Addressing Multiple Object Tracking with Segmentation Masks

Manuel Bendaña Gómez
Master Thesis — Master in Computer Vision
University Of Santiago de Compostela
Santiago de Compostela, Spain
manuel.bendana@rai.usc.es

Supervisors:
Manuel Mucientes Molina (manuel.mucientes@usc.es)
Victor Manuel Brea Sánchez (victor.brea@usc.es)
University of Santiago de Compostela

***Abstract*—Multiple Object Tracking (MOT) aims to locate all the objects from a video, assigning them the same identities across all frames. Traditionally, this problem was addressed following the Tracking by Detection (TbD) paradigm, using detections represented by bounding boxes. However, bounding boxes can contain information from several objects, something that does not happen with segmentation masks. This work takes the ByteTrack MOT system as a starting point. Our proposal, ByteTrackMask, integrates a class-agnostic segmentation method and a segmentation-based tracker in ByteTrack in order to rescue tracks that would have been lost. Results over validation sets of MOT challenge datasets provide improvements in MOT metrics of interest like MOTA, IDF1 and false negatives.**

***Index Terms*—Multiple object tracking, segmentation, deep learning**

I. Introduction

This Master Thesis is centered on Multiple Object Tracking. This topic consists of the detection of all the objects present in every frame of a video, maintaining their identities along time (Fig. 1) [1].

Nowadays, a problem like MOT is solved using Deep Learning strategies, having special relevance the use of Convolutional Neural Networks (CNNs) [2] and, more recently, Vision Transformers (ViT) [3]. Tracking by Detection (TbD) is a common approach followed by MOT systems. This type of solution is based on the idea shown in Figure 2, where the most relevant components referred to a given frame F_{t+1} are the following:

1) A detection network identifies all the objects present in a video frame with bounding boxes —det_{t+1} in Figure 2—, this is, rectangles that delimit the position in which each object is located.
2) Through data association techniques, detections det_{t+1} and the position of the tracks that were obtained for frame t —P_t in Figure 2– are combined, obtaining the position of the tracks for the actual frame P_{t+1}. In this way, we can save the trajectories for all objects in a video. Sometimes, an extra module makes an estimation of the positions P_t onto frame $t+1$ before association.

A lack of detection for a given track is marked as a miss, even if it is still present in the scene. In this situation, we would like to keep tracking the object, which would improve tracking metrics.

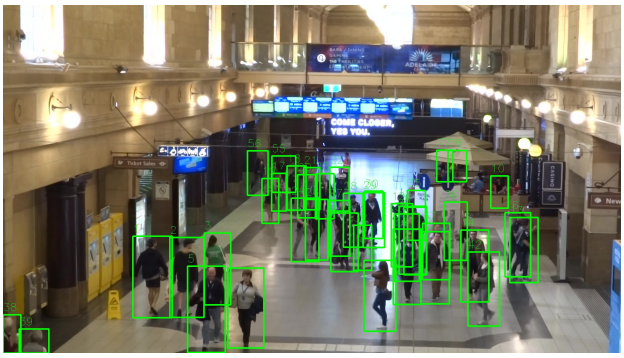

Fig. 1. Example of a MOT20 [1] video frame. Green boxes represent the bounding boxes of each of the objects present in that frame.

A solution for addressing this would be to predict the position of an object in the actual frame with a visual object tracker. Visual object trackers solve a different problem, as shown in Figure 3. Given an exemplar object E and a certain frame F_t, a visual object tracker locates the position of that exemplar in that frame P_t.

This approach is normally based on bounding boxes, however, this is a problem in situations like the one shown in Figure 4. Let us consider that object B is being tracked. Its bounding box is mainly influenced by information from object A. If we want to track B using said bounding box, an identity switch can be produced, this is, to start following object A instead of object B.

We hypothesize that segmentation masks would be more accurate in such a situation. Figure 4 is an example of how segmentation masks would be able to perfectly distinguish both objects even in the presence of occlusions. This may help to avoid confusion between objects and improve tracking metrics.

This type of solution introduces two main challenges. The first one is to generate the segmentation masks themselves. The second one is to track said masks. Our solution to these two challenges in this Master Thesis is to integrate a visual object tracking based on segmentation masks in the well-known MOT system ByteTrack [4]. The final system, which we name ByteTrackMask, includes the following contributions:

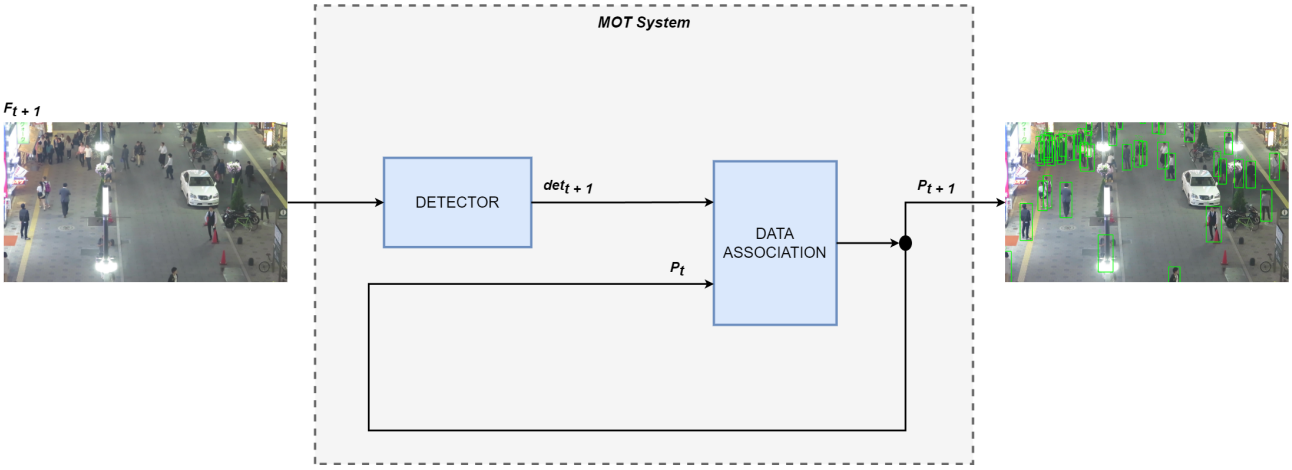

Fig. 2. Representation of the common structure that a MOT system has following the Tracking by Detection (TbD) paradigm.

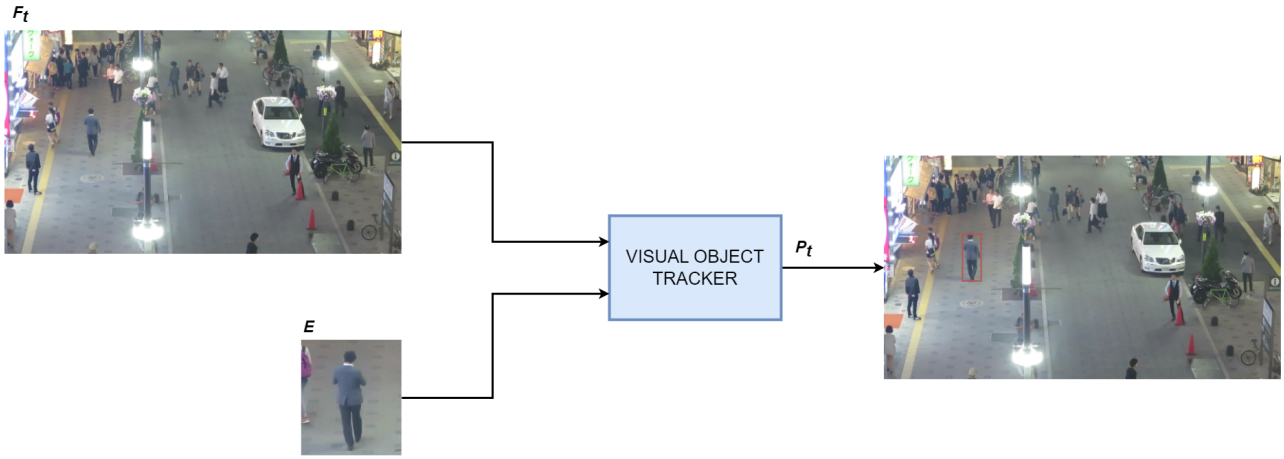

Fig. 3. Representation of the problem addressed by a Visual Object Tracker.

