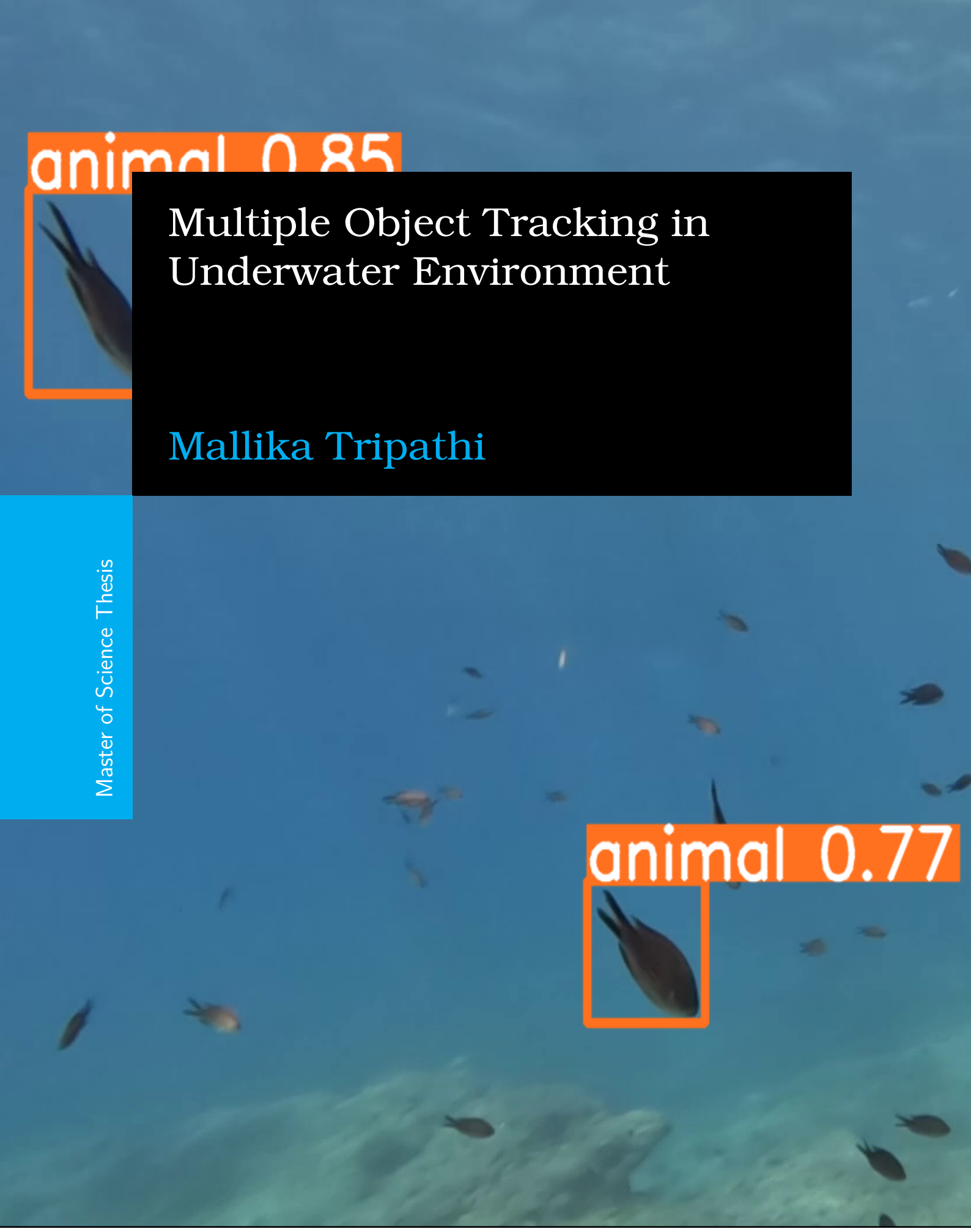


Multiple Object Tracking in Underwater Environment

Mallika Tripathi

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Multiple Object Tracking in Underwater Environment

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Mallika Tripathi

August 26, 2024

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

While various tracking algorithms have demonstrated effectiveness in terrestrial and aerial contexts, their performance in underwater settings remains unexplored. Object tracking in underwater videos presents unique challenges due to variable lighting, water turbidity, and unpredictable camera movement, all of which are likely to hinder the performance of traditional detection and tracking methods. Addressing this gap is crucial for applications such as marine biology research, underwater surveillance, and autonomous underwater vehicles.

This thesis first evaluates existing tracking algorithms, Simple Online and Real-time Tracking (SORT), ByteTrack, and Bag-of-Tricks SORT (BoT-SORT), each incorporating motion estimation and linear data assignment methods on a novel underwater video dataset with a moving camera. The thesis then improves on the SORT algorithm by adopting velocity estimation techniques, a formula-based and an optical flow-based, giving rise to two new algorithms, SORT-V and SORT-OF. Furthermore, the thesis proposes a novel tracker that utilises an Interacting Multiple Model (IMM) filter to estimate the location of the target object. The evaluation focuses on finding a balance between specific metrics, such as tracking accuracy, identity switches, and the number of tracked and lost trajectories. The results indicate that using velocity estimation techniques improves the tracking accuracy by 11% and tracks more objects by nearly halving the number of lost objects. Incorporating a constant acceleration model in the IMM filter gives the best result, with the highest tracking accuracy and with the least number of identity switches, all in real-time computational speed.

Table of Contents

Acknowledgements

I would like to first thank my supervisor Prof.dr.ir. Bart De Schutter for his invaluable guidance and support throughout this thesis. Our meetings provided me with both reassurance about my work and profound insights into this field of research. Additionally, I am grateful to him for imparting the importance of clarity in mathematical and technical writing, as well as in academic presentations. Thanks to his mentorship, I now approach reports and presentations with a new perspective.

A heartfelt thanks to my daily supervisor, Athina Ilioudi whose friendly and personal guidance has been instrumental throughout this journey. Your support allowed me the freedom to shape this thesis according to my interests and ambitions. I deeply valued the open dialogue and the fresh, creative perspectives you brought to our weekly meetings. Additionally, I am particularly grateful for the time and effort you dedicated to providing feedback on my work.

Divya and Kartikeya, thank you for being the greatest cheerleaders. A special thank you to CineUs for being the funniest group of people ever.

I am immensely grateful to all the friends I've made during this master's degree. A special mention to Chris for the endless coffees and ensuring we took enough study breaks, and to Bram and Nathan for making me feel right at home.

To my parents, I could not have made it this far without your never-ending support. To my brother, thank you for the memes. Thank you, Sanket, for helping me maintain my work-life balance. And a special mention to Simba, who brightens my life just by existing.

Lastly, I would like to thank 20-year-old Mallika, thank you for dreaming big and aiming high. We did it, we made it.

Delft, University of Technology — Mallika Tripathi
August 26, 2024

Chapter 1

Introduction

The Earth's oceans play a crucial role in weather regulation and serve as the largest natural carbon and heat sink. Maintaining the health of the oceans is synonymous with ensuring the overall well-being of our planet. Yet, up to 12 million metric tons of plastic is dumped into the oceans each year [19]. This is equivalent to more than 100000 blue whales every single year. By 2050, it is estimated that ocean plastic will outweigh all of the ocean's fish [29].

Plastics being synthetic polymers are notoriously durable and resistant to natural degradation, which contributes to their persistence in the environment. Over time, they can be broken down into smaller and smaller pieces by the sun, wind, and water but this then leads to a bigger issue of microplastics. Microplastics are ingested by marine life and have also now found their way into human lives. These can now also be found in the air and water around us, have reached the deep sea [38], and are present in large quantities in the frozen Arctic sea [28]. An average adult consumes approximately 2000 microplastic particles per year through salt alone [37].

Several organisations are actively engaged in efforts to retrieve plastic waste from the ocean, with a primary focus on addressing floating plastics, but scientists estimate that 14 million metric tons of ocean garbage rest on the seafloor [4]. To tackle this issue, an initiative funded by the European Union under the Horizon Europe Programme (Grant Agreement 101093822), SeaClear aims to eliminate seabed and floating marine pollution [34].

1-1 SeaClear

SeaClear — SEarch, identification, and Collection of marine Litter with Autonomous Robots — is an initiative aimed at solving the problem of underwater litter [34]. The project employs a fleet of autonomous, intelligent robots, including two underwater Remotely Operated Vehicle (ROV), one Unmanned Surface Vehicle (USV), and one Unmanned Aerial Vehicle (UAV) working collaboratively to monitor and collect marine litter.

Various devices, including cameras and forward-looking sonars, are employed on the observation ROV to detect litter and mark its location on a reference map. In the subsequent step,

Figure 1-1: Detecting trash underwater via SeaClear.

a larger collection ROV approaches each identified piece of litter on the map and re-detects each of these pieces with high precision before collecting it using a gripper mechanism [34].

1-2 Research Aim and Objective

The SeaClear team has effectively incorporated the capability to detect different objects underwater using a camera at this stage of development. The detection process utilizes a Convolution Neural Network (CNN) trained on SeaClear-specific data. This trained network can categorize objects into four distinct classes: plant, animal, trash, and ROV. The detection result is illustrated in Figure 1-1. The subsequent phase involves initiating the collection process for the detected trash. The collection ROV descends to collect the trash by continuously aligning with it on the seabed. Challenges arise at this step, especially in situations where multiple pieces of trash are clustered together. In such instances, the ROV is unable to pinpoint a specific target object for retrieval. Addressing this issue presents a set of intermediate questions that we aim to tackle in this study.

Challenge 1 Determining the process by which the ROV should prioritise and select objects for collection.

It is straightforward for us humans to retrieve multiple objects from the ground. After a quick scan, our brain is capable of computing somewhat of an optimal course to retrieve the objects. The easiest is to begin with the closest item and progressively continue picking up objects in a sequential manner. But it is not that straightforward for an autonomous ROV to navigate this task. The ROV needs various sensors, such as sonar, to understand the depth and distance of objects from their current location. It must calculate distances from each object every time it moves to identify the next target object for retrieval. Hence, it is easier for the ROV to also sequentially pick up the objects if each one is assigned a unique identity. A key aspect of this process involves ensuring that the ROV assigns a distinct identity to each

object it encounters. When the ROV moves to retrieve one object, others may temporarily leave its field of view. Upon re-entering its view, the ROV must recognize these objects as the same ones it previously observed, rather than assigning new identities. This ability to consistently track and identify objects is crucial for the ROV's motion planning, as it ensures accurate and efficient retrieval operations in dynamic environments.

Challenge 2 Assigning unique identities to objects.

Assigning a unique identity to each detected object in an initial image frame is a straightforward task. However, the target object must be identified in each subsequent frame and the same identity it had in the previous/initial frame must be allocated to it. This process must be followed consistently, even in situations where an object is temporarily obstructed for a few frames or exits and re-enters the frame.

The answer to this is tracking the objects continuously over time. The human mind effortlessly performs this as it can associate objects even after temporary disappearances. However, replicating this capability in a machine requires the application of various statistical and image-processing algorithms. Currently there exist multiple State-of-the-art (SOTA) object tracking algorithms, but all have been designed and implemented for conventional conditions above sea level [6, 27, 33]. A few examples can be found when object tracking algorithms were implemented underwater with stationary cameras for counting fish [42]. However, our case is unique, involving a moving underwater camera, diverse detection classes (trash, plant, animal, and ROV), and potential interruptions in object visibility due to the movement of the ROV creating disturbances at the seabed like moving sand and bubbles.

This thesis aims to initially implement existing object tracking algorithms on underwater data to evaluate their performance in this specific environment. Based on the results, the next step will involve introducing adaptations to these algorithms to enhance their efficiency and effectiveness in tracking underwater objects. This approach ensures that the algorithms are tailored to the unique challenges posed by underwater conditions, leading to more accurate and reliable tracking outcomes.

1-3 Thesis Organisation

In Chapter 2, a comprehensive overview of object-tracking techniques is presented, including a detailed discussion of the key steps involved in these processes. This chapter also explores the current SOTA methods for multiple object tracking. Chapter 3 introduces the novel dataset developed specifically for this research, detailing the methodology used to establish ground truth for the data. In Chapter 4, the focus shifts to the proposed modifications to an existing object-tracking algorithm, along with the introduction of a new tracking algorithm that incorporates multiple motion models. Chapter 5 presents the experimental results, with a focus on the specific use case, and provides a critical evaluation of the strengths and weaknesses of each approach.

The thesis concludes by summarising the key findings and offering recommendations for future research directions in Chapter 6.

Chapter 2

Overview of Object Tracking

Object tracking focuses on the continuous identification and monitoring of a target object's spatial and temporal changes within a video or image sequence. The primary goal is to follow the target's movement and maintain its unique identity over time despite variations in scale, orientation, lighting conditions, and occlusions [44]. This field has found widespread applications across diverse domains, ranging from surveillance and security to robotics, automotive systems, augmented reality, and beyond. Object tracking is pivotal in extracting meaningful information from dynamic scenes, enabling automated systems to make decisions, anticipate future actions, and respond accordingly.

2-1 Introduction

Object tracking can be categorised into online and offline tracking based on when and how the tracking algorithm processes the video data. Online tracking operates in real-time as the video frames are captured. It continuously processes each incoming frame and updates the object's position on the fly. Offline tracking, on the other hand, processes the entire video after it has been recorded. The algorithm analyzes the video as a whole rather than in real-time. Online tracking algorithms need to be memory-efficient and computationally lightweight to handle real-time constraints. They often prioritize speed over exhaustive analysis, while offline tracking has more flexibility in terms of computational resources and memory usage. It can afford to use more complex algorithms and analyze the video in greater detail since there is no time constraint. It also utilizes a batch of frames to process the data, observations from all the frames are required to be obtained in advance and are analysed jointly to estimate the final output. The former method is crucial for applications where real-time decision-making is essential, such as surveillance, robotics, and live video analysis while offline tracking is often used in scenarios such as post-event analysis, video summarisation, or offline processing for research purposes like tracking the yearly migration of animals and birds. Since the SeaClear initiative aims to collect trash as it is detected, the focus of this literature review will be on real-time algorithms for tracking.

Currently, real-time object-tracking algorithms can be divided into two groups based on the methods used for tracking, these being detection-based tracking and detection-free tracking. In the former, the system relies on an initial detection step to identify and locate objects in the scene. This strategy is also commonly referred to as tracking-by-detection. First, an object detector has to be trained in advance on specific kinds of targets, such as pedestrians, vehicles or face. Detection-free tracking requires manual initialisation of a fixed number of objects in the first frame, then subsequent tracking.

Feature	Detection-based	Detection-free
Initialisation	Automatic, imperfect	manual, perfect
No.of objects	Varying	Fixed at the start
Drawbacks	Performance depends on object detector	cannot handle new objects

Table 2-1: Comparing detection-based and detection-free tracking.

Detection-based tracking is more popular due to its ability to automatically handle the discovery of new objects and the termination of disappearing objects—a task that detection-free tracking struggles with. However, detection-free tracking is advantageous in that it does not rely on pre-trained object detectors and hence there is no requirement for large datasets to train a model for detection. SeaClear currently implements a trained neural network that is capable of detecting various objects such as plants, animals, trash, and Remotely Operated Vehicle (ROV) in the underwater environment. Examples of the detections are shown in Figure 1-1 and Figure 2-2. With the availability of a well-performing detection model, tracking-by-detection is a better method to use than the detection-free method since the drawback of the latter defeats the purpose of proposed object tracking. All the objects cannot be initialised manually at the start because we will be unaware of how many objects the ROV encounters. The detection and tracking of the objects will happen in real-time and hence we will be focusing on algorithms suitable to this approach.

Tracking-by-Detection The majority of the State-of-the-art (SOTA) object tracking algorithms are based on the tracking-by-detection method where the tracking is performed through continuous detection and association of objects. This method encompasses multiple stages, with each step employing a range of methods, including deep learning, signal processing, and statistical modeling. The various steps are discussed below [1, 44].

- Detection: Identifying the presence of an object in the initial frame of a video or image sequence. This can be done using various object detection or feature detection algorithms.
- Initialisation: Establishing a reference or model of the target object based on its appearance, features, or both by giving it a unique identity. This serves as the starting point for subsequent tracking.
- Localisation: Continuously estimating the object's position in subsequent frames, often through methods like motion estimation or feature matching.
- Data Association: Linking the detected object in the current frame with the previously identified target. This is a critical step in maintaining the object's identity over time.

Each of these steps can be addressed via different techniques depending on the ultimate use case. In the subsequent subsections, the above steps will be discussed in more detail giving examples of relevant work and keeping in mind an underwater, multi-class environment.

2-2 Object Detection

Object detection serves as a foundational step for object tracking when using the tracking-by-detection method, providing initial information about the objects in a scene. It is imperative to define the specific requirements for object detection by keeping in mind what needs to be tracked. Various examples are illustrated in Figure 2-1, showcasing different forms of detection. Different methods are suitable for each of these forms. The most commonly used form is the bounding box shown in Figure 2-1(a). Deep learning-based approaches have emerged as the predominant method for detecting objects in this format. A classic example is using trained Convolution Neural Network (CNN) [30] to extract features from images and hence detect objects. Subsequent advancements like the fast region-based (Fast R-CNN) [42] have improved the speed and accuracy of object detection.

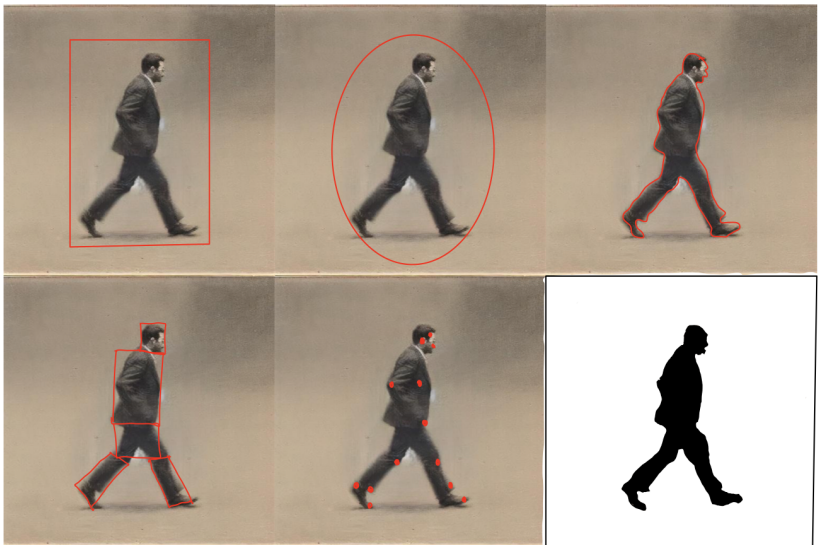


Figure 2-1: Illustration of object tracking forms. (a) bounding box, (b) ellipse, (c) contour, (d) articulation block, (e) interest point, (f) silhouette.

One of the most popular real-time object detection algorithms is the You Only Look Once (YOLO), known for its speed and efficiency. It processes the entire image in a single forward pass through a neural network by dividing the input image into smaller grid cells. Object detection is done on each of these smaller grids, making the process faster. A bounding box is presented at each detected box with a confidence score indicating the model's confidence that the box contains an object. The low confidence scores are filtered out at the end. The current detection model for SeaClear uses a YOLOv8 model, which is the latest iteration in the YOLO series, trained on SeaClear data. The model performance can be seen in Figure 2-2 where it detects fish close to the camera and a rope as trash. This image is from a video sequence

Figure 2-2: Detecting underwater objects in using the SeaClear YOLOv8 model on unseen data.

taken while scuba diving in Marseilles, France, and is not part of any dataset obtained or used by the SeaClear team. This shows that the model performs quite well on unseen, new data.

For contour or silhouette-type detection and tracking, the exact boundaries of the object have to be extracted from an image. Even though deep-learning-based methods are widely used, other traditional methods like optical flow or edge detection which uses colour and pixel information generally give better results. Dalal et al. [9] use a locally normalised histogram of gradient orientation features in a dense overlapping grid which is used for human detection. The same has been used successfully to track humans via drone [13].

Detecting interest points, illustrated in Figure 2-1(e) also known as keypoint detection, serves various purposes such as image stitching. It is instrumental in recognising specific big and small human movements such as running, walking, and swimming, as well as smiling, crying, and yawning [40]. A popular method in computer vision for detecting and describing local features in images is the Scale-Invariant Feature Transform (SIFT). This involves identifying distinctive key points and extracting descriptors to represent the local image structure. The key points are invariant to image rotation and scale and are robust across a substantial range of affine distortion, noise, and change in illumination [23]. The SIFT algorithm is discussed further in Section 2-3-2.

For this research, employing bounding box detection seems to be a suitable method for both object detection and tracking. This approach is preferred because it doesn't necessitate the identification of specific features of the trash. Additionally, the existing object detection model for SeaClear effectively detects objects and provides bounding box outputs, making it practical to focus on scenarios involving bounding boxes from now on.

2-3 Object Initialisation and Localisation

Once an object is identified, it must be assigned a unique reference or identity. This identity must remain consistent across all subsequent frames to ensure that the object detected in

earlier frames is recognised as the same in later ones. Two primary methods are typically employed for this purpose: motion estimation and feature matching. Motion estimation involves predicting the object's position and velocity using a motion model, while feature matching relies on comparing the object's appearance and key-points to track it. The choice between these methods depends on the specific use case. For instance, if the objective is to count the number of cars of different colors, feature matching would be more appropriate, as cars with similar colors would share visual features and could be grouped. However, this approach would be less effective for tracking the individual trajectories of each car. The following sections will explore these different methods for object localisation in detail, discussing their strengths and weaknesses in various scenarios.

2-3-1 Motion Estimation

In object tracking, motion estimation is crucial for predicting the future positions of objects, enabling real-time monitoring and analysis [35]. This approach is especially useful in situations where an object's appearance might change or when multiple objects share similar features, making traditional methods like feature matching less dependable. In such cases, motion estimation is more effective for tracking objects. The most widely used method for motion estimation, Kalman Filters, will be discussed in detail next.

Kalman Filter Kalman filter is a mathematical algorithm that estimates the state of a system in the presence of uncertainty and measurement noise. The filter achieves this by combining predictions of the system's state with actual measurements in an optimal way, considering the uncertainties associated with both the system's dynamics and the measurements [20]. The uncertainty related to the system dynamics, also known as the process noise, captures factors that are not accounted for in the state transition model. Measurement uncertainty accounts for the inaccuracies or disturbances present in the sensor measurements. It represents the discrepancy between the true system state and the observed measurements. Both these uncertainties are often modeled as a zero-mean random variable with a known covariance matrix. For a system with state x, state transition matrix F, control input matrix B, and z_k as the measurement at k time, the prediction steps are defined as

$$\hat{x}_{k|k-1} = F \cdot \hat{x}_{k-1|k-1} + Bu_k \tag{2-1}$$

for the state prediction, where $\hat{x}$ denotes the estimated state and the subscript $k|k-1$ represents time step k given the state estimate at the previous time step $k-1$. The covariance prediction step is given as

$$P_{k|k-1} = F \cdot P_{k-1|k-1} \cdot F^T + Q \tag{2-2}$$

Here Q is defined as the process noise covariance matrix and P is the covariance of the state estimate. The update step of the Kalman filter process comprises the Kalman gain update, state update, and covariance update. The Kalman gain adjusts the predicted state estimate based on the reliability of the measurements and the predicted state. It helps strike a balance between the predicted state and the observed measurements, giving more weight to the more reliable source. The Kalman gain is given by

$$K_k = P_{k|k-1} \cdot H^T \cdot (H \cdot P_{k|k-1} \cdot H^T + R)^{-1} \tag{2-3}$$

where H is the measurement matrix, and R is the measurement noise covariance matrix. The state update step is given by

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \cdot (z_k - H \cdot \hat{x}_{k|k-1}) \tag{2-4}$$

and the covariance update as

$$P_{k|k} = (I - K_k \cdot H) \cdot P_{k|k-1} \tag{2-5}$$

where I is an identity matrix.

In the case of object tracking, there is no control input, u_k. A challenge in using the Kalman filter for object tracking is the requirement for a precise discrete-time linear dynamic system. This system must possess a state transition matrix that accurately relates the current state and the subsequent state. Mathematically this is given as

$$x_{k+1} = Ax_k \tag{2-6}$$

with A being the state transition matrix.

Another challenge is defining the state vector x. Vasuhi et al. in [39], defined the state vector as $[p_x, p_v, v_x, v_y]^T$ for a single human tracking with (p_x, p_v) as the 2D coordinates of the object and (v_x, v_y) the velocity in each direction. Bewley et al. in [6] defined the state vector as $[u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$, where u and v represent the horizontal and vertical pixel location of the center of the target, while the scale s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively.

Creating a precise discrete-time linear state space model for entities like animals, fish, and trash, as relevant to our context, presents a considerable challenge. Additionally, the utilisation of Kalman filters is constrained by the necessity for the specified noise to follow a Gaussian distribution, and for the object model to be linear. Usually, the motion of a fish and floating trash along with the water current does not follow a linear motion. Finding an accurate linearisation for the nonlinear motion is of utmost importance for the Kalman filter giving satisfactory results.

2-3-2 Feature Matching

Feature matching is a widely used method for object localisation which focuses on the identification and correspondence of distinctive features between consecutive frames. The goal is to find points or regions in one image that correspond to the same points or regions in the previous image. These regions are generally features like corners, edges, or key points [36]. SIFT [23] is a widely used algorithm in computer vision for detecting and describing distinctive features in images. It is known for its robustness to changes in scale, rotation, and illumination. It begins by creating a series of blurred versions, called scale levels, of the original image using Gaussian smoothing [11]. In this process, the image is convoluted with a Gaussian function kernel. A Difference of Gaussians is computed by subtracting adjacent blurred images in the scale-space. Local extrema in the Difference of Gaussians images are identified as potential keypoint locations. The algorithm assigns orientations to these identified key points based on local gradients, ensuring rotation invariance. A local image patch

(a) Strong water currents. Notice that the objects in the background are not hazy.

(b) Illustrating the blueish-green tint in underwater images, part of the SeaClear dataset.

Figure 2-3: Images from the SeaClear dataset.

around each key point is taken, and a descriptor is generated to represent the key point's appearance. Keypoint descriptors from two images are matched by comparing their feature vectors. A common method is to use the Euclidean distance between the descriptors.

