DETECTING OCCLUSION FROM COLOR INFORMATION TO IMPROVE VISUAL TRACKING

Stephen Siena and B.V.K. Vijaya Kumar

Carnegie Mellon University
Electrical and Computer Engineering
5000 Forbes Avenue, Pittsburgh, PA, 15213

ABSTRACT

Visual tracking in unconstrained environments often involves following an object that exhibits a number of appearance changes from factors such as scale change, rotation, and illumination. Effective tracking requires adapting a tracker to the object's changing appearance over time. When a target becomes occluded by other objects in the scene, a naive tracker may end up learning the appearance of the occluding object. Our work introduces a method of detecting occlusion by considering the color profile of the target to prevent inappropriate tracker updates while the target is occluded. We show improved overlap and central location precision with three visual trackers when adding our hue-based occlusion detection to each tracking system.

***Index Terms*—** Tracking, color, correlation filters, occlusion

1. INTRODUCTION

Visual tracking – the task of following a target in video – is an important computer vision problem with numerous real-world applications. As tracking performance has improved through research advances, the difficulty of problems being considered has also increased. A very simple yet challenging paradigm in visual tracking is to track a target in an unconstrained environment based on an initial single image of that target. Target appearance can change due to factors such as scale and illumination changes, object rotations, or a target going out of view or getting occluded by other objects.

When a tracker must successfully operate across a wide range of videos without any calibration for individual sequences, trade-offs are necessary. Changes to a tracker may help on 20% of videos while harming the performance on 10% of other videos; this is to be expected and is largely unavoidable. One particular trade-off which must be made is that between a tracker's adaptability and a tracker's stability relative to noise or other objects in the scene. If a target undergoes quick changes, it makes most sense for a tracker to update its model rapidly to adapt to the target's shifting appearance. In other videos, the target may be occluded for a stretch of time, during which the same tracker with an aggressive adaptation rate may end up quickly learning the appearance of the occluding object and drift off the original target, even as it reappears.

Correlation filters (CFs) have become a popular approach for the challenging one-shot visual tracking problem[1]. CFs can be computed efficiently in the Fourier domain, often allowing correlation tracking to run faster than real time. Additionally, the training (and retraining) of CFs implicitly includes the area surrounding the target as negative training examples, thus giving the tracker more discriminative power. The MOSSE filter [2] originally proposed a CF design that allowed for efficient retraining with each new frame processed in a tracking sequence. More recently, the CSK tracker [3] presented an efficient kernelized correlation tracker. The KCF [4] and ACT [5] trackers extend correlation trackers to multi-channel features instead of only using intensity features. The DSST tracker [6] added an additional 1D CF to estimate a target's scale. All these correlation trackers improve upon previous works, but all still suffer from the problem of a fixed learning rate across all videos. Because of this, they are all prone to locking on to an occluded object and losing track of the actual target if the issue is not addressed.

Some trackers try to handle the problem of lost tracks by performing redetection during the course of tracking. The TLD tracker [7] operates with a tracker that provides high-confidence training examples to a detection model, and the tracking and detection models work together to correct the final tracking output. In correlation tracking, the LCT tracker [8] runs a random fern detector when the correlation tracker model outputs an estimate with low confidence. Both the TLD and LCT trackers work to recorrect tracking results with low confidence, but neither specifically address or estimate if the target is occluded. The CCT tracker [9] has a tracking model and a detection model, and suggests that the target is occluded when the two models disagree entirely. When this decision is made, the learning rate of the tracking model is reduced by a factor of 10. The intuition is that while a tracker may estimate the target to be where an occluding object is, a detector that runs independent of the target's previously esti-

mated location will not. While this is a valid approach, the CCT tracker is not directly trying to detect or estimate the presence of occlusion by the content of the target region estimated by the tracker, but rather it is estimating occlusion by measuring the disagreement between two models.

In contrast, this work proposes a technique for inspecting the target region estimated by an arbitrary tracker, and then adapting the tracker's learning rate based on that information. We propose that easily computed color features will be a powerful indicator of occlusion in color videos, and by utilizing those features we can modify the tracker's learning rate to avoid the problem of adapting to the appearance of an occluding object.

2. TRACKING-BY-DETECTION FRAMEWORK

The objective of visual tracking is to follow a target as it moves during a video sequence. Oftentimes only the bounding box in the initial frame is provided to the tracking system. Trackers must learn the appearance of the target from a single training example, and detect and adapt to a changing target appearance through the rest of the video.

