Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 39 (2014) 162 – 165

Procedia Computer Science

6th International conference on Intelligent Human Computer Interaction, IHCI 2014

Fusion of tracking techniques to enhance adaptive real-time tracking of arbitrary objects

Peter Poschmann[a,*], Patrik Huber[b], Matthias Rätsch[c], Joseph Kittler[b], Hans-Joachim Böhme[a]

[a] *University of Applied Sciences Dresden, Friedrich-List-Platz 1, D-01069 Dresden, Germany*
[b] *University of Surrey, Centre for Vision Speech and Signal Processing, Guildford, Surrey GU2 7XH, United Kingdom*
[c] *Reutlingen University, Alteburgstraße 150, D-72762 Reutlingen, Germany*

Abstract

In visual adaptive tracking, the tracker adapts to the target, background, and conditions of the image sequence. Each update introduces some error, so the tracker might drift away from the target over time. To increase the robustness against the drifting problem, we present three ideas on top of a particle filter framework: An optical-flow-based motion estimation, a learning strategy for preventing bad updates while staying adaptive, and a sliding window detector for failure detection and finding the best training examples. We experimentally evaluate the ideas using the BoBoT dataset[a]. The code of our tracker is available online[b].

Peer-review under responsibility of the Scientific Committee of IHCI 2014
Keywords: Adaptive tracking; Particle filter; Optical flow; Sliding window

1. Introduction

The goal of visual tracking is to determine the location of a target in each frame or to detect its disappearance. If the target is known beforehand, a classifier can be trained with all appearances and a reasonable number of negatives. However, if the target appearance is unknown or somehow not involved in the classifier training, then it might be missed. Algorithms that update the classifier while tracking are able to follow the target in these situations, adapting to the appearance, background, and recording conditions. This is useful in HCI scenarios, e.g. where a robot has to keep track of the person it is interacting with regardless of pose, (out-of-plane) rotations, illumination changes etc.

Adaptive trackers usually start with a single annotated frame that indicates the location of the target, e.g. by using a bounding box. The goal of the tracker is to estimate the new target position in each of the following frames and to detect when the target is missing. If the tracker does not adapt itself, then it might lose the target, but if it does, then it will introduce errors with each update [1]. We refer to Wu et al. [2] for a recent survey of adaptive tracking algorithms.

[a] "Bonn Benchmark on Tracking", http://www.iai.uni-bonn.de/~kleind/tracking/
[b] http://adaptivetracking.github.io/
* Corresponding author.
E-mail address: poschmann@htw-dresden.de

Peer-review under responsibility of the Scientific Committee of IHCI 2014
doi:10.1016/j.procs.2014.11.025

Typically, adaptive tracking algorithms use the estimated target position as the positive training example[3,4,5], but there are also approaches that use semi-supervised learning algorithms to determine the optimal positive training examples[6,7]. STRUCK[8] avoids the problem of assigning binary labels (target vs. background) altogether by learning the target displacement using structured output prediction. Because of costly computation, STRUCK only searches at the initial scale and therefore cannot adapt the bounding box size after the first frame. The tracker of Klein and Cremers[3] builds upon a particle filter for estimating the target state. The classification confidence of the target position is used to decide whether a boosted classifier is updated. Tracking-Learning-Detection (TLD)[4] combines an optical-flow-based tracker and a sliding window detector that correct each other's errors. Supančič III and Ramanan[9] proposed a tracker that re-evaluates past decisions and corrects errors made in previous frames, but was not designed to run in real-time. A very fast tracker was presented by Kolarow et al.[5]. To achieve more than real-time speed, they reduced the object model to a single sparse template that is created anew in every frame unless an occlusion is detected.

The core contribution of our work is the fusion of a particle-filter-based adaptive tracker with three enhancements and evaluating their influence on the tracking performance. (1) *Optical flow* incorporates the current measurement into the prediction, which leads to a better proposal distribution, so the particles can follow the target more closely even under rapid movements. (2) The introduction of a simple *learning condition* reduces drift by only updating with confident target locations. (3) Adding a *sliding window detector* increases the quality of the negative training examples, while also enabling fast re-detection and failure detection. By combining these ideas on top of an adaptive particle filter framework we obtain a robust real-time tracking algorithm.