A major drawback of this method is that it needs stable SIFT features that persist throughout the video, but this is unlikely to exist in underwater environments due to illumination changes and most importantly occlusion. Also, I suspect that similar-looking trash, fishes, and plants would have similar key points and descriptors confusing the algorithm and hindering tracking in a multi-object scenario. In the underwater environment, the true colour of objects is not captured by cameras and there is usually a blueish-green tint, as seen in Figure 2-3a and Figure 2-3b, making the colour histogram very biased to that feature. These considerations render the feature-matching technique unsuitable for this particular case, leading to a focus on motion estimation techniques for the remainder of the research.

2-4 Data Association

Data association refers to the process of linking observations or estimates obtained from sensors or algorithms to the corresponding real-world entities they represent. In the field of object tracking, the goal is to associate the detected objects from frame to frame and hence maintain a consistent identity for the object. For example, in the Kalman filter seen in section 2-3-1, the prediction of each object is to be associated with the detection in the next frame. This is done by a data association algorithm.

The simplest data association method is to associate each detection in the current frame with the nearest detection in the previous frame based on a distance metric, such as Euclidean distance or Mahalanobis distance [17]. The Euclidean distance is the distance between any two points, P and Q given as

$$d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} \tag{2-7}$$

and the Mahalanobis distance is a measure of the distance between a point P and a distribution D. For a point P from a distribution D with mean μ and covariance matrix Σ, the Mahalanobis distance is given by

$$D_M(P) = \sqrt{(P-\mu)^\top \Sigma^{-1}(P-\mu)}. \tag{2-8}$$

This is the multi-dimensional generalisation of measuring how many standard deviations away P is from the mean of D. While Euclidean distance is appropriate for uncorrelated variables, Mahalanobis distance is more suitable when there is a correlation between variables. It reduces to the Euclidean distance when the covariance matrix is the identity matrix. Effective data association for object tracking use cases ensures that the correct identity of objects is maintained across frames, even in the presence of challenges such as occlusions, appearance changes, and noise in complicated and crowded scenarios. For the case of object tracking, each object prediction in the current frame is compared with every object present in the previous frame and matched appropriately. This matching cannot be done with a distance metric but needs to be solved optimally.

The Hungarian algorithm [22], also known as the Kuhn-Munkres algorithm, is a commonly used method for data matching and solving multiple problems together. It is a combinatorial optimisation algorithm [12] that deals with finding the optimal assignment of a set of tasks to a set of agents, each with an associated cost or weight. In other words, the goal is to minimise the total cost of assigning tasks to agents while ensuring that each task is assigned to exactly one agent, and each agent is assigned to at most one task. This is mathematically represented as a cost matrix, where the rows correspond to tasks, the columns correspond to agents, and each matrix entry represents the cost of assigning a particular task to a specific agent. A common combinatory optimisation problem is the traveling salesman problem. The steps followed by the Hungarian algorithm are given below.

- Initialisation: Starting with a cost matrix that represents the costs or profits associated with assigning tasks to agents.
- Row Reduction: For each row of the cost matrix, the minimum cost in that row is subtracted from all the elements in the row. This ensures that at least one zero is present in every row.
- Column Reduction: For each column of the cost matrix, the minimum cost in that column is subtracted from all the elements in the column. This ensures that at least one zero is present in every column.
- Cover Columns and Rows: All columns and rows containing zeros are covered with a line. If the number of lines is equal to the size of the matrix, i.e. (n for a n*n matrix), the algorithm is done. Otherwise, the algorithm proceeds to the next step.
- Update Costs: The smallest element that is not covered by a line in the previous step is found and subtracted from all uncovered elements, and added to all elements that are already covered.
- Repeat: The last two steps are repeated till an optimal solution is reached.

In the case of data association for object tracking the cost matrix is computed for the Hungarian algorithm with different distance metrics such as the Euclidean distance, convolutional cost, Mahalanobis distance, etc.

2-5 Above-land Object Tracking Algorithms

As evident from the preceding sections, object-tracking is a multi-stage process and the selection of a specific algorithm at each step requires careful consideration. This section consolidates all the individual steps to engage in a discussion and comparison of diverse SOTA object-tracking algorithms.

2-5-1 IOU Tracker

The Intersection-over-union (IOU) tracker is a very simple method based on the assumption that the detector produces a detection per frame for every object to be tracked, i.e. there are barely any few missing detections [7]. Furthermore, it is assumed that detections of an object in consecutive frames have an unmistakably high overlap IOU which is commonly the case when using sufficiently high frame rates. The IOU is a popular metric used in object detection that calculates the amount of overlapping between two bounding boxes which are usually a predicted bounding box and a ground truth bounding box. This metric is defined as the ratio of the intersection of the two boxes' separate areas' to their combined areas. Mathematically this metric is defined as

$$\frac{|A \cap B|}{|A| \cup |B|} \tag{2-9}$$

where A and B are the two bounding boxes.

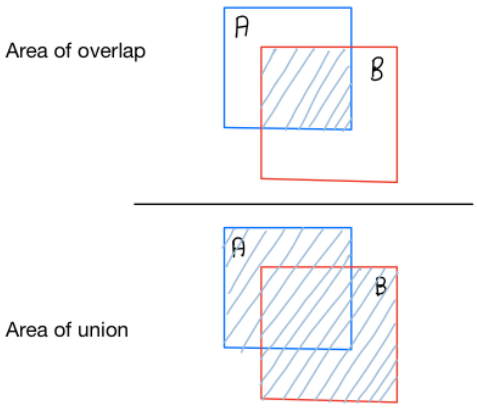


Figure 2-4: Visualising the IOU metric.

A simple IOU tracker continues tracking an object by associating it with the detection in the previous frame that has the highest IOU score, provided that this score surpasses a specified

threshold, σ_{IOU}. In [7], the performance is further improved by filtering out all tracks with a length shorter than a threshold, $t_{\min}$, and the ones without at least one detection with a confidence score above σ_h. Short tracks are removed because they usually root in false positives and add clutter to the output. Requiring a track to have at least one high-scoring detection ensures that the track belongs to a true object of interest while benefiting from low-scoring detections for the completeness of the track. Only the best-matching, unassigned detection is taken as a candidate to extend the track.

If the tracker is used online in conjunction with a state-of-the-art detector, the computational cost compared to the detectors becomes negligible. Therefore, tracks can be obtained at almost no additional computational cost from the detections. This algorithm outperforms the SOTA in terms of object tracking in a simple, non-crowded environment with only a fraction of the complexity and computational cost. With a good detection model available, a simple IOU tracker may perform well on the SeaClear data.

2-5-2 SORT

Simple Online and Real-time Tracking (SORT) [6] is one of the most widely used object-tracking algorithms. It is based on a tracking-by-detection framework which implies that the tracking system relies on first detecting objects in each frame of a video or image sequence and then associating those detections across frames to establish the trajectory of the object. Wojke et al. in [6], adopt a straightforward approach that avoids incorporating appearance features beyond the detection component. Object detection is achieved through a fast region-based CNN, which produces bounding boxes around objects. SORT relies solely on the bounding box's position and size for both motion estimation and data association. Motion estimation is then employed via a Kalman filter, utilising a linear constant velocity model that approximates the inter-frame displacements of the objects and remains independent of other objects and camera motion. The state of each object is characterised by a state vector defined as

$$[u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T. \tag{2-10}$$

Here u and v represent the horizontal and vertical pixel location of the centre of the object, while s and r represent the scale (area) and the aspect ratio of the objects's bounding box respectively.

Data association is done via the Hungarian algorithm where the assignment cost matrix is computed as the IOU distance between each detection and all predicted bounding boxes from the existing targets. A lower limit in the IOU is imposed to reject assignments where the detection to target overlap is poor. After associating an object with its detection, the detected bounding box is used to update the target state where the velocity components are solved optimally via a Kalman filter framework as explained in Section 2-3-1. If no detection is associated with the object, its state is simply predicted without correction using the linear velocity model.

The most important part of the algorithm is creating identities. When objects enter or leave the video, identities are assigned or removed. For new objects, the tracker is set up using the size and position of the bounding box of the detected object. Since the speed of the object is unknown at this point, it is assumed to as not moving and the velocity is set to zero. Hence the

covariance of the velocity component is set to a large value indicating uncertainty. The new tracker has a probationary period where it needs to be associated with additional detections. This is done to gather enough evidence and avoid tracking false positives. If an object is not detected for a certain number of frames tracking is terminated. to prevent the number of trackers from growing indefinitely and reduce errors caused by making predictions without corrections from the detector for an extended time. This parameter is set to 1 in [6]. This decision is based on the fact that the method used for predicting the object's future position is not very accurate given the constant velocity model is a poor predictor of the true dynamics and the main focus is on tracking from frame to frame. If an object that was previously lost reappears, the tracking resumes with a new identity. This approach simplifies the tracking process and is more concerned with short-term tracking rather than re-identifying objects over a longer period.

The majority of motion estimation-based algorithms have been built on top of the SORT to improve tracking performance by handling longer tracking and occlusion. Some of them are discussed next.

2-5-3 ByteTrack

Zhang et al. in [43] introduces a simple, effective, and generic data association method, Byte. In contrast to most algorithms that only utilize high-score detection boxes, here almost every detection box is retained and separated into high-score ones and low-score ones. The low-confidence detection boxes indicate occluded objects improving tracking performance.

Once objects are detected, the low-scoring and higher-scoring detections are segregated based on a set threshold. After segregating, the Kalman filter is employed to predict the new locations in the current frame for each track. The initial association involves applying the Hungarian algorithm between the high-score detection boxes and all the tracks T, just like the SORT algorithm. Tracklets that fail to match an appropriate high-score detection box, most probably due to occlusion, motion blur, or size changes, are excluded and stored separately in T_{remain}. A subsequent association is carried out between the low-score detection boxes and these unmatched tracks. Following this second association, still unmatched low-score detection boxes are removed as they are now considered background. The tracks left unmatched after the second association is placed into T_{lost}. For each track in T_{lost}, only when it exists for more than a certain number of frames is deleted from the tracks T. This is to ensure if an object re-enters the scene, it can be matched with a previous track in T_{lost} keeping its original identity. This ensures there are minimum ID switches.

A simple and strong tracker named ByteTrack is designed in [43] by equipping the high-performance detector YOLO [15] with the association method Byte.

2-5-4 BoT-SORT

Performance of the SORT-like approaches mainly depends on the quality of the predicted bounding box of the tracklet. In many complex scenarios, predicting the correct location of the bounding box may fail due to camera motion, which leads to low overlap between the two related bounding boxes and finally to low tracker performance. BoT-SORT [2] builds on top of

ByteTrack by additionally handling camera motion compensation. This is done by adopting conventional image matching to estimate the camera motion. The global motion compensation (GMC) technique used in OpenCV's [8] video stabilisation module with affine transformation is followed. This image registration method is suitable for revealing the background motion. First, the extraction of image key points [36] takes place, followed by sparse optical flow, which focuses on tracking a subset of points instead of all the pixels/features for feature tracking. The sparse matching techniques are leveraged to selectively focus on specific features in a scene, excluding the influence of dynamic objects and thus estimating the background motion more accurately.

Another variation of the algorithm is the BoT-SORT-ReID which enhances object tracking by additionally combining appearance features with motion cues, providing a more robust and accurate tracking system compared to traditional methods like SORT. BoT-SORT-ReID does this by incorporating appearance embeddings extracted from a pre-trained object reidentifying deep neural network. These embeddings are essentially unique visual signatures that describe the appearance of an object, capturing details such as color, texture, and shape. The appearance embeddings in BoT-SORT-ReID allow the algorithm to recognize and re-identify objects when they reappear or change direction, ensuring that the same object is tracked consistently throughout the sequence.

While appearance embedding is a valuable feature for tracking objects, it involves the complex task of extracting and comparing detailed object features to ensure consistent identification over time. This process is computationally demanding, particularly in real-time scenarios. Algorithms must quickly handle and process large volumes of data in high-dimensional spaces, which can strain computational resources and result in latency issues. Therefore, we have opted not to use this algorithm in our research, as it does not meet the real-time processing requirements we need.

2-6 Object Tracking Benchmarks

An object-tracking benchmark is a standardised evaluation framework used in the field of computer vision to assess and compare the performance of different object-tracking algorithms. These benchmarks typically consist of a set of video sequences with annotated ground truth which provides information about the correct position and extent of the objects being tracked in each frame. By using a common benchmark, researchers can objectively compare the strengths and weaknesses of different tracking algorithms. Along with the datasets, different benchmarks define different performance metrics based on specific use cases and requirements. They also provide a common format for both the ground truth and the output generated by tracking algorithms, facilitating straightforward comparisons.

There have been several object-tracking benchmarks for both single and multiple object tracking. The Object Tracking Benchmark (OTB) [41] and Visual Object Tracking (VOT) [21] were some of the first to do so, but both are focused on single object tracking. The most popular is the Multiple Object Tracking (MOT) benchmark [10], which has evolved into an annual challenge in computer vision and tracking research, introducing increasingly complex datasets each year. The challenge typically includes diverse scenarios, such as crowded scenes and occlusions, to assess the robustness and accuracy of tracking algorithms in various real-world conditions. Such benchmarks make it easier for researchers and developers to compare

and advance the SOTA algorithms. In total 22 different datasets were published between 2015 and 2023 for the MOT Challenge. Of these 22, only one, 3D-ZeF [32] is remotely relevant to the problem of underwater object tracking as it contains Zebrafish tracking in a laboratory environment. A laboratory environment has good lighting and fish tanks are not deep enough to accurately depict light distortion, and refraction that would occur in deep waters. There is also an absence of dirt and water currents which contributes heavily in a real underwater setting. Unfortunately, all the other existing tracking datasets are open-air and are based on single-category tracking focusing on either pedestrians or vehicles.

Since no publicly accessible dataset for underwater object tracking exists, the aim is first to develop a novel underwater multiclass multi-object tracking dataset. This dataset will encompass a variety of object trajectories, both short and long, smooth and irregular. It aims to represent diverse underwater conditions, including clear and murky water, and varying water currents.

2-7 Comparisons

The SOTA object tracking algorithms are compared in Table 2-2, highlighting the difference in the localisation and association steps and the advantages and disadvantages. Both the IOU tracker and SORT are lightweight and computationally efficient, but the motion prediction in SORT gives it an upper hand over the former. The association method is also optimal in SORT rather than just matching with the highest IOU score, this improves performance in a crowded scenario. When compared with SORT, ByteTrack gives a higher tracking accuracy and lesser numbers of identities on the MOT17 dataset [26]. This result suggests that a simple Kalman filter along with a long-range data association can achieve fewer identity switches when the detection boxes are accurate enough. There is no need for any deep learning models to help "remember" the object features to improve the tracking capabilities.

BoT-SORT and BoT-SORT-ReID give the best performance from all SORT-like algorithms on MOT20 datasets containing above-the-sea datasets like pedestrian tracking, but they still have several limitations. In scenes with a high density of dynamic objects, the estimation of the camera motion may fail due to a lack of background key points. Wrong camera motion may lead to unexpected tracker behavior. Another real-life application concern is the run time. Calculating the global motion of the camera can be time-consuming when large images need to be processed. With almost no underwater datasets for re-identification training a neural network that gives an accurate feature embedding generation for BoT-SORT-ReID is not feasible. Even if there were enough datasets, using appearance embeddings would not be ideal. Unlike humans, fishes of the same species and the same kind of trash will have the same features and similar feature embeddings. Despite these challenges, BoT-SORT achieves tracking accuracy score 63.3% on the MOT20 dataset. In comparison, ByteTrack is close behind, with a HOTA score of 61.3%. These metrics indicate that while BoT-SORT slightly outperforms ByteTrack, both offer strong tracking accuracy and are efficient, making them suitable for real-time applications. Meanwhile, the traditional SORT algorithm, though faster and simpler, generally lags behind in these dense tracking environments. It achieves a HOTA score of 45-50% on the MOT20 dataset, highlighting the advancements made by newer methods like BoT-SORT and ByteTrack.

Hence the SORT, ByteTrack, and BoT-SORT seem like good candidates for a satisfactory tracking performance along with less computation time. fish of the same species and similar debris would likely produce indistinguishable features.

	IOU	SORT	BoT-SORT
Detection	CNN	CNN	CNN
Localisation	Bounding box No motion prediction	Kalman Filter with scale and aspect ratio of the bounding box	Kalman Filter with height and width of the bounding box
Association	IOU scores	Hungarian Algorithm with IOU costs	Byte - consider both high and low scoring detecting boxes
Advantages	Computationally efficient	Lightweight	Camera motion compensation
Disadvantage	Performance dependent on detection model	Poor performance in occlusion and nonlinear motion	High image resolutions slow down the algorithm

Table 2-2: Comparing the SOTA object tracking algorithm based on motion estimation methods.

2-8 Summary

This chapter began by outlining the foundational steps in object-tracking algorithms, focusing on the tracking-by-detection approach. In this approach, object detection, localisation, and association are the primary stages, each with its own set of applicable algorithms. Localisation techniques such as motion estimation and feature matching play a critical role, while the Hungarian algorithm is highlighted for its effectiveness in solving linear assignment problems, thereby optimising data association.

The discussion then delved into various SOTA tracking algorithms, starting with the IOU tracker, which simplifies the tracking process by associating objects based solely on the highest IOU score. Following this, the SORT algorithm was explored. This algorithm enhances object localisation through Kalman filtering and uses an IOU score-based cost matrix within the Hungarian algorithm for more precise data association. The chapter then discussed ByteTrack, which builds on SORT by incorporating the Byte association step, which improves data association by considering both high-scoring and low-scoring detected bounding boxes.

Additionally, the chapter reviewed BoT-SORT and BoT-SORT-ReID, which introduce motion compensation to address camera stabilisation issues. These methods leverage OpenCV's video stabilisation module, using background key points to enhance the robustness of ByteTrack. The latter also incorporates an object re-identification neural network to enhance performance in occluded scenarios.

The chapter concluded by highlighting the limitations and challenges within the current object-tracking methodologies and set the stage for the next chapter. The forthcoming chapter will explore existing datasets for object tracking, evaluate their suitability for the specific use case, and develop a novel underwater dataset for future research and implementation steps.

Chapter 3

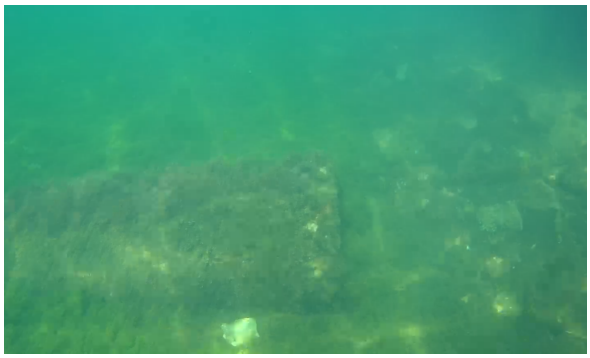
Novel Dataset for Underwater Object Tracking

This research primarily aims to implement and compare the tracking performance of several State-of-the-art (SOTA) algorithms, with a subsequent focus on enhancing object tracking in underwater environments. A substantial dataset is essential to conduct a fair evaluation of the algorithms. Since the objective is to track the objects, a collection of sequential images or videos is required. The model for detecting objects in SeaClear currently has four different classes, namely, animal, plant, Remotely Operated Vehicle (ROV), and trash. So we also want to achieve multi-class multi-object tracking underwater.

3-1 SeaClear Dataset

As seen in the previous chapter, the Multiple Object Tracking (MOT) Challenge serves as a significant collaboration among researchers worldwide, contributing groundbreaking efforts to improve object-tracking algorithms. However, the datasets available through the MOT benchmark comprise only of open-air conditions. An alternative dataset, UOT100 [31], includes 104 underwater video sequences with over 74000 annotated frames derived from both natural and artificial underwater videos, featuring a diverse range of distortions. While lacking trash-related content, this dataset includes videos with ground truth information for marine animals. Most sequences depict single objects in a frame, and the videos have been sourced from platforms like YouTube and National Geographic, with manual annotations provided

While this dataset can be used initially, it does not provide an ideal match for our purposes. For instance, SeaClear stands out with the challenge of tracking four different classes and a moving camera, which we have not yet come across while looking for datasets since even open-air datasets are based on fixed CCTV camera footage. Also, the image sequences in the UOT100 dataset are close to the surface, captured with good large camera equipment and hence have good quality with lots of light. The SeaClear team has conducted extensive tests using their ROV in diverse locations. The dataset derived from these tests will serve as the



(a) Light distortion underwater.

(b) Bubbles caused due to water currents.

Figure 3-1: Various characteristics specific to an underwater dataset.

(a) Multiple trash objects together in murky water.

(b) Single trash in clear water, with bounding box.

Figure 3-2: Different scenarios for trash collection from the SeaClear dataset.

definitive benchmark for evaluating algorithm performance in this research. SeaClear goes to the ocean floor to collect trash where light reaches very scarcely and after lots of refractions and reflection. As seen in Figure 3-1, light distortion, water currents, and additional motion due to these currents bring more uncertainty and challenges in this environment.

3-2 Ground Truth Annotations

Collecting and processing videos for tracking is not enough to make a dataset capable of being used for object tracking. The video sequences need to be annotated with ground truth information to compute performance metrics ensuring a definitive and comparable result rather than relying on intuition and the human eye. To ensure a consistent comparison, the videos will undergo a self-annotation process.

Instead of manually annotating all frames, the existing detection model, is coupled with a Python script, to generate initial ground truth annotations. While this automated process will be flawed, inaccuracies will be corrected through manual annotation after this. The subsequent steps for this process are outlined below.

- Detecting objects with their classes, confidence percentage, and bounding boxes using the existing You Only Look Once (YOLO) detecting model.

Figure 3-3: Ground truth and expected tracking results.

- Using the output from the detector, unique and consistent identities will be assigned to the objects. This will be done by comparing the class and bounding box location and size from the previous frame.

- Errors and missed detections will be corrected by hand after this process.

These ground truths will be stored in a `.txt` file and can be changed to any other format like `.JSON` if and when required. The format of the ground truths will be given as,

<Frame number><ID><Class number><left><top><width><height><Confidence>.

Each video will have a separate `.txt` file containing the ground truth annotations. <left> and <top> represent the x and y coordinates of the left-top corner of the bounding box respectively while the width and the height of the bounding box are represented by <width> and <height> respectively. After annotating the ground truth, the result is expected to be as seen in Figure 3-3.

The `Python` script for automatic annotations is given in Appendix B. After the annotation process, the different kinds of object trajectories seen in the dataset are given in Figure 3-4. Each subfigure corresponds to one video sequence. The pixel coordinates of the top left corner of the bounding box are plotted illustrating the two-dimensional location of the object and its corresponding trajectory. The trajectories in the third subfigure are illustrated in Figure 3-5 with the frame numbers on the x-axis. This 3D plot helps to see the gaps in the trajectories. The trajectories shown in Figure 3-4 are only from four of the videos. In total, 82 different object trajectories are annotated, with different length and motion characteristics.

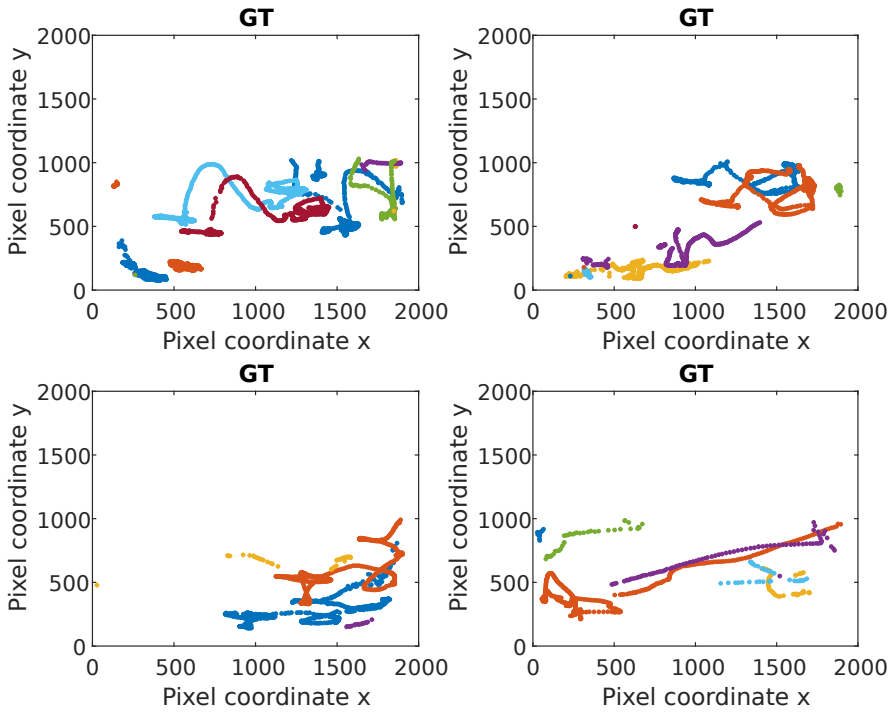


Figure 3-4: Ground truth trajectories in the 2D space. The pixel coordinates are plotted on the x and y-axis.

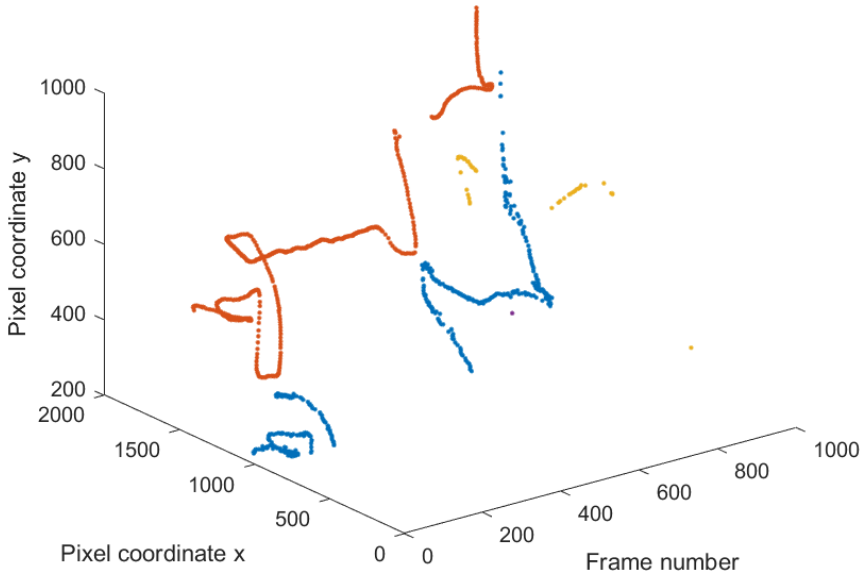


Figure 3-5: Ground truth trajectories of the third subfigure in Figure 3-4 in the 3D space. The pixel coordinates are plotted on the y and z-axis and the frame number on the x-axis.

