Proceedings of the 7th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'21)
~~Prague, Czech Republic~~ Virtual Conference - July, 2021
Paper No. MVML 101
DOI: 10.11159/mvml21.101

New Object Tracker Based On Adaptive Intensity Models of Object and Its Surroundings

Dorothy Gors[1], Robbert Hofman[1], Merwan Birem[1], Steven Kauffmann[1]
[1]Flanders Make
Gaston Geenslaan 8, B-3001 Leuven, Belgium
dorothy.gors@flandersmake.be; robbert.hofman@flandersmake.be; merwan.birem@flandersmake.be;
steven.kauffmann@flandersmake.be

Abstract **–** New developments on the object tracker topic are needed, so that reliable tracking systems can have value for industrial applications, like surveillance and assembly monitoring. This paper presents a new object tracker algorithm based on adaptive models of the intensity probabilities of the object and its surroundings. Using the tracked object contour in the previous frame and the object path allow to estimate a narrow search area, in which contours with high object probability are combined, after masking pixels with high surrounding probabilities away. Rules about the object contour area ensures that the tracked contour doesn't drift away between frames or spreads into the surroundings. If the tracking is lost, the contour prediction in combination with the surrounding estimation takes over, filling the gaps until the object intensity-based tracker leads the tracking again. The proposed tracker was contrasted against three of the available trackers in OpenCV (i.e. KCF, CRST and MOSSE). Their performances were evaluated on two different applications (i.e. drone tracking and part tracking in an assembly cell) based on the Intersection over Union (IoU)-metric and their processing time. The obtained results show that the proposed tracker is faster and more accurate.

Keywords**:** Computer vision, Contour prediction, Intensity-based models, Object tracking

1. Introduction

Object tracking is an important topic of computer vision. Especially in the fields of surveillance, traffic control, human-robot interaction and medical imaging [1, 2]. Tracking is the process of estimating the trajectory of an object in subsequent frames, given the bounding box (i.e. position and size) of the object in the initial frame. Closely related are feature detectors; those will just detect a specific object on each individual frame, based purely on matching features [3]. Pure trackers, like investigated in this study, aim to follow the trajectory as well as predict the future location of the object. This eliminates the risk of mistaking the tracked object for a visually similar object, hereby making the object tracking more stable. There are many variables affecting the performance of trackers, such as illumination variations, scale variations, object rotation, occlusion and background clutter [4] and camera movement. Currently there is a wide range of different tracking algorithms, but none can cope with all those problems simultaneously. Because of its relevance in numerous applications, an increasing amount of time and effort is dedicated to the challenging research about object trackers. In this study we will contrast a new tracker against three available trackers, and investigate if we come closer to a universal tracker.

1.1. Previous Work

Due to its widespread use, some trackers have already been integrated in popular libraries, such as OpenCV [3]. Those trackers are easy to use as several tutorials demonstrate [5, 6]. The most common used OpenCV trackers are: KCF [7], CRST [8] and MOSSE [9]. They all use filters on a restricted search area around the predicted location to estimate a bounding box comprising the object at its new location. However, they rely on different algorithms and filters and consequently, have their own strengths and limitations [5,6]. KCF uses Kernelized Correlation Filters in combination with an adaptive classifier built on patches with and without the object. It is a fast algorithm, with slightly lower tracking accuracy and bad occlusion recovery. The discriminative Correlation Filter with Channel and Spatial Reliability, CRST, trains on two features (HoG and Colornames). It is slower, but more accurate than KCF. Minimum Output Sum of Squared Error, MOSSE, uses adaptive correlation filters. It is a fast algorithm and robust against occlusion, light, scale and pose variations and non-rigid deformations, but at the expense of accuracy.

1.2. Proposed Work

We propose a method that tracks an object on a new frame, based on intensity and spatial information learned from the previous frames. Intensity models are used to find pixels whose intensities have a high probability to match the intensities of the tracked object. This search is restricted to the region where the object is supposed to be found. Monitoring the trajectory of the object along the previous frames, allows us to estimate the object's position on the new frame and to narrow down the search region. To reduce the risk of tracking the object as well as its surroundings even more, we work with two different intensity models: one characterizing the object and one the surroundings. This allows us to extract the surroundings from the search region. Additionally, when multiple contours are found, only a restricted set are combined into the final object contour. By controlling the area of the combined contour and the inter-contour distances, we avoid including contours covering the surroundings.