2. Tracking algorithm and its extensions

Our baseline algorithm is similar to the one of Klein et al.[10]. The main differences are in the motion model and choice of features and classifier. The tracking system estimates the state $\mathbf{x} = (x, y, s, \dot{x}, \dot{y}, \dot{s})^T$ of the target, which consists of position, size and change of these. The aspect ratio is fixed and will be set at the initial frame. A particle filter[11] estimates the probability distribution of the target state at time t using a set of particles $S_t = \{s_t^k\}$, $k \in \{1, \ldots, n\}$. Each particle $s_t^k = (\mathbf{x}_t^k, \pi_t^k)$ consists of a state $\mathbf{x}_t^k$ and an importance factor (weight) π_t^k, which is computed by the measurement model using the current frame. The target state is then calculated as the weighted mean over all particle states $\bar{\mathbf{x}}_t = \frac{1}{n} \sum_{k=1}^{n} \pi_t^k \mathbf{x}_t^k$. If the classifier score of the estimated target position falls below a threshold, the target is considered lost. This may happen in the case of occlusions, leaving the field of view or drifting away from the real target. Next, we describe three essential parts of the tracker and the enhancements on top of them.

2.1. Motion model

In the baseline tracker, we apply a simple constant velocity motion model to predict the new target state before incorporating the new measurement. The first extension is to estimate the actual motion of the target by computing the optical flow, resulting in a more accurate *optical-flow-based* motion model. We use the method of Kalal et al.[12], where the flow between the previous and current frame is estimated by a regular grid of points within the target bounding box. To reduce the likelihood of the points capturing the background, we changed the grid to an inset circle-like shape.

2.2. Classifier update

The classifier is re-trained using supervised learning. There are two key assumptions for generating new labeled training data: the smoothness of the trajectory and the uniqueness of the target. If the target's movement is fairly smooth, the tracker is able to follow it closely, which makes it possible to extract new positive training examples from the estimated target position in each frame. If the target is considered to be unique, then any training example extracted from the remainder of the frame has to have a negative label. As long as the tracker provides a target position, the classifier can be re-trained using the new training examples from the current frame. To prevent adapting to wrong objects such as occluders, we do not update the classifier if the estimated target location is classified as negative.

Updating the classifier immediately after each frame keeps it up-to-date and leads to a quick adaptation to changes, but also drifts away quickly in case of erroneous updates. Extending the tracker with a *learning condition*[3] prevents updates in uncertain situations. Our base tracker already has a weak learning condition, as our classifier is only

updated if the estimated target location is being classified as positive, which is equivalent to an SVM score threshold of 0. With this extension, we employ a higher threshold which leads to more conservative updates, so fewer frames are used to select new examples for re-training the classifier, and the likelihood for adding bad examples is reduced.

For training the classifier we use libSVM[13]. Because it uses batch learning, the time needed depends on the size of the training set. To limit the update time, we restrict the size of the training set[10], so there are at most 20 positive and 100 negative examples. The oldest negative training examples are replaced by strong negatives sampled from the current frame, while positive examples are replaced based on their classification confidence.

2.3. Measurement model

To compute the importance factor of a particle, we first extract 13-dimensional extended HOG features[14] from a gray-scale image pyramid. These features are fairly invariant to out-of-plane rotations, as they capture the shape of an object. A linear support vector machine (SVM) then computes a score $d_t^k = w_t \cdot \phi_t(s_t^k) + \rho_t$ for the particle s_t^k, where w_t is the weight vector and ρ_t is the bias of the SVM, and ϕ_t extracts the features from frame t. The last step is to transform this score into the measurement probability $\pi_t^k = \eta(1 + e^{-\lambda d_t^k})^{-1}$ of the particle using a sigmoid function with parameter λ controlling the slope. The normalization factor η ensures that all particle weights sum up to one.