3-3 Summary

This research aims to implement and compare the performance of various SOTA object-tracking algorithms, with a focus on enhancing tracking in challenging underwater environments. A key aspect of this study is the creation and annotation of a dataset crucial for fair algorithm evaluation. While the MOT20 benchmark is widely used for above-land tracking, particularly for vehicles, it is limited to open-air conditions and lacks relevance for underwater scenarios. Similarly, the existing UOT100 dataset, while containing useful underwater data, falls short in several aspects, including the absence of trash-related content and limitations in multi-class, multi-object scenarios. To address these gaps, the SeaClear project provides a custom dataset captured by an ROV on the ocean floor, presenting unique challenges such as low light, light distortions, and water currents. An initial automatic annotation process, using an existing YOLO detection model, will be refined manually to ensure accurate ground truth data for evaluating tracking performance. This annotated data will serve as the benchmark for the study, offering diverse object trajectories and scenarios that reflect real-world underwater tracking challenges.

Chapter 4

Novel Enhancements for Underwater Object Tracking

4-1 State-space Representation

As seen in the current State-of-the-art (SOTA) for object tracking, the state space is represented using the bounding box coordinates and speed of the object. The location of the object is represented using the center pixel, the aspect ratio, and the area of the bounding box. This was done keeping in mind the motion of a pedestrian or a car, both of which generally move in a two-dimensional plane similar to the camera setup. A CCTV camera is mounted on a building, capturing cars and pedestrians moving along the road. The spatial relationship between the cars or pedestrians remains constant, meaning that their aspect ratio— the ratio of the bounding box's width to its height—stays the same. Because the cars and pedestrians do not move vertically and the camera does not rotate, they appear as two-dimensional objects. The aspect ratio remains unchanged since the only variation is in their distance from the camera, which affects the width and height of the bounding box but not their ratio.

When a camera is in motion, it can move in any direction, particularly in the fluid environment of water, where objects like fish and debris can float and rotate freely. This means that the two-dimensional projection of an object, represented by a bounding box around it, will not maintain a consistent shape or structure. For example, a fish swimming toward the camera will appear with a different aspect ratio than the same fish swimming sideways. Similarly, even with stationary debris on the seabed, the camera's perspective changes as it moves above it.

Keeping this in mind, the first proposed change is to change the state vector of the object in the Kalman Filter. In the Simple Online and Real-time Tracking (SORT) algorithm and the following algorithms built on it, ByteTrack, BoT-SORT, the state space is defined as

$$[u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T. \tag{4-1}$$

Here u and v represent the horizontal and vertical pixel location of the centre of the object, while s and r represent the scale (area) and the aspect ratio of the objects's bounding box respectively.

We now propose a change, not taking into consideration the aspect ratio and the area of the bounding box, but just the width and the height. Also instead of taking the centre pixel location, the pixel location of the top left corner of the bounding box is considered. This makes the state vector an 8-variable vector defined as

$$[x, y, w, h, \dot{x}, \dot{y}, \dot{w}, \dot{h}]^T. \tag{4-2}$$

with (x,y) defined as the top left pixel coordinate of the bounding box, and w and h are the width and the height of the bounding box respectively. This is also the output of the convolutional neural network-based detection model used in SeaClear.

4-2 Velocity Estimation

Since the algorithms were designed keeping in mind pedestrian and vehicle movement, it is assumed to follow constant velocity. But this is not the case for objects underwater. The Remotely Operated Vehicle (ROV) does not move with a constant velocity and the water disturbances make this motion more unpredictable. For more accurate tracking of the object, improving the motion model is the focus.

The position and velocity estimates obtained using the Kalman filter are inaccurate because the object's motion does not follow a constant velocity model. During the update step of the Kalman Filter, the position estimates are corrected using measurements of the actual object position from the bounding box. However, since no velocity data is available, this correction only applies to the position estimates. As a result, the velocity estimates remain inaccurate, which can lead to errors in the overall estimates.

Two different velocity estimation methods are proposed and the result from these will be assumed to be the actual velocity. These estimates will be used as "sensor data" in the Kalman filter for the update step.

4-2-1 Analytical Formula

We start with considering the simplest definition of velocity, - the rate of change of position. No matter the motion of the object, linear, nonlinear, circular, etc, the velocity of a point mass can be described as

$$\dot{x}_k = \frac{x_k - x_{k-1}}{dt}, \tag{4-3}$$

where dt is the time difference. This can be extended to the single pixel describing the top left corner of the bounding box.

The aim is to use this velocity value as a sensor measurement in the update step of the Kalman filter. The update step does not occur at every frame, but only when a successful data association is achieved. The velocity estimates are hence calculated as

$$\dot{x}_{est} = \frac{x_{curr} - x_{prev}}{\texttt{time since update}}, \dot{y}_{est} = \frac{y_{curr} - y_{prev}}{\texttt{time since update}}, \tag{4-4}$$

with the `time since update` variable being the number of frames between two update steps of the same object identity and x_{prev} and y_{prev} being the x,y pixel locations of the bounding box at the last update step. The `time since update` variable is incremented by 1 at every predicted step of the Kalman filter. The pseudo-code for this method is given in Algorithm 1.

Algorithm 1 Formula-based velocity estimation with the Kalman filter update step.

bbox ← bounding box
kf ← Kalman Filter instance
x_{prev} ← initial x position
y_{prev} ← initial y position
time since update ← 0
while tracking **do**
  x ← bbox[0]
  y ← bbox[1]
  if bbox matched with object **then**
    $x_{\text{dot}} \leftarrow (x - x_{\text{prev}})$/time since update
    $y_{\text{dot}} \leftarrow (y - y_{\text{prev}})$/time since update
    kf.update(bbox, $x_{\text{dot}}, y_{\text{dot}}$)
    $x_{\text{prev}} \leftarrow x$
    $y_{\text{prev}} \leftarrow y$
    time since update ← 0
  else
    kf.predict()
    time since update ← time since update + 1
  end if
end while

4-2-2 Optical Flow

Optical flow [18] is an algorithm that estimates the motion of pixels and key points between consecutive frames by analysing the intensity changes. Optical flow is often represented by a vector field, where each vector component represents the horizontal and vertical motion (direction and velocity) of the pixel [25]. In optical flow, the brightness consistency assumption is used. This assumption states that the intensity of pixels does not change across consecutive frames. The intensity of a pixel in a frame at time step t at location (x, y) is equated to the intensity of a pixel at the next frame at a slightly different location and represented as

$$I(x, y, k) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{4-5}$$

From the Taylor-series expansion of Equation 4-5, assuming the movement to be small we get

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \text{H.O.T.} \tag{4-6}$$

Ignoring the higher-order terms and equating Equation 4-5 to Equation 4-6, an initial optical flow equation is derived as

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{4-7}$$

Dividing Equation 4-7 by Δt gives the optical flow equation

$$I_x \cdot v_x + I_y.v_y = -I_t \tag{4-8}$$

Equation 4-8 is also known as the gradient constraint with two unknowns (v_x, v_y), the velocities in the x and y direction, respectively. Under the assumption of constant brightness, other constraints are brought into play to solve Equation 4-8 to obtain the pixel velocities. Currently, two widely used algorithms are the Lucas-Kanade method and Horn-Schunck method which can be found in detail in [25].

We want to see if optical flow can be used to estimate the general velocity of motion and used with the Kalman filter. We will be discussing and analysing two popular methods to implement optical flow.

Sparse Optical Flow The first and most commonly used method is to start with finding good features to track and then calculate the optical flow of these features. An advantage of this method is that the features would be ideally persistent in the video or image sequence. When using this method, we use the built-in function in `OpenCV`, `goodFeaturesToTrack`. The optical flow of these features is calculated by solving for the displacement by minimising the difference in intensity values between corresponding patches in successive frames. This method is also known as the sparse optical flow [3].

(a) Frame 189

(b) Frame 199

Figure 4-1: Image Frames from a video sequence of the SeaClear dataset. The above are 10 frames apart.

This method is applied to the SeaClear dataset to evaluate its performance. In Figure 4-1, two images from a video sequence 10 frames apart are illustrated. The optical flow of the identified good features to track is shown in Figure 4-2. Notably, only two pixels are selected as good features to track, represented by the two blue lines. Aside from the text on the frame, these selected features do not correspond to any significant object of interest, such as the tire depicted in the frame.

(a) Frame 189

(b) Frame 199

Figure 4-2: Sparse Optical flow output of the image frames. The motion trajectories are shown in green line from the pixels of interest.

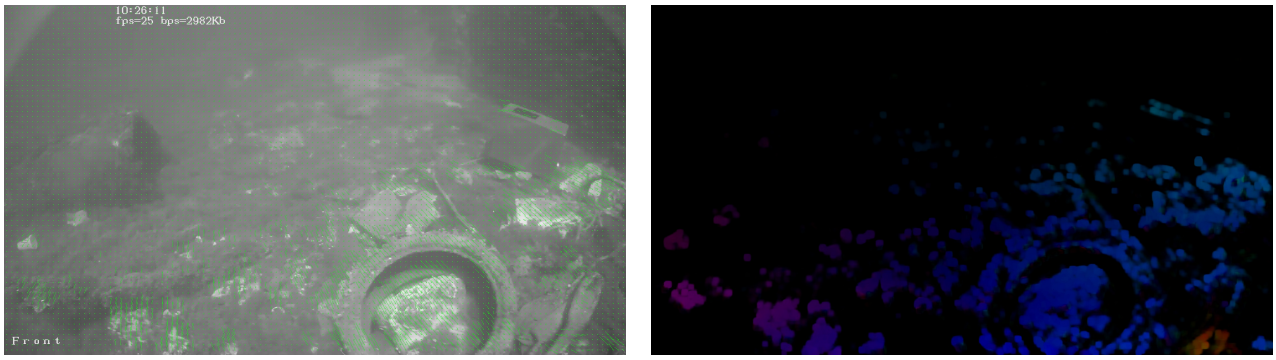
Dense Optical Flow Unlike sparse optical flow, which tracks a limited number of distinct feature points, dense optical flow aims to compute a motion vector for each pixel, providing a comprehensive view of how objects in the scene are moving. The `cv2.calcOpticalFlowFarneback` function in OpenCV implements the Farnebäck [14] dense optical flow algorithm, which is a robust and efficient method for computing dense optical flow. It employs polynomial expansion to approximate local pixel neighborhoods and utilizes a multi-scale approach with image pyramids to handle varying levels of detail and large displacements.

Implementing a dense optical flow on the same frame has no noticeable results yet again as seen in Figure 4-3. The actual image frame is converted to its grey variant and small green dots represent the motion vectors of the pixels. At the bottom right in Figure 4-3a, below the tire the optical flow gives some velocity arrows. We also check other frames of the same video with some more motion. In Figure 4-4 we see the dense optical flow gives a good result of the movement of pixels. A reason for this can be a faster motion and clearer difference in pixels. For example in frame 318, there are more "white" pixels from the plastics trash as compared to frame 189, in other words, this frame is closer to the trash, making the boundaries more clear, and giving more difference between adjoining pixels. Figure 4-4a shows the direction and intensity of the motion using green arrows and Figure 4-4b shows only the optical flow vectors, with the cooler colours being small in intensity and warmer colours more intense.

(a) Frame 189 with dense optical flow indicated by green dots and small green motion vectors.

(b) Frame 189 dense optical flow mask. No colours indicate a lack of motion.

Figure 4-3: Dense Optical flow and mask of image frame 189 of the same video sequence.



(a) Frame 318 with dense optical flow. The green vectors illustrate the motion of the pixels.

(b) Frame 318 dense mask. The colours indicate the direction of the motion.

Figure 4-4: Dense Optical flow and mask of frame 318 of the same video sequence.

With this, we can conclude that even though we cannot get motion information on specific features and pixels in a video sequence, more general optical flow information can be received using the dense optical flow method. This also confirms that using feature-based object-tracking algorithms would not work on underwater data.

The aim now is to use this general velocity information as the sensor data in the Kalman filter update step, just as we did in the previous subsection instead of using formula-based framework.

Algorithm 2 Using optical flow for velocity estimation with the Kalman filter update step.

bbox $\leftarrow$ bounding box
$kf \leftarrow$ Kalman Filter instance
$\text{img}_p \leftarrow$ previous gray image frame
img $\leftarrow$ current gray image frame
$s \leftarrow$ space between dense points
$fps \leftarrow 25$ {no. of frames per second}
if bbox matched with object **then**
  flow $\leftarrow$ cv2.calcOpticalFlowFarneback(img_p, img)
  $h, w \leftarrow$ img.shape[: 2]
  $y, x \leftarrow$ mesh grid$[s/2 : h : s, s/2 : w : s]$
  $fx, fy \leftarrow$ Transpose(flow$[y, x]$)
  $x_{\text{dot}} \leftarrow fx.\text{mean} * fps$ {Estimate velocity in x direction}
  $y_{\text{dot}} \leftarrow fy.\text{mean} * fps$ {Estimate velocity in y direction}
  kf.update(bbox, $x_{\text{dot}}, y_{\text{dot}}$)
  time since update $\leftarrow 0$
else
  kf.predict
  time since update $\leftarrow$ time since update $+ 1$
end if

4-3 Interacting Multiple Models

The Interacting Multiple Model (IMM) filter is a probabilistic algorithm used for state estimation to handle situations where the dynamics of a system can change over time. It is beneficial when multiple possible models can describe the evolution of a system and the true model is not known beforehand or changes dynamically [16]. The IMM filter maintains multiple dynamic models, each representing a different hypothesis about the system's behavior. These models could correspond to different modes of operation, such as constant velocity, turn, or other complex maneuvers. Since we are unaware of the exact motion model of the ROV using the IMM filter seems suitable.

Each model in an IMM filer has an associated probability that reflects how likely it is that the system is currently following that particular model with μ_k^j being the probability of the j^{th} model at time step k. These probabilities are updated over time based on the sensor measurements. This model probability is initialised by μ_0^j for each model j. There is also a mixing probability, $\mu_{k-1}^{i|j}$ for when the system was in mode i at time $k-1$ given that it is in mode J at time step k defined as

$$\mu_{k-1}^{i|j} = \frac{\mu_{k-1}^i \pi_{ij}}{\sum_{l=1}^{M} \mu_{k-1}^l \pi_{lj}}, \tag{4-9}$$

where π_{ij} is the transition probability from model i to model j and $\sum_{l=1}^{M} \mu_{k-1}^l \pi_{lj}$ is the normalising factor. The mixed state estimate and covariance are calculated as

$$\hat{x}_{k-1|k-1}^{0j} = \sum_{i=1}^{M} \mu_{k-1}^{i|j} \hat{x}_{k-1|k-1}^{i} \tag{4-10}$$

and

$$\hat{P}_{k-1|k-1}^{0j} = \sum_{i=1}^{M} \mu_{k-1}^{i|j} \left[\hat{P}_{k-1|k-1}^{i} + \left(\hat{x}_{k-1|k-1}^{i} - \hat{x}_{k-1|k-1}^{0j} \right) \left(\hat{x}_{k-1|k-1}^{i} - \hat{x}_{k-1|k-1}^{0j} \right)^T \right] \tag{4-11}$$

respectively, with $\hat{x}^i$ being the estimated state vector of model i and P^i the covariance. Predictions are made independently using each model, j. The initial mixed state estimate, $\hat{x}_{k-1|k-1}^{0j}$, is passed forward in time, based on the model dynamics, outputting the model state prediction and covariance update as

$$\hat{x}_{k|k-1}^{i} = A(i) \hat{x}_{k-1|k-1}^{0i} \tag{4-12}$$

and

$$P_{k|k-1}^{i} = A(i) P_{k-1|k-1}^{0i} A^{\mathrm{T}}(i) + Q \tag{4-13}$$

respectively. Sensor measurements are then used to update the state estimates for each model. The update process includes calculating the likelihood of the measurements given other models and this is defined as

$$\Lambda_k^j = \frac{1}{\sqrt{(2\pi)^m \left| S_k^j \right|}} \exp(-\frac{1}{2} (Z_k^j)^T \cdot (S_k^j)^{-1} \cdot Z_k^j), \tag{4-14}$$

with innovation Z_k^j and innovation covariance S_k^j of model j at time step k. The innovation and its covariance is defined as

$$Z_k^j = z_k - H \cdot x_k^j \tag{4-15}$$

and

$$S_k^j = H^j \cdot P_k^{0j} \cdot (H^j)^T + R \tag{4-16}$$

respectively. Here z_k is the measurement from the sensor at time step k, H^j is the measurement matrix of model j, and R is the measurement noise matrix. The mode probabilities are then updated using the likelihood as

$$\mu_k^j = \frac{\Lambda_k^j \sum_{i=1}^{M} \mu_{k-1}^i \pi_{ij}}{\sum_{l=1}^{M} \Lambda_k^i \sum_{i=1}^{M} \mu_{k-1}^i \pi_{il}}. \tag{4-17}$$

After the update step, the IMM filter combines information from all models to produce a final estimate of the system's state. This combination is weighted by the probability of each model, reflecting how likely it is that the system is following that particular model.

Models Up until now, we have been applying the widely used constant velocity model. However, we can confidently state that an underwater object would not adhere to this motion. Therefore, the aim is to implement the IMM filter to introduce another model, the constant acceleration model, to introduce variation and evaluate how well it describes the motion. In this approach, the model will have no constraints on estimating the object's velocity. This would increase the number of states in the system. The state vector for the constant acceleration model is,

$$[x, y, w, h, \dot{x}, \dot{y}, \dot{w}, \dot{h}, \ddot{x}, \ddot{y}]^T \tag{4-18}$$

The width and the height are not considered here for the constant acceleration and are still considered to change with constant velocity. We are more interested in making sure that the top left corner of the bounding box follows different motions. and that the width and the height of the bounding box do not contribute to the overall trajectory of the object.

This motion model for a constant acceleration model is

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ w_{k+1} \\ h_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{w}_{k+1} \\ \dot{h}_{k+1} \\ \ddot{x}_{k+1} \\ \ddot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & \frac{1}{2}dt^2 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & \frac{1}{2}dt^2 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ w_k \\ h_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{w}_k \\ \dot{h}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} \tag{4-19}$$

We will be using an IMM filter with two different motion models, constant velocity and constant acceleration, and evaluate the outcome.

A constraint of the IMM filter is that the number of states in all models should be the same, for consistent mixing of state estimates. To make sure this is the case, we have to make changes in the constant velocity model to incorporate the states $\dot{x}$ and $\dot{y}$. This is done by making all the entries of the last two columns and rows of the state transition matrix 0. This gives a modified constant velocity model given as

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ w_{k+1} \\ h_{k+1} \\ \dot{x}_{k+1} \\ \dot{y}_{k+1} \\ \dot{w}_{k+1} \\ \dot{h}_{k+1} \\ \ddot{x}_{k+1} \\ \ddot{y}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ w_k \\ h_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{w}_k \\ \dot{h}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} \tag{4-20}$$

4-4 Summary

In this chapter, several modifications to existing object-tracking algorithms are proposed to enhance performance in underwater environments. The first modification involves altering the state vector used in the state space representation of an object. Instead of relying solely on area and a fixed aspect ratio, the state vector is expanded to include the width and height of the bounding box. This allows the bounding box to dynamically adjust to any rectangular shape.

The next proposed modification introduces velocity estimation techniques to improve sensor measurements during the update step of the Kalman filter. Two methods are utilised for velocity estimation. The first method calculates velocity using an analytical formula based on the rate of change in position, derived from the previous pixel location, current position, and the time interval between measurements. The second method employs dense optical flow applied to the image sequence, capturing the overall motion across the frame. Optical flow is a technique in computer vision that calculates the motion of objects in a scene by analysing the changes in pixel intensity between consecutive video frames. It provides a vector field representing the direction and speed of motion for each pixel.

To support a wider range of motion models without restricting velocity, an IMM filter is implemented. An IMM filter is an advanced estimation algorithm that combines multiple models to predict and update the state of a dynamic system. It switches between different motion models, weighting them based on their likelihood, to improve accuracy in tracking targets with varying motion patterns. This research combines two distinct motion models: constant velocity and constant acceleration. This approach is necessary due to the uncertainty associated with the nature of underwater motion.

The next chapter will provide a detailed illustration and discussion of the algorithms' performance.

Chapter 5

Experimental Results

In this chapter, we will begin by implementing various State-of-the-art (SOTA) object tracking algorithms, originally designed for above-land conditions, on an underwater dataset. Following this, we will evaluate the performance of the newly proposed algorithms using the same dataset and conduct an extensive comparison. Prior to discussing the results, it is crucial to comprehend the mathematical metrics employed to evaluate the performance of object-tracking methods.

5-1 Performance Parameters

While it is desirable to condense the performance of object-tracking algorithms into a single metric for straightforward comparison, this approach is not feasible. Evaluating the performance of a tracking algorithm involves considering several aspects. First, it is important to determine how many objects the algorithm can track. However, this raises further considerations: Does this refer to the total number of objects tracked, regardless of whether they are tracked for just a second or for an extended period? Or does it pertain to how many objects are accurately tracked for the longest duration? Additionally, one must consider how frequently the algorithm switches the identity of an object or fails to track the object across frames. In a multi-class scenario, another factor is whether all classes are tracked equally, or if the algorithm excels in tracking one class while performing poorly with another. These varied aspects cannot be encapsulated by a single numerical value. One might want to provide more informative performance estimates by detailing the types of errors the algorithms make, which precludes a clear ranking. Hence various metrics need to be taken into consideration when looking at the overall performance of a tracking algorithm. Some of the various performance metrics used in object tracking are discussed below with a $\uparrow$ or $\downarrow$ symbol indicating if a higher or a lower score means better performance.

Intersection-over-union (IOU)($\uparrow$) As discussed in Equation 2-9, IOU between two bounding boxes is the ratio of the area of intersection of the boxes to the total area covered by the

boxes. A higher IOU implies better object detection as it implies that the area of the union of the two bounding boxes is closer to the area of the intersection. This metric is widely used in object detection and data association. It is usually also used to calculate the cost matrix in the Hungarian algorithm.

True Positive(TP)(↑) In the context of object tracking, true positive refers to a situation where the tracking system correctly identifies and associates a tracked object with its corresponding ground truth or actual object in the video frames.

False Positive(FP)(↓) A false positive occurs when the tracking system erroneously identifies and associates a region or object as a target object when in reality, there is no corresponding object or the object is different from what was identified.

False Negative(FN)(↓) A false negative represents a failure in the tracking system to identify an existing object. A good result is expected to have as few false positives and false negatives as possible.

Precision(↑) Precision assesses the accuracy of positive predictions generated by a model. From all the instances predicted as positive, it quantifies the proportion which are genuinely positive. Mathematically this is defined as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{5-1}$$

Recall(↑) Recall quantifies a model's effectiveness in capturing all relevant instances. It signifies the model's capability to accurately identify genuine positive instances among the entire set. This is defined as

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Nagatives}}. \tag{5-2}$$

Though it is desirable to have high precision and recall, there is generally a trade-off between the two. A tracker with high precision and low recall is cautious. It is confident about the positive instances it tracks but might miss a substantial number of actual positives. On the flip side, a tracker with high recall and low precision casts a broader net. It tracks most of the actual positive instances but is prone to false positives.

Identity switches(↓) This metric is defined by the number of times the identity of a target object changes to a different previously tracked object or a new identity. The number of identity switches from henceforth is denoted as IDSW.

MOTA(↑) Multi-object Tracking Accuracy (MOTA) is the most widely used figure to evaluate a tracker's performance [5]. The main reason for this is its expressiveness as it combines three sources of errors defined above, false negative, false positive, and identity switches. This is numerically calculated using

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{G}_t} \tag{5-3}$$

formula. The ground truth is defined as G. The variable t is the frame index.

Detection Accuracy(↑) Detection Accuracy (DetA) measures the alignment between the set of all predicted detections and the set of all ground-truth detections. A localisation threshold, α, is defined above which two detections are considered to intersect. Detection accuracy at a localisation threshold is defined as

$$\text{Det}\, A_\alpha = \frac{|\text{TP}|}{|\text{TP}| + |\text{FN}| + |\text{FP}|}. \tag{5-4}$$

Associaltion Accuracy(↑) Association measures how well a tracker links detections over time into the same identities, given the ground-truth tracks. The intersection between two tracks is measured as the number of True Positive matches between the two tracks, called True Positive Associations (TPA). Any remaining detections in the predicted track, which are either matched to other ground-truth tracks or none at all, are False Positive Associations (FPA), and any further remaining detections in the ground-truth track are referred to as False Negative Associations (FNA). Just as for detection accuracy, a localisation threshold is defined above which the tracks are considered to match. The association accuracy at a certain localisation threshold α is given by

$$\text{AssA}_\alpha = \frac{1}{|\text{TP}|} \sum_{c \in \{\text{TP}\}} \frac{|\,\text{TPA}(c)|}{|\,\text{TPA}(c)| + |\,\text{FNA}(c)| + |\,\text{FPA}(c)|}. \tag{5-5}$$

HOTA(↑) Higher Order Tracking Accuracy (HOTA) is the geometric mean of the detection, association and localisation accuracy. HOTA at a certain localisation threshold is given as

$$\text{HOTA}_\alpha = \sqrt{\text{DetA}_\alpha \cdot \text{AssA}_\alpha}. \tag{5-6}$$

It is the only monotonic metric as opposed to metrics such as MOTA, which overemphasizes detection [24]. Integrating this value over all localisation thresholds from 0 to 1 as

$$\text{HOTA} = \int_{0<\alpha\leq 1} \text{HOTA}_\alpha \tag{5-7}$$

gives the final value which can be used to look at tracking performance without looking at other metrics.

Mostly Tracked(↑) This is the number of ground-truth tracks that have the same label for at least 80% of their life span.

Mostly Lost(↓) This is the number of ground truth tracks having the same identity for at most 20% of their respective life spans.

Fragmentation(↓) Defined by the number of times a ground truth track is interrupted by missed detections.

Depending on the use case, a trade-off between accuracy and speed has to be chosen since the speed of most accurate trackers is considered too slow for real-time applications. For the SeaClear use case, a preference is given to the HOTA metric as that is the only single metric that reflects a good balance between detection accuracy, localisation accuracy, and association accuracy. With this, the number of identity switches and mostly tracked and mostly lost trajectories will be seen when comparing the performances of various trackers. For a deeper comparison of performance, the recall and precision metrics will be taken into consideration.