With this approach we seek to obtain a tracker that doesn't just return a rough estimation of the bounding box comprising the object, but a detailed delineation of the actual object's contour. Furthermore, with this algorithm there should be more robustness against variation in appearance (intensity, scale and pose) and occlusions. The method will also prevent the tracker from drifting, as the tracker cannot easily lock on the surroundings instead of the real object.

The remaining of this paper is organized as follows: Section 2 describes how intensity models are derived and used to identify regions that are likely to belong to the tracked object. Section 3 presents how the position of the object is predicted in a new frame, based on the object's previous trajectory. Section 4 describes how the different identified regions are combined into the final object contour. In Section 5, the proposed framework is validated on two different applications and evaluated against three existing trackers. Section 6 presents the conclusions and ideas for future work.

2. Intensity Models

An object is represented by several pixels in a frame, each with their own pixel intensities. An intensity model then expresses the probability that a certain pixel intensity can be found in the object. The object can then be identified in a frame as the contour that comprises pixels for which the intensity model gives high probabilities. In this paper we propose a histogram-based model, where the normalised histogram of the pixel intensities within an object's contour is used to represent the intensity distribution of that object. Using colour images means that we need to deal with a three-dimensional intensity space (i.e., RGB). It is common to generalise this into the one-dimensional grayscale space, however this average operation will result in the loss of information. Therefore, we designed a new one-dimensional space in which the distinction between red, green and blue is not lost, but instead similar red, green or blue hues are expressed with a single intensity value (equation 1).

$$I = \left(\frac{R}{8}\right)_{round} + 32 * \left(\frac{G}{8}\right)_{round} + 32^2 * \left(\frac{B}{8}\right)_{round} \quad with\ R, G, B\ \in [0 \cdots 255] \tag{1}$$

An intensity model can then be initiated or updated from the transformed RGB intensities of the pixels within the object's contour. When an object tracker initializes, it is given a contour to track. This input can be provided either manually or by an external object detection algorithm. The pose of the object changes during tracking, which causes a shift in the object's intensity distribution. Also, erroneous detection, where the object is only partly included in the contour or where the surroundings are included as well, introduces errors in the intensity distribution. To avoid those issues, there is chosen for an adaptive intensity model. After each frame the intensity model is updated as the weighted average of the current model and the normalised histogram derived from the current tracked object. The weight of the new histogram into this, is a compromise between a quick response to fast changing intensity appearance of the object and the stability against mis delineation of the contour at the current frame.

At a new frame the adapted intensity model is used to identify the regions with high probability of being part of the tracked object. It cannot be fully excluded that intensities of the surrounding appear in this intensity model. Moreover, when parts of the surroundings get covered by the tracked contour over multiple frames, those intensities can supress the object intensities in the model. In that case the tracked contour will drift away from the true object. To prevent this,

two intensity models are made: one of the object and one of the surroundings. In the restricted search region (section 3) the intensity histogram within the contour is used to update the object's model, while the histogram of the intensities of the remaining pixels is used to adapt the surroundings' model. At a new frame an object mask and surrounding mask can be derived by thresholding the pixel probabilities from the respective models. Both masks are used to define the object contour (section 4). Depending on the situation, the exclusion of the surroundings can have more importance than the inclusion of the object, or vice versa, in order to obtain a good object contour. This can be obtained by setting different threshold for both masks.

3. Search Region

Searching for regions with similar intensities as the object within the entire frame would lead to many false positives. Therefore, we predict the location of the object and search for the object contour in an area around this location. Additionally, reducing the search area will save in computation time.

The translation of the tracked object during the last five frames is monitored. The average translation is performed on the detected contour of the previous frame, to obtain a prediction of the contour at the new frame. This contour is then dilated to obtain a search region that most likely covers the new location of the object. The dilation radius needs to be high enough to compensate a possible misprediction, however small enough to restrict the search area.