The third extension to our tracking algorithm is a combination with a sliding window detector. The latter is responsible for a fast re-detection in case of a tracking failure or after occlusions, and finding the strongest negative training examples in the background without relying on random selection. A disadvantage is the additional computational burden. To accommodate this, the *sliding-window-based* measurement model can get the SVM score directly from the responses generated by the sliding window approach instead of computing features and score on a per-particle basis. That even speeds up the whole tracking compared to the original version, because when using a linear SVM, computing the classifier responses over the image is just a convolution on each layer of the feature pyramid. The disadvantage is a coarser measurement model, as the resolution of the measurements then is a HOG cell instead of a pixel. This also leads to a worse quality of the positive training example, because it might be few pixels off, which results in an inaccurate classifier and the tracker will eventually drift. Therefore, we extract the features of the estimated position directly from the gray-scale image pyramid, just as without the addition of the sliding window approach.

3. Experiments

The evaluation is done using the BoBoT dataset[10,3], consisting of 13 image sequences. They show pedestrians and arbitrary objects, featuring challenging conditions like lighting changes, partial and full occlusions, out-of-plane rotations, changing background, and distracting objects. The overlap of the bounding boxes (ratio of intersection to union) is computed for each frame and averaged over the whole sequence, resulting in a score. To compensate for the random nature of particle filters, each sequence was tested 20 times with our tracker and the results were averaged.

The optical flow (OF) improves the prediction of the new target position, but fails if the target is subject to out-of-plane rotations (seq. A) or is partially occluded (seq. I, Ja). The learning condition (LC) reduces bad updates. This is especially notable when combining both ideas (OF+LC), as optical flow leads to the particles following the occluder, but the score threshold prevents the adaptation and enables re-detection of the correct target after the occlusion. Both extensions increase the performance. When adding the *sliding-window-based* measurement model on top (OF+LC+SW), the tracker runs more than twice as fast (60 fps compared to 25 fps), but the performance decreases on average. This is because the localization accuracy suffers from the coarse resolution of the measurements, which consequently decreases the quality of the positive training examples.

We compare our variations with the adaptive particle filter tracking of Klein and Cremers[3], the color based hyper-real-time tracker of Kolarow et al.[5], TLD of Kalal et al.[4] and the self-paced learning based tracker (SPLTT) of Supančič III and Ramanan[9]. For the first two trackers, we took the results from the respective papers. The last four scores of[5] are missing, because the evaluation was done using an older version of the dataset. For better comparison we additionally show the average scores of the first nine sequences. For the other two trackers, we used the code that is openly available[c]. Detailed results are shown in Table 1.

[c] https://github.com/zk00006/OpenTLD and https://github.com/jsupancic/SPLTT-Release

Table 1: Comparison of tracking algorithms. BT = Base Tracker, OF = Optical Flow, LC = Learning Condition, SW = Sliding Window

Seq.	average overlap [%]								
	BT	OF	LC	OF+LC	OF+LC+SW	Klein [3]	Kolarow [5]	Kalal [4]	Supančič [9]
A	76.6	53.4	71.5	80.2	80.3	62.0	69.2	46.7	80.8
B	72.9	74.2	77.3	81.6	81.7	83.7	80.4	58.4	74.5
C	76.9	76.6	76.8	77.1	71.1	92.1	67.9	64.6	62.4
D	68.4	79.6	74.7	79.9	61.3	80.9	80.6	67.6	72.8
E	79.1	81.5	80.2	84.5	71.3	85.5	86.0	79.3	78.9
F	53.7	58.4	55.0	58.6	58.6	64.4	56.1	47.9	57.5
G	59.5	63.4	55.4	63.1	31.2	74.5	87.1	59.0	45.6
H	84.3	87.2	86.3	88.4	88.2	96.1	98.3	86.1	86.0
I	62.0	51.1	71.3	69.7	67.8	83.3	77.7	43.3	57.3
Ja	48.3	42.4	75.8	80.4	78.6	83.1	-	35.1	58.8
Jb	68.2	71.6	74.3	77.7	81.9	80.6	-	53.0	58.3
K	49.2	69.5	49.5	68.8	77.0	84.1	-	55.1	69.2
L	71.8	80.5	74.6	81.4	63.8	85.8	-	39.4	63.2
avg (A-I)	70.4	69.5	72.1	75.9	67.9	80.3	78.1	61.4	68.4
avg (A-L)	67.0	68.4	71.0	76.3	70.2	81.2	-	56.6	66.6