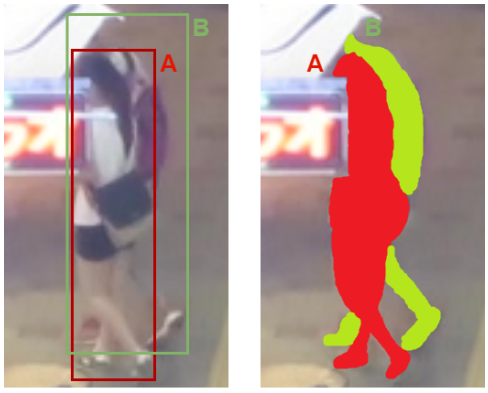

Fig. 4. Example of a situation in a MOT17 dataset video in which two objects have a big overlap, but their precise segmentation masks would still be distinguishable.

- We integrate ByteTrack MOT system with a new module that helps to keep tracking objects without a detection associated.
- We include the Segment Anything Model (SAM) [5] to generate segmentation masks in a class-agnostic fashion in order to provide masks for those tracks of interest.
- We incorporate a visual object tracking method that works with segmentation masks —AOT [6]—, taking as exemplars the masks generated with SAM.

Our main hypothesis is that tracking those masks will boost the performance of the base ByteTrack.

II. State of the Art

A. Multiple Object Tracking

Drawing conclusions and guidelines from state-of-the-art MOT results is not straightforward, as difficulties appear at different levels. Object detection leads to issues like deformations, total or partial occlusions, background confusion and illumination. More tracking-specific issues like identity switches or rapid direction changes are also hard to tackle [7].

In this Master Thesis we have focused on ByteTrack [4], which introduces a two-stage association mechanism under the

name of BYTE. BYTE runs a first stage focused on matching with high confidence detections followed by a second stage centered on the low confidence ones. ByteTrack is the result of a strong detector alongside this novel association method. In contrast, other tracking by detection solutions only keep high score detection boxes in a single stage.

Besides ByteTrack, other proposals have been emerging. For example, GHOST [8] is introduced as a system that tries to preserve ideas from old TbD methods, analyzing where these models failed and solutions to increase performance by means of appearance features and a simple motion model. StrongSORT [9] is another system, designed with the goal of, revising initially a previous classic tracker as DeepSORT [10], addressing problems of missing association and missing detection. For that, it is introduced: an appearance-free link model for doing association without appearance information, and a Gaussian-smoothed interpolation to address missing detections. StrongSORT tries to serve as a new baseline for comparison between different MOT systems —specifically in the TbD field.

We want to improve ByteTrack performance with our ByteTrackMask proposal. This is the point in which segmentation masks and tracking of those masks appears: we want to integrate them in the MOT system.

B. Segmentation

Another problem inside Computer Vision which has special interest is segmentation. We can define it as the classification of the pixels of an image [11]. If we are interested in classifying pixels with semantic labels, this is, in categorizing each pixel in a certain class, e. g. pedestrians, this is known as semantic segmentation. On the other hand, if we are interested in the distinction of the pixels of individual objects —for example, pixels that belong to each specific person—, then we talk about instance segmentation. Panoptic segmentation refers to a combination of both ideas [11]. As in our case we want to segment each object individually in order to track them with masks, the topic of interest for our work is instance segmentation.

There are multiple strategies and networks for performing segmentation, from convolutional neural networks or fully convolutional networks to encoder-decoder based ones. It was common to find segmentation alongside detectors, like in the case of Mask R-CNN [12]. This approach allows to detect objects in an image, and at the same time generate a high-quality segmentation mask for each instance. For that, based on Faster R-CNN detector [13], which had two outputs for classification and bounding box regression, Mask R-CNN adds a specific third branch for providing the mask.

The proposal that has a lot of interest nowadays, and that is the one that also caught our attention in this work, is Segment Anything Model [5], abbreviated as SAM. Based on ideas from Natural Language Processing, a model for promptable segmentation is created. One of the benefits of this method is its class-agnostic nature, this is, for mask generation the class is not taken into account. Therefore, we are able to use this model with no need of a specific training for MOT challenge datasets. SAM has been applied to many tasks with success, like zero-shot single point mask generation, instance segmentation, edge detection and text prompting.

The idea is, after checking the feasibility of using SAM for segmenting objects in images of MOTChallenge datasets, to integrate masks into ByteTrack. We have to take into account that in many cases bounding boxes contain information from other objects. But if we regard to the correct segmentation masks, objects can be distinguishable. Taking those masks for tracking may avoid confusions that ByteTrack could have with boxes.

C. Visual Object Tracking

Another problem related with object tracking is Visual Object Tracking or VOT [14], which refers to the idea of, given a reference frame and an object present in that frame, predict if it is present in subsequent frames of the video and its position. It is important to distinguish it from MOT, as we have to indicate in this case the position of the object in a first, reference frame. In addition, each object is managed independently, although some proposals that have been appearing in recent years [6], [15], [16] consider the possibility of working with multiple instances in parallel.

In this work we focus in segmentation-based approaches for VOT. In that field, there is a tracker based on transformers [3] leading the state of the art actually, AOT, which means Associating Objects With Transformers. AOT essentially includes [6]:

- An ID assignment mechanism for joint association and decoding of multiple targets. It includes an identification embedding to assign each target a different identification vector, and an identification decoder that predicts probabilities for all the targets.
- A Long Short-Term Transformer (LSTT) framework, which allows hierarchical multi-object matching and backpropagation.

Beginning with this idea, DeAOT [15] is also proposed as an evolution. It further improves AOT achievements and, compared to its predecessor:

- Rethinks hierarchical propagation from AOT, decoupling it in 2 parallel branches: an object-agnostic visual branch, and an object-specific ID branch.
- Includes a more efficient module for propagation, based on Single-Head Attention mechanisms, named Gated Propagation Module (GPM).

The interesting idea proposed in this series of trackers and their performance also in scenarios like the ones in MOTChallenge datasets led us to use them for segmentation-based tracking.

III. ByteTrack background

We give now some details of ByteTrack, which architecture is shown in Figure 5. It is a MOT system formed by many components, following the structure of Figure 2:

- Any object detector can be introduced into ByteTrack architecture.
- Before data association, a Kalman Filter [17] is applied for positions of tracks in the previous frame. Kalman Filter does an estimation of the position in which the tracks would be located in the actual frame.
- The association is based on a novel method proposed by the same authors [4], called BYTE, which has two association steps —or rounds— as mentioned in section II.

We will consider always the new frame to predict as $t+1$, and the previous frame as t. Taking this into account, when we have to make a prediction for frame $t+1$, detections are firstly computed —det_{t+1} in Figure 5. A confidence score is provided with each of them. Detections with a score over a certain threshold τ will be considered of high confidence and used at the first association step —det_hc_{t+1} in Figure 5. The rest of them, will be the ones of lower confidence, used at the second step —det_lc_{t+1} in Figure 5. In parallel to the detector, Kalman Filter takes object positions from frame t to estimate their location in $t+1$ —$track_{t+1}$ in Figure 5. Here is the point where the association begins:

- In a first step, high confidence detections det_hc_{t+1} and track estimations $track_{t+1}$ are matched by means of the Hungarian algorithm [18]. The detection-track pairs that were matched are associated and therefore we can consider those tracks as solved —P_1st_{t+1} in Figure 5. Unmatched tracks $u\_track\_1st_{t+1}$ are sent to the second step, and unmatched detections u_det_{t+1} are initialized as new tracks —P_new_{t+1} in Figure 5.
- In a second step, low confidence detections det_lc_{t+1} and the remaining tracks $u\_track\_1st_{t+1}$ are matched again using the Hungarian algorithm. Matched pairs are associated and therefore those tracks will be solved —P_2nd_{t+1} in Figure 5. No further action is performed for unmatched detections, and unmatched tracks are marked as lost. If after a number of frames N those tracks are not recovered with posterior matches, they are deleted.

Hungarian algorithm [18] is a well-known optimization method. As an input to it in ByteTrack, a matrix is provided, representing tracks in its rows and detections in its columns. In each position, the intersection over union (IoU) similarity metric is computed for the pair of bounding boxes (last track position, detection). Optimization process is performed afterwards, providing as a result the best set of possible matches between tracks and detections. It is not necessary that they all match, in fact, unmatched tracks and detections are also provided in separate lists.

