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