The predicted contour is used to initialize the intensity-based search for the tracked contour at the current frame. However, when this intensity-based search fails, the prediction of the movement of the object can propagate to the next frame. In that case, the predicted contour keeps translating in between frames, until the intensity-based search gets track of the object again. Since the prediction errors increase over time, the dilation radius will increase with each frame in which the tracking is lost. This is called a prediction phase. Often the prediction phase lasts only one or two frames, before the normal tracking procedure takes over. When tracking is lost for too long, the change that the predicted contour deviates from the true contour increases. The chance, when a contour with high object probability is finally detected, that it actually belongs to the true object, decreases. Therefore, when the prediction phase lasts longer than three frames, the tracker switches to a searching phase. In this phase, the dilatation radius does not increase further. No new contour is predicted based on the estimated motion, instead the same location as in the previous frame (i.e. the last frame of the prediction phase) is used. Once reaching the searching phase, it can be recommended to break the tracking and perform an object detection instead, such that the tracking can continue properly.

4. Object Tracking

In previous sections it is explained how at each frame a search region can be defined, based on the previous movement of the object, and how intensity models can be used within that region to identify regions with high probability of being object or surroundings. In this section, those techniques come together into the actual framework of the proposed object tracking. Initializing the object tracker happens when an object is detected by an external method. In the same frame, the pixels within the detected contour are used to initialize the object intensity model. The intensity distribution of the pixels surrounding this contour, at a distance of maximal the dilation radius, becomes the initial surroundings model. Afterwards, the object tracking proceeds in three steps: (1) contour prediction, (2) object regions delineation with surroundings-based correction, and (3) restricted combination of regions into object contour. This framework is illustrated in figure 1.

In the first step a prediction of the contour is made, based on the object's estimated movement. A dilation of this contour then forms the search region (section 2).

Secondly, the RGB values in the search region are transformed by eq. 1 into new intensity values. After which they are converted into an object and surroundings probability map, using the object and surroundings intensity model, respectively. Then a distance-based correction, based on the predicted contour, is performed on those maps. Within the object probability map, the probability values outside the predicted contour are decreased, according the distance from the predicted contour (see eq. 2). Opposite, within the surrounding probability map, the probability values within the predicted contour are decreased, according to the distance from the boundary of the predicted contour (see eq. 3). After this correction, an object

and surrounding mask are thresholded out of those maps. The surrounding mask is subtracted from the object mask to obtain the object regions. During this entire procedure morphological operations are used to smoothen the results.

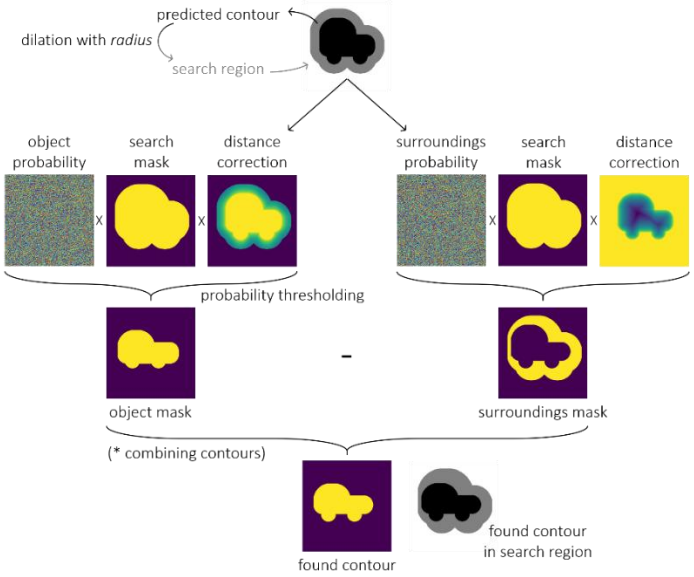

Fig. 1: Object Tracking Framework.

```
Cnts ← sort(Cnts, min(|cnt, centre|))
Cnts_combined = {},   bigEnough = False
for all cnt in Cnts do
    calculate a_new, A_new(= w_new * h_new) from {cnt ∪ Cnts_combined}
    if bigEnough then
        if w_new > w_i−1 + radius or h_new > h_i−1 + radius then
            break
    if |cnt, Cnts_combined| ≤ 25 then
        if a_new ≤ a_max and A_new ≤ A_max then
            Cnts_combined ← {cnt ∪ Cnts_combined}
        else
            if bigEnough = False and isSearching == False then
                repeat algorithm without delineated search region
            break
        if a_new ≥ a_min and A_new ≥ A_min then
            bigEnough = True
if bigEnough = False then
    repeat algorithm with inversed surroundings mask
```