Correlation tracking has been frequently used for the problem of single-frame training, because it can implicitly learn the appearance of the target while using the surroundings as negative training examples, and this process can be done efficiently in the Fourier domain. We will be illustrating our occlusion detection on three publicly available correlation trackers: the CSK [3] and KCF [4] trackers, which estimate only the central location of the target at a fixed scale, and the DSST [6] tracker, which attempts to estimate the scale as well as the central location of a target. All three trackers follow the same fundamental process. After learning the appearance of the target on the first frame, the tracker detects the most likely location (and in the case of DSST, the most likely scale) of the target in the next frame and retrains the model with the new, imperfect detection. Retraining is done by linear interpolation, with a small weight given to the new information relative to the previously learned model (the three trackers give between 2-7.5% weight to the new frame).

It is important to note that the model is updating every frame; there is no measure of confidence or tracking quality used to decide if the model should be updated. In frames where the object is occluded and there is no "correct" location, the model ends up learning the appearance of something other than the original target. In a few frames, this may only degrade the quality of the tracker, but given a long enough series of frames with occlusion, the model will drift off of the original target and begin to track the occluding object.

3. OCCLUSION DETECTION FRAMEWORK

As mentioned in Sec. 2, many correlation trackers do not adapt the learning rate according to any feedback from the tracking system. In practice, this is difficult; lower confidence in the tracking results may be the result of occlusion, but it may also indicate that the appearance of the target is changing rapidly. In the case of occlusion it would be desirable to reduce or completely stop model adaptation, whereas increasing the rate of model adaptation would be desirable in the case of a rapidly changing target appearance. Feedback from correlation trackers makes distinguishing between these two scenarios very difficult, which motivates the need for a separate way of identifying occlusion.

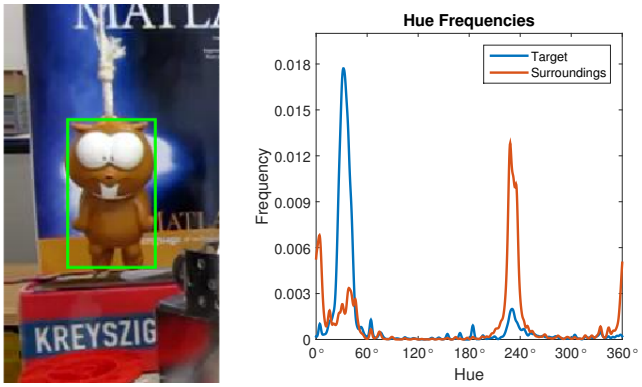

Fig. 1: Portion of initial frame of 'lemming' track with target shown (left), and probability distribution of the hues found in the initial frame (right). The two largest peaks correspond to brown and blue hues.

The video tracking problem consists of tracking objects with difficulties including in-plane and out-of-plane rotations, scale changes, and illumination variation [1]. Given all the possible changes a target may undergo after the tracker is initialized with a single training image, we suggest that texture features will not be sufficient in distinguishing between an occluding object and a rapidly changing target appearance. Rather, we propose that the *hue profile* of a target will remain static over the course of a short video, and if it is distinct from the target's surroundings, will discriminate between a tracker following the target and a tracker that has lost the target behind an occluding object. When occlusion has been detected, the target can halt model adaptation until the object reappears.

3.1. Training

Just as the tracker is initialized on the single ground truth frame available, the target's hue profile is also learned solely on the first frame. Given a RGB video with an initial bounding box, the target and surroundings are converted from RGB values to hue values, according to the transformation from RGB to HSV color space. This transformation is chosen because hue values capture the pigment of the object better than any individual RGB value ('saturation' and 'value' capture the intensity and brightness of the pigment, which are prone to change due to shadows or other illumination changes).

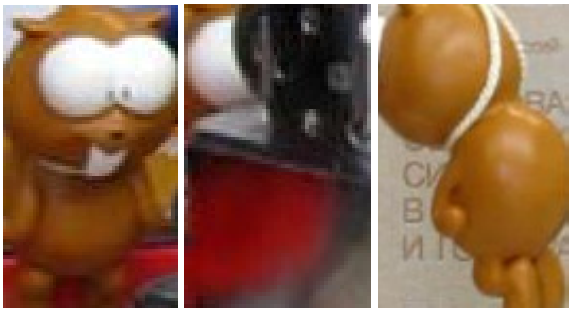

Fig. 2: Target regions based on ground truth labels in three 'lemming' track frames. Frame 296 (left) has $OS_{raw} = -0.73$, while the occluded target in frame 338 (middle) has $OS_{raw} = 1.13$. Frame 971 (right) has $OS_{raw} = -1.88$.

