PROJECT PLANNING & SCOPING

Effects of de-hazing on USOD performance

Jakub S. Zareba
Jsz22@bath.ac.uk

Contents

1. Introduction

In underwater conditions images are degraded due to poor lighting, colour absorption and light scattering, all of which are amplified as depth increases. These degradations are particularly harmful to computer vision systems, particularly those developed and trained on terrestrial datasets which aren't affected by the same degradations.

A common approach to addressing the domain shift is to retrain or redesign models using underwater specific datasets and architecture. This approach is often effective, but it often increases architectural complexity, computational requirements and implementation costs, which may limit suitability for embedded or real-time systems.

This project investigates whether image de-hazing can work as an alternative to domain specific specialisation for underwater salient object detection. The performance of a generic segmentation model trained on underwater data will be compared under three conditions:

1. Using raw underwater data
2. Using de-hazed images as inputs to shift the domain at runtime
3. Using de-hazed images as both training and input data

These results will then be compared to a domain specific Underwater Salient Object Detection (USOD) model.

Performance will be assessed using pixel-level saliency metrics such as F-measure, mean absolute error (MAE) and Intersection over Union (IoU), along with system level measures including inference latency, parameter count and computational resource usage. The goal is to determine whether pre-processing based enhancement methods can compete in performance with more specialised, complex and domain specific models, in environments where resource constraints are critical.

2. Research Questions

This project aims to answer these questions in no particular order:

- Can image de-hazing improve USOD performance to a level comparable to that achieved by a specialised model.
- How do different classes of de-hazing methods, including deep learning based, restoration based and enhancement based, affect object detection performance?
- Does image preprocessing via dehazing provide a more computationally efficient approach than specialised transformer based USOD models, when evaluated considering realistic system constraints?
- Can de-hazing without retraining provide a comparable performance improvement?

3. Methodology

3.3 Overview

This project evaluates the impact of image de-hazing as a preprocessing step for USOD. The experiments are designed to compare:

1. A generic segmentation based SOD model trained on raw underwater images
2. The same model using de-hazed images as the input
3. The same model trained on de-hazed images
4. A domain specific model as a specialised benchmark

Performance will we assessed using both pixel-level and system-level metrics.

3.2 Dataset

Experiments evaluations and training will be conducted using the USOD10K dataset. The USOD10K dataset provides pixel-level annotated underwater images for SOD tasks. The dataset will be divided into training, validation and test subsets according to the protocol specified in the original benchmark to ensure comparability with existing work

3.3 Models

3.3.1 Generic Baseline Model

A standard encoder-decoder segmentation architecture will be used as the generic SOD baseline, such as U-Net. The model will be trained on raw underwater images to establish baseline performance before enhancement or specialised architecture.

3.3.1 De-hazing at Run Time

Selected de-hazing methods will be applied to images that are evaluated by a generic SOD model trained on raw images. The de-hazing methods will be selected from three categories:

- Restoration (physics) based
- Enhancement based
- Deep learning based

For each method the generic segmentation model will only see the de-hazed images during inference.

Image quality improvement will be verified using standard metrics such as PSNR and UCIQUE prior to evaluation.

3.3.1 De-hazing Trained Model

The same methods that were used in the De-hazing at Run Time test will be used on training data and the evaluation data. The same model as used for the baseline will be used trained using de-hazed images before being used for SOD.

3.3.1 Domain Specific Model

A transformer based underwater saliency architecture will be used as the specialised comparator, TC-USOD. This model was designed explicitly for USOD and will be trained and evaluated following the protocol described in its original publication.

3.4 Evaluation Metrics

3.4.1 Saliency Performance Metrics

Pixel level performance will be evaluated using:

- F-measure
- MEA
- IoU

These metrics quantify mask accuracy and alignment with ground truth annotations

3.4.1 Image Quality Metrics

To confirm that de-hazing has been successful, enchancement quality will be measured using:

- Peak Signal to Noise Ratio (PSNR)
- Underwater Image Colourfulness Evaluation (UCIQE)

These metrics will not be used to determine success of the USOD but will provide context for the performance changes

3.4.1 System Level Metrics

To evaluate the practical suitability of the methods the following metrics will be used:

- Inference duration (ms per image)

- Model parameter count
- GPU memory usage
- Floating point operations (FLOPs), if measurable

This will enable a comparison of the trade-offs between the different SOD methods.