Intersection over Union, also known as *Jaccard Index*, is a metric that compares the similarity between two arbitrary shapes [19], being computed as follows:

$$IoU(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

Where A and B represent the areas of each of the objects. IoU can take values between 0 and 1. In the worst case, this is, if A and B are disjoint, intersection would be zero, therefore, IoU would take a result of 0. In the best case, this is, if A and B are exactly equal, then intersection and union would have the same value, therefore IoU would take a result of 1. Also, this metric can be computed both for bounding boxes and segmentation masks. We have used IoU in this work in both situations.

The final list of tracks with a position in frame $t+1$ obtained with ByteTrack will be composed of: tracks matched in a first round P_1st_{t+1}, unmatched detections from first round that are initialized as new ones P_new_{t+1}, and tracks matched in a second round P_2nd_{t+1}. They will be subsequently used in Kalman Filter for the next frame estimations.

IV. Our approach: ByteTrackMask

Taking as reference ByteTrack, we propose to modify data association, adding a third round after the first two associations as shown in Figure 6.

Originally, tracks that were not matched in the second association step, $u\_track\_2nd_{t+1}$ in Figure 6, were directly marked as lost. As a result, we could not be tracking an object which is present on the video. In that situation, we would like to keep tracking the object, but after the two first rounds we do not have any extra detection that we can use for matching. Here is where segmentation masks-based tracking approach appears: all tracks still lost from the second association step are sent to a new module —3rd round estimation with masks, as shown in Figure 6— that will work with them and their masks.

Therefore, with this approach we want to avoid losing tracking of objects which are still present in the video, either permanently or temporally. On posterior frames, we can keep those objects tracked either with our third round or in the first/second rounds from ByteTrack.

This new third round of the association using masks is represented in Figure 7, which, at the same time, is divided in sub-parts or steps. Firstly, we need to obtain a segmentation mask for the object —mask extraction block from Figure 7— , which can be generated with SAM or from the previous prediction if a tracking with AOT was already running for that object. Once we get a mask, the prediction step is performed, in which AOT estimates the new position of the object regarding its mask —mask prediction block from same Figure 7. A last control step is necessary to check if the mask is coherent —track termination check in Figure 7. If so, the new position of the bounding box is computed and assigned to the track and therefore it will be rescued from being lost.

A. Mask extraction

For the mask extraction process, given an unmatched track after second round in frame $t+1$, we firstly check if there is a previous mask available in frame t. In other words, we check if there is an AOT instance active for tracking the corresponding mask for that object, since we will only have them active when performing tracking in this third round. If so, we take the mask from frame t for the next block —Get AOT prediction

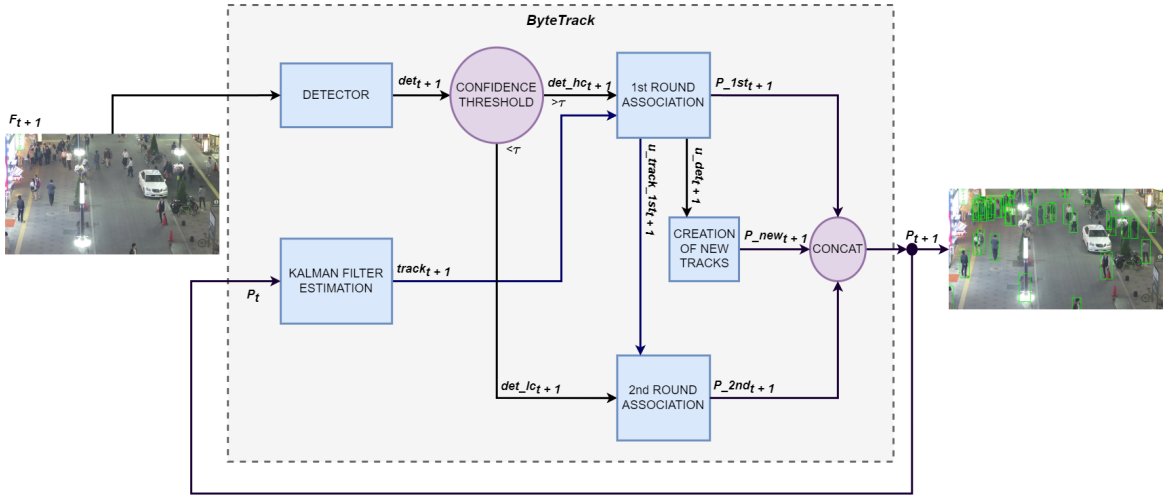

Fig. 5. Architecture of ByteTrack with the main components of interest for our work [4].

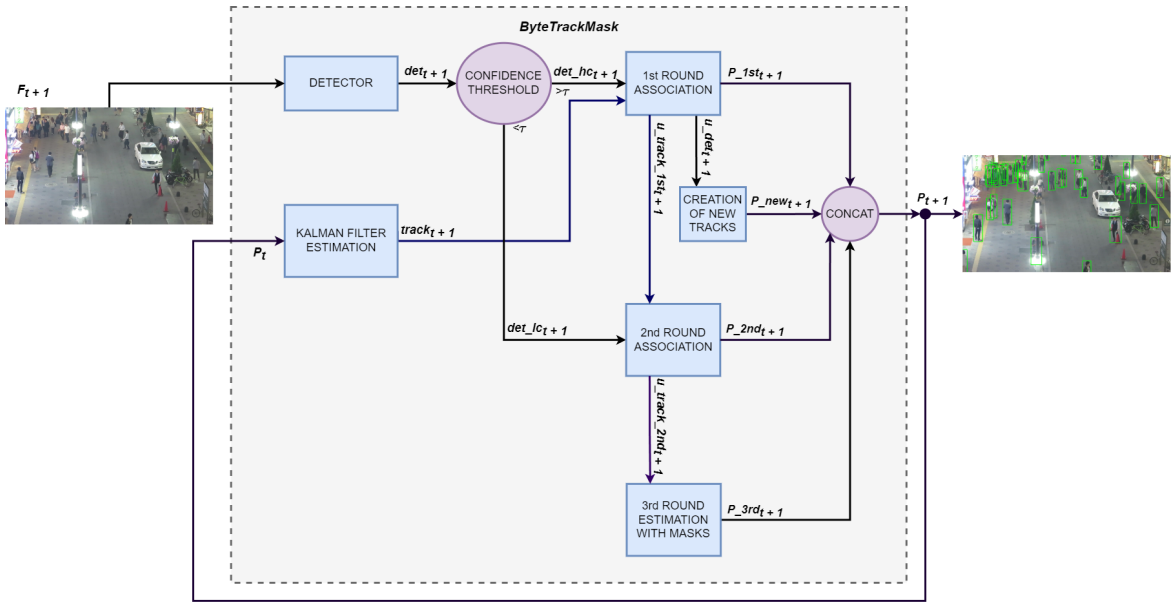

Fig. 6. Architecture of our approach —which we name ByteTrackMask— proposed to evaluate.

in Figure 7. If not, this is, if we have just lost the track from first/second round, we will need to generate a mask —mask generation in Figure 7.

For this mask generation process we need to know how SAM works. It follows an encoder-decoder strategy with multiple components, shown in Figure 8:

- An image encoder allows us to create an embedding from the input, that only needs to be run once per image in order to begin prompting.
- A prompt encoder allows to process the prompt that is given in order to generate the mask. This prompt, as can be seen in Figure 8, can be simply a point at which we know the object is present —or a set of points—, a bounding box surrounding it, or even text. Apart from this, there is the possibility of providing a mask, which is embedded in a different way —using convolutions, as also shown in Figure 8.
- Mask decoder is used to map all the embeddings into a mask, which is given as an output of the model.

It is worth noting that SAM allows to generate three masks per prompt, to represent the whole object, or a part or a subpart of the object. Each of the masks is accompanied by a confidence score provided by SAM, as an estimation of the IoU [5].

Our mask generation options are detailed in Figure 9. Figure 10 shows the full process that we propose. We can directly provide the bounding box assigned to the track in frame t as a prompt to SAM. A more refined second alternative consists on extracting a set of masks using only one point as a prompt.