5-2 Above-water Object Tracking Algorithms

The SOTA object tracking algorithms discussed in Chapter 2 will be implemented using the novel dataset created. Based on the metrics discussed above, the performance of these trackers will be compared. Different parameters will be chosen and tuned to optimise the performance of each algorithm. By adjusting these parameters, the goal is to enhance HOTA, reduce identity switches, and improve the overall tracking consistency. This process of parameter tuning will involve a systematic exploration of various settings to identify the optimal configuration that yields the best results on the novel dataset created for underwater object tracking.

5-2-1 IOU Tracker

The IOU tracker is the simplest tracker of all. Only detections given from the SeaClear detection module are used to track the objects without any motion estimation techniques. The detections are compared in each frame with the previous frame using the IOU score. The algorithm waits for successful associations of T_{min} frames to start the tracking process, this is done to reduce false positives. It is also required that the track has at least one detection with a confidence score above σ_h, which ensures that it belongs to a true object of interest while also benefiting from low-scoring detections for the completeness of the track. There is also a matching threshold, σ_{IOU}. Only an IOU score above this threshold is considered to be a match. Detections below a certain confidence score σ_l are not considered.

These parameters can be tuned according to the use-case requirements and in order to improve the performance. Starting with initial parameters as $\sigma_h = 0.8, \sigma_{iou} = 0.5, \sigma_l = 0.35$ and tuning the T_{min} parameter. This means that only detections with confidence scores above 0.35 will be considered. For a successful match, the IOU score has to be larger than 0.5 and each object track should have one detection confidence score over 0.8. The tracking results with these parameters are shown below.

As seen from Table 5-1, with an increase in T_{min} the detection accuracy decreases but the association accuracy increases. Keeping the value of T_{min} above 0 would mean that the first

T_{min}	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	ML	Frag
10	57.804	71.719	46.589	99.966	101	58	18	43	25
20	57.67	70.5	47.174	99.965	87	49	18	45	23

Table 5-1: IOU tracker results with $\sigma_{iou} = 0.5$.

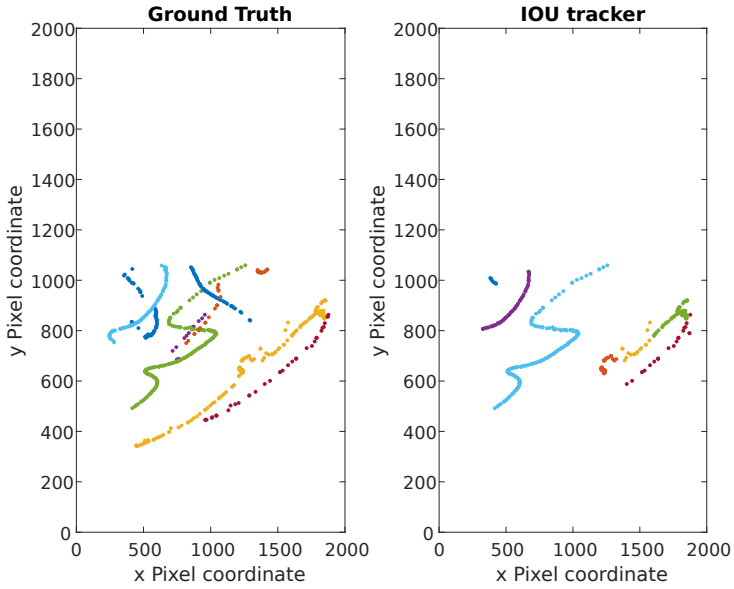


Figure 5-1: Ground truth and IOU tracker results for a video sequence with multiple object trajectories.

T_{min} occurrences of an object will always be missed since the track is not initialised as of yet, but keeping this parameter is important to filter out false positive detections. So it is obvious that as T_{min} increases the detection accuracy will always decrease. Additionally, it is also observed that the identity switches decrease along with an overall number of total identities, while the number of mostly lost trajectories increases. This is another consequence of increasing T_{min}, when more detections will be missed, there will be more lost trajectories. Next, the σ_{iou} parameter is decreased to 0.3, to make the matching criteria more lenient and see how the tracking performance changes.

T_{min}	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	ML	Frag
10	59.168	72.815	48.079	99.971	101	56	21	41	24
20	59.022	71.576	48.67	99.97	87	46	21	43	22

Table 5-2: IOU tracker results with $\sigma_{iou} = 0.3$.

On decreasing σ_{iou}, the value of HOTA increases along with mostly tracked (MT) trajectories even though the number of identities remains the same when compared to Table 5-1. The number of identity switches decreases with an increase in all three, detection, association, and localisation accuracy. Even though with a decrease, overall tracking performance is improved, the IOU matching parameter will not be decreased further as this would mean that even smaller IOU scores would lead to matches. Next, we tune the parameter σ_l for the

detection threshold. This parameter is reduced to 0, again making the overall tracker more lenient.

σ_l	T_{min}	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	ML	Fragments
0.35	10	59.168	72.815	48.079	99.971	101	56	21	41	24
	20	59.022	71.576	48.67	99.97	87	46	21	43	22
0.0	5	60.912	74.328	49.917	99.96	105	52	24	38	38
	10	60.887	73.907	50.161	99.96	94	46	23	41	36
	25	60.659	72.473	50.772	99.963	77	37	22	44	34

Table 5-3: IOU tracker results on tuning σ_l and T_{min}, keeping $\sigma_{iou} = 0.3$.

In general, the tracking performance improves with a decrease in σ_l, which was unexpected. Typically, detections with lower confidence scores are prone to being incorrect. The only observed decrease in performance is in localisation accuracy and fragmentations. This suggests that most detections with low scores are still valuable and do not significantly increase false positives.

However, the increase in fragmentations indicates that these low-scoring detections may not maintain a consistent trajectory. In this instance, we adjusted the T_{min} parameter to 25 instead of 20 to ensure the object remains consistently associated for at least one second before initiating tracking. Given a frame rate of 25 frames per second, a T_{min} value of 25 is more appropriate. Although the HOTA score was highest for $T_{min} = 5$, the number of identity switches was significantly lower for $T_{min} = 25$. Therefore, we selected the optimal parameters as $\sigma_l = 0.0$, $\sigma_h = 0.8$, $\sigma_{iou} = 0.3$, and $T_{min} = 25$. The tracking performance is also visually shown for some object trajectories in Figure 5-1 and Figure 5-2. Each of these figures represents one video and each unique object trajectory is represented with a different colour.

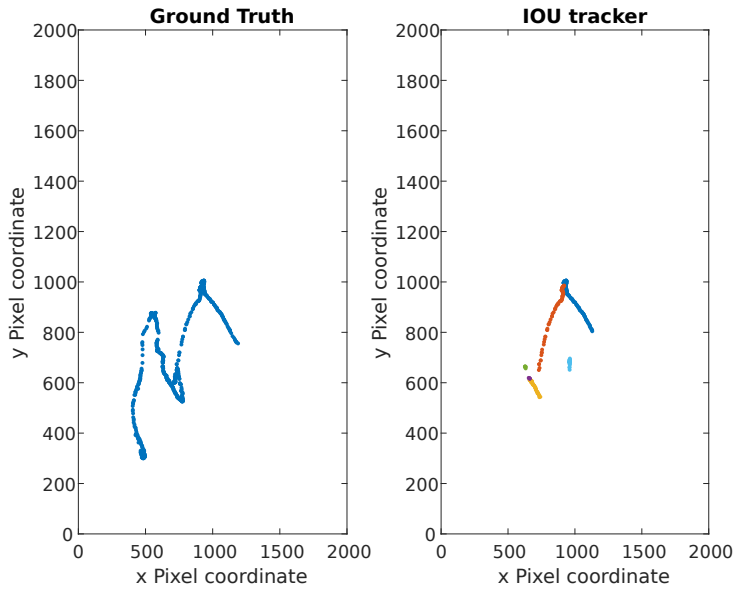


Figure 5-2: IOU tracker result for a single object trajectory.

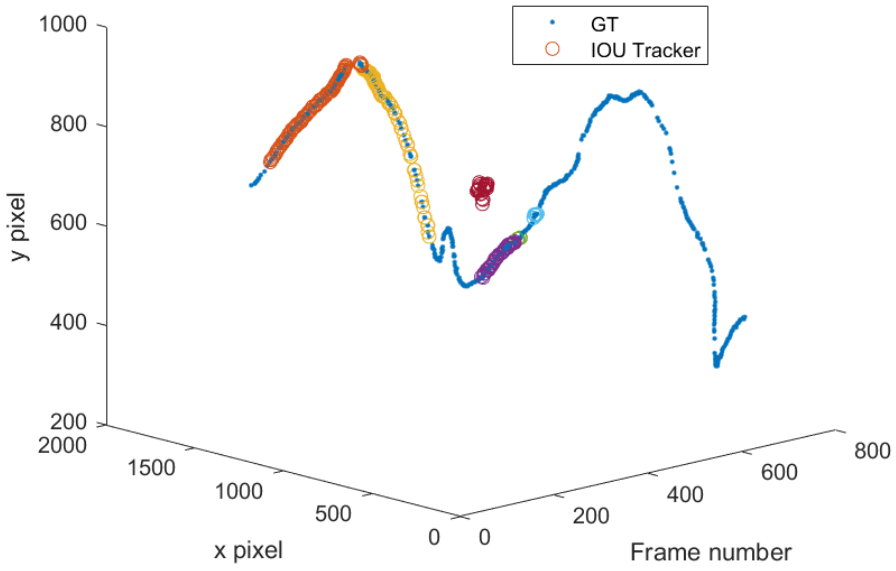


Figure 5-3: IOU tracker result for the single object trajectory shown in Figure 5-2 in a 3-D plot with the frame numbers on the x-axis, and the pixel coordinates on the y-z plane.

Since the IOU tracker is only dependent on the detection model and there is no motion estimation or prediction, whenever a detection is missed it is unable to continue to track the object. This is also visible in Figure 5-1, where the tracker is only able to track consistent, continuous object trajectories. For trajectories with missed detections in between, there are many identity switches. This can also be seen in the 3-D plot, Figure 5-3, with the frame numbers being on the x-axis and the image space on the y-z plane. The dot represents the ground truth, and the circle represents the IOU tracker. This 3-D graph is the same trajectory illustrated in Figure 5-2 but now also includes the frame numbers. With more gaps in the ground truth, the tracker is unable to follow the trajectory, the identity keeps changing, and towards the end of the trajectory when there are more gaps, because of the parameter T_{min} the tracker waits for a certain amount of successful association to start tracking and miss a big chunk of the trajectory and eventually is unable to track at all.

5-2-2 SORT

The Simple Online and Real-time Tracking (SORT) is the basis of a majority of the current SOTA object tracking algorithms. Here the motion estimation of the object is done using a Kalman Filter. The min hits parameter, just like the case for T_{min} for the IOU tracker helps to reduce the number of false positive tracks by requiring a new object to be detected consistently over a number of consecutive frames before it is considered a valid track. Another parameter here is the max-age, the number of frames for which an existing track is kept alive and predicted even without having any successful associations. In real-world scenarios, objects might temporarily disappear from view due to occlusions or missed detections by the object

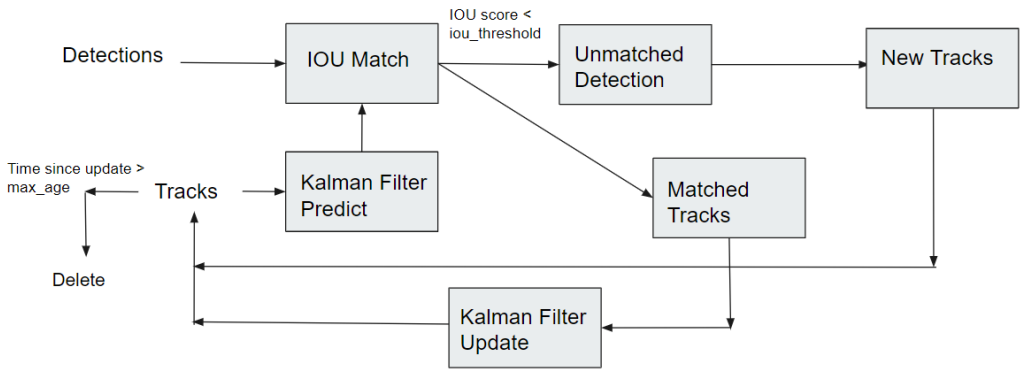


Figure 5-4: SORT algorithm illustrated in steps.

detection algorithm. The max-age parameter allows the tracker to maintain these objects for a few frames even when they are not detected, thereby providing robustness against short-term occlusions or detection failures. If a detection is not associated with a track in a given frame, the counter for that track is incremented. When the counter exceeds the max-age value, the track is considered lost and is subsequently deleted from the tracking system. Like the IOU tracker, there is an IOU threshold below which a match is not considered. An overview of the SORT algorithm is given in Figure 5-4.

The tuning of the parameters starts with IOU threshold = 0.3, min hits = 15. The same IOU threshold is set to the same as the one considered for the IOU tracker.

Max Age	HOTA	DetA	AssA	LocA	IDs
3	58.273	68.195	49.829	95.521	133
10	59.544	66.548	53.307	95.529	110
20	59.458	66.415	53.259	95.555	107
30	59.464	66.466	53.229	95.555	106

Table 5-4: SORT results with IOU threshold = 0.3, min hits = 15.

There is barely any noticeable change in performance when increasing the max-age. Hence the next set of metrics is looked at, the identity switches and mostly tracked and lost trajectories. When comparing identity switches, there are 59 switches with a max age of 3, while this number decreases to 36 with increasing the max age to 30. Although the number of mostly tracked (15) and mostly lost (33) remains the same in both cases, the reduction in identity switches is evident in the smaller total number of identities. This reduction in identity switches is attributed to the fact that since the trajectories are kept alive and predicted for a longer time, objects undergoing occlusion can be re-associated to the same identity again if the motion estimation is accurate.

Keeping the max-age constant at 30, the min hits parameter is tuned. As seen in Table 5-5, lower value of min-hits give the best performance. The first three rows correspond to the max-age of 30. As the min-hits increase, the HOTA score decreases, along with a significant drop in detection accuracy. This outcome was anticipated because a longer initialisation period for tracks results in more missed detections. Consequently, there is a substantial number of lost

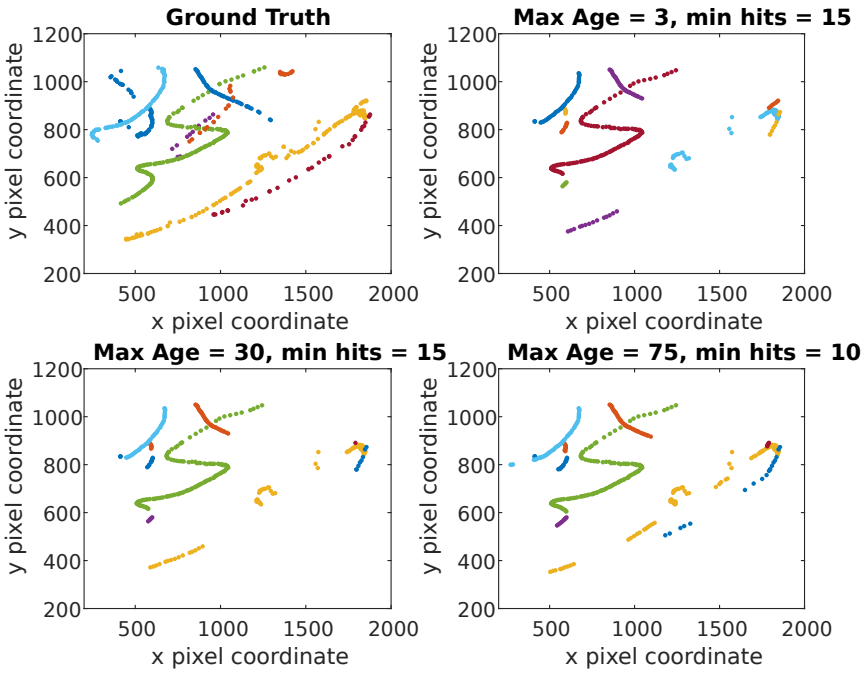


Figure 5-5: Comparing SORT tracker with varying parameters with the ground truth for multiple object trajectories, overlapping each other.

tracks (39) despite a reduction in fragments.

Min hits	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	PT	ML	Frag
10	62.127	71.127	54.298	95.42	122	44	18	35	29	70
15	59.464	66.466	53.229	95.555	106	36	15	34	33	56
25	55.978	60.147	52.123	95.739	93	34	11	32	39	50
min hits = 10 max age = 75	61.838	70.515	54.26	95.43	117	39	19	32	31	69

Table 5-5: SORT results with IOU threshold - 0.3, changing the min hits, max age = 30 if not mentioned otherwise.

The max-age is increased to 75 while keeping the minimum hits at 10. This adjustment ensures that the algorithm can continue tracking an object even if it is occluded or leaves the scene and re-enters after three seconds. The performance with a max-age of 75 is comparable to that with a max-age of 30, with a decrease in identity switches. This is a good tradeoff between the identity switches and the HOTA.

The tracking results are also depicted visually in Figure 5-5 and Figure 5-6. The scenario with a max-age of 75 demonstrates the ability to track objects for a longer duration while preserving the same identity. This is chosen to be the best performing point of the SORT algorithm. But there is still a large scope for improvement as seen from Figure 5-6. For trajectories with more gaps, the algorithm is unable to track successfully even though the object is kept alive. This issue can be attributed to an inaccurate motion model, which results in predictions of the object's location that are not close to the actual position.

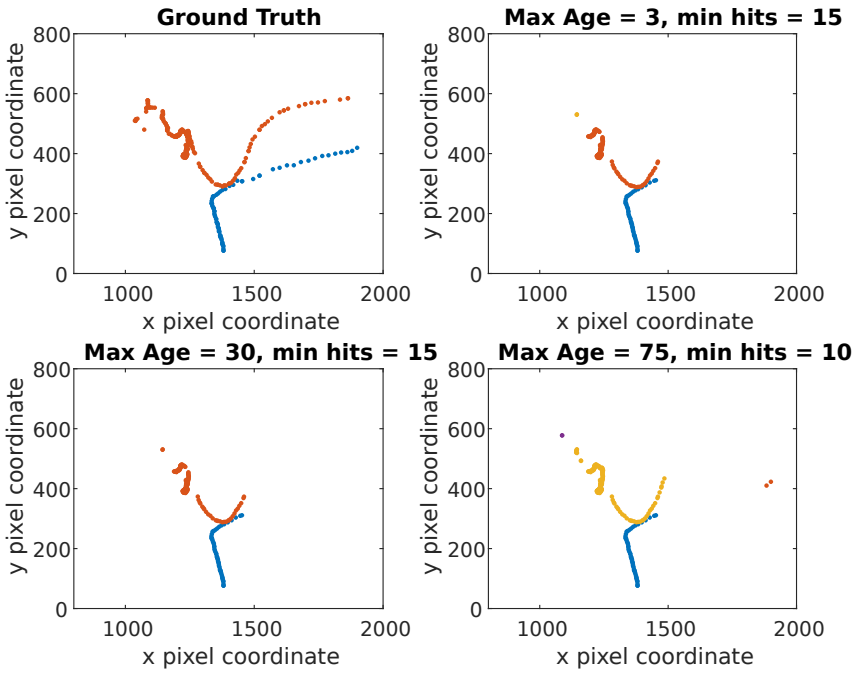


Figure 5-6: Comparing SORT tracker with varying parameters with the ground truth for multiple object trajectories, with more missed gaps.

5-2-3 ByteTrack

ByteTrack builds up on the SORT algorithm by using a different association method, Byte. In this case, low and high detection scores are utilised differently. Once objects are detected, the low-scoring and higher-scoring detections are segregated based on a threshold, the track threshold. After segregating, the Kalman filter is employed to predict the new locations in the current frame for each track. The initial association involves applying the Hungarian algorithm between the high-score detection boxes and all the tracks, just like the SORT algorithm. A subsequent association is carried out between the low-score detection boxes and the unmatched tracks from the first association. Following this second association, still unmatched low-score detection boxes are removed as they are now considered to be part of the background. Same as for SORT, the tracks without successful associations are kept alive for a max-age number of frames and are assumed to be lost afterward and hence deleted.

The max-age will be kept at a constant of 75 throughout the tuning process based on the results of SORT. First, the tracking confidence threshold is tuned keeping the match threshold constant at 0.9.

Track Threshold	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	PT	ML	Frag
0.5	65.94	75.101	59.734	89.831	157	59	28	25	29	107
0.6	64.811	72.414	58.067	94.015	106	46	25	28	29	83
0.7	62.111	65.024	59.369	94.348	72	25	15	27	40	71

Table 5-6: ByteTrack performance metrics with varying track threshold.

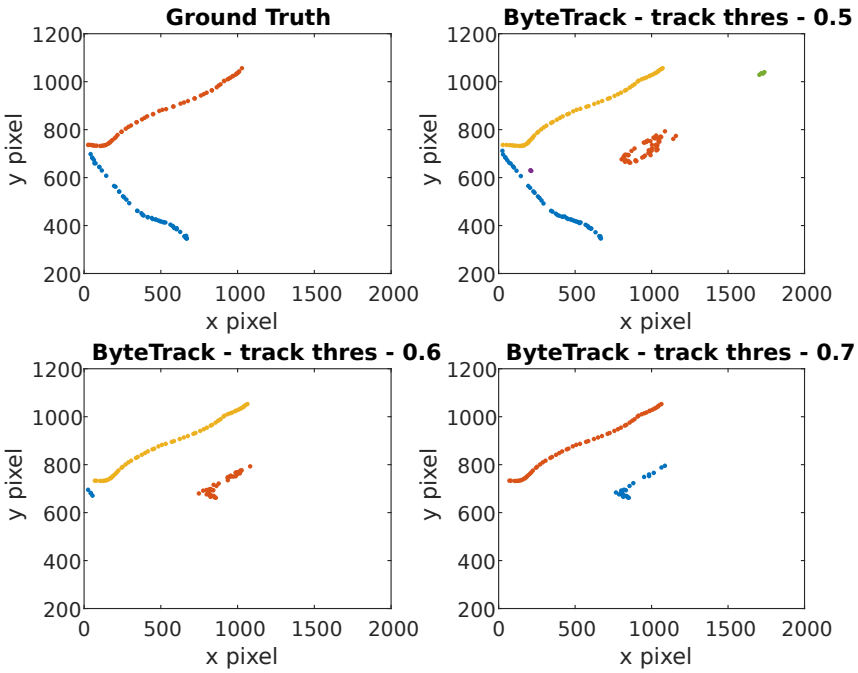


Figure 5-7: Object tracking using ByteTrack with varying track threshold. A tradeoff between accurate tracking and false positives is required.

Increasing the track threshold leads to decreased tracking performance. A threshold of 0.5 yields the best HOTA but significantly reduces localisation accuracy, and gives more identity switches and increased fragmentation. This can be attributed to more false positives, this is also illustrated in Figure 5-7, Figure 5-8 and Figure 5-9. A threshold value of 0.6 provides a balanced performance, offering a tradeoff between false positives and effective tracking.

In Figure 5-8, the lower threshold of 0.5 tracks the ground truth perfectly without any identity switches but also detects and tracks another object which in reality is not present, while the higher threshold of 0.7 misses most of the true detections. Finding a mid-ground with the threshold of 0.6, we see that the true object trajectory is tracked but with an identity switch. There is still a false positive detection but very smaller compared to the 0.5 case. In Figure 5-9 we again see the lower threshold having larger false positives while 0.6 does not.

Since the 0.6 gives a midway we choose this to be the best performing. Next keeping a constant track threshold = 0.6, the match threshold is varied to see how the performance changes.

Match Threshold	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	PT	ML	Frag
0.7	61.463	73.581	51.421	94.046	135	74	24	30	28	88
0.8	63.048	74.118	53.706	94.003	121	59	25	29	28	86
0.9	64.811	72.414	58.067	94.015	106	46	25	28	29	83

Table 5-7: ByteTrack results

Increasing the match threshold increases the tracking performance consistently, with a very

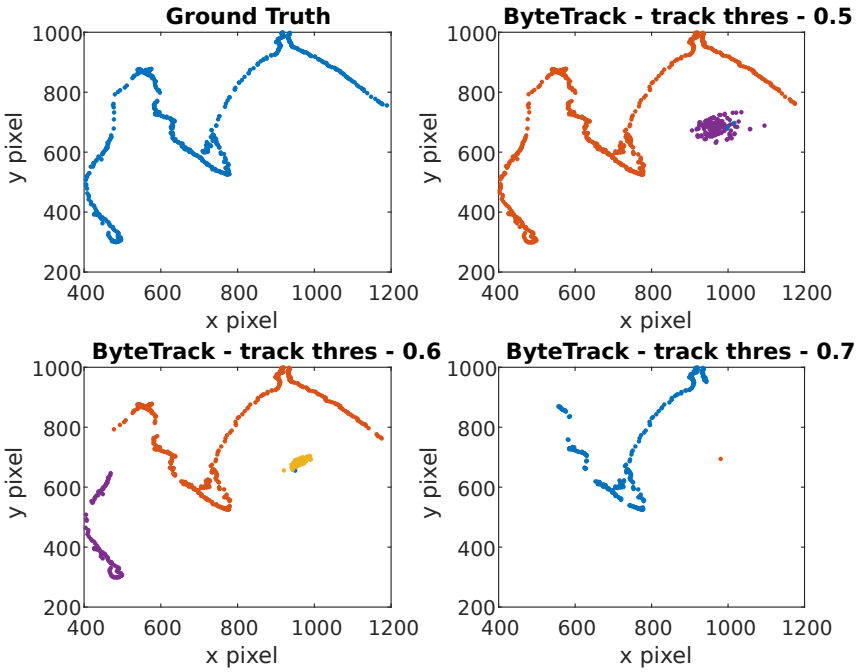


Figure 5-8: Object tracking using ByteTrack with varying track threshold. The results show high false positives.

