Visual Multiple Object Tracking with Kernelized Correlation Filters

Francisco Granda Utreras - Student Number: 1006655941

***Abstract*—Multiple object tracking (MOT) is currently a pivotal aspect in the development of robotics and its applications in many fields. For successful practical applications, a MOT algorithm should achieve high levels of accuracy with reasonable computational time to be suitable for real-time conditions. Currently, solutions with convolutional neural networks (CNNs) and deep-feature based trackers accomplished exceptional accuracy results with a trade-off in processing speed. This trade-off will reduce accuracy under real-time streaming conditions due to the fast changes of a real-time environment. On the other side, the application of kernelized correlation filters (KCFs) for single object tracking achieved competitive accuracy levels with fast processing speed. In this paper, a KCF multiple object tracker was built to take advantage of the fast processing speed and was implemented with background subtraction techniques to improve overall accuracy when handling occlusion and scaling. For experimentation, a friendly graphical user interface was implemented to acquire the initial conditions of the tracker. The algorithm was tested with sequences of the Visual Tracking Benchmark and showed promising results while maintaining the fast characteristics of KCF, meaning that further developments can be introduced without significant reductions in processing speed.**

***Index Terms*—Multiple Object Tracking, Kernelized Correlation Filter, Background Subtraction.**

I. Introduction

Visual multiple object tracking continues to be a fundamental part of research and applications in many fields like robotics, computer vision and the aerospace domain. [1] Currently, convolutional neural networks and deep-feature learning algorithms have raised to this challenge accomplishing excellent results in available benchmarks and real-time streaming environments. For instance, the Multiple Object Tracking Benchmark [2] with their 2020 version has on its top leaderboards documented methods involving graph neural networks like GSDT [3] that reaches a 67.1% multiple object tracking accuracy (MOTA) [4] but remains with a slow processing speed of 0.9 frames per second (FPS). Similarly, CSTrack [5] uses CNNs based object detection and re-identification (ReID) to achieve a 65.0% MOTA with an improvement in speed to 4.5 FPS.

These documented methods represent some of the best approaches to solve the MOT problem, but their performance in online streaming conditions will not remain identical because slow computational speeds will result in the skipping of processed frames and applications like autonomous driving will not be entirely successful due to the high-speed changes in the environment.

The author is with the University of Toronto Institute for Aerospace Studies. (UTIAS)
E-mail: francisco.granda@mail.utoronto.ca

With these observations in mind, faster approaches have been tested with less expensive computational efforts to obtain competitive results that could be used in simpler practical applications. For these reasons, the results of a fast KCF single object tracker seen in [6], represent an attractive starting point to solve the MOT problem.

Their algorithm is simplified by the flowchart in Fig. 1, with important features like the tracking of an object without a previous learning stage, meaning that a set of initial conditions related to the desired object in the first image frame is required to initialize the algorithm and start the tracking of the object. The KCF tracker was tested with 50 sequences of the Visual Tracking Benchmark [7] showing a mean precision of 73.2% and a mean of 172 FPS. However, this method has limitations in the form of static bounding boxes that do not adapt to scale and difficulties when dealing with the occlusion of the tracked object.

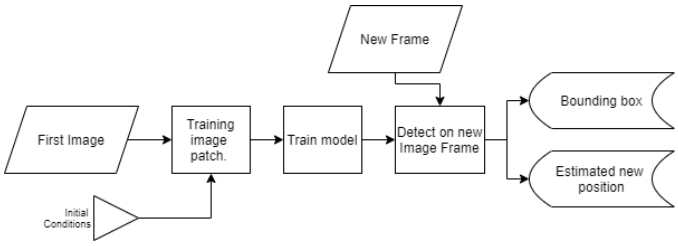

Fig. 1. KCF basic algorithm flowchart. The initial conditions create a patch to train the model. The new frame is included to detect the previous patch, then output a bounding box and an estimated new position for the next iteration

In this paper, we use the base KCF tracker in a parallel pipeline methodology to enable multiple object tracking in the same sequence and test the reliability of implementing multiple KCF trackers simultaneously and their impact on computational speed. Additionally, background subtraction techniques were used to improve the performance of the tracker by associating blob candidates to each object, enabling adaptive bounding boxes and an increased robustness when dealing with occlusion.