As we can see in the example of Figure 9, a bounding

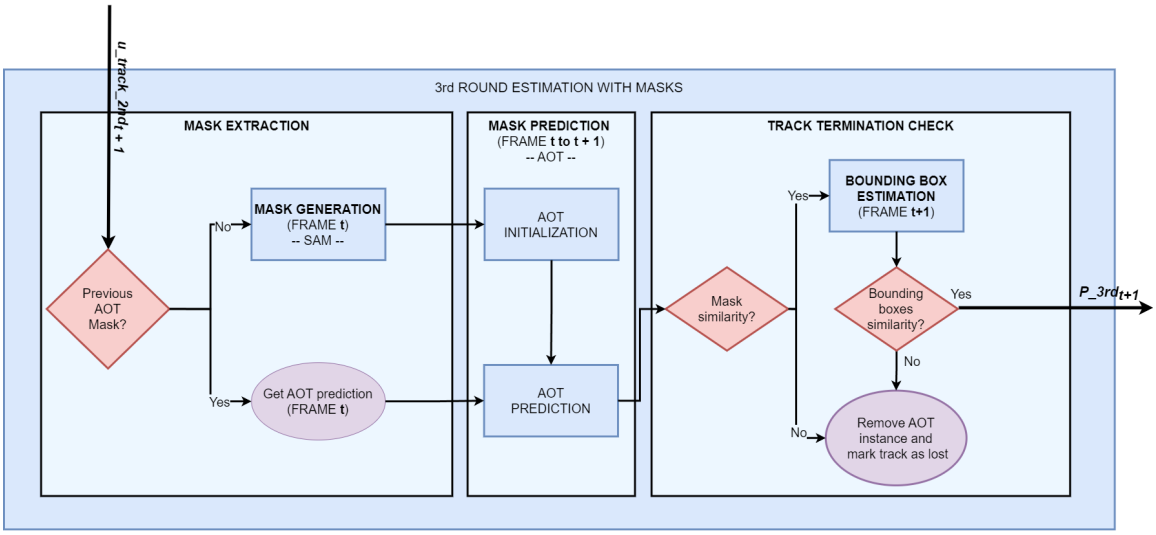

Fig. 7. Details of the third round proposed for ByteTrackMask in order to try to achieve improvements.

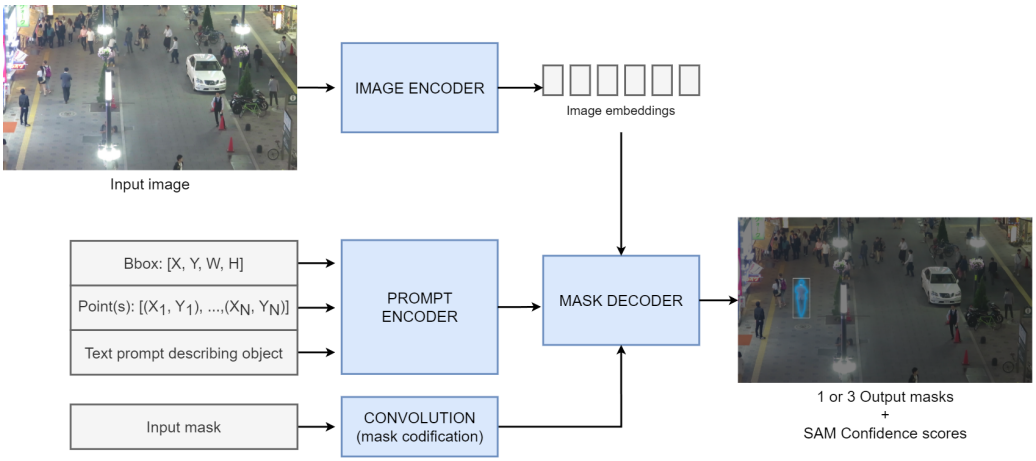

Fig. 8. Architecture of SAM including the possible prompts that can be an input to the model [5].

Fig. 9. Example of a mask generated with SAM using the bounding box as a prompt (left) or a correct point located inside the object (right).

box can introduce information from other objects as well, making the segmentation process difficult, something that does not happen when providing a correct point from the object. Moreover, frame t represents the moment before losing the object in first/second round. In that situation, object can be occluded —like in the scene shown in Figure 9— and it may be hard to extract the correct mask for an architecture like the one from SAM and a bounding box prompt.

Centering in the second option of the point prompt, we need to provide a point inside the object to generate a correct mask —like the shown in Figure 9. For that, we consider a point grid, having as reference the bounding box that we would use as a prompt in the first option. All three possible masks are extracted for each point of the grid.

After SAM execution, we get a set of candidate masks —for a $M \times M$ grid, $M \cdot M \cdot 3$ masks—, also in case we use the alternative of generating three masks from a single bounding box —although in that option only 3 would be obtained. Therefore, we need a filtering process in order to select the most correct mask for each object. For that, we propose the following filters, that are also shown in Figure 10

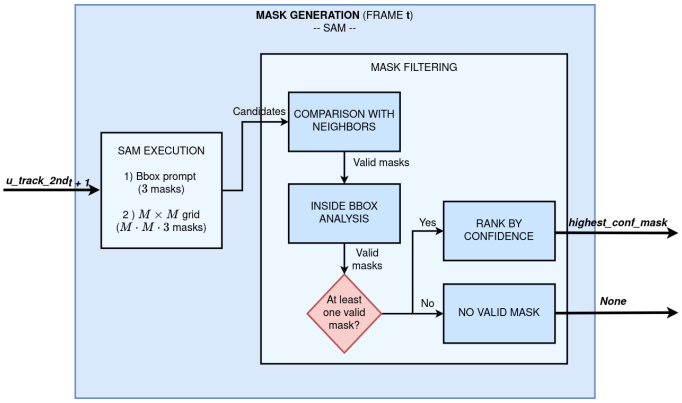

Fig. 10. Detail of the mask generation process inside our third round proposal for ByteTrackMask.

inside Mask Filtering block, in this order:

1) **Comparison with neighbors**: select neighbor tracks and generate a mask using SAM and their bounding boxes —from frame t— as prompts. This will give an idea of the position of each of the neighboring objects. With that, we can filter out those candidate masks that overlap with any of the neighbors over a threshold α. Overlapping measure will be Intersection over Union (IoU) between each candidate mask and a frame mask representing a concatenation of the masks of each of the neighbor tracks.
2) **Inside bbox (bounding box) analysis**: we want to ensure that candidates have the most pixels of the mask within the reference bounding box —the one that indicates the position of the track in frame t. Proportion of pixels inside the bounding box must be greater than a certain value β. We are not interested in segmenting objects mostly present outside that rectangle.
3) **Rank by Confidence**: in case more than one mask passes the previous filtering process, we return the mask with the highest SAM confidence score. We introduce the previous filters since SAM does not take into account those conditions in the calculation of the confidence score, thus eliminating masks that we do not want to be selected.

After this process, we can have two situations. In the first one, a mask is finally selected —with the rank by confidence—, case in which we advance in the process of this third round. The second one occurs when no mask is able to pass the two proposed filters —no valid mask, shown in Figure 10—, a case in which we will not continue trying to save that track as we do not have a mask that we consider good enough to associate it.

B. Mask prediction

We will only achieve this stage if, after mask generation step, a mask has been successfully generated by SAM or if we have a mask predicted with AOT in previous frame —this is, we have an active AOT instance running for that object.

If we are on the first situation, this means that we have not began tracking with AOT. Therefore we need to initialize a new instance of the tracker providing the mask obtained in mask generation block as reference —AOT initialization, as we show in Figure 7. After initialization, or having an active instance of the tracker, we will make the prediction for the new frame $t+1$ —AOT prediction, as we show in Figure 7.

C. Track termination check

Once mask is predicted, we need to have some control in order to avoid letting non reliable AOT predictions go by and stop tracking process in that situation. For this, we propose a third stage in which two filters are introduced to check consistency between two consecutive frames. Both are represented in Figure 7:

- **Mask similarity**: two masks between two consecutive frames, although different, should be similar. Therefore, we can measure the IoU between them and keep masks which IoU is over a certain threshold γ.
- **Bounding box similarity**: we can apply the same process that we do with masks to bounding boxes: they should also be similar between consecutive frames. In particular, the bounding box that was assigned for the track in the previous frame t, and the one that we would generate for the track in $t+1$ taking into account the predicted mask. Similarity is again measured with IoU, that should be above a δ threshold.