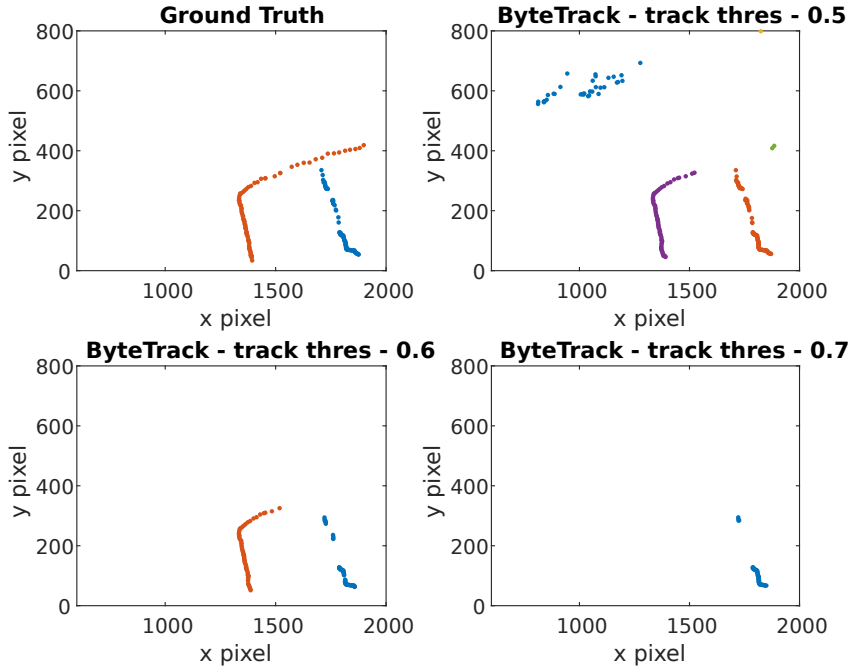


Figure 5-9: Object tracking using ByteTrack with varying track threshold. With lower false positives, there are also lower true positives.

big increase in association accuracy. The detection accuracy does decrease because a higher matching threshold would mean some lower detections would be missed as background. So the match threshold would be kept at 0.9 for detections with higher confidence scores, giving the best performance with a track threshold of 0.6.

5-2-4 BoT-SORT

The BoT-SORT algorithm enhances ByteTrack by incorporating camera motion compensation. This is performed by extracting and matching features between consecutive frames, estimating the camera's movement using a transformation matrix, and then warping the frames to neutralize this movement. This process was introduced to accurately track objects by isolating their motion from the camera's motion.

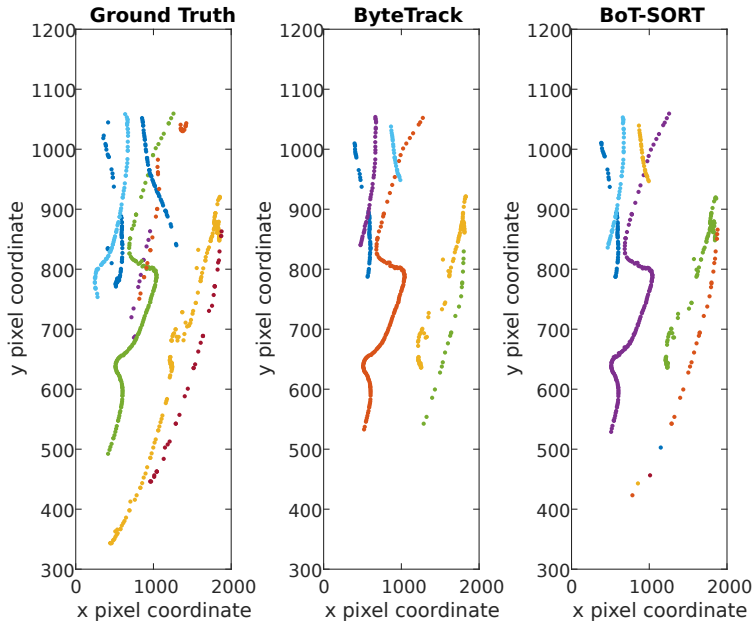


Figure 5-10: Tracking performance of BoT-SORT compared with ByteTrack. BoT-SORT has more detections but is unable to associate them with the same object.

This algorithm introduces no new parameters. The same parameters that yielded the best results for ByteTrack are used to ensure the technique is applied effectively.

Algorithm	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	IDs
ByteTrack	64.811	72.414	58.067	75.76	92.698	59.488	94.67	94.015	106
BoT- SORT	67.754	79.79	57.537	82.398	95.981	58.276	97.719	97.149	165

Table 5-8: Comparing tracking metrics of BoT-SORT with ByteTrack.

Just from looking at the HOTA value of the two trackers, BoT-SORT does indicate an improved performance. Notice that the association recall has decreased while the association

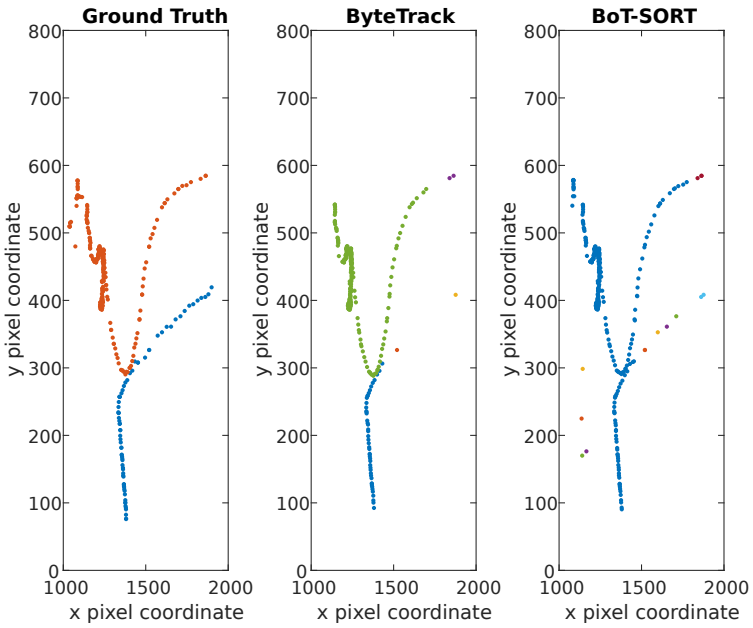


Figure 5-11: Tracking performance of BoT-SORT compared with ByteTrack. Data association is worse with BoT-SORT.

precision increases. This indicates that the algorithm is more conservative in making associations. It is more accurate with the associations it makes (higher precision), but it is also more likely to miss some true associations (lower recall). This trade-off implies that the algorithm is tuned to reduce false positives at the expense of missing some true positives. Another important observation is the significant increase in the overall number of identities. Looking more in-depth in Table 5-9, we see there is no difference in the number of trajectories tracked and lost between ByteTrack and BoT-SORT. However, there is an increase in the number of identity switches and fragmentation which leads to an overall increase in identities. This shows that even though more objects are detected, there is not a consistent association between them.

By utilising visual trajectories, we observe and analyse the results in more detail. In Figure 5-10 and Figure 5-11, objects in a trajectory are detected for longer by BoT-SORT than ByteTrack, but the identities keep switching rapidly. There is a higher number of detections that can also be observed from the detection accuracy value in Table 5-8, but association accuracy is reduced which is clear from the visual trajectories. In other cases like Figure 5-12 there is no noticeable change when compared to ByteTrack.

Algorithm	HOTA	IDs	IDSW	MT	PT	ML	Frag	FPS
ByteTrack	64.811	106	46	25	28	29	83	2112.8
BoT-SORT	67.754	165	80	26	27	29	92	23.2

Table 5-9: Comparing number of trajectories tracked between BoT-SORT and ByetTrack.

A critical consideration is the computational time of the algorithm. The inclusion of additional image processing and optical flow calculations for camera motion compensation results in a

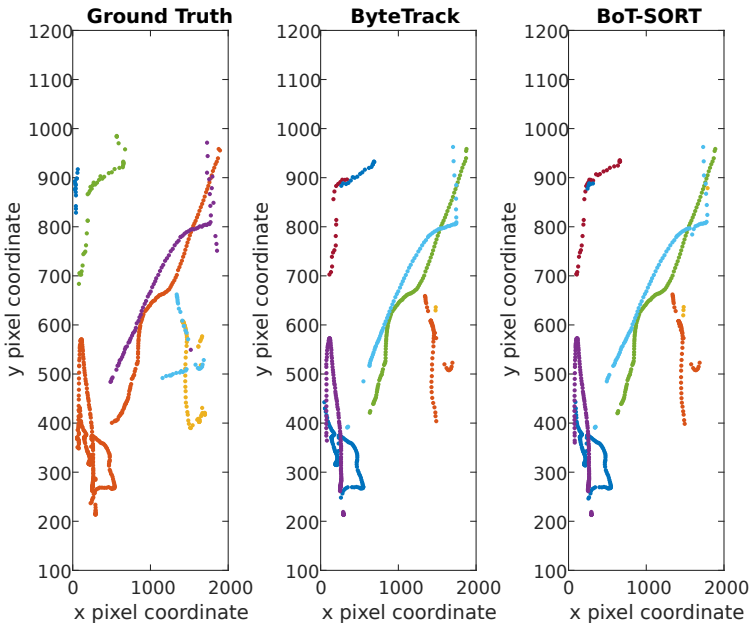


Figure 5-12: Tracking performance of BoT-SORT compared with ByteTrack. A similar result is seen, with no improvement by BoT-SORT.

computational speed that is approximately 100 times slower than ByteTrack. Consequently, BoT-SORT is not suitable for real-time or online tracking applications.

5-3 Proposed Algorithms for Underwater Object Tracking

After implementing the existing above-water object tracking algorithms on the underwater dataset, the goal is now to determine if the proposed algorithms perform better. The expectation is that there should be higher HOTA with more mostly tracked trajectories. If the proposed algorithms are suitable for underwater tracking, there should also be an increase in the association accuracy.

5-3-1 Velocity Estimation

Both the formula-based and the optical flow-based velocity estimation techniques are applied to the SORT algorithm, giving two new algorithms namely, SORT-V and SORT-OF respectively. From Table 5-10, there is a clear increase in performance when velocity estimation is implemented. The localisation accuracy almost reaches 99%. This can also be seen from Figure 5-13. The SORT algorithm when tracking the object in orange in the ground truth tracks an approximate linear version of the actual trajectory. When velocity estimation techniques are introduced the true nature of the trajectory is tracked, increasing the localisation accuracy.

The formula-based velocity estimation technique is also implemented with ByteTrack to give rise to a new algorithm ByteTrack-V. Comparing this new algorithm to ByteTrack, we see

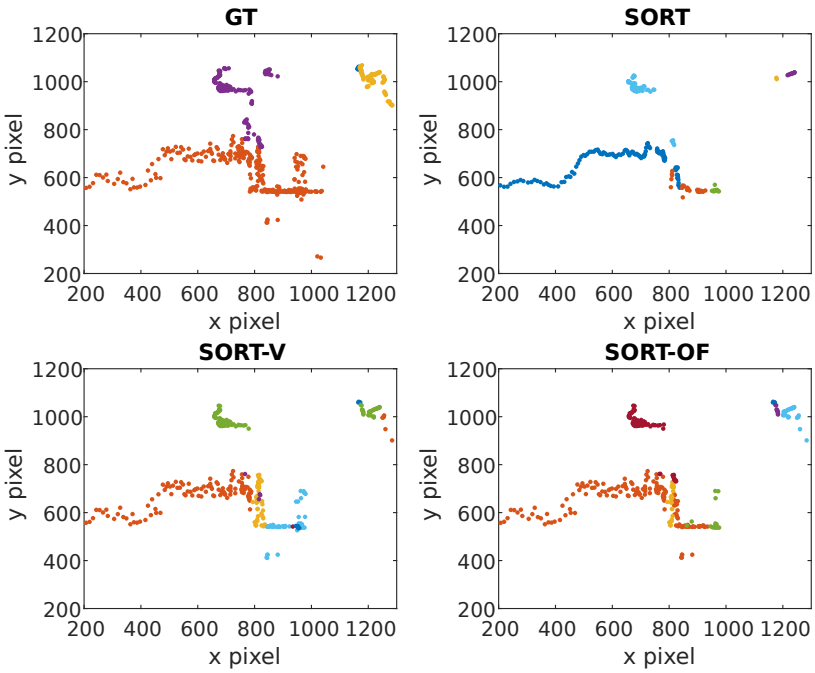


Figure 5-13: Visual ground truth trajectories with tracking results from SORT, SORT-V, and SORT-OF.

Algorithms	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	IDs
SORT	61.838	70.51	54.26	72.56	95.26	55.118	96.729	95.43	117
SORT-V	67.445	81.738	55.651	83.421	97.591	55.936	98.798	99.866	171
SORT-OF	68.619	81.525	57.757	83.197	97.594	58.119	98.791	99.845	156
ByteTrack	64.811	72.414	58.067	75.76	92.698	59.488	94.67	94.015	106
ByteTrack-V	67.106	80.971	55.615	83.106	96.917	56.056	98.072	98.768	120

Table 5-10: Comparing results of the proposed velocity estimation based trackers with SORT and ByteTrack.

an improvement in the tracking performance, and higher detection accuracy but a lower association accuracy. The association precision increasing indicates that the model's positive predictions are becoming more accurate. It means fewer false positives are being made. In other words, when the model does associate a positive result, it is more likely to be correct. But there is also a decrease in association recall meaning that the model is identifying fewer of the actual positive instances. It misses more true positives, which translates to an increase in false negatives. So ByteTrack-V is missing more true positives, but the ones that it associates are going to be correct.

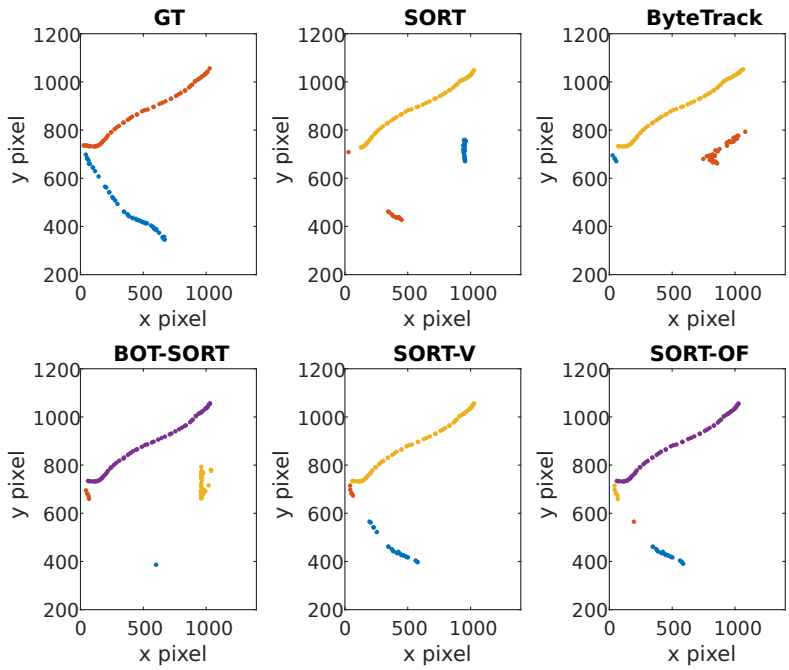


Figure 5-14: On comparing with SOTA object tracking algorithms, proposed velocity estimation techniques successfully resolve false positives.

When comparing SORT-V and ByteTrack-V, the HOTA scores and the number of mostly tracked trajectories are similar for both, but there is a significant difference in the number of mostly lost trajectories. This discrepancy is evident in Table 5-11, which shows the number of partially tracked (PT) trajectories. While ByteTrack-V exhibits fewer identity switches and less fragmentation compared to SORT-V, this is partly due to ByteTrack-V's inability to track 28 trajectories, whereas SORT-V fails to track only 14. The detection, association, and localisation accuracy of ByteTrack-V are all lower than SORT-V. From this point forward, we will focus solely on SORT-V and not ByteTrack-V. Despite the use of an improved association method in ByteTrack-V, it has not yielded better results.

Algo	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	PT	ML	Frag
SORT-V	67.445	81.738	55.651	99.866	171	69	30	38	14	110
ByteTrack-V	67.106	80.971	55.615	98.768	120	50	26	28	28	82

Table 5-11: Comparing number of tracked trajectories with SORT-V and ByteTrack-V.

Compared to ByteTrack and BoT-SORT, the performance of velocity estimation techniques shows significant improvement. Both formula-based and optical flow-based velocity estimation methods effectively eliminate false positives. This can be visually seen in the below figures. All algorithms are also compared numerically in Table 5-12. The HOTA score increases gradually, but it is also crucial to consider additional performance metrics from the table.

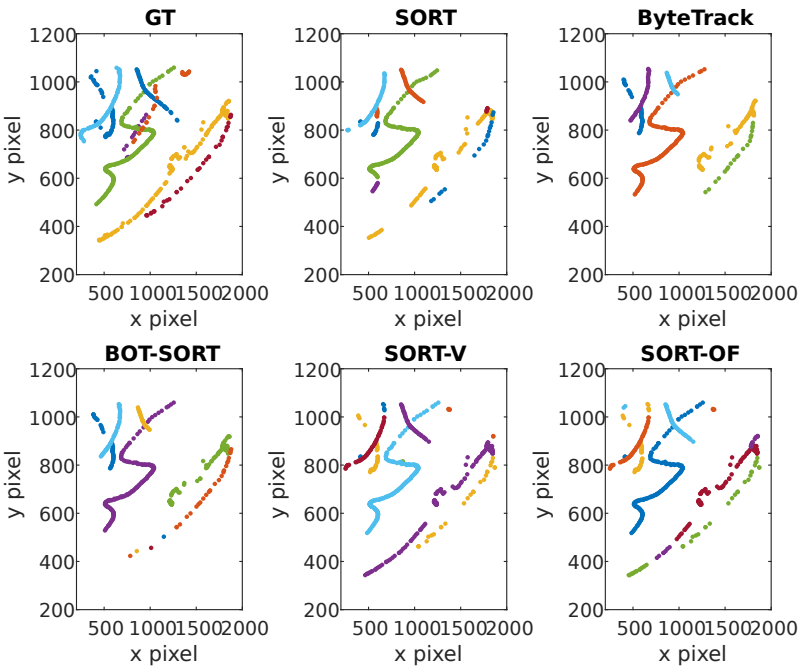


Figure 5-15: Velocity estimation techniques tracking for longer and correctly associating those trajectories.

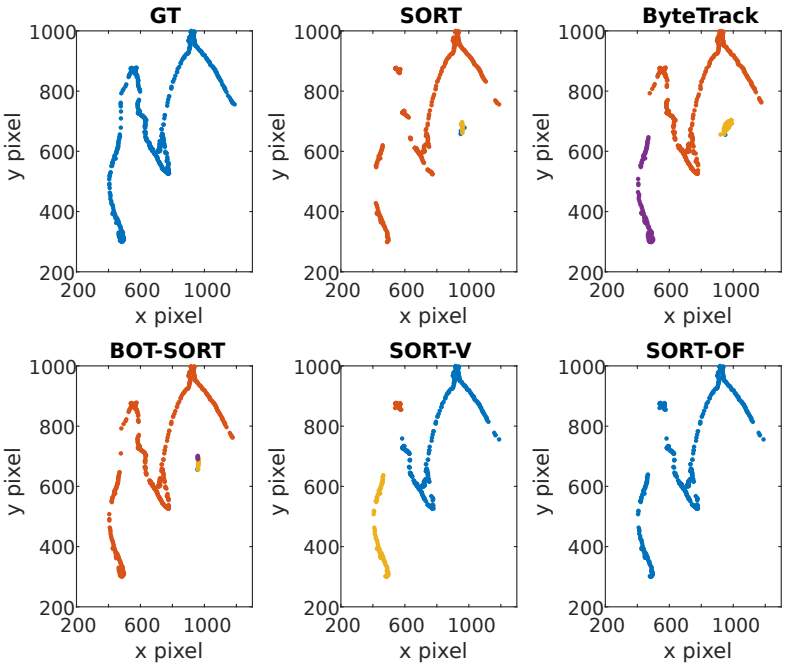


Figure 5-16: Velocity estimation techniques compared with SOTA object tracking algorithms.

Specifically, when examining the mostly tracked (MT) and mostly lost (ML) trajectories, the

velocity estimation algorithms reduce the number of mostly lost trajectories by nearly half. This represents a substantial performance improvement.

Algorithm	HOTA	DetA	AssA	LocA	IDs	IDSW	MT	PT	ML	Frag
SORT	61.838	70.51	54.26	95.43	117	39	19	32	31	69
ByteTrack	64.811	72.414	58.067	94.015	106	46	25	28	29	83
BoT-SORT	67.754	79.79	57.537	97.149	165	80	26	27	29	92
SORT-V	67.445	81.738	55.651	99.866	171	69	30	38	14	110
SORT-OF	68.619	81.525	57.757	99.845	156	65	29	38	15	111

Table 5-12: Comparing novel SORT-V and SORT-OF to other SOTA object tracking algorithms.

5-3-2 Interacting Multiple Model Filter

To account for different motion models, the Interacting Multiple Model (IMM) filter is implemented with the SORT algorithm, resulting in a novel tracker, SORT-IMM. The IMM filter is implemented with two different models, a constant velocity and a constant acceleration model. The parameters of the SORT algorithm remain the same, but the transition probabilities have to be defined. These are the probabilities of switching from a constant velocity model to a constant acceleration model and vice versa. The probability of not transitioning, ie. continuing in the same model is defined as α, and hence the probability of transitioning to another model is defined as $100 - \alpha$. The transition probability matrix is defined below.

$$p = \begin{bmatrix} \alpha & 100 - \alpha \\ 100 - \alpha & \alpha \end{bmatrix} \tag{5-8}$$

Different initial transition probabilities were assigned to the two models to see how the performance varied.

α	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	IDSW	MT	ML	Frag
50	68.71	81.64	57.82	83.33	97.58	58.22	98.69	99.85	62	30	13	112
75	68.93	81.06	58.62	82.73	97.56	59.01	98.75	99.86	52	30	13	114
95	68.50	81.47	57.60	83.18	97.54	57.99	98.67	99.87	60	29	16	102

Table 5-13: Object tracking performance of SORT-IMM with different transition probabilities.

From all the performance parameters, the best-performing tracker is when the value of $\alpha = 75$, meaning that starting from a constant velocity model, the probability of staying in a constant velocity model is 75% and transitioning to a constant acceleration model is 25%. The same is true for the starting in a constant acceleration model, staying there would be 75% probable, and transitioning to constant velocity is 25% probable. There is a noticeable increase in the association performance, which is desirable. There are fewer identity switches, with the highest number of mostly tracked trajectories. An example can be seen in Figure 5-17, where the object indicated with orange colour in the ground truth is tracked the furthest as compared to the other IMM-based trackers with different α value.

Another parameter of the IMM filter is the model probability. The model probabilities in the IMM filter represent the likelihood that the system is operating under each model at any given

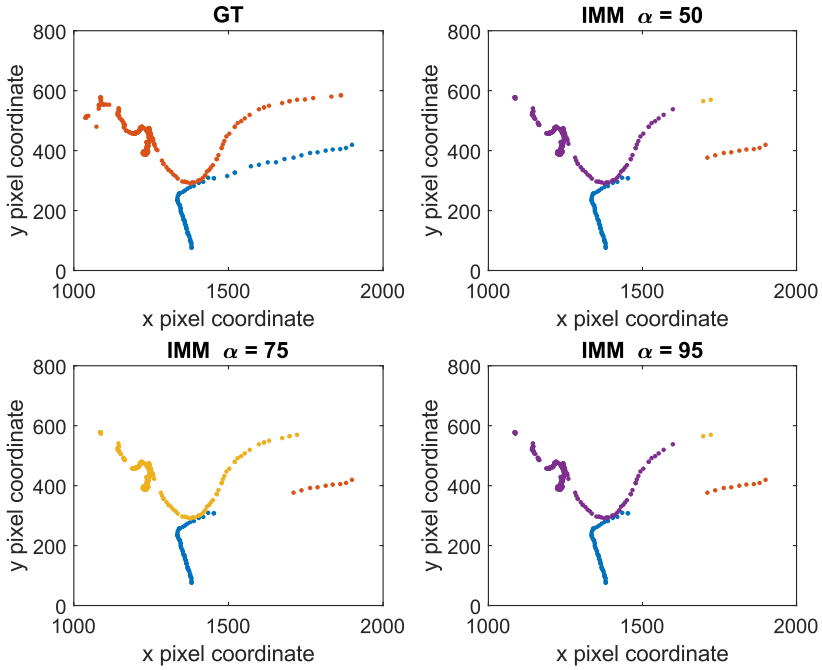


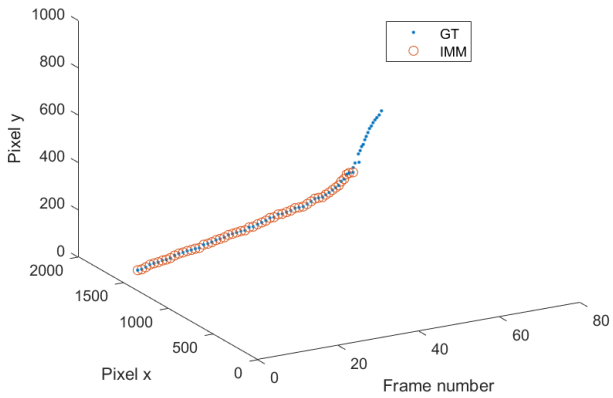
Figure 5-17: Ground truth vs SORT-IMM tracker with different transition probabilities.

time. These probabilities are updated at each time step based on the measurement likelihoods and the transition probabilities between models. The initial model probabilities are set to 50-50 because, in practice, the exact motion of the Remotely Operated Vehicle (ROV) is unknown, and there is no preference for one model over the other. For both trajectories in Figure 5-17, we see in detail how the model probabilities change with the likelihood throughout the tracking in Figure 5-18b and Figure 5-19b.

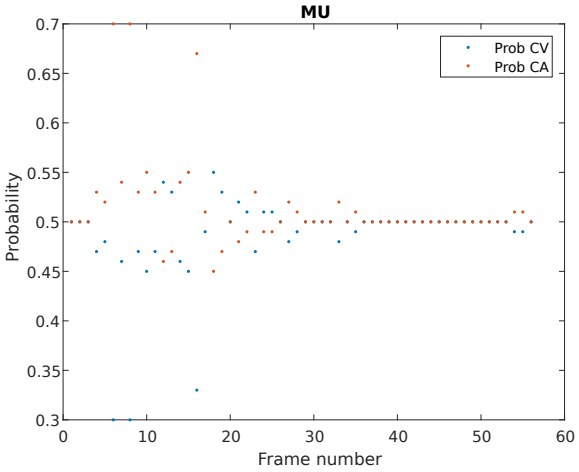
In Figure 5-18 the x-axis represents frame numbers, while the y and z axes represent the pixel coordinates x and y, respectively. The model probabilities are depicted in Figure 5-18b, starting at 0.5, fluctuating, and then returning to 0.5. A similar pattern is observed for the second object's trajectory in Figure 5-19b. This indicates that neither the constant velocity nor the constant acceleration model accurately describes the true motion throughout the life of the object, as the probabilities linger around the 0.5 value consistently.