While we could increase the performance of our tracker, it does not outperform all the others. The goal of Kolarow et al. was to create a very fast tracker, so they did not aim for re-detectability after long occlusions. This disadvantage is not reflected within the results, because the dataset does not contain a sequence with this problem. The results for Kalal et al. and Supančič III and Ramanan are a bit below the others, however they did not optimize their parameters towards the BoBoT dataset, as the others have done while testing different variations of their trackers against it.

4. Conclusion and further work

We proposed the fusion of a particle filter with other tracking techniques to increase the robustness of adaptive tracking. We compared the variants and showed the applicability of the resulting tracker in real-world scenarios. While the *optical-flow-based motion model* and the *learning condition* improve the performance, the *sliding-window-based measurement model* increases the efficiency. The choice of the learning threshold is quite crucial - if it is too low or too high, then there might be bad updates or none at all, depending on the conditions of the video. Therefore, in future work we will explore the possibility of having an adaptive threshold or find other ways around this problem.

References

1. Matthews, I., Ishikawa, T., Baker, S.. The template update problem. *IEEE Trans on PAMI* 2004;**26**(6):810–815.
2. Wu, Y., Lim, J., Yang, M.H.. Online object tracking: A benchmark. In: *Proc. of CVPR*. 2013, p. 2411–2418.
3. Klein, D.A., Cremers, A.B.. Boosting scalable gradient features for adaptive real-time tracking. In: *Proc. of ICRA*. 2011, p. 4411–4416.
4. Kalal, Z., Mikolajczyk, K., Matas, J.. Tracking-learning-detection. *IEEE Trans on PAMI* 2012;**34**(7):1409–1422.
5. Kolarow, A., Brauckmann, M., Eisenbach, M., Schenk, K., Einhorn, E., Debes, K., et al. Vision-based hyper-real-time object tracker for robotic applications. In: *Proc. of IROS*. 2012, p. 2108–2115.
6. Stalder, S., Grabner, H., Van Gool, L.. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In: *Proc. of ICCV Workshops*. 2009, p. 1409–1416.
7. Babenko, B., Yang, M.H., Belongie, S.. Robust object tracking with online multiple instance learning. *IEEE Trans on PAMI* 2011; **33**(8):1619–1632.
8. Hare, S., Saffari, A., Torr, P.H.S.. Struck: Structured output tracking with kernels. In: *Proc. of ICCV*. 2011, p. 263–270.
9. Supančič III, J.S., Ramanan, D.. Self-paced learning for long-term tracking. In: *Proc. of CVPR*. 2013, p. 2379–2386.
10. Klein, D.A., Schulz, D., Frintrop, S., Cremers, A.B.. Adaptive real-time video-tracking for arbitrary objects. In: *Proc. of IROS*. 2010, p. 772–777.
11. Isard, M., Blake, A.. Condensation–conditional density propagation for visual tracking. *Int J on CV* 1998;**29**(1):5–28.
12. Kalal, Z., Mikolajczyk, K., Matas, J.. Forward-backward error: Automatic detection of tracking failures. In: *Proc. of ICPR*. 2010, p. 2756–2759.
13. Chang, C.C., Lin, C.J.. LIBSVM: A library for support vector machines. *ACM Trans on IST* 2011;**2**:27:1–27:27.
14. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.. Object detection with discriminatively trained part-based models. *IEEE Trans on PAMI* 2010;**32**(9):1627–1645.