If track is not able to pass any of the filters, it will be marked as lost and not maintained in this frame. In other case —both filters passed— we would assign the new bounding box as the position of the track in $t+1$ —Bounding box estimation in Figure 7. As the system works with bounding boxes, position needs to be given to the track. This is done taking into account three elements:

- The last position —bounding box in frame t.
- The mask generated for frame t.
- The mask estimated for frame $t+1$.

We managed different options for updating the bounding box. They are shown in Figure 11 for a problematic situation in which we have a partial occlusion on the object of interest. Firstly, the most straightforward, would consist on assigning the bounding box that surrounds the mask predicted in frame $t+1$ —top-right image in Figure 11. However, bounding boxes are expected to represent where the object is located even in their occluded parts. As a result, if the object is partially visible —for example, just a head—, the bounding box would be different with respect to how it should be.

An intermediate step that was performed consisted on moving the center of the last position onto the center of the mask estimated for frame $t+1$ —bottom-left image in Figure 11. However, we would have a problem due to a similar reason than in the previous case: if the object has occlusion, we would displace the bounding box to the center of the visible part of the object, which may not coincide with the real center of the object —maybe occluded.

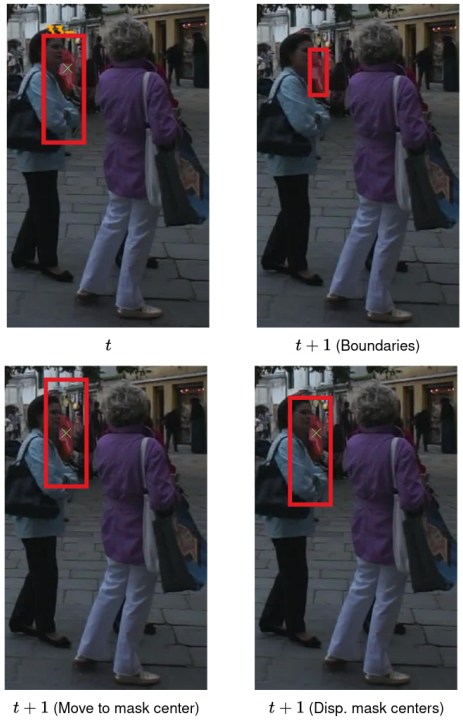

Fig. 11. Detail of the bounding box estimation process alternatives. If we want to perform a more realistic displacement, basing it on the difference between the mask centers from previous and actual frames is the option which provides better results.

We need to estimate better how the object displaced, and for doing so we propose to move the center of the bounding box at frame t following the displacement between the center of the masks of frames t and $t+1$ —bottom-right image in Figure 11. In this way, even if the object is occluded, movement of the bounding box would be more realistic than in previous proposals.

V. Experiments and results

Regarding the experimentation made for this Master Thesis, we will show our main results for the proposed model, as well as an ablation study performed with different components. At the same time, interesting results that were extracted from a preliminary study stage will be shown.

A. Datasets

All testings made with our integration in ByteTrack were addressed in MOTChallenge datasets. MOTChallenge is a relevant challenge introduced in order to standardize MOT methods evaluation. In this work, we have used 2 datasets: MOT17, based on previous MOT16 [20] with improved ground-truth annotations; and MOT20 [1], the most difficult as it is based on crowded scenarios. Besides these two, another dataset named Multiple Object Tracking and Segmentation (MOTS) [21] was used, as it contains a ground-truth with segmentation masks. All these datasets have two subsets: one for training and another for testing. However, in ByteTrack implementation [22] we can find scripts that allow us to split MOT17 and MOT20 training sets into two, one part for training and one part for validation.

TABLE I
Information of MOT17 and MOT20 validation subsets videos: length —number of frames—, resolution and density —average number of objects per frame.

DATASET	VIDEO	LENGTH	RESOLUTION	DENSITY
MOT17	MOT17-02	300	1920 x 1080	31
	MOT17-04	525	1920 x 1080	45,3
	MOT17-05	418	640 x 480	8,3
	MOT17-09	262	1920 x 1080	10,1
	MOT17-10	327	1920 x 1080	19,6
	MOT17-11	450	1920 x 1080	10,5
	MOT17-13	375	1920 x 1080	15,5
MOT20	MOT20-01	214	1920 x 1080	46,32
	MOT20-02	1391	1920 x 1080	55,62
	MOT20-03	1202	1173 x 880	130,42
	MOT20-05	1657	1654 x 1080	194,98

In our case, most of the experiments were performed in MOT17 [20] validation half, although some extra evaluations were done in MOT20 [1] validation half to have another reference. It is important to mention that for using test set we would need to compute the results and upload them onto an evaluation server with limited amount of submissions per proposal. Take into account that annotations of this set are not publicly available. For this work, our goal was to check the performance of our module and different variants of it. Therefore, submitting results onto MOT servers would be impractical and at the same time not purely correct, as we could start to compare variants over the test set. That comparison would be expected to be done in validation subsets.

Table I shows resolutions and lengths of each of the videos of each validation subset. We can see in this table the differences between MOT17 and MOT20 in number of videos, frames and objects per frame, expecting more crowded scenarios in MOT20 dataset. Resolutions vary in each video, but it is common to have 1920 x 1080.

For evaluating how well a method performs in both datasets, many metrics are proposed. In this work we will focus on MOTA, IDF1, FP and FN.

FP and FN stand for the number of false positives and false negatives. MOTA is the abbreviation of Multi Object Tracking Accuracy, and it is one of the most common metrics used to evaluate how well does a tracker perform [1]. The following equation defines how it is computed:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \quad (2)$$

Where FN and FP stand for false negatives and false positives, respectively, IDSW represents the number of identity switches that were produced and GT represents the number of ground truth objects. In addition, t indicates the frame. As we can see, MOTA is obtained dividing the sum of false positives, false negatives and identity switches by the total amount of ground truth objects per frame, and subtracting that value from 1. Therefore, this metric can take values as a percentage in the

interval $(-\infty, 100]$, having the possibility of a negative MOTA in case the number of errors is too high [1].

IDF1 metric stands for ID F1 score, and it represents the ratio of correctly identified detections over the average number of ground-truth and computed detections. IDF1 allows to rank trackers in a single scale balancing ID precision and recall [23]. Its equation can be represented as follows:

$$IDF1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \tag{3}$$

Where IDTP represents the number of true positive IDs, IDFP is the number of false positive IDs and IDFN is the number of false negative IDs. This metric can take percentage values in the interval [0, 100]. In the worst case —0 true positives— IDF1 would take a result of zero, and in the best case —neither false negatives nor false positives—, a perfect value would be achieved.

B. Implementation details

Before showing the results, we present some implementation details. All the implementation was developed in Python language, taking into account that we have libraries like PyTorch, optimized for performing Deep Learning operations, or other libraries that we know from many subjects of this master like NumPy, OpenCV, Matplotlib and Torchmetrics.

Python was also the language used in the different GitHub repositories that we needed for implementing our module. We include a detailed list of them in this section:

- **ByteTrack** [22]: the code of the MOT system is taken as reference for our work.
- **AOT Series Frameworks** [24]: it contains the implementation of AOT, including DeAOT version.
- **Segment Anything** [25]: the implementation of Segment Anything Model, other of the components that was integrated in our work.
- **MOTS tools** [26]: the implementation of some tools that allow us to evaluate performance in the Multiple Object Tracking and Segmentation (MOTS) challenge.
- **COCO API** [27]: it is a repository containing APIs in different languages for managing COCO dataset annotations [28]. COCO is a very famous dataset inside visual recognition, that can be used for tasks such as object detection, captioning or object segmentation. In particular, with this API we have the possibility of saving and loading segmentation mask annotations to or from a file, thanks to its encoding and decoding tools.
- **TrackEval** [29]: it contains code for, given a ground-truth and corresponding results files from a MOT system, evaluate performance with a wider set of metrics than the one that a direct execution of ByteTrack has.