This is not always the case. Another object trajectory with the ground truth and the SORT-IMM output is illustrated in Figure 5-20. The tracker closely follows the ground truth and the model probability of the constant acceleration model remains at 1 for multiple frames. This indicates that this model consistently describes the actual motion of the object, resulting in continuous tracking.

The same can be seen in other object trackers, Figure 5-21a shows the tracking result and Figure 5-21b shows the probability of the tracker, we see that the model probability for the constant acceleration model is consistently higher than that of the constant velocity model implying that this model describes the actual motion more accurately over the constant velocity model, though still not fully accurate. Another tracking result from the SORT-IMM is shown in Figure 5-22a with the corresponding model probabilities in Figure 5-22b. Here

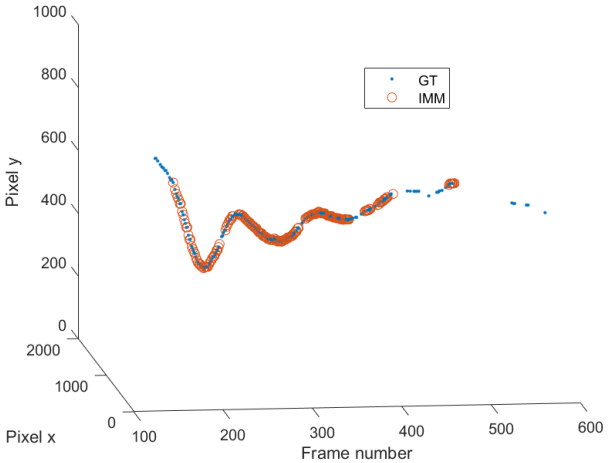


(a) SORT-IMM tracking result with the ground truth for a single trajectory.

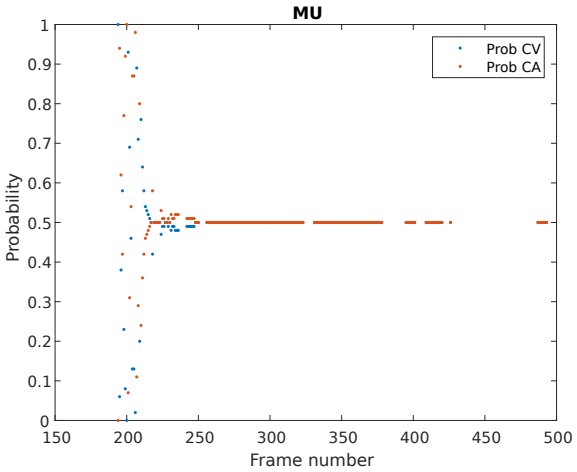


(b) Model probability values throughout the tracking duration.

Figure 5-18: SORT-IMM tracking results along with model probabilities for the blue trajectory in the ground truth in Figure 5-17.

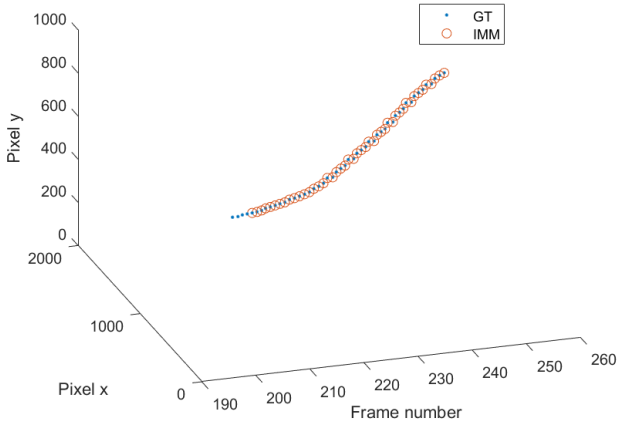


(a) SORT-IMM tracking output with the ground truth

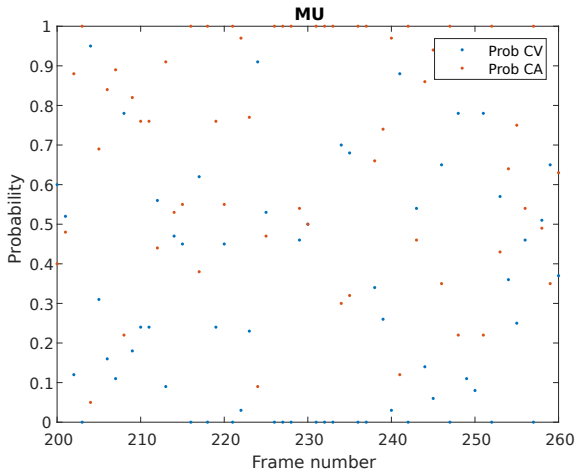


(b) Model probability values throughout the tracking duration.

Figure 5-19: SORT-IMM tracking result along with model probabilities for the orange trajectory in the ground truth in Figure 5-17.



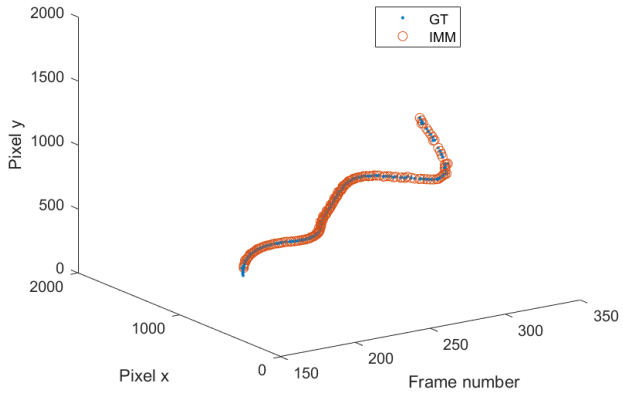
(a) SORT-IMM tracking output with the ground truth.



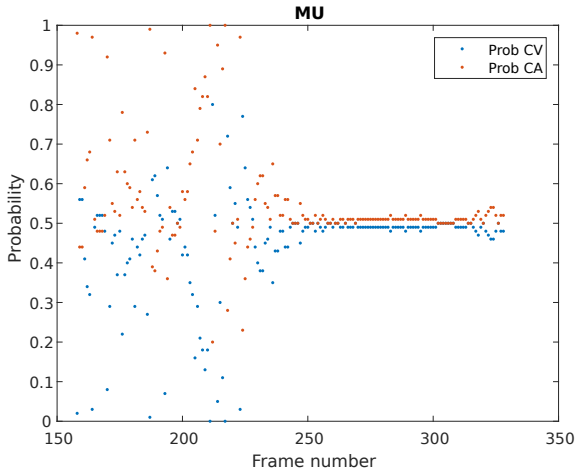
(b) Model probabilities throughout the tracking horizon.

Figure 5-20: SORT-IMM tracker consistently tracking the ground truth.

the tracking is not consistent and the model probabilities linger around the 0.5 value meaning that none of the motion models is preferred over the other.



(a) SORT-IMM tracking output with the ground truth.

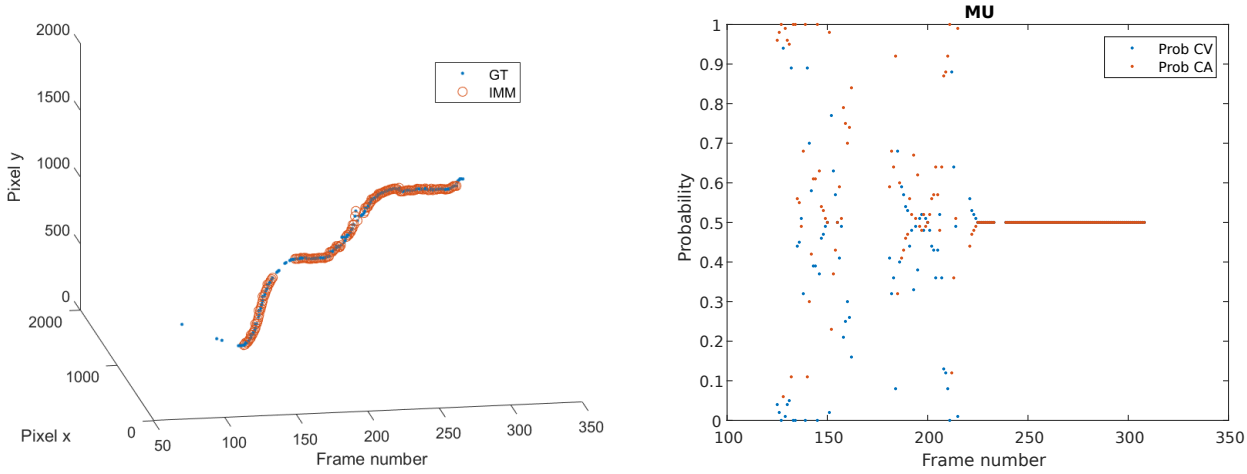


(b) Model probabilities throughout the tracking horizon.

Figure 5-21: SORT-IMM tracker consistently tracking the ground truth. In the end, due to more gaps, tracking performance is not exact.

In conclusion, the constant acceleration model emerges as a superior choice over the constant velocity model for certain trajectories, particularly in dynamic environments where motion patterns are complex and variable. The constant velocity model, while useful in some scenarios, often oversimplifies motion dynamics, leading to potential inaccuracies. In contrast, the constant acceleration model captures changes in velocity over time, providing a more accurate representation of an object's movement. When integrated into IMM filters, the constant

acceleration model further enhances the system's ability to adapt to uncertain or changing motion models, making it a valuable tool for underwater motion estimation.



(a) SORT-IMM tracking output with the ground truth.

(b) Model probabilities throughout the tracking horizon.

Figure 5-22: SORT-IMM tracker consistently tracking the ground truth. The constant acceleration model is preferred till frame number 220, after which both models are equally probable.

5-4 Comparisons and Conclusions

In this section, the performance of all the previously mentioned trackers will be compared using the performance parameters. First, the comparison of HOTA, detection, association, and localisation accuracy metrics reveals that the newly proposed SORT-IMM gives the highest HOTA value at 68.93. This high score results from its superior association and localisation accuracy. While its detection accuracy is not the highest, it is only 0.7 less than the top-performing tracker. From this, it can be concluded that object tracking algorithms rely heavily on the optimal performance of all sub-algorithms, detection, association, and localisation equally.

Tracker	HOTA	DetA	AssA	LocA	IDSW	MT	ML	Frag	FPS
IOU	60.66	72.47	50.77	99.6	37	22	44	34	155604
SORT	61.83	70.52	54.26	95.43	39	32	31	69	2694.9
ByteTrack	64.81	72.41	58.07	94.02	46	25	29	83	4489.9
BoT-SORT	67.75	79.79	57.54	97.15	80	26	29	92	23.2
SORT-V	67.44	**81.74**	55.65	99.8	69	30	14	110	3158.3
SORT-OF	68.62	81.52	57.76	99.84	65	29	15	111	1.8
ByteTrack-V	67.10	80.97	55.61	98.77	50	26	28	82	4517.6
SORT-IMM	**68.93**	81.06	**58.62**	**99.86**	52	**30**	**13**	114	966.4

Table 5-14: Comparing performance metrics of SOTA tracking algorithms with the newly proposed.

Considering other performance metrics, the SORT-IMM tracker does not exhibit the lowest number of identity switches, instead, the IOU and SORT trackers give the fewest identity switches. This is unexpected given their lower association accuracy. This reduction in identity switches can be attributed to both these trackers missing approximately 44 and 31 objects, respectively. As a result, instead of tracking the original 82 ground truth identities, they only managed to follow around 38 and 51 trajectories, respectively. Hence it is important to consider that the IOU tracker has 37 identity switches when tracking 38 objects, and the SORT tracker has 39 identity switches when tracking 51 objects. In this context, the SORT-IMM provides the best overall performance, with the highest number of mostly tracked objects at 30, the fewest mostly lost trajectories at 13, and 52 identity switches. Although the algorithm ByteTrack-V shows fewer identity switches, it misses nearly 28 trajectories. The algorithm SORT-OF also gives great tracking performance with the second highest HOTA only 0.3 behind the SORT-IMM tracker and the second highest mostly tracked trajectories, only 1 less than the SORT-IMM. This performance can be seen in Figure 5-25, with other algorithms giving identity switches and false positives, the SORT-OF gives the same result as the SORT-IMM tracker.

Another important metric is the computational speed of the algorithms. This is an important metric to keep in mind to be able to achieve online and real-time tracking. The metric is only for the tracking algorithm and does not include the computational time to detect objects. All algorithms were run using the Nvidia GeForce MX570 GPU. Both methods using optical flow are not suitable to be used in an online, real-time setting. BoT-SORT can be used with a more powerful computing system, but the SORT-OF is just at 1.8 frames per second. So even though it has good performance and is close to the SORT-IMM tracker, the computational speed is very slow. These two algorithms will hence not be taken into consideration.

When examining detailed performance metrics, competition emerges between the SORT-IMM and the SORT-V. The latter demonstrates slightly higher detection recall and precision, with margins of 0.3 and 0.03, respectively, compared to the IMM-based tracker. However, the IMM tracker excels in association recall, achieving 59.01 compared to 55.94 for SORT-V. Association recall measures how well predicted trajectories cover ground-truth trajectories; a low association recall indicates that a tracker splits an object into multiple predicted tracks. The higher association recall of the SORT-IMM tracker suggests it consistently associates objects without identity switches, resulting in fewer false negatives.

Association precision, which measures how well-predicted trajectories adhere to tracking the same ground-truth trajectories, relates to the number of false positives. Introducing velocity estimation or a different motion model significantly impacts detection accuracy, leading to increases in both recall and precision, thereby reducing false negatives and false positives.

The performance of all algorithms implemented and discussed is also visually compared. The SORT-IMM tracker is most capable of tracking non-linear motion as seen in Figure 5-23. It is the only algorithm consistently able to track the loop starting approximately at pixel coordinate $(2000, 1000)$. Similarly also seen in Figure 5-24, the SORT-IMM and the SORT-V trackers are the only ones able to track the ground truth trajectories shown in the ground truth graph in yellow and maroon. This shows that having the constant velocity model is not enough to describe the motion and additional information about the velocity or more defined motion model is required. It can also be concluded that the constant acceleration model can help describe the object trajectories better though not perfectly.

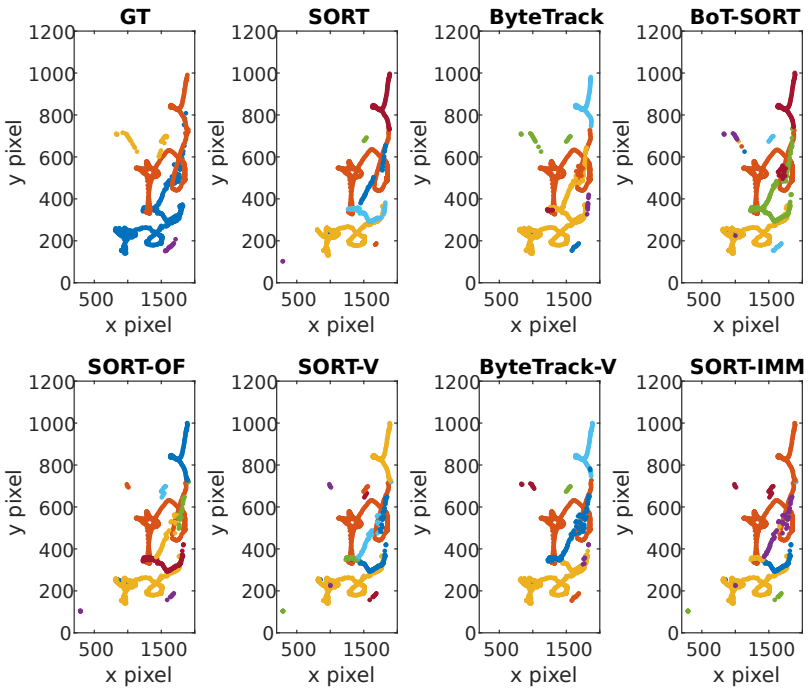


Figure 5-23: Illustrating the superior capabilities of the SORT-IMM tracker when tracking non-linear trajectories.

As presented in Table 5-16, all implemented algorithms have different advantages and disadvantages over each other. One can start adding various different motion models to an IMM-based tracker, but the more models added, the computational time increases. Though the ByteTrack algorithm has the lowest identity switches, it has a large number of false positives. This is also why it has the lowest association precision. The choice of algorithm depends on the user's needs and the specific requirements of the use case. If minimising identity switches is a priority and false positives are less of a concern, ByteTrack would be a suitable option. Meanwhile, if high detection and precision are required then SORT-V is the algorithm to go for. In conclusion, it is not possible to develop a single algorithm that will be universally effective for tracking across various environments.

In Table 5-15, the last two rows have the metrics of the best performing multiple object tracking algorithm part of the Multiple Object Tracking (MOT) Challenge. As discussed earlier, this dataset includes pedestrian and vehicle data captured by a fixed camera. The proposed algorithms in this thesis—SORT-V, ByteTrack-V, and SORT-IMM—demonstrate higher HOTA values. However, the association accuracy does not match that of the benchmark algorithms. Examining these benchmark values reveals that the overall performance of SOTA object tracking algorithms in the research community remains relatively low, indicating substantial room for research and improvement. Most algorithms rely on a linear constant velocity motion model, whereas real-world objects can move in various unpredictable ways. The three novel algorithms that account for real velocity and multiple models proposed in this thesis, result in significantly improved performance and include more accurate object tracking and higher localisation accuracy.

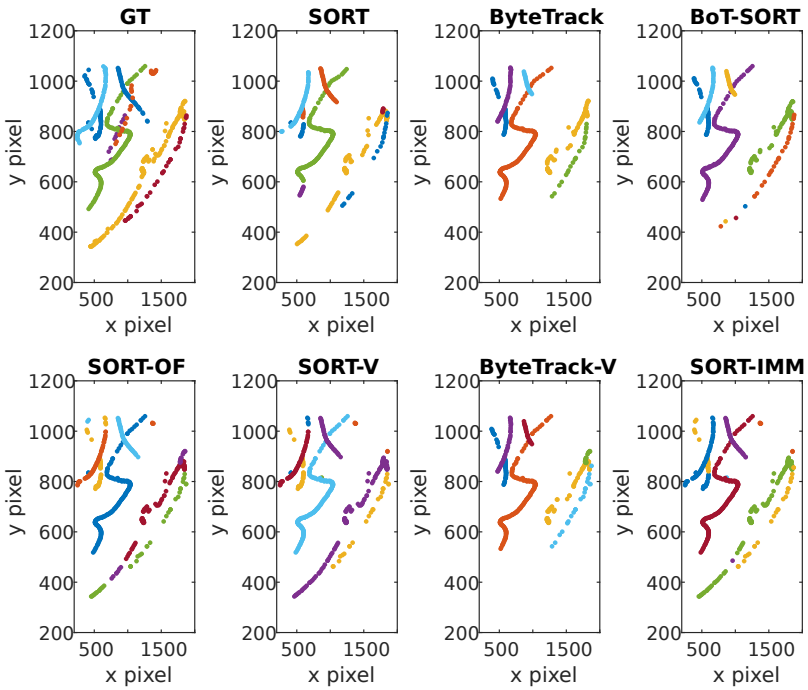


Figure 5-24: Comparing tracking performance of all implemented algorithms. SORT-IMM and SORT-V both give equally good results.

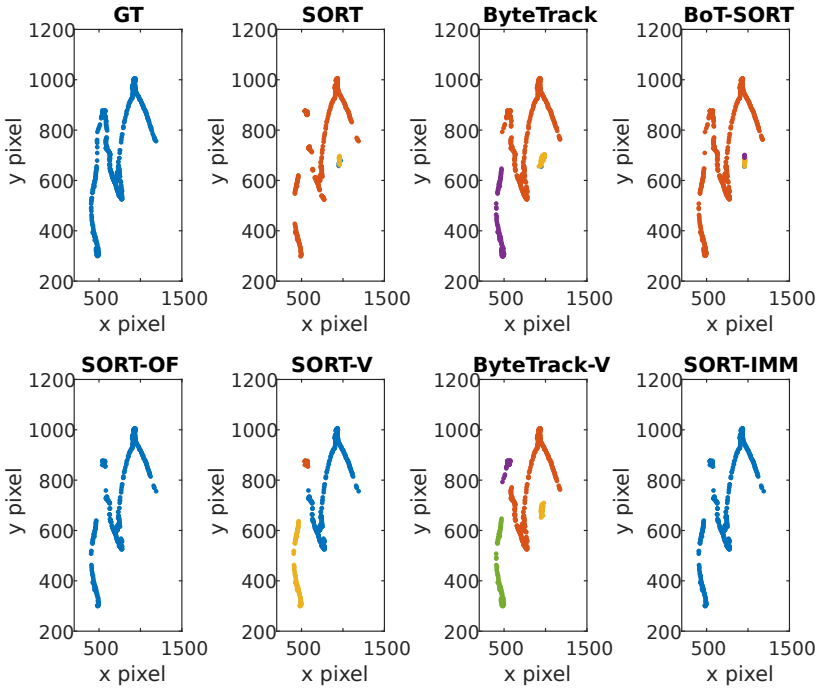


Figure 5-25: Comparing tracking performance of all implemented algorithms. SORT-IMM and SORT-V both give equally good results.

Tracker	HOTA	DetA	AssA	LocA	DetRe	DetPr	AssRe	AssPr
SORT	61.83	70.52	54.26	95.43	72.57	95.26	55.12	96.73
ByteTrack	64.81	72.41	58.07	94.02	75.76	92.70	58.29	94.67
SORT-V	67.44	**81.74**	55.65	99.8	**83.42**	**97.59**	55.94	**98.80**
ByteTrack-V	67.10	80.97	55.61	98.77	83.11	96.92	56.06	98.08
SORT-IMM	**68.93**	81.06	**58.62**	**99.86**	82.73	97.56	**59.01**	98.75
MOT-20 BEST	63.0	64.0	62.2	84.2	69.4	79.5	67.7	78.4
MOT-20 Online	62.7	63.5	62.2	84.1	69.5	78.4	67.3	78.9

Table 5-15: Comparing tracking metrics of implemented algorithms with the benchmark.

	SORT	ByteTrack	BoT-SORT	SORT-V	SORT-OF	IMM
Advantages	Lightweight	Lowest identity switches	Camera stabilisation	Highest detection accruacy	Measure actual pixel velocity	Incorporate multiple motion models
Disadvantage	Poor performance in complex scenarios	High false positives	Not suitable for realtime tracking	High identity switches	Not suitable for realtime tracking	More models increase computational time

Table 5-16: Comparing the SOTA object tracking algorithm based on motion estimation.

5-5 Summary

In this chapter, first different metrics used to compare the performance of various object-tracking algorithms are discussed. The HOTA metric is highlighted as it is unbiased towards any specific sub-algorithm used in object tracking and, therefore is used for performance comparison. Additionally, HOTA is the only monotonic metric, meaning that an increase in the metric always indicates improved tracking. HOTA combines detection, localisation, and association accuracy into a geometric mean. Metrics such as the number of mostly tracked and mostly lost trajectories are also considered, corresponding to trajectories tracked for at least 80% and at most 20% of their duration, respectively. Furthermore, identity switches are taken into account.

The IOU tracker which does not have any motion modeling and localisation gives the poorest performance. It is entirely dependent on the detection model and hence whenever a detection is missed the tracker cannot associate the object with the same identity anymore. The SORT algorithm gives good results when the trajectories are linear and not crowded. ByteTrack algorithm has the lowest identity switches but gives many false negatives. BoT-SORT has additional camera motion compensation capabilities on top of ByteTrack, but this is only suitable for stabilising the vibrations in a stationary camera and not for a fully moving camera. It has higher identity switches when compared with ByteTrack and lower association accuracy, which is an important metric we want to improve. Implementing velocity estimation techniques and changing the state-space representation, two new tracking algorithms, SORT-V and SORT-OF improve on all metrics. Both track almost 30 object trajectories and bring down the number of missed trajectories from 29 to 15. Even though more trajectories are being tracked, the number of identity switches is reduced.

Finally, the SORT-IMM tracker gives the best overall tracking result especially when tracking complex trajectories, giving the highest HOTA score at 68.93%, giving a rise of 11% when compared to the SORT algorithm and 13.5% when compared to the IOU tracker. The IMM tracker also has the highest association and localisation accuracy, and second highest detection accuracy, only 0.6 behind the highest. When compared to the current best performing above-water object realtime object tracking algorithm, the HOTA score is increased by almost 10%.

Chapter 6

Conclusions and Future Recommendations

With little to no public research on multiple object tracking in underwater scenarios, the aim of this thesis was to first develop a novel dataset to facilitate more research on this topic. The next objective was to implement existing State-of-the-art (SOTA) object tracking algorithms on this new dataset. Additionally, two newly proposed velocity estimation techniques were implemented, and a new multi-model tracker was introduced.

6-1 Conclusions

Making the new dataset for underwater videos was important to take into consideration the special problems faced in this uncertain environment. This dataset contained strong water currents, occluded objects, and various motions of the Remotely Operated Vehicle (ROV).

Velocity Estimation All SOTA object tracking algorithms followed a constant velocity model for motion estimation since they were used for above-water tracking, but such a motion is not followed underwater. The algorithms used Kalman Filters to predict the location of the object. Positions were used as measurements during the update step of the Kalman Filter. This update step made sure the variance matrices were updated using the correct location of the object but only for the position parameters. This inadequacy highlighted the need for more robust velocity estimation techniques tailored to the unique characteristics of underwater motion. To address this, three novel trackers are proposed: SORT-V, SORT-OF, and ByteTrack-V. These trackers incorporate advanced velocity estimation methods to more accurately capture the erratic and non-linear motion of underwater objects.

SORT-V and ByteTrack-V used formula-based velocity calculation at each update step of the Kalman Filter. This calculated velocity served as a sensor measurement to update both velocities and positions during the update step. In contrast, SORT-OF utilised optical flow to

calculate pixel velocities. Dense optical flow provided pixel velocities, and the average value was taken to determine the general flow of motion. This average velocity measure was then used in the Kalman Filter update step, leading to the proposal of a new algorithm, SORT-OF.