Finally, a graphical user interface (GUI) was implemented to manually select the desired objects in the first image frame and automatically obtain the initial conditions for the initialization of the algorithm. We tested our KCF multiple object tracker with sequences of the Visual Tracking Benchmark [7] in two stages. The first stage did not include background subtraction and only used the base KCF to track multiple objects. The final stage with background subtraction showed noticeable qualitative improvements over the base algorithm while maintaining fast computational speeds.

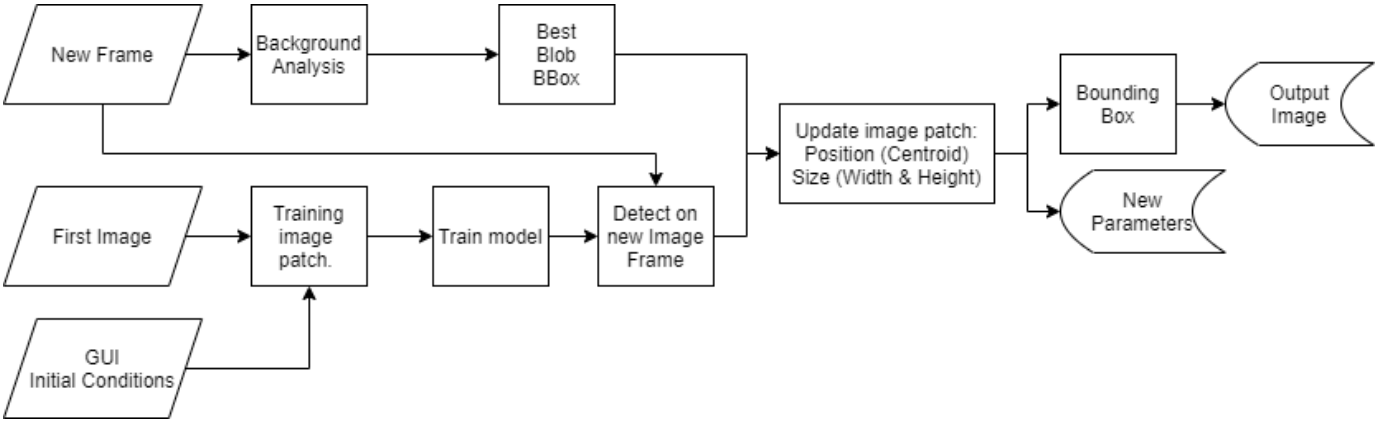

Fig. 2. Workflow of the proposed method for the tracking of multiple objects in a video sequence. The updated new parameters are used for the next iteration of the KCF tracker and the training of the model.

II. Methodology

In this section each stage of the tracker will be explained following the flowchart of our proposed algorithm displayed in Fig. 2.

A. KCF tracker

In addition to the details of Fig. 1, it is also important to notice that the KCF tracker presented in [6], used two feature descriptors: raw pixels and histogram of oriented gradients (HOG). The results presented in the introduction were associated with the HOG feature descriptor and the parameters described in Table I.

TABLE I
KCF tracker parameters with histogram of oriented gradients.

KCF Parameters	With HOG
Feature bandwidth σ	0.5
Adaptation rate	0.02
Spatial bandwidth s	$\sqrt{mn}/10$
Regularization λ	10^{-4}

In this table, n and m refer to the width and height of the desired object bounding box, measured in HOG cells.

B. Graphical User Interface

To start, as shown in Fig. 3, the GUI asks for the number of objects that will be tracked. For this, it will present a prompt to set this number and continue to show the first image frame of the sequence where rectangle selections can be made as an initial bounding box of the objects. The GUI will determine the centroid, width and height information of each bounding box and use it to initialize a KCF tracker for each object.