Experimentation is normally expensive, therefore, we needed access to GPUs in order to perform general purpose operations thanks to CUDA [30], the parallel computing platform by NVIDIA. We were able to get access to two GPGPU computation servers from Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS) at the University of Santiago de Compostela: `ctgpgpu6` and `ctgpgpu11`. Normally, we used the first one, which has the following properties:

- 2 processors Intel Xeon Silver 4214.
- Four GPUs:
  - One NVIDIA Quadro P6000 with 25 GB.
  - One NVIDIA Quadro RTX 8000 with 50 GB. This is the GPU we commonly used in all experimentation. In particular, for the results shown in this report, it was the GPU in which those experiments were run.
  - Two NVIDIA A30 with 25 GB each.
- CentOS Linux Operating System, in version 7.7, including CUDA driver in version 12.2.

Regarding details from the execution, we have taken the weights of ByteTrack ablation model —available in their GitHub [22]. It was trained using CrowdHuman and half of the training set of MOT17, as the other half was used for validation. YOLOX [31] was selected as the object detector, with YOLOX-X backbone. Regarding ByteTrack specific parameters, they are aligned with the default values used in their proposal, including confidence τ for classifying detections into high and low confidence, which was set to 0,6. The number of frames N that a track is preserved after being lost and before setting it as deleted is also included and it is set to 30.

For SAM, weights were downloaded from their GitHub as well [25], in particular for the largest backbone size (ViT-H), being trained in the large 11 million images dataset SA-1B —proposed also by their authors [5].

In the case of AOT, weights were taken again from GitHub [24], and they were the ones of the DeAOT model which yielded best results, this is, DeAOT-L with SwinB transformer backbone. It was trained in YouTube-VOS [32] and DAVIS [33] datasets after a pre-training with many static image datasets.

Regarding our thresholds, we have tried to remain with the same global value of 0,9 for β, γ and δ, being inverted in the case of α as it is used in the comparison with neighboring masks and we want to check if there is a minimum overlap with them —therefore, α is set to 0,1. We have generated SAM masks using the grid approach, following the idea proposed in Section IV for implementing the module. Masks were generated with a 3 x 3 point grid, having a total of $3 \cdot 3 \cdot 3 = 27$ masks for each case.

We have also not considered the inference tricks introduced in ByteTrack. Some later works [8], [9] compared their results with ByteTrack, but they showed different scores for that system with respect to the ones originally presented in [4]. This occurs as this MOT system uses an offline interpolation that fills lost frames of tracks, which leads to an extra increase in the results. Moreover, ByteTrack introduces a custom parameter setting depending on the videos from the MOTChallenge datasets, in order to further improve its results —as an example, value N was changed from the default 30 depending on the video. We follow in our experimentation the

TABLE II
AVERAGE IOU BETWEEN THE MASKS OBTAINED FROM BOUNDING BOXES OF DIFFERENT ASSOCIATION ROUNDS OF BYTETRACK AND CORRESPONDING GROUND-TRUTHS PAIRED THROUGH HUNGARIAN ALGORITHM.

VIDEO	IoU 1st	IoU 2nd	IoU 3rd
MOT17-02	0,73	0,34	0,31
MOT17-05	0,64	0,19	0,35
MOT17-09	0,72	0,34	0,38
MOT17-11	0,76	0,24	0,36
TOTAL	0,72	0,29	0,34

TABLE III
AVERAGE IOU BETWEEN THE MASKS OBTAINED FROM BOUNDING BOXES THAT WOULD BE USED IN A THIRD ROUND OF BYTETRACK AND CORRESPONDING GROUND-TRUTHS PAIRED THROUGH HUNGARIAN ALGORITHM. MASKS WERE GENERATED WITH DIFFERENT GRID SIZES, AND HIGHEST IOU WAS SELECTED.

VIDEO	3x3	5x5	15x15	30x30	Automatic
MOT17-02	0,65	0,67	0,69	0,7	0,65
MOT17-05	0,48	0,53	0,55	0,56	0,51
MOT17-09	0,53	0,57	0,61	0,62	0,57
MOT17-11	0,52	0,54	0,57	0,58	0,55
TOTAL	0,55	0,59	0,61	0,62	0,57
TIME	1h 11m	1h 15m	1h 52m	4h 5m	1h 48m

good practices of these works [8], [9] and we do not take into account neither tuning nor interpolation.

C. Preliminary studies

We began experimentation by evaluating the use of SAM in situations from a challenge like MOT, using for that MOT17 dataset. We could evaluate the quality of generated masks thanks to MOTS dataset, since it has ground-truth masks. Moreover, videos and objects from this dataset are also present in MOT17.

A first evaluation was made based entirely in the use of SAM prompted by bounding box coordinates. We measured the mask quality that we could extract from different detections of ByteTrack. There are detections of high confidence which are used in the first association step, whilst others of lower confidence were taken for the second step. When thinking of going onto a third round, having to generate a mask that comes from a track which last detection was taken on first or second round could make a difference, and that is what we tried to check in this experiment.

We executed ByteTrack normally over MOT17 and extracted masks for detections with SAM. However, not all objects from MOT17 have a ground-truth represented in MOTS. Therefore, to perform the most correct as possible evaluation, we needed to match detections with ground-truth. For that, we took the set of detections from each round, and we employed a matching using the Hungarian algorithm [18] between detections and MOT17 ground-truth bounding boxes. At the same time, there is a correspondence between MOT17 and MOTS IDs. As a result, we could match detections with MOT17 ground-truth bounding boxes and MOTS masks. That is how we were able to do comparisons based on IoU, in particular, on the mean mask IoU of all objects available through all MOTS frames. In Table II we show the average results obtained per each video we could evaluate and each round.

We also added to this table a third column, which represents the average IoU of masks extracted for objects that would go to a third step evaluation with our new module. We did this in order to get a reference of mask quality in that point as well. In this case, we needed to use bounding boxes and ground-truth from the last frame in which the object was tracked.

Results show how there is a clear influence on the detection quality. From the first round, average IoU is over 70%. We have to take into account that IoU computes similarity over all individual pixels. Even lower values of IoU can be representative of accurate masks, as a difference in any pixel makes IoU lower. However, in 2nd round detections we move onto specially low values: an average of less than 30% —less than a half of the first round average. Candidates that would go into third round seem to be in a similar situation, having their mean value around 34%.

Considering these results, specially from the bounding boxes related with third round —those for which masks are generated in ByteTrackMask—, we thought on alternative ways to generate masks. This is how we ended up testing with point prompts. For that, we considered grids of different sizes, generated the three possible masks with each grid point as a single prompt, and evaluate the best IoU for each object —that of the mask with the most overlap with ground-truth. With these results we can know how much quality we are able to improve with this prompting, although mask selection is not performed unlike our module for ByteTrackMask. Table III shows us the results of testing different grid sizes.

We have tested two types of grid sizes: a fixed grid size per bounding box —for which we have considered different values—, and an automatic grid defined regarding in this case to ground-truth masks, to check whether it is better to adapt the grid depending on the expected size of the mask. As we can see in Table III, there are not too much differences in global IoU between using a fixed 3×3 grid or a 30×30 grid —less than 0,1. However, in computation times differences start to appear with the biggest grid sizes, therefore, it is not worth generally to try to use more points. Automatic sizing does not give a big improve as well, needing also more time. This result, taking into account the comparison with the results using bounding boxes in Table II, made us to select a fixed 3×3 grid for our ByteTrackMask approach.

D. Ablation study

An ablation study was carried out in order to ensure that the different modules that we introduced onto our ByteTrackMask contributed to improve performance. In this way, we achieve the results which are shown in Table IV.

First row of the table represents our baseline model, being our reference result. The next two rows represent results of

TABLE IV
RESULTS OF ABLATION STUDY OF BYTETRACKMASK. FIRST ROW REPRESENTS OUR BASELINE. MG, MS AND TERM STAND FOR MASK GENERATION, MASK SELECTION, AND TERMINATION, RESPECTIVELY. SC STANDS FOR SAM CONFIDENCE, AND FILT FOR FILTER.