3.5 Experimental Procedure

1. Train generic model on raw images
2. Test generic model on raw images
3. Test generic model on de-hazed images
4. Train generic model on de-hazed images, one for each method
5. Test trained generic model on de-hazed images
6. Train or fine-tune the domain specific TC-USOD
7. Test TC-USOD on raw images
8. Compare performance across saliency, image quality and computational resource usage.
9. Analyse trade-offs between saliency performance and resource overhead

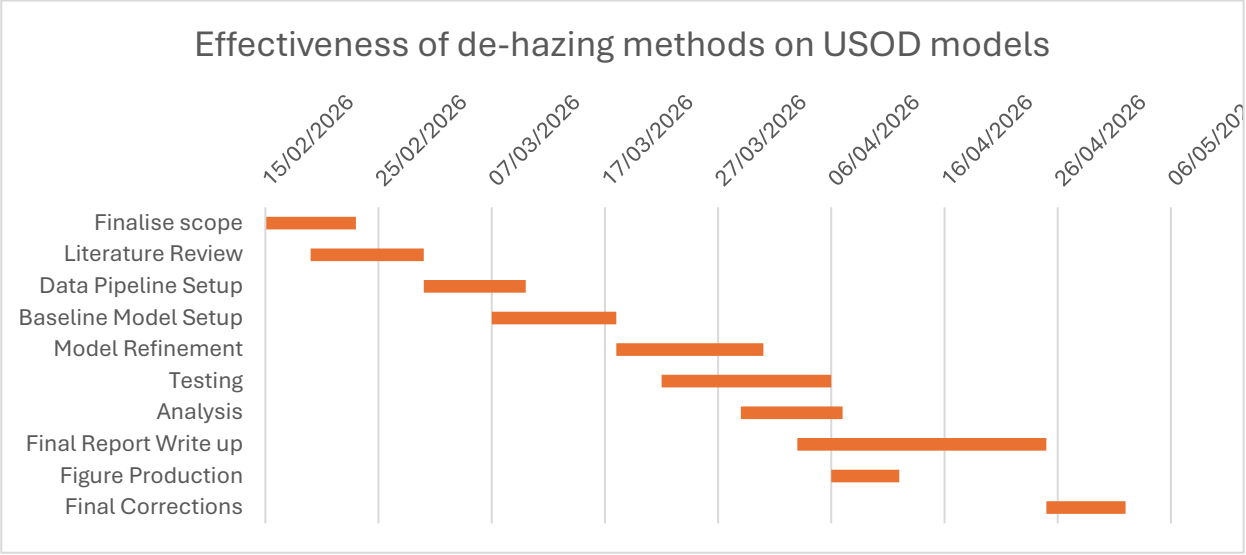
Statistical consistency will be kept by using identical dataset splits for training, validation and testing.

4. Technical Infrastructure

All experiments will be implemented using PyTorch in a Linux based development environment. GPU acceleration will be used for both training and inference. The primary workstation will have an AMD GPU with ROCm support. PyTorch's hardware agnostic code allows for easy implementation across GPU backends without the need for architecture changes.
To preserve consistency and reproducibility, Docker containers will be used t o define fixed software environments, including specific versions of Python, PyTorch and their supporting libraries. Identical container configuration will be used in all the experiments.
As a contingency the university's AI workstation, equipped with an NVIDIA GPU can be used. Due to PyTorch being backend agnostic no architectural changes should be necessary if switching from ROCm to CUDA.

5. Timeline

The Gantt chart below outlines the timeline I anticipate for each major component of the project. Tasks are sequenced according to logical dependencies, with overlap included where feasible.



The timeline does not extend the full duration of the project to accommodate potential delays, especially during experimentation or model training. This structure should allow for structured progression and enough flexibility to accommodate unexpected circumstances.

6. Risk Assessment

Several risks may impact the completion time of the project.

6.1 Training Time and Computational Constraints

Model training may take longer than expected due to hardware limitations or increased complexity. To mitigate this simpler models will be prioritised over the more computationally intensive versions.

6.2 Implementation and Debugging

Unexpected software or integration issues may slow development. Early implementation of a baseline pipeline reduces this risk and allows for incremental modification.

6.3 Data Quality Issues

Dataset inconsistencies or preprocessing difficulties may affect performance or delay experimentation. Early dataset choice and validation reduce the impact of this.

6.4 Scope Creep

Expanding the project beyond the defined objectives would rapidly consume time and reduce the depth and quality of work carried out. Strictly defining the models and tests planned will reduce this risk.