C. Background Subtraction

Next, foreground detection was applied to each image frame with the help of MATLAB's [8] foreground detector function [9], [10]. Knowing that by this point the KCF tracker was already initialized, the number of training frames was set to 10 and a binary mask was obtained with information of the pixels corresponding to background and foreground.

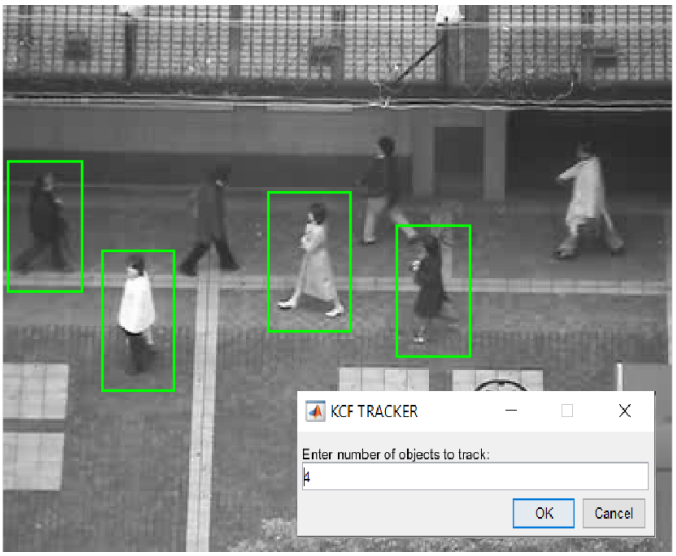

Fig. 3. GUI when tracking 4 objects in sequence "Subway" of the Visual Tracking Benchmark.

Then, morphological operations were applied to this mask at each frame to improve the isolation of blobs from the background and avoid treating fragmented parts of one blob as different objects. These operations include closing (dilation followed by erosion) and hole filling. A sample result can be seen in Fig. 4.

D. Blob Analysis

To filter out very small blobs in the binary mask, each blob's pixel area was compared and validated only if it was greater than half the area of the smallest bounding box selected in the KCF tracker. This way, each valid blob was associated to a bounding box and their centroid, width and height information was stored for future use.

The next step was data association of these blobs with the selected objects in the KCF tracker. To do this, the size of the bounding box of the KCF object was imposed to the centroid

Fig. 4. Binary mask of 20th frame of sequence "Subway" of the Visual Tracking Benchmark.

of each blob to create multiple blob candidates. Then the bounding box of each blob candidate was compared to the KCF bounding box by computing the overlap ratio given by:

$$Overlap = \frac{area(box_1 \bigcap box_2)}{area(box_1 \bigcup box_2)} \quad (1)$$

With the knowledge that only one object is present at each KCF bounding box, only one candidate with the highest overlap ratio was selected.

Finally, to deal with occlusion and the intersection of multiple objects in the same space, the obtained candidate was only associated to the KCF object if the overlap ratio was greater than a threshold value of 0.5. This threshold value was initially set to 0.5 because it is the recommended value to use in other methods like Non-maxima suppression (NMS) [11] and was empirically validated for this algorithm with tests on various sequences of the Visual Tracking Benchmark [7].