Fig. 2: Algorithm for combining contours.

$$d^* = 1 - 0.5 * \sqrt{distance_{from\ predicted\ contour}/radius} \tag{2}$$

$$d^* = 1 - \sqrt{2 * distance_{from\ predicted\ surroundings}/radius} \tag{3}$$

no large area variation expected

$$a_{min} = \max(0.2 * a_{init}, 0.5 * a_{i-1}), \quad a_{max} = \min(2 * a_{init}, 2 * a_{i-1}) \tag{4}$$

$$A_{min} = \max(0.25 * A_{init}, 0.5 * w_{i-1} * h_{i-1}),$$
$$A_{max} = \min(2 * A_{init}, (w_{i-1} + 0.35 * radius) * (h_{i-1} + 0.35 * radius)) \tag{5}$$

large area variation expected

$$a_{min} = 0.5 \cdot a_{i-1}, \quad a_{max} = 2 * a_{i-1} \tag{6}$$

$$A_{min} = 0.5 * w_{i-1} * h_{i-1},$$
$$A_{max} = (w_{i-1} + 0.35 * radius) * (h_{i-1} + 0.35 * radius) \tag{7}$$

As a result, one or more object regions are obtained. Considering them all as part of the object is unwise. It is unavoidable that there are regions that don't cover the real object. In order to exclude those, we start including regions from the centre of the predicted contour. In general, adding regions stops as soon as the next addition results in a combined contour's area that exceeds some critic values. Two additional rules are introduced to further concentrate the object contour; (1) when the newly added region causes a sudden increase in the growing contour's size, this region is excluded and the adding procedure stops and (2) when the next region is too far from the growing contour, this region is discarded, while the procedure continues. An algorithmic representation, depicted in figure 2, gives more detail about this combining procedure. There are two metrics to express the contour's area: a = the number of pixels and A = the area of the contour's bounding box. The critical minimal and maximal value, from both, depends on the area at initialisation (a_{init}, A_{init}) and at the previous frame (a_{i-1}, A_{i-1}). Eq. 4 and 5 define a_{min}, a_{max}, A_{min} and A_{max} when object's distance towards the camera, and hence its perspective, isn't expected to vary much. While eq. 6 and 7 exclude the dependency towards initialisation, to define the critical values more suited to track objects moving in free space. Contrasting the final object's area against critical minimal values gives an indication that the combined contour doesn't cover the entire object, in which case additional procedures can be initialised. The situation where including an additional region directly causes the contour's area to increase from 'not big enough' to 'too large', can be contained by constraining this additional region within the predicted contour. In that case, the entire procedure is repeated, but excluding the dilatation operation performed at the first step. It is also possible that the object contour is too small (or even non-existing), because few pixels with sufficient high object probabilities where identified. In that case an object contour can be forced, by solely

relying on the surrounding mask. The surrounding mask will then be directly subtracted from the predicted contour, instead of the nearly absent object mask, to create the object regions. After which those regions can be combined using the same combining algorithm. Since, the resulting object contour isn't based on the object's intensities, it is not safe to continue this alternative procedure for more than a few sequential frames.

After finding the object contour, the intensity histograms within the contour and within its surrounding enclosed by the search region are used to update the intensity models (section 2), and the location of the tracked object is used to predict the contour's position at the following frame. If, however, the object tracking fails to delineate an object contour based on the intensity models, it goes into the prediction phase (section 3).