Figure 1 shows a portion of the first frame, as well as the probability distribution of the hues found in both the target region and surrounding region. These distinct hue probability distributions serve as the cue we use to identify occlusions. Because the hues from a single frame will provide a small and noisy sample, we smooth the probabilities with a Gaussian filter. From these distributions we compute the log-likelihood that a given hue belongs to the target as

$$\mathcal{L}(\text{target}|H) = \log\left(\frac{P(H|\text{target}) + \varepsilon}{P(H|\text{background}) + \varepsilon}\right) \quad (1)$$

where H denotes the hue value and ε is a small value to compensate for the large range of hues not present in the initial frame. In all our experiments we set $\varepsilon = 10^{-4}$.

3.2. Occlusion Detection

The scope of our occlusion detection is solely to adjust the learning rate of a tracking system. In each frame, the tracker will find the most likely location (and possibly the most likely scale). The target location in the nth frame can be denoted as (x_n, y_n, w_n, h_n), where (x_n, y_n) represent the top-left corner of the rectangular region, and (w_n, h_n) represent the width and height of the region in the nth frame. We compute a raw occlusion score (OS) by averaging the hue likelihoods of the pixels in that region as follows:

$$OS_{raw}(n) = -\frac{\sum_{i=x_n}^{x_n+w_n}\sum_{j=y_n}^{y_n+h_n}\mathcal{L}\left(\text{target}|H(i,j)\right)}{w_n h_n} \quad (2)$$

Examples of high and low raw occlusion scores shown in Figure 2 demonstrate how the scores are easy to interpret.

Beyond learning the likelihood of hues belonging to the target as shown in Eq. 1, we also use the initial frame to normalize occlusion scores of subsequent frames as follows:

$$OS(n) = OS_{raw}(n) - OS_{raw}(1) \quad (3)$$

Normalizing $OS(n)$ by the raw occlusion score from the ground truth frame compensates for differences between targets from different sequences and makes it easier to pick a single decision threshold across all videos.

Once the $OS(n)$ is calculated for a given tracking result, the decision to change the learning rate or not must be made. In our experiments, we use a very simple scheme. If $OS(n)$ is higher than some threshold, the learning rate is set to zero and the tracker model will not be updated. If $OS(n)$ is below the threshold, the learning rate of the tracker remains at its original setting.

4. EXPERIMENTAL SETUP

We demonstrate our hue-based occlusion detection on a subset of the CVPR 2013 Visual Tracker Benchmark [1]. We use a subset of 20 video sequences containing 21 targets. This subset contains all RGB sequences containing occlusion.

All three trackers tested [3, 4, 6] are unmodified from their available open-source versions, with the exception of the occlusion detection that can affect the learning rate in a given frame. All experiments are performed on an Intel(R) Core(TM) i5-4210U CPU using a single core at 1.70 GHz with 8G RAM. The computation spent on occlusion detection results in a moderate decrease in speed across the three trackers. The CSK tracker moves from 168 to 127 frames per second (FPS), the KCF tracker moves from 118 to 102 FPS, and the DSST tracker drops slightly from 20.8 to 19.3 FPS.

The occlusion detection has a different threshold for each tracker, but this value is constant across all videos for the given tracker. With different feature descriptors, different components, and most importantly different learning rates between the three trackers, tuning the occlusion detection separately for each one is not unexpected.

5. RESULTS

Overall results showing the impact of our hue based occlusion detection can be seen in Figure 3. There are noticeable improvements when using the CSK and DSST trackers, and a small improvement when using the KCF tracker. Given the small reduction in tracking speed when using the occlusion detection, the moderate improvements to tracking quality appear to be a favorable trade-off.

A strictly quantitative analysis of tracking performance can provide an incomplete picture of the improvements or problems introduced by our occlusion detector. The quantitative results shown in Figure 3 are calculated on a per frame basis for each sequence. This means that successfully handling occlusion (or any challenging factor) at the end of a sequence will be given significantly less weight than successfully handling an occlusion present at the beginning of a video.