E. Update Stage and Display

At this point, each object of the KCF tracker has two possible states within each iteration:

- *KCF object associated with a blob* - if the previous stage successfully associated a KCF object with a blob candidate, the information about the original bounding box of that blob will be incorporated to update the centroid and size of the estimated bounding box for the next iteration. This will be accomplished by performing a weighted average for the x-y components (P(x,y)) of the centroids as well as the widths (W) and and heights (H) of both bounding boxes.
  To assign the weights correctly, we acknowledge that KCF is the main algorithm for tracking, so we assigned a unit weight for this estimation. Then, to weight the estimation of the blob candidate, we used the overlap ratio (O_{ratio}) value obtained in the previous stage. The complete update is described by:

$$P_{New}(x,y) = \frac{P_{KCF}(x,y) + O_{ratio}(P_{Blob}(x,y))}{1 + O_{ratio}} \quad (2)$$

$$W_{New} = \frac{W_{KCF} + O_{ratio}(W_{Blob})}{1 + O_{ratio}} \quad (3)$$

$$H_{New} = \frac{H_{KCF} + O_{ratio}(H_{Blob})}{1 + O_{ratio}} \quad (4)$$

- *KCF object with no association* - in this case, there is not a blob candidate associated to a KCF object so the information provided by KCF will be used unchanged in the next iteration.

Next, it was necessary to handle the situation of an object leaving the scene. For this, we observed the centroid x-y position and if it was outside the bounds of the image frame a counter was initialized. When the counter reached 5 consecutive frames of the object being outside the scene, the track was considered lost and the display was updated without this bounding box.

III. Experiments

For this section, it is important to understand that our proposed algorithm only tracks objects that were selected by the user in the initial image frame. There is not a detection stage for new objects in subsequent frames, which is a constraint that will be discussed later. For this reason we followed the evaluation presented in [6], and various sequences from the benchmark [7] were selected to test our tracker and present results.

It is important to notice that the sequences in [7] do not include ground truth data for every object because it is focused to single object tracking algorithms, which is why an exact MOTA evaluation will be discussed in the future work section of this paper. However, this dataset was selected because it contains useful sequences with objects that are already present in the first image frame and move throughout the video.

In addition, the benchmark [7] includes details for attributes like: illumination variation, scale variation, occlusion and others for each sequence.

A. Sequences

The sequences selected for evaluation are presented in Table II, where we also include the attributes that are associated to each sequence.

TABLE II
SELECTED SEQUENCES FROM VISUAL TRACKING BENCHMARK.

Sequence	Resolution	Number of Frames	Attributes
Basketball	576x432	725	IV, OCC, DEF, BC
Bolt	640x360	350	OCC, DEF, IPR
Crossing	360x240	120	SV, DEF, FM, BC
Subway	352x288	176	OCC, DEF, BC
Walking	768x576	412	SV, OCC, DEF

IV: illumination variation. OCC: occlusion. DEF: deformation. BC: Background Clutters. IPR: In-Plane Rotation. SV: scale variation. FM: fast motion.

B. Results and Analysis

In this section, results will be presented for the second stage of evaluation by starting with the most simple sequence in our dataset and continue to more complex scenarios according to their attributes and general observation of the scene. Comparisons with the first stage and the base algorithm will be presented.

- *Crossing* - in this sequence, two pedestrian objects can be selected to be tracked as shown in Fig. 5.

Fig. 5. 20th frame of Crossing sequence with tracking displayed.

  This sequence includes illumination variation by showing sun and shadow spots. Also scale variation and deformation due to the objects being pedestrians walking up the image. The tracking of both objects was successful in this sequence, even with the presence of fast moving cars that partially intersect with the tracked objects in the image space. A difference with the first stage was that a first glace of the adaptive bounding boxes of our tracker was observable but not entirely due to the short number of frames.

- *Walking* - in this sequence, two pedestrian objects and a car object can be selected to be tracked as shown in Fig. 6.
  This sequence includes scale variation and object deformation similarly to the previous sequence. But now occlusion was introduced when the pedestrian objects passed behind a lamp post. In the first stage, the track of the second pedestrian was lost and the bounding box stayed with the lamp post. In the second stage the tracking of all objects was successful, solving the occlusion problem and clearly showing the adaptive bounding boxes as the pedestrians continued to walk left of the image.