5. Validation

The performance of our object tracker is evaluated on two different applications. For the first experiment, videos of the drone-vs-bird detection challenge (WOSDETC 2020) [10] are used, where the ground-truth annotations serve to initialise the drone tracker and later to validate the tracked drone contour. For the second experiment, videos of an assembly operation were collected to validate the tracker in an industrial setting. Here, the ground truth was manually annotated. The performance is compared with results obtained with trackers found in the OpenCV library: KCF, CRST and MOSSE. Tracking results of the different methods are contrasted in terms of the Intersection over Union (IoU)-metric and their processing time. The IoU value is calculated as the ratio between the area of overlap and the area of union. Area of overlap is the overlapping area between the tracked bounding box and the ground-truth bounding box, and area of union is the total area covered by both bounding boxes. If the proposed tracker goes into the searching phase (i.e. when the predicting phase continues for more than three successive frames), the tracking is aborted and the ground-truth data is used to re-initialise the tracker, such that tracking can continue. The same goes for when the OpenCV trackers identifies an interrupted tracking. This results in an additional evaluation metric: the number of frames with effective tracking (regardless if the tracking is accurate or not).

5.1. Drone Tracking

In the first experiment we applied our tracker and the three OpenCV trackers on videos of drones. We set our tracker to identify drone regions with an intensity probability higher than 0.005 and surrounding regions with an intensity probability higher than 0.01. The histogram distribution of each frame contributes with a weight of 0.10 to the adaptive intensity models. A dilation radius of 100 pixels is used to expand the search area around the predicted drone location. Since the drone can move freely in the open air, we set for eq. 6 and 7 as drone area conditions. Figure 3 shows snapshots of the four investigated videos. Here, an advantage of our method can already be observed: OpenCV trackers identify only the object's bounding box, while our method can delineate its contour. Between the three OpenCV trackers, the CRST tracker outperforms the other two. While KCF and MOSSE trackers fail to keep track of the drone after some frames, the CRST tracker keeps track until the end (with the exclusion of the third video). However, as Figure 4 shows and Table 1 quantifies, the CRST tracker is not tracking the drone on video 4 (IoU = 0.01), but fails to identify this mistake and keeps the computational cost going.

5.2. Part Tracking in Assembly Cell

In the second experiment we applied the different trackers on a video of an assembly operation. Three scenes of the video are subjected to the trackers. In each case another assembly part is being picked up and inserted into the main assembly. The intensity probability thresholds to distinguish regions of the assembly parts and their surrounding regions are both set to 0.005. The histogram distribution of each frame contributes with a weight of 0.10 to the adaptive intensity models. A dilation radius of 50 pixels is used to expand the search area around the predicted assembly part location. The operator handles the parts, so no large changes in the distance between part and camera (and hence no large changes in the part's perceived area) are expected, therefore we set for eq. 4 and eq. 5 as part area conditions. Figure 5 shows snapshots of the three investigated scenes. The difficulty of this scene is the reflective gray surfaces of parts handled in scene 2 and 3, the occlusion by the hands of the operator, and background clutter due to color similarity between the gray surface of the parts and the gray assembly-kit or the brown-red surface of the parts (scene 1 and 2) and the hands. Both the MOSSE and KCF trackers can keep track of the parts in less than 50% of the frames. For the frames they do track, the mean IoU is between 0.01 and 0.51

(Figure 6). This indicates that those trackers drift away from the ground truth. Ours and the CRST tracker perform better, with mean IoU values listed in Table 2. However, for scene 3, dealing with a highly reflective surface, our tracker close to the ground truth, while the CRST tracker drifts away. Figure 6 shows clearly a steep decrease of the IoU at 28 of the third scene for the CRST tracker. The tracking percentage of 100% for the CRST tracker indicates that this fails to recognise this false positive result. Our tracker is better in recognising its tracking error, so that a new (performed by an object detector) can be triggered, allowing to proceed the tracking more accurately. Our tracking is around 95%, however, this trigger for reinitialization allows us to maintain a high IoU value for a longer period of time. Regarding the computation time, our tracker is globally 5.3 times faster than the CRST tracker.

Fig. 3: Frames from the four drone videos with our tracked contour (blue) and the provided ground truth (red).

Table 1: Statistics of drone tracking with our method and CSRT.