MG	MS	TERM	MOTA ↑	IDF1 ↑	FP ↓	FN ↓
-	-	-	76,7	79,4	**3355**	9066
Bbox	SC	Yes	76,9	78,9	3809	8490
Bbox	SC+Filt	Yes	76,8	79,4	3462	8892
Grid	SC+Filt	No	70,5	74,9	7909	**7807**
Grid	SC	Yes	72,7	77,5	6103	8454
Grid	SC+Filt	Yes	**77**	**79,5**	3416	8804

using bounding boxes for mask generation instead of a point grid, achieving improvements over the baseline in MOTA and also in false negatives. The difference between them is that in one case (row 2) we directly take higher confidence masks, and in the other one (row 3) we apply our filtering step that in this case is not contributing positively in MOTA —take into account that filtering is made in this case for only three masks. However, we have to consider that the model that is based only on SAM confidence, although it provides a bigger drop in false negatives, has also a significant increase in false positives. That increase does not allow to achieve a better result despite reduction in false negatives. As a result, we desire to not introduce too much false positives at the same time that the number of false negatives is decreased.

This idea is better accomplished when regarding to grid-based segmentation masks, with which we are able to further improve MOTA. We have tested in this case without doing termination check (row 4) or the selection filtering (row 5) which in the case of bounding boxes prompt was not helpful. However, both experiments resulted in a huge drop in performance with respect to the baseline, although getting a considerable decrease of false negatives —reaching the lowest values of all models. Once again, a big increase in false positives was obtained as well, causing MOTA drop. In the case of bounding boxes, we were generating 3 masks only, therefore, selection of masks was not crucial as in this case, in which we are considering a 3×3 point grid, having 27 masks in total. Here, removing that filtering stage is really harmful compared to the case of bounding boxes. Consider also that with a point prompt, a mask can perfectly cover a bigger amount of the image as no boundaries are provided like when using a bounding box.

Our final model is highlighted in gray. As we can see, all the metrics are improved except for the false positives, something consistent as we try at the same time to remove false negatives. In this case, all the components are included on the model and help to achieve a gain in performance, justifying their integration.

E. ByteTrackMask main results

We finally introduce the main results obtained in ByteTrackMask. In table V, we can see a direct comparison of the baseline ByteTrack results with ByteTrackMask approach added by ourselves, video by video and in total for different metrics, in MOT17 and MOT20 datasets. Take into account that we are using the same weights in evaluation with both datasets. Nevertheless, the case of MOT20 is useful to confirm the conclusions that we can extract from the experimentation over MOT17 dataset.

In green color results in which we are improving are highlighted. As we can see, there is a MOTA global improve over the ByteTrack baseline, as well as in the case of false negatives, which are reduced. Focusing on MOT17, we are able to reduce the total amount of false negatives in most of the videos, having interesting increases in MOTA: in video MOT17-04, in which an improve of 0,7 is made, and in MOT17-09, with a 0,6 MOTA improve. IDF1 is also improved, in one tenth. If we attend to our MOT20 evaluation, we can see a bigger improvement in all metrics compared to MOT17. Specially, a higher number of false negatives are reduced, which makes sense as in this dataset, more crowded than MOT17, the baseline number is really higher and we are able to reduce it by more than 6000. All the videos are able to reduce false negatives, having also interesting MOTA increases in two of them: MOT20-03 by 0,5 and MOT20-05 by 0,7. Total IDF1 has also a higher increase with respect to MOT17 experiment: 0,6.

At the same time, the number of false positives increases globally in both datasets, however, this is something expected: the more examples are introduced in third round, the more false positives are likely to appear. In any case, the decrease on false negatives has more weight than the increase of false positives, something that explains the increases on MOTA metric in both datasets.

Regarding computation times, we have to mention that we are not looking for real-time in this work, therefore, code was not optimized at all. It was expected that experimentation took more than baseline. In fact, ByteTrack baseline in MOT17 ran in our hardware at 12 FPS, meanwhile our ByteTrackMask ran at only 2 FPS. MOT20 has a similar frame drop, from 11 FPS in baseline to less than 1. However, our main goal in this work was to analyze if ByteTrackMask was helpful for improving performance in terms of the metrics shown in Table V, something that we can confirm.

We also show in this report the behavior of ByteTrackMask when it rescues an object that would be normally lost. Figure 12 shows examples from different videos. We show how we are able to mantain tracks active in a third round whilst ByteTrack would lose them normally. Even in some cases objects are tracked again in first or second round after using our third round. Thanks to this behavior, false negatives are reduced and consequently, MOTA is increased.

VI. CONCLUSIONS

In this work we propose ByteTrackMask, a MOT system based on ByteTrack [4], aiming to achieve improvements over it. ByteTrackMask includes a novel third association round for tracks that do not have a detection matched after ByteTrack first and second rounds. In this new step, segmentation masks are generated for the immediately previous frame with SAM

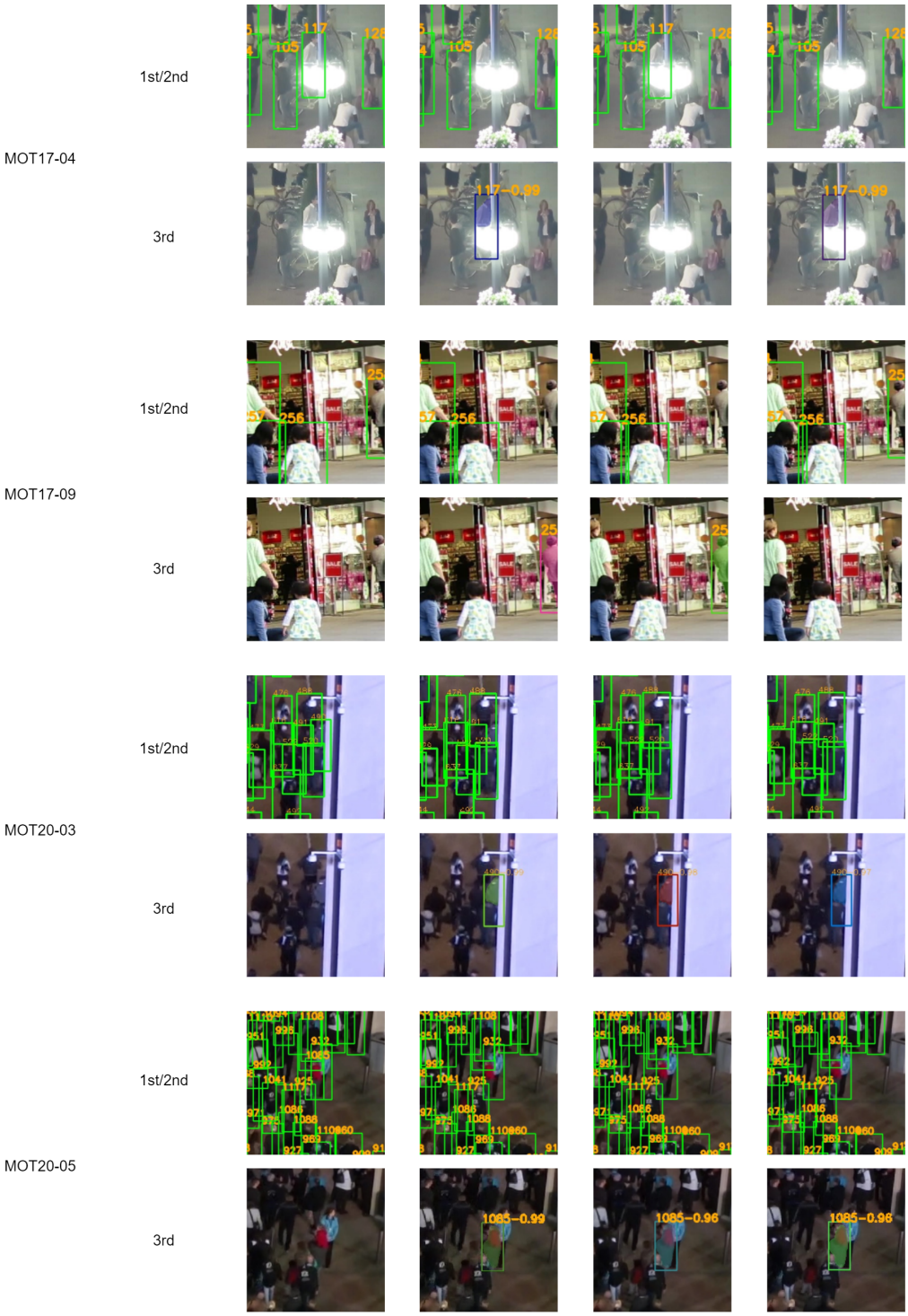

Fig. 12. Examples of ByteTrackMask rescuing tracks that would be lost in normal ByteTrack behavior. For each video, both ByteTrack normal output of first and second round and the third round output are presented separately.