- *Subway* - in this sequence, seven pedestrian objects can be selected to be tracked as shown in Fig. 7.
  The first stage showed the incorrect tracking of five of the seven objects due to occlusion and background clutter in the scene. On the other hand, the second stage showed a successful tracking of all seven objects through the entire sequence. Considering that the sequence does not present scale variance or a moving image frame, this improvement over the base algorithm is directly related to the correct association of blobs and KCF objects.

Fig. 6. 30th frame of Walking sequence with tracking displayed.

Fig. 7. 15th frame of Subway sequence with tracking displayed.

- *Bolt* - in this sequence, eight athlete objects can be selected to be tracked as shown in Fig. 8.
  This sequence presented great difficulty to track the athletes in both the first and second stages of the evaluation. There was not a noticeable improvement of the proposed algorithm over the first results, and this can be directly associated to the fact that this sequence include in-plane rotation, meaning that there is a movement of the image frame resulting in a non-static background that would not work properly with our tracker when trying to discern between foreground and background pixels.
  Then, without adequate blobs to improve the estimation of the position of our objects, the algorithm would rely entirely on the results of KCF. We observed that the

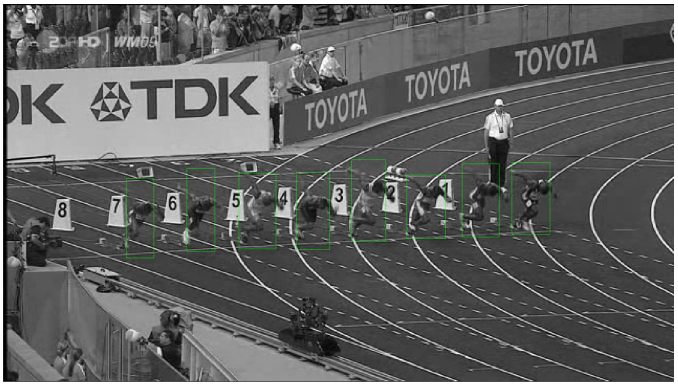

Fig. 8. 1st frame of Bolt sequence with tracking displayed.

KCF algorithm successfully tracked only four of the eight possible athletes. This poor result was directly associated to the deformation of the objects across each frame, where the athletes would rapidly change their pose with their arms and legs resulting in inconsistent image patches to be detected in the subsequent frames.

- *Basketball* - in this case, not every object can be tracked across the entire sequence because some of them exit and reenter the scene. However, the benchmark [7] proposed one player that is observed during the entire sequence which can be seen in Fig. 10.
  This sequence represents the most challenging scenario for our tracker because in addition to background clutter and the occlusion of almost all objects, the image frame is also non-static which complicated the extraction of blobs similarly to the case of the 'Bolt' sequence.
  The tracker was tested with various objects and obtained inconsistent results even for the object in Fig. 10 in both stages. In [6], the correct tracking of the object was accomplished because the algorithm was initialized with the ground truth data for the initial frame. This result was possible in our evaluation stages but using the same constraint without noticeable improvements between stage one and stage two.

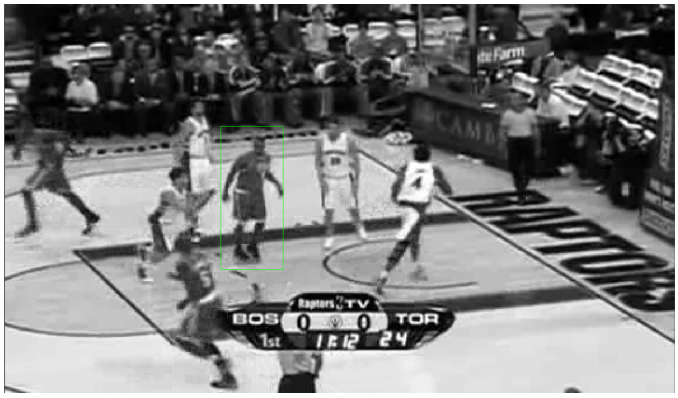

Fig. 10. 1st frame of Basketball sequence with tracking displayed.