	IoU (mean – std)		calculation time (mean – std)[ms]		Tracked frames [%]	
	Ours	CSRT	Ours	CSRT	Ours	CSRT
video 1	0.60 – 0.19	0.33 – 0.13	21.66 – 9.98	61.07 – 23.62	99.48	99.87
video 2	0.36 – 0.18	0.59 – 0.07	12.11 – 12.78	75.29 – 8.02	98.85	99.86
video 3	0.26 – 0.26	0.09 – 0.25	12.63 – 17.30	4.07 – 10.21	92.08	13.86
video 4	0.33 – 0.29	0.01 – 0.09	22.77 – 26.02	57.95 – 19.96	90.00	99.75

Table 2: Statistics of part tracking with our method and CSRT.

	IoU (mean – std)		calculation time (mean – std)[ms]		Tracked frames [%]	
	Ours	CSRT	Ours	CSRT	Ours	CSRT
scene 1	0.79 – 0.18	0.75 – 0.17	19.64 – 4.77	97.16 – 18.53	98.77	100
scene 2	0.61 – 0.28	0.71 – 0.12	12.64 – 4.17	80.08 – 11.31	92.78	100
scene 3	0.62 – 0.26	0.37 – 0.46	15.10 – 9.28	69.87 – 5.13	95.77	100

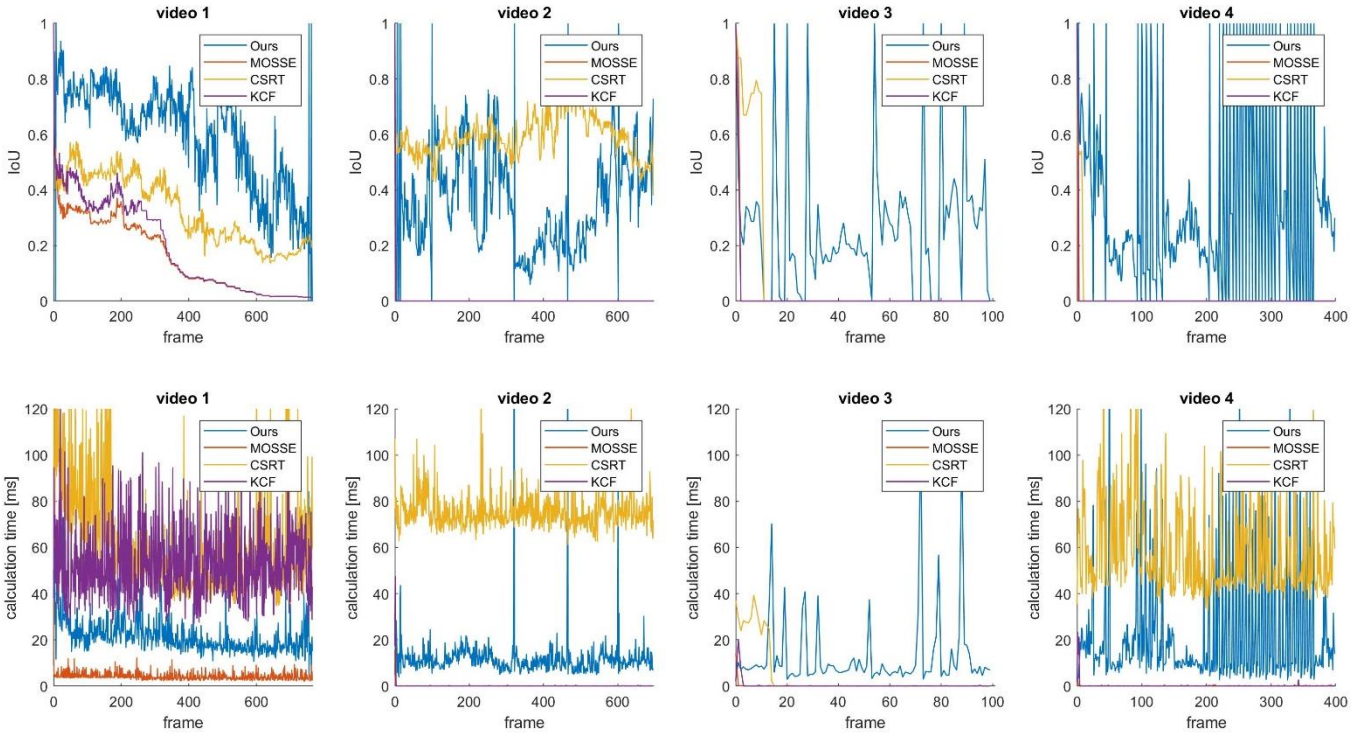

Fig. 4: IoU and calculation time of four drone trackings with our method, MOSSE, CSRT and KCF.

