run in real time. Combining enhancement and tracking in a real-time pipeline would require hardware acceleration, architectural compression, or a fundamentally different approach to on-device inference.

7.2 Future Work

There are several directions for future research following from the limitations and findings of this thesis:

The most direct extension would be to fine-tune the evaluated enhancement models on a representative subset of WebUOT-1M before applying them to the full test set. Zero-shot evaluation as performed here captures baseline generalization performance, but domain-specific fine-tuning

could substantially improve both enhancement quality and downstream tracking performance. This would also allow a fairer comparison between methods, as differences in performance may currently reflect differences in training data distribution rather than differences in architectural capability.

A second direction is to evaluate additional tracking models alongside deAOT to determine whether the observed disconnect between enhancement and tracking performance is specific to deAOT's feature extraction and memory propagation strategy, or whether it holds across a broader class of trackers. Methods that rely on different visual representations, such as correlation filter-based trackers or recent transformer-based single object trackers, may respond differently to the color and contrast manipulations introduced by enhancement.

Third, it would be valuable to develop or apply evaluation metrics that measure image quality specifically from the perspective of downstream computer vision tasks rather than human perception. Task-oriented quality metrics that quantify the distinctiveness or stability of object features across frames would provide a more principled basis for evaluating underwater image enhancement in AUV applications, and could help bridge the gap between enhancement research and practical deployment. The results of this thesis suggest that this is a necessary step toward establishing reliable evaluation frameworks for UIE in the context of autonomous underwater systems.

Finally, future work should investigate the computational efficiency of enhancement methods more directly in an AUV context. This includes profiling inference latency on embedded GPU hardware representative of AUV systems, exploring quantization and pruning strategies to reduce model size and inference time, and evaluating whether lightweight enhancement models sacrifice tracking performance relative to the more complex architectures evaluated here.

References

[1] Ahmad Shahrizan Abdul Ghani and Nor Ashidi Mat Isa. Underwater image quality enhancement through integrated color model with rayleigh distribution. *Applied Soft Computing*, 27:219–230, 2015.

[2] Kancharagunta Kishan Babu, Ashreen Tabassum, Bommakanti Navaneetha, Tenneti Jahnavi, and Yenka Akshaya. Underwater image enhancement using generative adversarial networks: A survey. *Journal of Marine Science and Engineering*, 2022.

[3] Jennifer A. Cardenas, Zahra Samadikhoshkho, Ateeq Ur Rehman, Alexander U. Valle-Pérez, Elena Herrera-Ponce de León, Charlotte A.E. Hauser, Eric M. Feron, and Rafiq Ahmad. A systematic review of robotic efficacy in coral reef monitoring techniques. *Marine Pollution Bulletin*, 202:116273, 2024.

[4] Xuelei Chen. Psnr-ssim-uciqe-uiqm-python. https://github.com/xueleichen/PSNR-SSIM-UCIQE-UIQM-Python, 2021. Accessed: 2026-06-28.

[5] R. Dinakaran, L. Zhang, C.-T. Li, A. Bouridane, and R. Jiang. Robust and fair undersea target detection with automated underwater vehicles for biodiversity data collection. *Remote Sensing*, 14(15):3680, 2022.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. pages 5967–5976, 07 2017.

[8] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016.

[9] Md Raqib Khan, Priyanka Mishra, Nancy Mehta, Shruti S. Phutke, Santosh Kumar Vipparthi, Sukumar Nandi, and Subrahmanyam Murala. Spectroformer: Multi-domain query cascaded transformer network for underwater image enhancement. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1443–1452, 2024.

[10] Chongyi Li et al. An underwater image enhancement benchmark dataset and beyond. *IEEE Transactions on Image Processing*, 29:4376–4389, 2020.

[11] Jie Li, Katherine A. Skinner, Ryan M. Eustice, and Matthew Johnson-Roberson. Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images. *arXiv preprint arXiv:1702.07392*, 2017.

[12] Niki Martinel. ce-vae-underwater-image-enhancement. https://github.com/iN1k1/ce-vae-underwater-image-enhancement, 2024. Accessed: 2026-06-28.

[13] John J. McCann. Retinex theory. In Ronnier Luo, editor, *Encyclopedia of Color Science and Technology*, pages 1118–1125. Springer, 2016.

[14] K. McLAREN. Xiii—the development of the cie 1976 (l* a* b*) uniform colour space and colour-difference formula. *Journal of the Society of Dyers and Colourists*, 92(9):338–341, 1976.

[15] Karen Panetta, Chen Gao, and Sos Agaian. Human-visual-system-inspired underwater image quality measures. *IEEE Journal of Oceanic Engineering*, 41(3):541–551, 2016.

[16] Yanting Pei, Yaping Huang, Qi Zou, Yuhang Lu, and Song Wang. Does haze removal help CNN-based image classification? In *Proceedings of the European Conference on Computer Vision Workshops (ECCVW)*, 2018.

[17] Long Peng, Chao Zhu, and Lijun Bian. U-shape transformer for underwater image enhancement. *IEEE Transactions on Image Processing*, 32:3066–3079, 2023.

[18] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos, 2024.

[19] Robert Schultz. Partitions on ALICE. https://pubappslu.atlassian.net/wiki/spaces/HPCWIKI/history/37519504/Partitions+on+ALICE, 2026. Accessed: 2026-06-28.

[20] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

[21] Atif Sultan, Lyes Saad Saoud, Mahmoud Elmezain, Mohamed Heshmat, Lakmal Seneviratne, and Irfan Hussain. Autonomous robotic systems for coral reef monitoring: Review and open research issues. *Ecological Informatics*, 92:103511, 2025.

[22] Y. Tang, T. Iwaguchi, and H. Kawasaki. Underwater image enhancement by transformer-based diffusion model with non-uniform sampling for skip strategy. *arXiv preprint arXiv:2309.03445*, 2023.

[23] Miao Yang and Arcot Sowmya. An underwater color image quality evaluation metric. *IEEE Transactions on Image Processing*, 24(12):6062–6071, 2015.

[24] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating objects with transformers for video object segmentation. *CoRR*, abs/2106.02638, 2021.

[25] Zongxin Yang and Yi Yang. Decoupling features in hierarchical propagation for video object segmentation, 2022.

[26] H. R. Sheikh Z. Wang, A. C. Bovik and E. P. Simoncelli. "image quality assessment: from error visibility to structural similarity". *IEEE Trans. Image Process., vol. 13, no. 4, pp. 600–612*, 2004.