SORT-V and SORT-OF gave improved tracking results when compared to Simple Online and Real-time Tracking (SORT), ByteTrack, and BoT-SORT with the least number of lost trajectories and higher Higher Order Tracking Accuracy (HOTA) scores, but SORT-OF had a very low computational speed of 1.8 fps. ByteTrack-V on the other hand did not give any improved performance, with still missing around 28 trajectories and giving a low association accuracy of 55%.

Motion Models No matter how much we can improve the update step, an incorrect motion model will always lead to inaccurate tracking. So Interacting Multiple Model (IMM) filter was used instead of the normal Kalman Filter in the SORT algorithm. The IMM filter maintains multiple dynamic models, each representing a different hypothesis about the system's behavior. These models could correspond to different modes of operation. Since we are unaware of the exact motion model of the ROV using the IMM filter seems suitable. Two different motion models were implemented in the IMM filter the constant velocity model and the constant acceleration model. Initial model probabilities were set at 50% for each, indicating no preference between the models. The best performance was achieved with a transition probability of 25% between models and 75% within the same model. This was implemented with the SORT algorithm and a new algorithm SORT-IMM is proposed. This gave the best tracking result, with the highest HOTA, the most number of trajectories tracked, and the least number of lost trajectories.

With this result, it can be concluded that using above-water trackers for underwater tracking is not suitable and needs to be adjusted. Using a constant velocity with a constant aspect ratio motion model does not describe the motion well. Even getting a correct velocity measure during the Kalman filter update step can improve the tracking performance a lot.

In summary, this thesis presents a significant advancement in the field of underwater object tracking. By moving beyond the limitations of linear constant velocity models and integrating novel velocity estimation techniques, we have developed trackers that offer reliable and precise tracking performance in challenging underwater environments. These contributions lay the groundwork for further research and development in underwater video analysis, with potential applications in marine biology, underwater robotics, and environmental monitoring.

6-2 Recommendations

With the rapid pace of technological advancement, there is a growing temptation to employ increasingly complex, learning-based algorithms that can adapt and learn in real-time. However, incorporating such techniques as deep neural networks which usually use optical flow calculations, and feature extraction, often leads to significant increases in computational load and processing time, which may not be practical or efficient for this application. This suggests that simply making algorithms more complicated is not necessarily the best path forward. Looking ahead, it is crucial to direct research efforts toward refining the motion model within the Kalman filter, rather than developing a new algorithm from scratch.

Enhancing the accuracy and reliability of the motion model is essential for achieving better performance and more accurate predictions. The SORT algorithm is capable of giving good tracking performance for an accurate motion model. Future studies should prioritize developing more sophisticated and precise motion models that can improve overall system performance without unnecessarily complicating the algorithm. Incorporating additional sensors on the ROV, such as high-precision accelerometers and gyroscopes, will give accurate data on the motion of the ROV. The data from these sensors can be used as real measurements for the Kalman filter update step instead of using velocity estimation techniques. This will improve real-time velocity estimations by providing more comprehensive motion data.

Once a larger dataset has been collected, the ground truth trajectories can also be leveraged to develop a more accurate motion model through offline learning. This refined motion model can then be integrated with the commonly used constant velocity model within the IMM filter. By doing so, it would be possible to significantly enhance the filter's ability to predict and adapt to the dynamic and often unpredictable nature of underwater environments. This approach could lead to more robust and reliable predictions, ultimately improving the overall performance of the system in challenging conditions.

By concentrating on improving the motion model, future research can lead to more efficient and effective solutions that maintain a balance between complexity and computational efficiency.

Appendix A

Conference Paper Draft

A draft conference paper has been written on the research presented in this MSc Thesis. The paper is written in the format of IEEE Oceanic Engineering Journal format.

Multiple Object Tracking in Underwater Environment

Mallika Tripathi, Athina Ilioudi, Bart De Schutter *Fellow, IEEE*

***Abstract*—While various tracking algorithms have demonstrated effectiveness in terrestrial and aerial contexts, their performance in underwater settings, which presents unique challenges due to variable lighting, water turbidity, and unpredictable camera movement, remains unexplored. Addressing this gap is crucial for applications such as marine biology research, underwater surveillance, and autonomous underwater vehicles. This research first evaluates existing tracking algorithms on a novel underwater video dataset with a moving camera and then proposes velocity estimation techniques, a formula-based and an optical flow-based, to enhance tracking performance giving rise to two new algorithms. Furthermore, the thesis proposes a novel tracker that simultaneously utilizes multiple motion models to estimate a target object's location. The results indicate that using velocity estimation techniques improves tracking accuracy by 11%, and tracks more objects by nearly halving the number of lost objects. Incorporating another motion model, a constant acceleration model, gives the best result, with the highest tracking accuracy, and the least number of identity switches all in real-time computational speed.**

***Index Terms*—Multiple Object Tracking, Kalman filter, Velocity Estimation, Optical Flow, Interacting Multiple Models.**

I. Introduction

The Earth's oceans play a crucial role in weather regulation and serve as the largest natural carbon and heat sink [18]. Maintaining the health of the ocean is synonymous with ensuring the overall well-being of our planet. Yet, up to 12 million metric tons of plastic is dumped into the oceans each year [9]. To tackle this issue, an initiative funded by the European Union, SeaClear aims to eliminate seabed and floating marine pollution [15]. The project employs a fleet of autonomous, intelligent robots working collaboratively to monitor and collect marine litter. Various devices, including cameras and forward-looking sonars, are employed to detect litter before collecting it using a gripper mechanism. The SeaClear team has effectively incorporated the capability to detect different objects underwater using a camera at this stage of development. The detection process utilizes a Convolutional Neural Network (CNN) trained on SeaClear-specific data. This trained network can categorize objects into four distinct classes: plant, animal, trash, and ROV. The detection result is illustrated in Figure 1. The subsequent phase involves initiating the collection process for the detected trash.

The ROV descends to collect the trash by continuously aligning with it on the seabed. Challenges arise at this step, especially in situations where multiple pieces of trash are clustered together. In such instances, the ROV is unable to pinpoint a specific target object for retrieval. To address this issue, the proposed solution is for the ROV to sequentially collect the objects. For this, it is imperative to assign a unique identifier to each detected object. A key aspect of this approach is object tracking, where the ROV ensures that every item it encounters is given a distinct identity, allowing for continuous monitoring and precise retrieval over time.

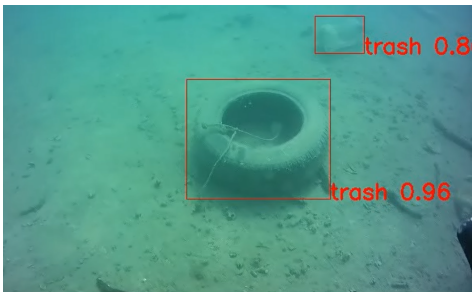


Fig. 1. Detecting trash underwater with the SeaClear system.

Currently there exist multiple state-of-the-art (SOTA) object tracking algorithms, but all have been designed and implemented for conventional conditions above sea level [4], [12], [16]. A few examples can be found when object tracking algorithms were implemented underwater with stationary cameras for counting fish [20]. However, the scenario under study is unique, involving a moving underwater camera, diverse detection classes, and potential interruptions in object visibility due to the movement of the ROV creating disturbances at the seabed like moving sand and bubbles.

In this paper, we address the need to develop object-tracking algorithms specifically for the unique challenges of underwater tracking. Specifically, the contributions of this paper are the following:

- We present a novel dataset for underwater multiple-object tracking.
- We incorporate velocity estimation techniques with the existing SOTA object tracking algorithm using analytical formula and optical flow.
- We employ multiple motion models for more accurate state estimation.

This paper is structured as follows. Section II offers an overview of the essential preliminary knowledge of object tracking. In Section III, various SOTA object tracking algorithms are discussed with their advantages and disadvantages, along with the rationale for developing a novel dataset for

This work was supported by SeaClear, European Union's Horizon 2020 Research and Innovation Programme under Grant 871295 and by SeaClear2.0, European Union's Horizon Europe innovation action programme under Grant 101093822.

underwater tracking. Section IV details the proposed methodologies, introducing optical flow and Interacting Multiple Model (IMM) filters. Section V covers the implementation and results. Finally, Section VI presents the conclusions and potential avenues for future work.

II. Preliminaries

The majority of the SOTA object tracking algorithms are based on the tracking-by-detection method where tracking is performed through continuous detection and association of objects. This method encompasses multiple stages, with each step employing a range of methods, including deep learning, signal processing, and statistical modeling. The various steps are discussed below [1], [21].

- Detection: Identifying the presence of an object in the initial frame of a video or image sequence. This can be done using various object detection or feature detection algorithms.
- Initialisation: Establishing a reference or model of the target object based on its appearance, features, or both by giving it a unique identity. This serves as the starting point for subsequent tracking.
- Localisation: Continuously estimating the object's position in subsequent frames, often through methods like motion estimation or feature matching.
- Data Association: Linking the detected object in the current frame with the previously identified target. This is a critical step in maintaining the object's identity over time.

One of the most popular real-time object detection algorithms is the You-Only-Look-Once (YOLO) [7], known for its speed and efficiency. It processes the entire image in a single forward pass through a neural network by dividing the input image into smaller grid cells. A bounding box is presented at each detected box with a confidence score. The current detection model for SeaClear uses a YOLOv8 model, trained on SeaClear data. Once the object is identified, it must be given a unique reference or identity. In all subsequent frames, it is crucial to consistently assign the same initial identity to the object, affirming that it is the same object detected in the preceding frames. In the context of object tracking, motion estimation serves as a foundation for predicting the future positions of objects and hence enabling real-time association [17]. This process is particularly valuable in scenarios where the appearance of the object may change, or where multiple objects with similar features are present, making traditional methods like feature matching less reliable.

III. Related Work

A. Object Tracking Algorithms

Simple, Online, and Realtime Tracking (SORT) [4] is one of the most widely used object-tracking algorithms. SORT relies solely on the bounding box's position and size for both motion estimation and data association. Motion estimation is employed via a Kalman filter, utilizing a linear constant velocity model that approximates the inter-frame displacements of the objects and remains independent of other objects and camera motion. Zhang et al. [19] introduced a simple and effective data association method, Byte. In contrast to most algorithms that only utilize detection boxes with high confidence scores, here almost every detection box is retained and separated into high-score ones and low-score ones. Utilizing this data association method with SORT gives another SOTA object tracker ByteTrack. BoT-SORT [2] builds on top of ByteTrack by additionally handling camera motion compensation. This is done by adopting conventional image matching to estimate the camera motion.

Since the current object tracking algorithms were designed keeping in mind pedestrian and vehicle movement, it is assumed to follow constant velocity. However, the ROV does not move with a constant velocity and the water disturbances make this motion more unpredictable. The effectiveness of SORT-like tracking methods is heavily influenced by the accuracy of the predicted bounding box for the tracklet. If the motion model is not accurate, this results in a reduced overlap between corresponding bounding boxes, subsequently diminishing the tracker's overall performance. So it is imperative to develop a precise discrete-time linear motion model for entities like animals, fish, and trash, which poses significant challenges. In practice, the movement patterns of fish and floating debris, driven by water currents, cannot be modeled using first principle equations. Therefore, employing a range of velocity estimation techniques and integrating multiple motion models is the proposed approach of this paper.

B. Datasets

There have been several object-tracking benchmarks for both single and multiple object tracking. The most popular is the Multiple Object Tracking (MOT) benchmark [5], which has evolved into an annual challenge in computer vision and tracking research, introducing increasingly complex datasets each year. In total, 22 different datasets were published between 2015 and 2023 for the MOT Challenge. Of these 22, only one, 3D-ZeF [13] is remotely relevant to the problem of underwater object tracking as it contains Zebrafish tracking in a laboratory environment. A laboratory environment has good lighting and fish tanks are not deep enough to accurately depict light distortion, and refraction that would occur in deep waters. There is also an absence of dirt and water currents which contributes heavily in a real underwater setting. Unfortunately, all the other existing tracking datasets are open-air and are based on single-category tracking focusing on either pedestrians or vehicles.

IV. Methodology

A. Dataset Synthesis

Since no publicly accessible dataset for underwater object tracking exists, the aim is first to develop a novel underwater multi-class multi-object tracking dataset to represent diverse underwater conditions, including clear and murky water, and varying water currents. The SeaClear team has conducted extensive tests using their ROV in diverse locations. The dataset derived from these tests will serve as the definitive

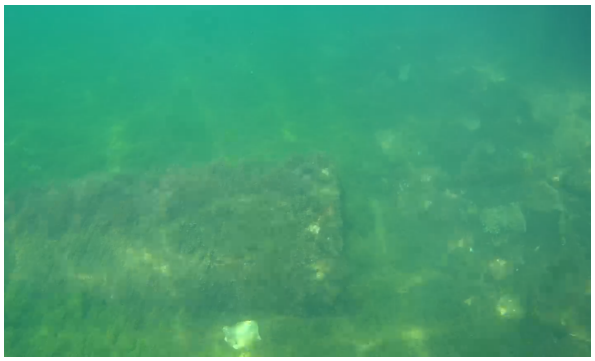


Fig. 2. Light distortion underwater.

Fig. 3. Bubbles caused due to water currents leading to occlusion.

benchmark for evaluating algorithm performance in this research. SeaClear goes to the ocean floor to collect trash where light reaches very scarcely after refracting and reflecting multiple times. As seen in Figure 3, light distortion, water currents, and additional motion due to these currents bring more uncertainty and challenges in this environment.

Collecting and processing videos for tracking is not enough to make a dataset capable of being used for object tracking. The video sequences need to be annotated with ground truth information to compute performance metrics ensuring a definitive and comparable result rather than relying on intuition and the human eye. To ensure a consistent comparison, the videos will undergo a self-annotation process. Instead of manually annotating all frames, the existing detection model is coupled with a Python script, to generate initial ground truth annotations. While this automated process will be flawed, inaccuracies will be corrected through manual annotation after this. In total, 82 different object trajectories are annotated, with different length and motion characteristics. The characteristics of the trajectories after annotation are given in Table I. Trajectories are categorized as short if they are less than 3 seconds, medium if they range between 3 and 7 seconds, and long if they exceed 7 seconds. The long trajectories are also the only objects in the video sequence, while most short and medium trajectories have other objects in their surroundings.

B. Velocity Estimation

For more accurate tracking of the object, improving the motion model is the focus. The motion model in the Kalman

TABLE I
OVERVIEW OF THE OBJECT TRAJECTORIES IN THE NEWLY CREATED DATASET.

Trajectory length	Number of objects
Short ($\leq$ 3s)	43
Medium	32
Long ($\geq$ 7s)	7

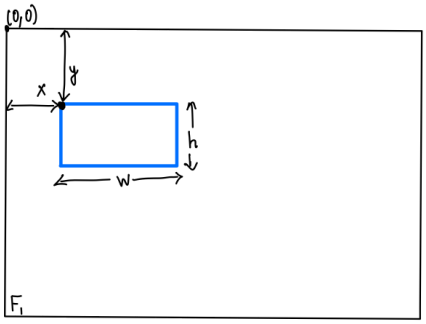


Fig. 4. Bounding box coordinates in an image.

filter is defined as

$$X_{k+1} = A \cdot X_k \tag{1}$$

with X as the state vector and A matrix as the system matrix or state transition matrix. This matrix follows a constant velocity model. The position and velocity estimates obtained using this model can be inaccurate as the real motion of the ROV does not adhere to a constant velocity model. However, during the update step of the method, the position estimates are corrected using data or measurements of the actual object position, using the bounding box. However, since there is no data about the velocity, this correction step does not apply to the velocity estimates, leading to persistent inaccuracies in the velocity estimates which can lead to inaccuracies in the overall estimates. Two different velocity estimation methods are proposed and the result from these will be assumed to be the actual velocity. These estimates will be used as "sensor data" in the Kalman filter for the update step.

1) Formula Based: We start with considering the simplest definition of velocity, the rate of change of position. No matter the motion of the object—whether linear, nonlinear, circular, or otherwise—the velocity of a point mass can be described as

$$\dot{x}_k = \frac{x_k - x_{k-1}}{dt}, \tag{2}$$

where dt is the time difference. This can be extended to the single pixel describing the top left corner of the bounding box. The aim is to use this velocity value as a sensor measurement in the update step of the Kalman filter. The update step does not occur at every frame, but only when a successful data association is achieved.

The velocity estimates are hence calculated as

$$\dot{x}_{est} = \frac{x_{curr} - x_{prev}}{\texttt{time since update}}, \tag{3}$$

$$\dot{y}_{est} = \frac{y_{curr} - y_{prev}}{\texttt{time since update}}, \tag{4}$$

with the `time since update` variable being the number of frames between two update steps of the same object identity and x_{prev} and y_{prev} being the x, y pixel locations of the bounding box at the last update step. The `time since update` variable is incremented by 1 at every predicted step of the Kalman filter. The pseudo-code for this method is given in Algorithm 1.

Algorithm 1 Formula-based velocity estimation with the Kalman filter update step.

```
bbox ← bounding box
kf ← Kalman Filter instance
x_prev ← initial x position
y_prev ← initial y position
time since update ← 0
while tracking do
  x ← bbox[0]
  y ← bbox[1]
  if bbox matched with object then
    x_dot ← (x − x_prev)/time since update
    y_dot ← (y − y_prev)/time since update
    kf.update(bbox, x_dot, y_dot)
    x_prev ← x
    y_prev ← y
    time since update ← 0
  else
    kf.predict()
    time since update ← time since update + 1
  end if
end while
```

2) Optical Flow: Optical flow [8] is an algorithm that estimates the motion of pixels and key points between consecutive frames by analysing the intensity changes. Optical flow is often represented by a vector field, where each vector component represents the horizontal and vertical motion (direction and velocity) of the pixel [11]. In optical flow, the brightness consistency assumption is used. This assumption states that the intensity of pixels does not change across consecutive frames. The optical flow equation is given as

$$I_x \cdot v_x + I_y.v_y = -I_t, \tag{5}$$

where I_x and I_y are the partial derivatives of the image intensity $I(x, y, t)$ of the spatial coordinates x and y. I_t is the partial derivative of the image intensity with respect to time, t. Equation 5 is also known as the gradient constraint which has two unknowns (v_x, v_y), the velocities in the x and y direction, respectively, in one equation. Under the assumption of constant brightness, other constraints are brought into play to solve Equation 5 to obtain the pixel velocities. Currently, two widely used algorithms are the Lucas-Kanade method and Horn-Schunck method presented in [11]. The aim now is to use this general velocity information as the sensor data in the Kalman filter update step, just as we did in the previous section instead of using the formula-based framework.

Algorithm 2 Using optical flow for velocity estimation with the Kalman filter update step.

```
bbox ← bounding box
kf ← Kalman Filter instance
img_p ← previous gray image frame
img ← current gray image frame
s ← space between dense points
fps ← 25 {no. of frames per second}
if bbox matched with object then
  flow ← cv2.calcOpticalFlowFarneback(img_p, img)
  h, w ← img.shape[: 2]
  y, x ← mesh grid[s/2 : h : s, s/2 : w : s]
  fx, fy ← Transpose(flow[y, x])
  x_dot ← fx.mean ∗ fps {Estimate velocity in x direction}

  y_dot ← fy.mean ∗ fps {Estimate velocity in y direction}

  kf.update(bbox, x_dot, y_dot)
  time since update ← 0
else
  kf.predict
  time since update ← time since update + 1
end if
```

C. Interacting Multiple Model

The Interacting Multiple Model (IMM) filter [6] is a probabilistic algorithm used for state estimation to handle situations when multiple possible models can describe the evolution of a system and the true model is not known beforehand or changes dynamically. Each dynamic model represents a different hypothesis about the system's behavior. Since we are unaware of the exact motion model of the ROV, using the IMM filter is suitable. Each model in an IMM filer has an associated probability that reflects how likely it is that the system is currently following that particular model at the time step. These probabilities are updated over time based on the sensor measurements. After the update step, the IMM filter combines information from all models to produce a final estimate of the system's state. This combination is weighted by the probability of each model.

The aim is to implement the IMM filter to introduce another model, the constant acceleration model, and evaluate how well it describes the motion. This would increase the number of states in the system. The state vector for the constant acceleration model is

$$X = [x, y, w, h, \dot{x}, \dot{y}, \dot{w}, \dot{h}, \ddot{x}, \ddot{y}]^T. \tag{6}$$

with (x,y) defined as the top left pixel coordinate of the bounding box, and w and h are the width and the height of the bounding box respectively. The width and the height are not considered here for the constant acceleration and are still considered to change with constant velocity. We are more interested in making sure that the top left corner of the bounding box follows different motions and that the width and the height of the bounding box do not contribute to the overall trajectory of the object. We use an IMM filter with

two different motion models, constant velocity and constant acceleration, and evaluate the outcome.

The motion model for a constant acceleration model is

$$X_{k+1} = A \cdot X_k \tag{7}$$

with the A matrix defined as

$$\begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & \frac{dt^2}{2} & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 & \frac{dt^2}{2} \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

and X being the state vector as defined in Equation 6. The constant velocity model follows the same structure, except the double differential term in the last two columns of the first two rows is 0.

V. Implementations and Results

The Higher Order Tracking Accuracy (HOTA) [10] metric is a comprehensive evaluation measure used in multi-object tracking tasks. It aims to provide a balanced assessment of both detection and association accuracy, addressing some of the limitations of traditional metrics like MOTA (Multi-Object Tracking Accuracy) [3] and IDF1 (Identity F1 Score) [14]. To compare performance, the HOTA metric is considered alongside the number of mostly tracked (MT) and mostly lost (ML) trajectories.

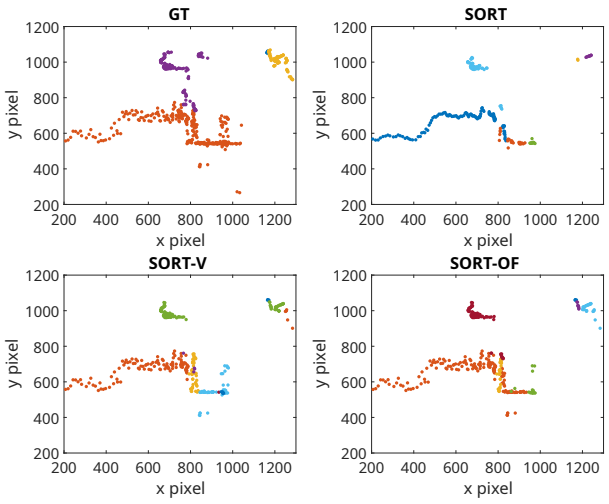


Fig. 5. Ground truth trajectories with tracking results from SORT, SORT-V, and SORT-OF.

A. Velocity Estimation

Both the formula-based and the optical flow-based velocity estimation techniques are applied to the SORT algorithm, giving two novel algorithms namely, SORT-V and SORT-OF respectively. From Table II, there is a clear increase in performance when velocity estimation is implemented. The localization accuracy reaches 99%. This can also be seen in Figure 5. The SORT algorithm when tracking the object in orange in the ground truth tracks an approximate linear version of the actual trajectory. When velocity estimation techniques are introduced, the true nature of the trajectory is tracked, increasing the localization accuracy and overall tracking accuracy.

Even when compared to ByteTrack and BoT-SORT, the performance of velocity estimation techniques shows significant improvement. Both formula-based and optical flow-based velocity estimation methods effectively eliminate false positives. This can be seen visually in the Figure 6. All algorithms are also compared numerically in Table III. The HOTA score increases gradually, but it is also crucial to consider additional performance metrics. Specifically, when examining the mostly tracked and mostly lost trajectories, the velocity estimation algorithms reduce the number of mostly lost trajectories by nearly half. This represents a substantial improvement in tracking performance.

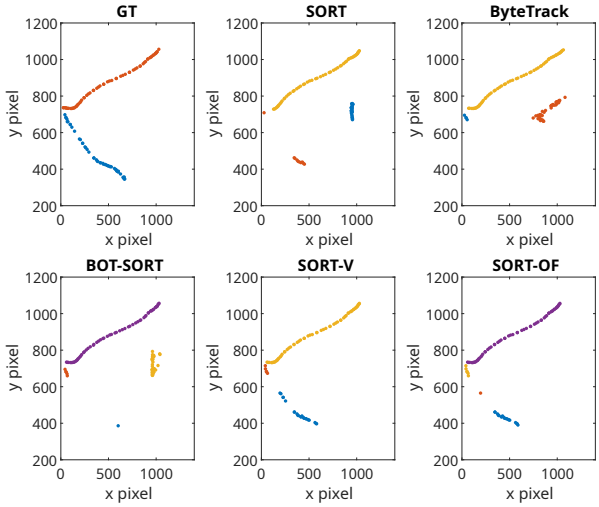


Fig. 6. Velocity estimation techniques can resolve false positives.

B. IMM

To account for different motion models, the IMM filter is implemented with the SORT algorithm, resulting in a novel tracker, SORT-IMM. The IMM filter is implemented with two different models, a constant velocity and a constant acceleration model. The probabilities for switching between a constant velocity model and a constant acceleration model, and vice versa, must be defined. The probability of remaining in the same model is denoted as α, and the probability of

TABLE II
COMPARING RESULTS OF THE PROPOSED VELOCITY ESTIMATION-BASED TRACKERS WITH SORT.

Algorithms	HOTA	DetA	AssA	LocA	IDs
SORT	61.838	70.51	54.26	95.43	117
SORT-V	67.445	81.738	55.651	99.866	171
SORT-OF	68.619	81.525	57.757	99.845	156

transitioning to a different model is given by $100 - \alpha$. This leads to the following transition matrix:

$$p = \begin{bmatrix} \alpha & 100 - \alpha \\ 100 - \alpha & \alpha \end{bmatrix}. \tag{9}$$

When implementing the SORT-IMM with various values of α, the best performance is achieved when $\alpha = 75$. This means that if the tracker starts in a constant velocity model, there is a 75% probability of remaining in that model and a 25% probability of transitioning to a constant acceleration model. Similarly, if the tracker starts in a constant acceleration model, there is a 75% probability of staying in that model and a 25% probability of switching to the constant velocity model. Another parameter of the IMM filter is the model probability which represents the likelihood that the system is operating under each model at any given time. These probabilities are updated at each time step based on the measurement likelihoods and the transition probabilities between models. The initial model probabilities are set to 50-50 because, in practice, the exact motion of the ROV is unknown, and there is no preference for one model over the other.