TABLE V
COMPARISON PER VIDEO BETWEEN BYTETRACK BASELINE AND OUR BYTETRACKMASK FOR MOT17 AND MOT20 VALIDATION DATASETS.

VIDEO	ByteTrack				ByteTrackMask			
	MOTA ↑	IDF1 ↑	FP ↓	FN ↓	MOTA ↑	IDF1 ↑	FP ↓	FN ↓
MOT17-02	53,6	58,4	1024	3510	**53,7**	57,6	1039	**3483**
MOT17-04	88,1	90,9	1287	1580	**88,8**	**91,5**	1320	**1369**
MOT17-05	77,1	76,4	144	598	77,1	76,4	144	598
MOT17-09	83,2	77,9	10	464	**83,8**	**78,2**	11	**444**
MOT17-10	70	68,6	303	1446	69,9	68,6	312	**1445**
MOT17-11	69,8	70,8	512	842	69,8	70,8	515	**839**
MOT17-13	77,3	83,4	75	626	77,3	83,4	75	626
TOTAL - MOT17	76,7	79,4	3355	9066	**77**	**79,5**	3416	**8804**
MOT20-01	67,3	76,4	1203	2282	66,8	75,3	1277	**2260**
MOT20-02	67,9	58,5	7197	21903	66,6	58,4	8649	**21631**
MOT20-03	68,9	75,9	18455	41417	**69,4**	75,9	19127	**39787**
MOT20-05	62	65,4	47649	72963	**62,7**	**66,6**	49599	**68638**
TOTAL - MOT20	65,1	67,9	74504	138565	**65,5**	**68,5**	78652	**132316**

class-agnostic segmentation method [5] and predicted for the following ones with AOT visual object tracker [6], [15]. Segmentation masks are used as they are more easily able to delimit different objects, meanwhile bounding boxes can contain information from various objects inside them. This third association step is designed to avoid losing tracks that represent an object that is still present on a video. In this way, we avoid them from counting as false negatives.

Results have shown how we are able to achieve improvements in MOT17 and MOT20 challenges, using for that the corresponding validation subsets. In particular, with this approach we are able to decrease false negatives, providing as a result increases in metrics like MOTA and IDF1. All in all, this work allowed us to confirm that segmentation masks are helpful for a MOT system like ByteTrack when it comes to rescuing tracks that would be lost in the normal functioning of it, validating our main hypothesis.

VII. FUTURE WORK

Although we were able to confirm initial hypothesis, more work needs to be done. Firstly, taking into account visual information we could extract, we suspect that improvements achieved could be higher by using more refinements. A first open line of work would be to adjust the proposed system. For example, checking visual outputs we have seen how in many cases objects do not pass any of the filters of our termination control because masks are correct but small. As a consequence, a slight difference in a few pixels can lead to a big variation in IoU. An alternative or refined version of the metric that takes this into account could be proposed. In addition to this, optimizing the implementation to try to search for real-time performance can be a possibility.

Moreover, we have presented a new module for the third association step with the idea of, when a new track is lost, generating the mask using the last reference frame before missing it. Nevertheless, in many cases last frame is not the best reference for objects —even with point prompts— as they may be occluded. Another open line of work would consist on proposing an alternative module. In fact, we are planning on the possibility of a dual-multi-object-tracking that maintains in parallel bounding boxes and segmentation masks of all the objects during all time. In this way, when a track is lost, we can have a mask reference that has been tracked during previous frames.

Lastly, a third open line of work could be centered on changing the models used for segmentation and segmentation-based tracking. SAM and AOT have also their problems. For example, AOT tends to degrade tracking quality with time. Moreover, we are not always able to get the correct mask with SAM despite trying to be precise in the prompt. Other proposals, maybe already presented or that will appear in the future, may help to achieve a better performance. Another possibility would be to modify the architecture of these components, which could lead to a training over them. In any case, taking into account the positive results of this work, we have many options for going on in the future.

ACKNOWLEDGEMENTS

I would like to thank my family for being a key support during all these months which I spent working for this Master Thesis. Specially, my parents and my sister, having a special memory for the ones that left us in the last years.

In addition, thanks to CiTIUS (Centro Singular de Investigación en Tecnoloxías Intelixentes) for providing me tools and space for working during this time and for the future. Also thanks for having me there with the *Concurso público de méritos, para la contratación de personal investigador que se inicie en la labor investigadora dentro de los correspondientes programas científicos de los centros singulares de la USC*, together with the University of Santiago de Compostela. Of course, thanks to my supervisors Manuel Mucientes and Victor Brea for all the support and help which was crucial, as well as to Lorenzo Vaquero, part of the team, who has given me a lot of help for continuing with the work.

Finally, thanks to my friends, in particular, to the colleagues from this Master, to my colleagues at CiTIUS and on the other hand, to all friends I have from university and before; thank you for being there all this time!

REFERENCES

[1] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A benchmark for multi object tracking in crowded scenes. *CoRR*, abs/2003.09003, 2020.

[2] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[4] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. *CoRR*, abs/2110.06864, 2021.

[5] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *CoRR*, abs/2304.02643, 2023.

[6] Zongxin Yang, Yunchao Wei, and Yi Yang. Associating Objects with Transformers for Video Object Segmentation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[7] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[8] Jenny Seidenschwarz, Guillem Brasó, Ismail Elezi, and Laura Leal-Taixé. Simple cues lead to a strong multi-object tracker. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13813–13823, 2022.

[9] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. StrongSORT: Make DeepSORT Great Again. *IEEE Transactions on Multimedia*, 25:8725–8737, 2023.

[10] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pages 3645–3649. IEEE, 2017.

[11] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image Segmentation Using Deep Learning: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3523–3542, 2022.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988. IEEE Computer Society, 2017.

[13] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 91–99, 2015.

[14] Matej Kristan et. al. The tenth visual object tracking VOT2022 challenge results. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors, *Computer Vision - ECCV 2022 Workshops - Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part VIII*, volume 13808 of *Lecture Notes in Computer Science*, pages 431–460. Springer, 2022.

[15] Zongxin Yang and Yi Yang. Decoupling Features in Hierarchical Propagation for Video Object Segmentation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[16] Yuanyou Xu, Zongxin Yang, and Yi Yang. Integrating Boxes and Masks: A Multi-Object Framework for Unified Visual Tracking and Segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9738–9751, October 2023.

[17] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[18] Mike B. Wright. Speeding up the hungarian algorithm. *Comput. Oper. Res.*, 17(1):95–96, 1990.

[19] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 658–666. Computer Vision Foundation / IEEE, 2019.

[20] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016.

[21] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. MOTS: multi-object tracking and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 7942–7951. Computer Vision Foundation / IEEE, 2019.

[22] Yifu Zhang. Bytetrack. https://github.com/ifzhang/ByteTrack, 2024.

[23] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, volume 9914 of *Lecture Notes in Computer Science*, pages 17–35, 2016.

[24] Zongxin Yang. Aot series frameworks in pytorch. https://github.com/yoxu515/aot-benchmark, 2024.

[25] Meta Research. Segment anything. https://github.com/facebookresearch/segment-anything, 2024.

[26] Visual Computing Institute. mots_tools. https://github.com/VisualComputingInstitute/mots_tools, 2024.

[27] cocodataset. Coco api. https://github.com/cocodataset/cocoapi, 2024.

[28] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, volume 8693 of *Lecture Notes in Computer Science*, pages 740–755. Springer, 2014.

[29] Jonathon Luiten. Trackeval. https://github.com/JonathonLuiten/TrackEval, 2024.

[30] Jayshree Ghorpade, Jitendra Parande, Madhura Kulkarni, and Amit Bawaskar. GPGPU processing in CUDA architecture. *CoRR*, abs/1202.4347, 2012.

[31] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021.

[32] Ning Xu, Linjie Yang, Yuchen Fan, Dingcheng Yue, Yuchen Liang, Jianchao Yang, and Thomas S. Huang. Youtube-vos: A large-scale video object segmentation benchmark. *CoRR*, abs/1809.03327, 2018.

[33] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbelaez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *CoRR*, abs/1704.00675, 2017.