Table 1 shows tallies of human-labeled "errors," which can simply be understood as instances where the tracker com-

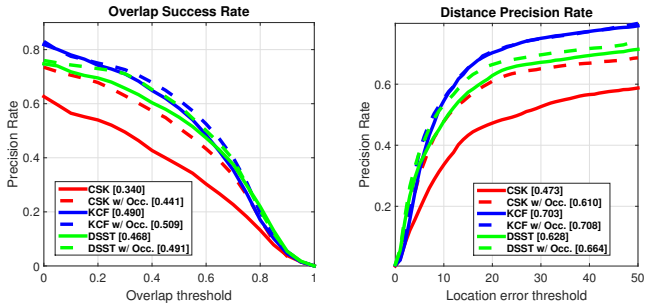

Fig. 3: Overlap success and central location precision plots on subset of benchmark dataset [1]. Results are from 21 sequences with color images and labeled occlusions. Numbers in bracket refer to area under the curve and 20 pixel error threshold precision, respectively.

Tracker	Tracking Errors		
	Total	Corrected	Introduced
CSK [3]	17	4 (24%)	1
KCF [4]	11	3 (27%)	1
DSST [6]	11	4 (36%)	1
Total	39	11 (28%)	3

Table 1: Qualitative results on 21 targets from 20 tracks with RGB images and tagged as having occlusion. Total errors refers to the baseline tracking performance, while the "fixed" and "introduced" columns represent errors that are corrected or brought on by use of the hue-based occlusion detection.

pletely loses the target. After a tracker makes one of these mistakes, any overlap with the target, or the exact central location error should be considered purely incidental. In contrast, a tracker that drifts of the center of the target, but still follows the target's movements in the sequence would not be labeled an error, despite producing suboptimal performance.

Table 1 shows that across the three trackers used, there are 39 significant errors in the sequences tested. A majority of these mistakes are directly tied to occlusions in these sequences, but the tally includes errors from all challenging factors (examples of these are found in the 'ironman' and 'matrix' sequences). Of these 39 errors, 11 are corrected with the hue-based occlusion detection. Of these 11 corrected errors, 10 can be attributed to occlusion. In contrast, the addition of the occlusion detection only introduces 3 errors into the tracking results. On a varied set of sequences where any change to the tracking system can expect to result in gains and losses on different videos, an 11-3 tradeoff is very encouraging for such a simple scheme of finding occlusion and changing a tracker's learning rate. Figure 4 shows three examples of errors that were corrected with the addition of the occlusion detection, and one of the three errors that were introduced (in fact, two of the three errors introduced were on the 'skating1' sequence).

(a) 'jogging' sequence, CSK tracker

(b) 'lemming' sequence, KCF tracker

(c) 'girl' sequence, DSST tracker

(d) 'skating1' sequence, KCF tracker

Fig. 4: Examples of corrections (a-c) due to occlusion detection, and one error (d). Red denotes original tracker and green denotes occlusion detection is included. Note the change in the stage lighting in the 'skating1' sequence.

6. CONCLUSIONS

Our work introduces a simple yet effective process for identifying occlusion of tracking targets in RGB videos. Despite the simplicity of the technique and the fact that it is only used to alter the learning rates of the tracker, the hue-based approach shows quantitative improvement on all 3 correlation trackers tested and shows that many errors are corrected upon inspection of individual tracking outputs. These performance gains come at very little cost to the tracking speed of each system.

Extending the use of the hue-based occlusion detection is worth exploration. The importance of pixel hues can be weighted by the spatial location within a bounding box, or by the saturation of the particular hue. Additionally, while we only use the occlusion detection to affect the tracker's learning rate, the promising results show that it may be worth investigating its use in estimating the track itself.

7. REFERENCES

[1] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang, "Online object tracking: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[2] David S Bolme, J Ross Beveridge, Bruce Draper, Yui Man Lui, et al., "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 2544–2550.

[3] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *proceedings of the European Conference on Computer Vision*, 2012.

[4] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2015.

[5] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 2014, pp. 1090–1097.

[6] Martin Danelljan, Gustav Häger, Fahad Khan, and Michael Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*, 2014.

[7] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas, "Tracking-learning-detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 7, pp. 1409–1422, 2012.

[8] Chao Ma et al., "Long-term correlation tracking," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[9] Guibo Zhu, Jinqiao Wang, Yi Wu, and Hanqing Lu, "Collaborative correlation tracking," in *Proceedings of the British Machine Vision Conference (BMVC)*, September 2015, pp. 184.1–184.12.