Comparing the HOTA, detection, association, and localization accuracy metrics in Table III, it can be seen that the newly proposed SORT-IMM gives the highest HOTA value at 68.93. This high score results from its superior association and localization accuracy. While its detection accuracy is not the highest, it is only 0.7 less than the top-performing tracker. From this, it can be concluded that object tracking algorithms rely heavily on the optimal performance of all sub-algorithms, detection, association, and localization equally.

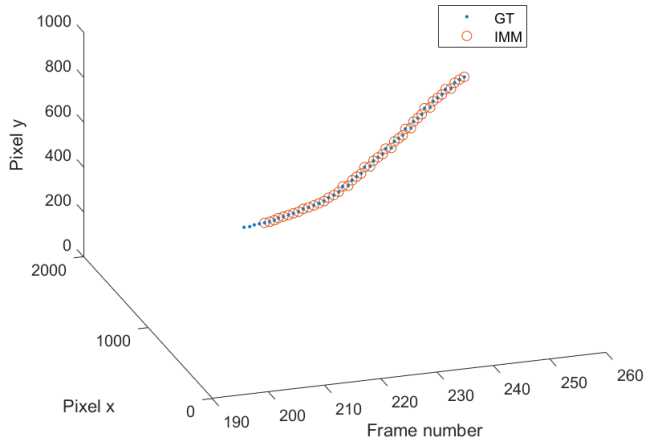


Fig. 7. SORT-IMM tracking output with the ground truth.

Object trajectories with the ground truth and the SORT-IMM output are illustrated in Figure 7. The IMM-based tracker closely follows the ground truth. The model probability of the constant acceleration model remains at 1 for multiple frames as seen in Figure 8. This indicates that this model consistently describes the actual motion of the object, resulting in continuous tracking.

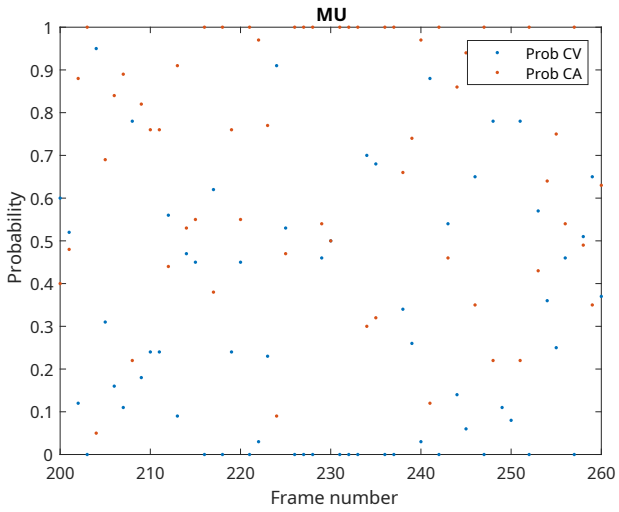


Fig. 8. Model probabilities throughout the tracking horizon.

Considering other performance metrics, the SORT-IMM tracker does not exhibit the lowest number of identity switches, instead, the SORT tracker gives the fewest identity switches. This is unexpected given the lower association accuracy. This reduction in identity switches can be attributed to the tracker missing approximately 31 objects. As a result, instead of tracking the original 82 ground truth identities, it only managed to follow around 51 trajectories. Hence, it is important to consider that the SORT tracker has 39 identity switches when tracking 51 objects. In this context, the SORT-IMM provides the best overall performance, with the highest number of mostly tracked objects at 30, the fewest mostly lost trajectories at 13, and 52 identity switches. The algorithm SORT-OF also gives good tracking performance with the second highest HOTA only 0.3 behind the SORT-IMM tracker and the second highest mostly tracked trajectories, only 1 less than the SORT-IMM. This performance can be seen in Figure 10, with other algorithms giving identity switches and false positives, SORT-OF gives the same result as the SORT-IMM tracker.

Another important metric is the computational speed of the algorithms. This is an important metric to keep in mind to be able to achieve online and real-time tracking. The metric is only for the tracking algorithm and does not include the computational time to detect objects. All algorithms were run using the Nvidia GeForce MX570 GPU. Both methods using optical flow are not suitable to be used in an online, real-time setting as seen from the computation time in Table III. BoT-SORT can be used with a more powerful computing system, but the SORT-OF is just at 1.8 frames per second. So even though it has good performance and is close to the SORT-

IMM tracker, the computational speed is very slow and not suitable for real-time tracking.

When examining detailed performance metrics, competition emerges between the SORT-IMM and the SORT-V. The latter demonstrates slightly higher detection recall and precision, with margins of 0.3 and 0.03, respectively, compared to the IMM-based tracker. However, the IMM tracker excels in association recall, achieving 59.01 compared to 55.94 for SORT-V. Association recall measures how well predicted trajectories cover ground-truth trajectories, a low association recall indicates that a tracker splits an object into multiple predicted tracks. The higher association recall of the SORT-IMM tracker suggests it consistently associates objects without identity switches, resulting in fewer false negatives.

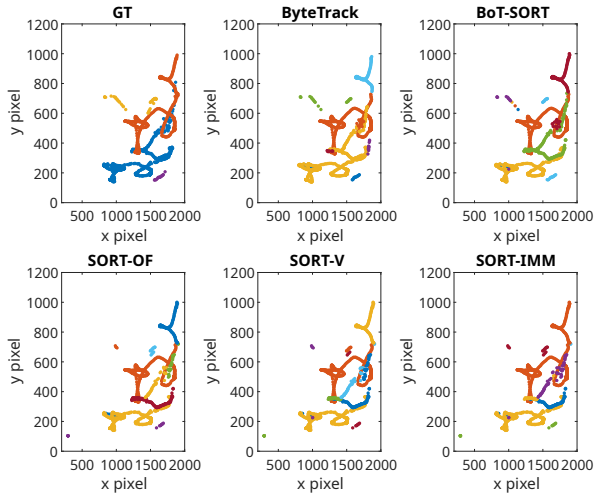


Fig. 9. Comparing tracking performance of all implemented algorithms. SORT-IMM tracking nonlinear trajectories the best.

The performance of all algorithms implemented and discussed is also visually compared. The SORT-IMM tracker is most capable of tracking non-linear motion as seen in Figure 9. It is the only algorithm consistently able to track the loop starting approximately at pixel coordinate $(2000, 1000)$. This shows that having the constant velocity model is not enough to describe the motion and additional information about the velocity or more defined motion model is required. It can also be concluded that the constant acceleration model can help describe the object trajectories better though not perfectly.

VI. Conclusions and Future Work

With little to no public research on multiple object tracking in underwater scenarios, this research aims to first develop a novel dataset to facilitate more research on this topic. In addition, existing SOTA object tracking algorithms are implemented on this new dataset. Additionally, two newly proposed velocity estimation techniques are implemented, and a new multi-model tracker is introduced. Two novel trackers are proposed, SORT-V and SORT-OF, both incorporate advanced velocity estimation methods to more accurately capture the erratic and non-linear motion of underwater objects. SORT-V uses formula-based velocity calculation at each update step of the Kalman Filter. In contrast, SORT-OF utilizes optical flow to calculate pixel velocities. This calculated velocity serves as a sensor measurement to update both velocities and positions during the update step. Both give improved tracking results when compared to SORT, ByteTrack, and BoT-SORT with the least number of lost trajectories and higher HOTA scores, but SORT-OF had a very low computational speed of 1.8 fps.

No matter how much we can improve the update step, an incorrect motion model will always lead to inaccurate tracking. So IMM filter is used instead of the normal Kalman Filter in the SORT algorithm. Two different motion models are implemented in the IMM filter the constant velocity model and the constant acceleration model. Initial model probabilities are set at 50% for each, indicating no preference between the models. The best performance is achieved with a transition probability of 25% between models and 75% within the same model. This gives the best tracking result, with the highest HOTA, the most number of trajectories tracked, and the least number of lost trajectories. With this result, it can be concluded that using above-water trackers for underwater tracking is not suitable and needs to be adjusted. Using a constant velocity with a constant aspect ratio motion model does not describe the motion well. Even getting a correct velocity measure during the Kalman filter update step can improve the tracking performance a lot. In summary, this paper presents a significant advancement in the field of underwater object tracking. By moving beyond the limitations of linear constant velocity models and integrating novel velocity estimation techniques, we have developed trackers that offer reliable and precise tracking performance in challenging underwater environments. These contributions lay the groundwork for further research and development in underwater video analysis, with potential applications in marine biology, underwater robotics, and environmental monitoring.

Enhancing the accuracy and reliability of the motion model is essential for achieving better performance and more accurate predictions. Future studies should prioritize developing more sophisticated and precise motion models to improve overall system performance without unnecessarily complicating the algorithm. Incorporating additional sensors on the ROV, such as high-precision accelerometers and gyroscopes, will give accurate data on the motion of the ROV. The data from these sensors can be used as real measurements for the Kalman filter update step instead of using velocity estimation techniques. This will improve real-time velocity estimations by providing more comprehensive motion data.

Once a larger dataset has been collected, the ground truth trajectories can also be leveraged to develop a more accurate motion model through offline learning. This refined motion model can then be integrated with the commonly used constant velocity model within the IMM filter. By doing so, it would be possible to significantly enhance the filter's ability to predict and adapt to underwater environments' dynamic and often unpredictable nature. This approach could lead to more robust and reliable predictions, ultimately improving the system's overall performance in challenging conditions.

By concentrating on improving the motion model, future

TABLE III
COMPARING PERFORMANCE METRICS OF SOTA TRACKING ALGORITHMS WITH THE NEWLY PROPOSED.

Tracker	HOTA	DetA	AssA	LocA	DetRe	DetPr	AssRe	AssPr	IDSW	MT	ML	Frag	FPS
SORT	61.83	70.52	54.26	95.43	72.56	95.26	55.19	96.73	39	32	31	69	2694.9
ByteTrack	64.81	72.41	58.07	94.02	75.76	92.70	57.49	94.67	46	25	29	83	4489.9
BoT-SORT	67.75	79.79	57.54	97.15	82.40	95.99	58.28	97.72	80	26	29	92	23.2
SORT-V	67.44	**81.74**	55.65	99.8	**83.42**	**97.60**	55.94	98.78	69	30	14	110	3158.3
SORT-OF	68.62	81.52	57.76	99.84	83.20	97.59	58.12	**98.79**	65	29	15	111	1.8
SORT-IMM	**68.93**	81.06	**58.62**	**99.86**	82.73	97.56	**59.01**	98.75	52	**30**	**13**	114	966.4

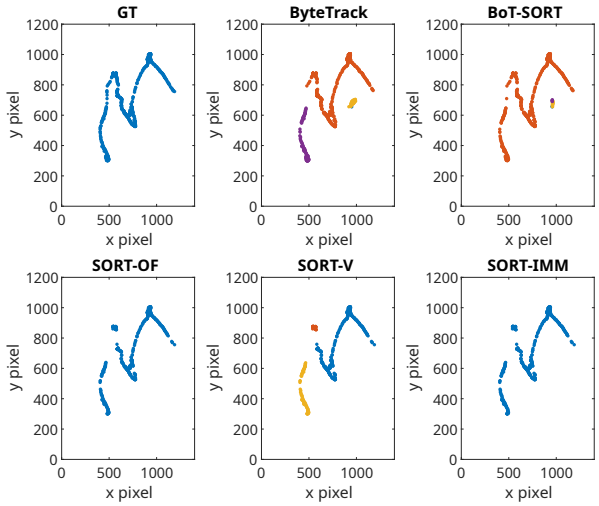


Fig. 10. Comparing tracking performance of all implemented algorithms. SORT-OF and SORT-V both give equally good results.

research can lead to more efficient and effective solutions that maintain a balance between complexity and computational efficiency.

REFERENCES

[1] Abdul Rajjak, S. & Kureshi, A. Recent Advances in Object Detection and Tracking for High Resolution Video: Overview and State-of-the-Art. *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*. pp. 1-9 (2019)

[2] Aharon, N., Orfaig, R. & Bobrovsky, B. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. (arXiv,2022)

[3] Bernardin, K. & Stiefelhagen, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP J. Image Video Process.*. **2008** pp. 1-10 (2008)

[4] Bewley, A., Ge, Z., Ott, L., Ramos, F. & Upcroft, B. Simple Online and Real-time Tracking. *2016 IEEE International Conference On Image Processing (ICIP)*. pp. 3464-3468 (2016)

[5] Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S. & Leal-Taixé, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. *Int. J. Comput. Vis.*. **129**, 845-881 (2021,4)

[6] Genovese, A. The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets. (https://secwww.jhuapl.edu/techdigest/Content/techdigest/pdf/V22-N04/22-04-Genovese.pdf), Accessed: 2024-7-22

[7] Ge, Z., Liu, S., Wang, F., Li, Z. & Sun, J. YOLOX: Exceeding YOLO Series in 2021. (arXiv,2021)

[8] Horn, B. & Schunck, B. Determining Optical Flow. *Artif. Intell.*. **17**, 185-203 (1981,8)

[9] Jambeck, J., Geyer, R., Wilcox, C., Siegler, T., Perryman, M., Andrady, A., Narayan, R. & K. L. Law Plastic Waste Inputs from Land Into the Ocean. *Science*. **347**, 768-771 (2015), https://www.science.org/doi/abs/10.1126/science.1260352

[10] Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixe, L. & Leibe, B. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. (2020,9)

[11] Mahdavi, H. Optical Flow Based Moving Object Detection and Tracking for Traffic Surveillance. (Zenodo,2013,9)

[12] Mirunalini, P., Jaisakthi, S. & Sujana, R. Tracking of Object in Occluded and Non-Occluded Environment using SIFT and Kalman filter. *TENCON 2017 - 2017 IEEE Region 10 Conference*. (2017,11)

[13] Pedersen, M., Haurum, J., Bengtson, S. & Moeslund, T. 3D-ZeF: A 3D Zebrafish Tracking Benchmark Dataset. *ArXiv:2006.08466[cs]*. (2020), https://arxiv.org/abs/2006.08466, arXiv: 2006.08466

[14] Ristani, E., Solera, F., Zou, R., Cucchiara, R. & Tomasi, C. Performance measures and a data set for multi-target, multi-camera tracking. (arXiv,2016)

[15] SeaClear Website. (https://seaclear-project.eu/about-main/about-seaclear), Accessed: 29-11-2023

[16] Salhi, A., Moresly, Y., Ghozzi, F., Yengui, A. & Fakhfakh, A. Modeling from an Object and Multi-object Tracking System. *2016 Global Summit On Computer And Information Technology (GSCIT)*. pp. 80-85 (2016)

[17] Shariat, H. & Price, K. Motion Estimation with More than Two Frames. *IEEE Transactions On Pattern Analysis And Machine Intelligence*. **12**, 417-434 (1990)

[18] United Nations The ocean – the world's greatest ally against climate change — United Nations. (United Nations)

[19] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W. & Wang, X. ByteTrack: Multi-object Tracking by Associating Every Detection Box. (arXiv,2021)

[20] Zhang, X., Zeng, H., Liu, X., Yu, Z., Zheng, H. & Zheng, B. In Situ Holothurian Noncontact Counting System: A General Framework for Holothurian Counting. *IEEE Access*. **8** pp. 210041-210053 (2020)

[21] Zhao, Y., Shi, H., Chen, X., Li, X. & Wang, C. An Overview of Object Detection and Tracking. *2015 IEEE International Conference On Information And Automation*. pp. 280-286 (2015)

Appendix B

Python File for Ground Truth Annotation

```
1
2  import os
3  import configparser
4  from ultralytics import YOLO
5  import pickle
6  directory = "Video8_00021983"
7  gt_file = "gt.txt"
8  det_file = "det.txt"
9  requirement = "Seaclear"
10 save_path = "predictions.pickle"
11
12 # if requirement == "Seaclear":
13 #     model = YOLO('best.pt')
14 #     video_path = video
15 # else:
16 #     video_path = 0
17 #     model = YOLO("yolov8n.pt")
18
19 # Parent Directory path
20 parent_dir = "C:/Users/malli/Desktop/Systems and control/THESIS/Data"
21 folder_path = os.path.join(parent_dir, directory)
22 if not os.path.exists(folder_path):
23         os.makedirs(folder_path)
24 gt_path = os.path.join(folder_path, gt_file)
25 det_path = os.path.join(folder_path, det_file)
26 # result=model.predict(video)
27 with open (save_path, 'rb') as file:
28     result = pickle.load(file)
29 file.close()
30 frame = 0
31 id_prev ={}
```

```
32 frame_prev = None
33 unique_id = 0
34 prev_frame_id ={}
35 det_file = open(det_path, 'w')
36 with open(gt_path, 'w', newline='') as out_file:
37     for r in result:
38         frame = frame + 1
39         box =0
40         for b in r.boxes:
41             matching_boxes =[]
42             cls = int(b.cls.item())
43             conf = b.conf.item()
44             print('%d,%d,%.2f,%.2f,%.2f,%.2f,%.2f,%d,%d,%d'%(frame,-1,b.
                   xywh[0][0].item(),b.xywh[0][1].item(),b.xywh[0][2].item(),
                   b.xywh[0][3].item(),conf,cls,1,1),file=det_file)
45             if conf < 0.35 :
46                 break
47             bb_left = b.xywh[0][0].item()
48             bb_top = b.xywh[0][1].item()
49             bb_width = b.xywh[0][2].item()
50             bb_height = b.xywh[0][3].item()
51             bb_area = bb_height*bb_width
52             if frame == 1:
53                 new_object = True
54             else:
55                 for prev_b in frame_prev.boxes:
56                     if cls == int(prev_b.cls.item()):
57                         matching_boxes.append(prev_b)
58                 if matching_boxes == []:
59                     new_object = True
60                 for matched_b in matching_boxes:
61                     bb_left_diff = abs(bb_left - matched_b.xywh[0][0].
                           item())
62                     bb_top_diff = abs(bb_top - matched_b.xywh[0][1].item
                           ())
63                     bb_width_diff =  abs(bb_width - matched_b.xywh[0][2].
                           item())
64                     bb_height_diff =  abs(bb_height - matched_b.xywh
                           [0][3].item())
65                     #bb_area_diff = abs(bb_area - (bb_width_match*
                           bb_width_match))
66                     # new_object = False
67                     if bb_left_diff < 100 and bb_top_diff < 100 and
                           bb_width_diff <100 and bb_height_diff < 70 and
                           matched_b.conf.item() in prev_frame_id.keys():
68                         id = prev_frame_id[matched_b.conf.item()]
69                         new_object = False
70                         break
71                     else:
72                         new_object = True
73             if new_object == True:
74                 unique_id = unique_id + 1
75                 id = unique_id
```

```
76                # id =1
77                print(frame, " ", id, " ", bb_left, " ", bb_top, " ",
                      bb_width, " ", bb_height, " ", conf, " ", cls, " ", 1, " "
                      , 1)
78                print('%d,%d,%.2f,%.2f,%.2f,%.2f,%.2f,%d,%d,%d'%(frame,id,
                      bb_left,bb_top,bb_width,bb_height,conf,cls,1,1),file=
                      out_file)
79                seq_len = frame
80                id_prev.update({conf : id})
81            frame_prev = r
82            prev_frame_id = id_prev
83            id_prev ={}
84    out_file.close()
85    det_file.close()
```

Bibliography

[1] S. S. Abdul Rajjak and A. K. Kureshi. Recent Advances in Object Detection and Tracking for High Resolution Video: Overview and State-of-the-Art. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–9, 2019.

[2] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. 2022.

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision*, 56(3):221–255, 2004.

[4] J. Barrett, Z. Chase, J. Zhang, M. M. B. Holl, K. Willis, A. Williams, B. D. Hardesty, and C. Wilcox. Microplastic Pollution in Deep-Sea Sediments from the Great Australian Bight. *Front. Mar. Sci.*, 7, Oct. 2020.

[5] K. Bernardin and R. Stiefelhagen. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP J. Image Video Process.*, 2008:1–10, 2008.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple Online and Real-time Tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016.

[7] E. Bochinski, V. Eiselein, and T. Sikora. High-Speed Tracking-by-detection Without using Image Information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, Aug. 2017.

[8] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[10] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé. MOTChallenge: A Benchmark for Single-camera Multiple Target Tracking. *Int. J. Comput. Vis.*, 129(4):845–881, Apr. 2021.

[11] J. P. F. D'Haeyer. Gaussian Filtering of Images: A Regularization Approach. *Signal Processing*, 18(2):169–181, Oct. 1989.

[12] D.-Z. Du, P. Pardalos, X. Hu, and W. Wu. Introduction. In *Introduction to Combinatorial Optimization*, Springer optimization and its applications, pages 1–11. Springer International Publishing, Cham, 2022.

[13] O. Escobar Díaz, A. Sánchez López, M. B. Bernábe Loranca, and R. González Velázquez. Real Time Drone Object Tracking Using Histograms of Oriented Gradients and Particle Filters. In *2016 Fifteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 35–40, 2016.

[14] G. Farnebäck. Two-Frame Motion Estimation based on Polynomial Expansion. In *Image Analysis*, Lecture notes in computer science, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[15] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: Exceeding YOLO Series in 2021. 2021.

[16] A. F. Genovese. The Interacting Multiple Model Algorithm for Accurate State Estimation of Maneuvering Targets. https://secwww.jhuapl.edu/techdigest/Content/techdigest/pdf/V22-N04/22-04-Genovese.pdf. Accessed: 2024-7-22.

[17] H. Ghorbani. Mahalanobis Distance and its Application for Detecting Multivariate Outliers. *Facta Univ. Ser. Math. Inform.*, page 583, Oct. 2019.

[18] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artif. Intell.*, 17(1-3):185–203, Aug. 1981.

[19] J. R. Jambeck, R. Geyer, C. Wilcox, T. R. Siegler, M. Perryman, A. Andrady, R. Narayan, and K. L. Law. Plastic Waste Inputs from Land Into the Ocean. *Science*, 347(6223):768–771, 2015.

[20] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *J. Basic Eng.*, 82(1):35–45, Mar. 1960.

[21] M. Kristan, J. Matas, A. Leonardis, T. Vojir, R. Pflugfelder, G. Fernandez, G. Nebehay, F. Porikli, and L. Čehovin. A Novel Performance Evaluation Methodology for Single-Target Trackers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2137–2155, Nov 2016.

[22] H. W. Kuhn. The Hungarian Method for the Assignment Problem. *Nav. Res. Logist. Q.*, 2(1-2):83–97, Mar. 1955.

[23] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, Nov. 2004.

[24] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. Sept. 2020.

[25] H. Mahdavi. Optical Flow Based Moving Object Detection and Tracking for Traffic Surveillance. Sept. 2013.

[26] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A Benchmark for Multi-Object Tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831.

[27] P. Mirunalini, S. M. Jaisakthi, and R. Sujana. Tracking of Object in Occluded and Non-Occluded Environment using SIFT and Kalman filter. In *TENCON 2017 - 2017 IEEE Region 10 Conference*. IEEE, Nov. 2017.

[28] R. W. Obbard, S. Sadri, Y. Q. Wong, A. A. Khitun, I. Baker, and R. C. Thompson. Global Warming Releases Microplastic Legacy Frozen in Arctic Sea Ice. *Earths Future*, 2(6):315–320, June 2014.

[29] Ocean Conservation. Ocean Pollution and Conservation. https://www.conservation.org/stories/ocean-pollution-facts#references. Accessed: 22-01-2024.

[30] K. O'Shea and R. Nash. An Introduction to Convolutional Neural Networks. 2015.

[31] K. Panetta, L. Kezebou, V. Oludare, and S. Agaian. Comprehensive Underwater Object Tracking Benchmark Dataset and Underwater Image Enhancement With GAN. *IEEE Journal of Oceanic Engineering*, 47(1):59–75, 2022.

[32] M. Pedersen, J. B. Haurum, S. H. Bengtson, and T. B. Moeslund. 3D-ZeF: A 3D Zebrafish Tracking Benchmark Dataset. *arXiv:2006.08466[cs]*, 2020. arXiv: 2006.08466.

[33] A. Salhi, Y. Moresly, F. Ghozzi, A. Yengui, and A. Fakhfakh. Modeling from an Object and Multi-object Tracking System. In *2016 Global Summit on Computer and Information Technology (GSCIT)*, pages 80–85, 2016.

[34] SeaClear. Seaclear website. https://seaclear-project.eu/about-main/about-seaclear. Accessed: 29-11-2023.

[35] H. Shariat and K. Price. Motion Estimation with More than Two Frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):417–434, 1990.

[36] J. Shi and Tomasi. Good Features to Track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[37] UNDP. Microplastics on Human Health: How Much do they Harm Us? –undp.org. https://www.undp.org/kosovo/blog/microplastics-human-health-how-much-do-they-harm-us#:~:text=Recent%20evidence%20indicates%20that%20humans,such%20as%20beer%20and%20salt. [Accessed 22-01-2024].

[38] L. Van Cauwenberghe, A. Vanreusel, J. Mees, and C. R. Janssen. Microplastic Pollution in Deep-Sea Sediments. *Environ. Pollut.*, 182:495–499, Nov. 2013.

[39] S. Vasuhi, M. Vijayakumar, and V. Vaidehi. Real Time Multiple Human Tracking using Kalman Filter. In *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–6, 2015.

[40] S. Wu, O. Oreifej, and M. Shah. Action Recognition in Videos Acquired by a Moving Camera using Motion Decomposition of Lagrangian Particle Trajectories. In *2011 International Conference on Computer Vision*, pages 1419–1426, 2011.

[41] Y. Wu, J. Lim, and M.-H. Yang. Object Tracking Benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[42] X. Zhang, H. Zeng, X. Liu, Z. Yu, H. Zheng, and B. Zheng. In Situ Holothurian Noncontact Counting System: A General Framework for Holothurian Counting. *IEEE Access*, 8:210041–210053, 2020.

[43] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. ByteTrack: Multi-object Tracking by Associating Every Detection Box. 2021.

[44] Y. Zhao, H. Shi, X. Chen, X. Li, and C. Wang. An Overview of Object Detection and Tracking. In *2015 IEEE International Conference on Information and Automation*, pages 280–286, 2015.

List of Acronyms

CNN	Convolution Neural Network
DetA	Detection Accuracy
IMM	Interacting Multiple Model
IOU	Intersection-over-union
HOTA	Higher Order Tracking Accuracy
MOT	Multiple Object Tracking
MOTA	Multi-object Tracking Accuracy
ROV	Remotely Operated Vehicle
SIFT	Scale-Invariant Feature Transform
SORT	Simple Online and Real-time Tracking
SOTA	State-of-the-art
YOLO	You Only Look Once