To continue, each sequence was tested starting with one object and incremented up to five. Then, the processing time was recorded to compute the processing speed. The results are summarized in Fig. 9 and Table III.

We notice in Fig. 9 that there is a considerable variation in the first FPS value between 'Crossing', 'Subway' and the remaining sequences. This is observed because if we recall Table II, 'Crossing' and 'Subway' are the sequences with the lower resolution of the bunch, resulting in faster initial processing speeds. However, this variation is reduced with the increment of objects where we find that with five tracked objects, there is a standard deviation value of only 9.12 FPS. The values for more than five objects remained partially constant in mean and standard deviation, which is why this analysis is focused on the first five objects.

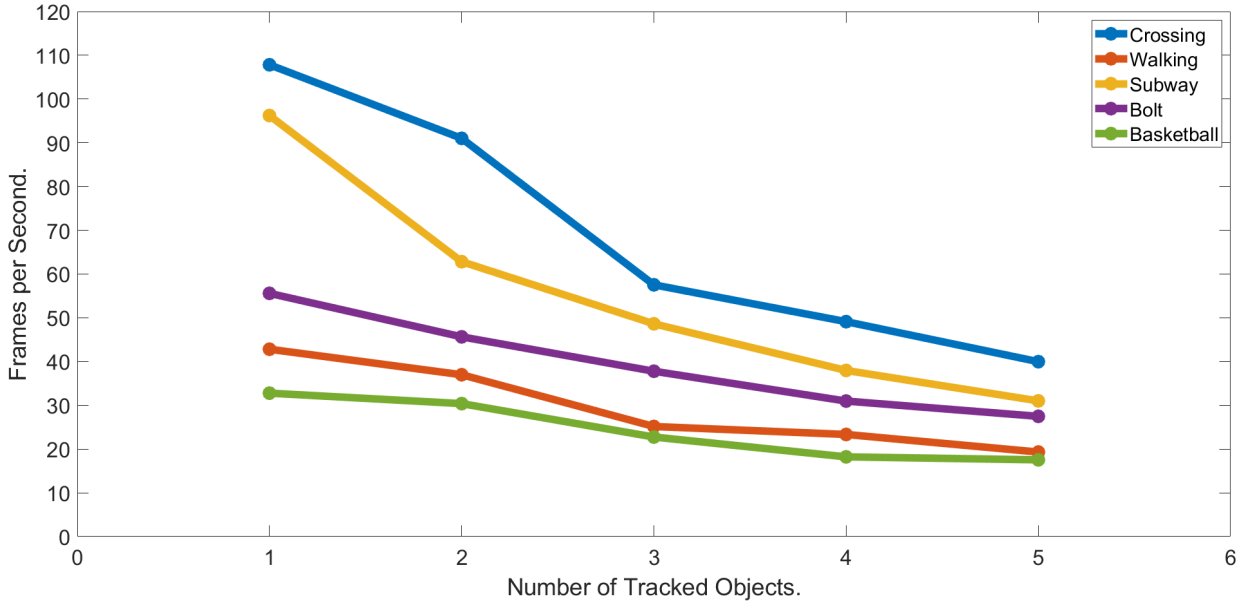

Fig. 9. Change in processing speed with respect to the number of tracked objects. For sequences with less than five objects, redundant trackers were initialized for the same object to obtain this chart.

TABLE III
PROCESSING SPEED: FRAMES PER SECOND.

Sequence	Number of Tracked Objects				
	1	2	3	4	5
Basketball	32.78	30.39	22.75	18.22	17.54
Bolt	55.59	45.65	37.79	30.97	27.48
Crossing	107.83	91.01	57.54	49.12	39.97
Subway	96.23	62.83	48.63	37.97	31.01
Walking	42.82	36.99	25.16	23.34	19.32
Average	56.05	44.81	32.48	27.27	23.38

All values are rounded to two decimal places.