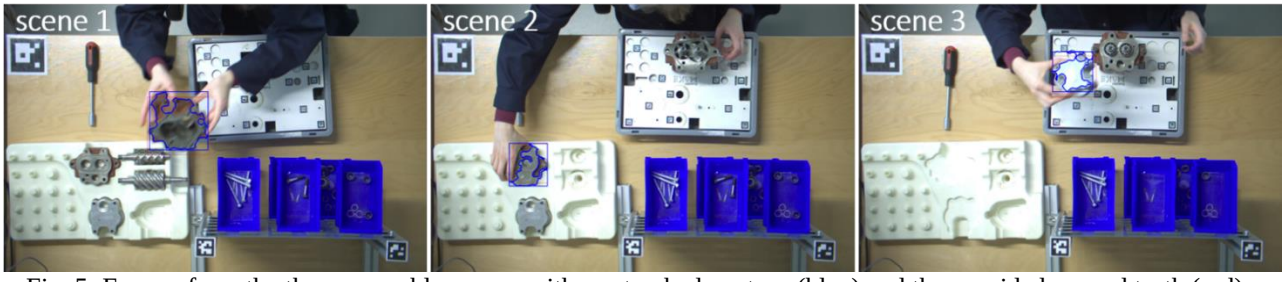

Fig. 5: Frames from the three assembly scenes with our tracked contour (blue) and the provided ground truth (red).

6. Conclusions

In this paper we conclude that our adaptive intensity-based tracker extended with contour merging and estimation performs better than the three investigated OpenCV trackers in terms of accuracy and speed; 5 times faster and an increased accuracy of 200% in challenging cases. We succeed in building a more universal tracking, that is more stable in difficult situations with large variations in object scale and pose, occlusion and background clutter. Additionally, our method didn't drift away or stopped before drifting could occur. This algorithm also returned contours, instead of just a bounding box. We observed similar contrast between the KCF, CSRT and MOSSE as mentioned in the OpenCV tutorials, with the smallest calculation time for MOSSE, followed by KCF, and the highest accuracy for CSRT. The robustness of MOSSE for hard situations, however, we didn't observe.

Future work is to validate the proposed technique with more industrial relevant scenarios. Some industrial challenges are multi-object tracking, variety of materials and environment, occlusions with other objects or obstacles and varying light conditions. In the performed experiment no sudden changes (direction or acceleration) in the object's trajectory was observed. In order to make our tracker more robust for sudden action, it is needed to improve the object trajectory prediction.

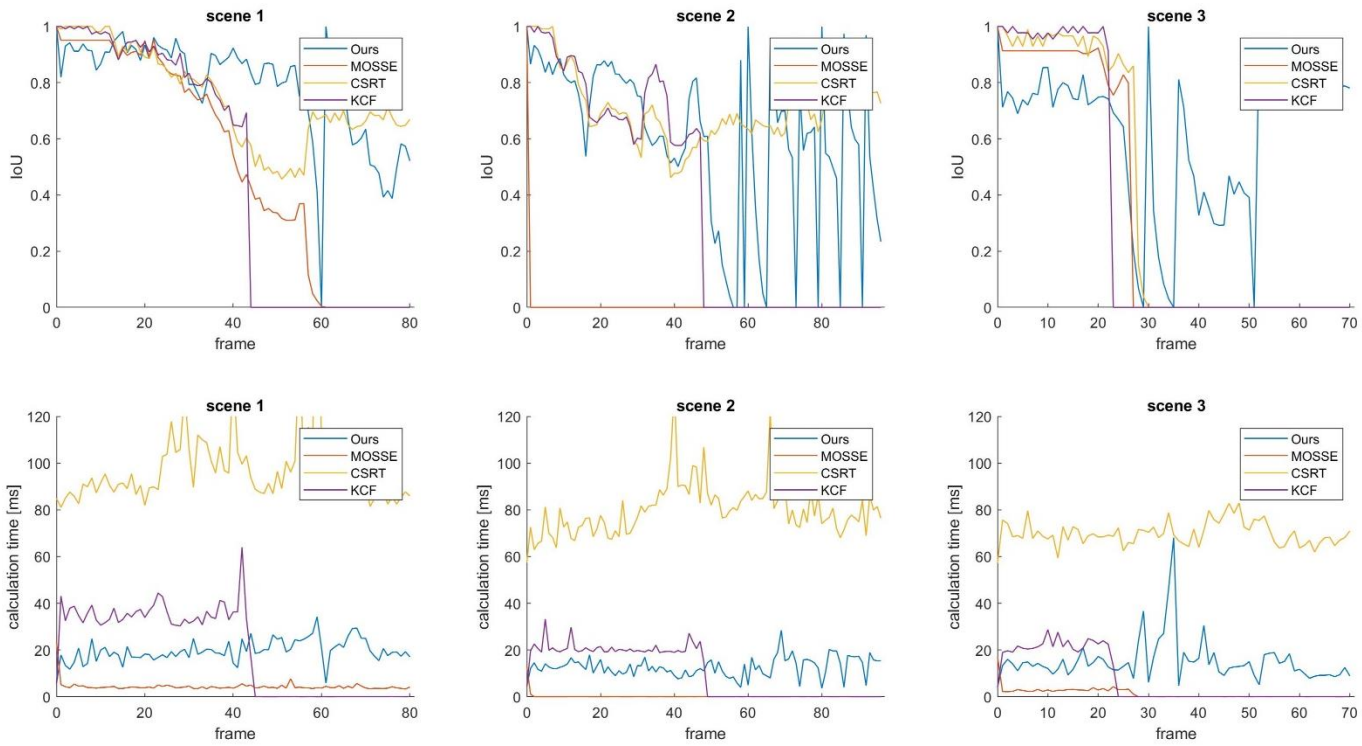

Fig. 6: IoU and calculation time of three part trackings with our method, MOSSE, CSRT and KCF.

Acknowledgements

This research is supported by VLAIO (Flanders Innovation & Entrepreneurship) Flanders Make, the strategic centre for the manufacturing industry in Flanders, within the framework of the FAMAR ICON-project.

References

[1] Q. Wang, F. Chen, W. Xu, and M. Yang, "An experimental comparison of online object-tracking algorithms," *Optical Engineering + Applications.*, 2011

[2] K. Cannons, "A Review of Visual Tracking," *Technical Report CSE-2008-07.*, York University, Canada, 2008.

[3] P. Janku, K. Koplik, T. Dulik, and I. Szabo, "Comparison of tracking algorithms implemented in OpenCV," *MATEC Web of Conferences.*, vol. 76, CSCC 2016.

[4] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," *IEEE TPAMI.*, 2015.

[5] S. Mallick. (2017, February 13). Object Tracking using OpenCV (C++/Python) [Online]. Available: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/

[6] A. Rosebrock. (2018, July 30). OpenCV Object Tracking [online]. Available: https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/

[7] J.F. Henriques, R. Caseiro, P. Martins and J. Batista, "High-Speed Tracking with Kernelized Correlation Filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, pp. 583–596, 2015.

[8] A. Lukežič, T. Vojíř, L. Cehovin, J. Matas and M. Kristan, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability," *International Journal of Computer Vision (IJCV).*, 2018.

[9] D.S. Bolme, J.R. Beveridge, B.A. Draper and Y. Man Lui, "Visual object tracking using adaptive correlation filters," *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*, pp. 2544–2550, June 2010.

[10] WOSDETC 2020. (2020, January 4). Drone-vs-Bird Detection Challenge [Online]. Available: https://wosdetc2020.wordpress.com/drone-vs-bird-detection-challenge/