Additionally, in all sequences the greatest variation of speed is found between a quantity of one object and two objects, while the remaining variations remain similar. This observation occurs due to the nature of the code implementation of the algorithm. For one object the code deals only with vectors and matrices, while for more than one object cell arrays are created to store the vectors and matrices associated to each object.

Finally, the variations of speed were computed and averaged across all sequences to obtain a value of $9.99\ [fps/object]$ that will represent the cost in processing speed for each of the first five additional objects in the tracker.

These results were obtained using an ASUS Zephyrus M workstation that includes an Intel Core i7-8750H processor clocked at 2.88 GHz, an NVIDIA GeForce GTX 1070 GPU and 16 GB of RAM memory.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented a multiple object tracker algorithm that combined the efforts of a basic KCF single object tracker with background subtraction and blob analysis. Additionally, a simple GUI was created to facilitate the testing of the prototype. The proposed tracker significantly improved the results over a KCF only algorithm by handling occlusion problems with blob association and introduced the feature of adaptive bounding boxes to better handle scale variation in the sequences. While exceptional results were obtained in sequences with occlusion and scale variation like 'Walking' and 'Subway', the limitations of this tracker were observed in 'Bolt' and 'Basketball'. A sequence with a non-static image frame and a moving background will enforce our tracker to only work with KCF results and basically go back to baseline performance even if the sequence does not include more complex attributes.

This tracker obtained a cost value of $9.99\ [fps/object]$ for the first five objects and an average of 56.05 FPS for the first object. If we compare our average FPS speed to the one presented in [6], we notice that there is an important difference that can be attributed to the variation in the hardware used for experimentation. This means that even better results for the cost value can be obtained with more competitive hardware and optimization of this prototype tracker. For these reasons, a KCF multiple object tracker is proven to be a reliable alternative to solve the MOT problem with the limitations described in this work and fast processing speeds.

For future work, the implementation of a detection stage for new objects will be pivotal to evaluate this algorithm with more complicated benchmarks like MOTChallenge [2] and obtain results associated to accuracy, precision and recall. This will enable us to properly compare our tracker to other state-of-the-art methods. Finally, with the fast properties of our tracker, more complex techniques can be introduced to work with this algorithm and improve the estimation stage of the tracker. These modifications would be focused to overcome the limitations of non-static backgrounds while still being suitable for real-time streaming conditions.

REFERENCES

[1] W. Luo and J. Xing, "Multi-task learning for dense prediction tasks: A survey," *arXiv preprint arXiv: 1409.7618*, 2017.
[2] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixe, "Mot20: A benchmark for multi object tracking in crowded scenes," *arXiv:2003.09003*, 2020.
[3] Y. ang, X. Weng, and K. Kitani, "Joint object detection and multi-object tracking with graph neural networks," *arXiv:2006.13164*, 2020.
[4] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *Image and Video Processing, 2008(1):1-10*, 2008.
[5] C. Liang, Z. Zhang, Y. Lu, X. Zhou, and B. Li, "Rethinking the competition between detection and reid in multi-object tracking," *arXiv:2010.12138*, 2020.
[6] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence, 37(3):583–596*, 2015.
[7] Y. Wu, J. Lim, and M. H. Yang, "Online object tracking: A benchmark," *CVPR*, 2013.
[8] MATLAB, *version 9.6.0.1 (R2019a)*. Natick, Massachusetts: The MathWorks Inc., 2019.
[9] P. Kaewtrakulpong and R. Bowden, "An improved adaptive background mixture model for realtime tracking with shadow detection," *In Proc. 2nd European Workshop on Advanced Video Based Surveillance Systems*, 2001.
[10] C. Staufferand and W. Grimson, "Adaptive background mixture models for real-time tracking, computer vision and pattern recognition," *IEEE Computer Society Conference on, Vol. 2, pp. 2246-252 Vol. 2*, 1999.
[11] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Improving object detection with one line of code," *arXiv: 1704.04503*, 2017.