2010 International Conference on Pattern Recognition

Interest point based tracking

Werner Kloihofer
Research and Development Image Processing
Center Communication Systems GmbH
Vienna, Austria
Email: w.kloihofer@centersystems.com

Martin Kampel
Computer Vision Lab
Vienna University of Technology
Vienna, Austria
Email: martin.kampel@tuwien.ac.at

***Abstract*—This paper deals with a novel method for object tracking. In the first step interest points are detected and feature descriptors around them are calculated. Sets of known points are created, allowing tracking based on point matching. The set representation is updated online at every tracking step. Our method uses one-shot learning with the first frame, so no offline and no supervised learning is required. Following an object recognition based approach there is no need for a background model or motion model, allowing tracking of abrupt motion and with non-stationary cameras. We compare our method to MeanShift and Tracking via Online Boosting, showing the benefits of our approach.**

***Keywords*-Tracking and surveillance; Object detection and recognition**

I. Introduction

Object tracking algorithms are tools for the extraction of information from temporal image sequences. Computers allow for an automatic or semi-automatic analysis to help to deal with the acquired surveillance data. Trajectories estimated by object tracking are the basis for higher-level algorithms in automated surveillance or for instance in human-computer interaction – like controlling a computer program simply by moving an item in your hand.

With an interest point based method, we chose a method from the field of object recognition. While conventional tracking approaches focus on exploiting temporal coherence assumptions, an object recognition based approach is less influenced by abrupt object motion, which occurs with low frame rate video (LFR) material or non-stationary cameras. Objects move too far from one frame to the next, therefore it is necessary to use methods which are able to identify objects without relying on motion prediction only. Low frame rate (LFR) sequences are characterized by abrupt change of appearance and fast motion. We consider 5 frames per second as low frame rate, which also corresponds to the definition used in [1].

A comprehensive survey of object tracking is given in Yilmaz et al. [2]. Using their taxonomy, tracking methods can be divided into 3 groups: Point tracking, kernel tracking and silhouette tracking. Our method belongs to the kernel tracking section, even though we are working with point descriptors instead of an object region description.

One way to reduce computational complexity with local descriptors is to apply them only to a subset of the image. The points in this subset are called interest points. A comprehensive summarization of interest point detectors is given in [3] and [4]. The first step of such a method is to locate appropriate interest points. The repeatability of this procedure is an important quality criterion. To allow successful matching between different images it is necessary to detect key points at exactly the same locations. Then a feature descriptor is locally calculated at these points. The matching step compares two images with detected feature vectors around interest points. One simple option is to compare all pairs of interest point descriptors using Euclidean distance.

We chose SURF [5] as the basis for our tracking algorithm since it provides a better performance compared to SIFT while allowing faster calculation, which is necessary for developing a real-time tracking algorithm. SURF is already used for tracking by a recent publication of Ta et al. [6], but the authors focus more on the efficient organization of the keypoint extraction, while we take the keypoints and their feature descriptors as given. In [7] a workflow similar to ours is used, but with an online EM algorithm for modeling the relation between point and object motion and a maximum likelihood method to estimate object motion. Moreover they incorporate object structure with a graph matching approach.

Another way to solve the problem of having to calculate descriptors for the whole image is to use Online Boosting [8]. In this method, features are only calculated for a small fraction of a larger underlying feature space by doing online feature selection. This allows fast classification and updating without having to restrict the number of possible features. Tracking with Multiple Instance Learning [9] follows a similar approach.

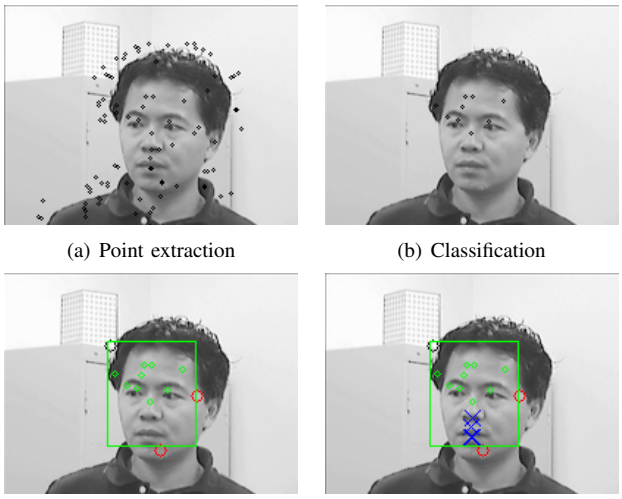
In Section 2 we first describe our tracking method. Next we show our experiments in Section 3. We conclude in Section 4 with a summary and an outlook on future work.

II. Object Tracker

Given a bounding box for initialization and a set of interest points $p_{0..k}$ with their corresponding feature vectors

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.866

$f_{0..k}$ for the first frame, we separate the interest points into two sets S_{fg} and S_{bg}, with an initial weight of w_{init}. The two sets of interest points with their feature descriptors and their weights are the learned knowledge on which tracking is based. Additionally we calculate the relative positions within the object's bounding box for all points in the foreground set, in order to determine the new bounding box based on a set of points. This knowledge is updated in every following tracking step to allow adaption to changing object appearance. Our sets of foreground and background points have no fixed size, but are limited by the upper bounds $n_{fg,max}$ and $n_{bg,max}$.



(a) Point extraction (b) Classification

(c) Bounding box and weight update (d) Replacing points

Figure 1. Workflow

First of all, the interest points are divided into three pairwise disjoint sets S_{pos}, S_{neg} and $S_{unmatched}$ (and $S_{pos} \cup S_{neg} \cup S_{unmatched} = E$). This is done by matching them to the foreground and background sets S_{fg} and S_{bg}.

After these sets are created the bounding box R_t is determined. This is done by calculating the upper left corner of the rectangle using the relative positions of the interest points, which are known from the first frame in which they were added to the set S_{fg}. For all points in S_{pos}, which corresponds to a subset of S_{fg} because of the matching step, the upper left corner of the rectangle is predicted. The average of those predictions is used for determining the bounding box R_n with the size of R_{init}.

Using this bounding box, all points in S_{pos} are classified into 2 sets TP (true positives) and FP (false positives) according to their location. The weights of all matched points in TP are updated using $w = w + w_{TP}$. The points in S_{neg} are also classified into FN (false negatives) and TN (true negatives) and the weights of the points in TN are updated using $w = w + w_{TN}$. The points in FP and FN receive a penalty by subtracting w_{FP} or w_{FN} respectively. After this step, all weights in both sets S_{fg} and S_{bg} are decreased using $w = w - w_{Cool}$. This cooldown makes sure that points which are rarely matched are getting a low weight which allows replacing them later on.

After these classification steps are done the update step is performed. All matched points which were correctly classified get their feature vectors updated using $f_t = \alpha f_t + (1 - \alpha) f_{t-1}$. Then new candidates are inserted into the sets S_{fg} and S_{bg} by either adding them, when the set has not reached $n_{fg,max}$ or $n_{bg,max}$ yet, or by replacing existing points. For replacement in S_{fg} three input parameters are important: The number of replacements to be made n_{torepl}, the set of new candidate points P_{new} and the set of candidates for replacement P_{old}. Then $\min(n_{torepl}, |P_{new}|, |P_{old}|)$ elements are replaced by choosing randomly (uniformly distributed) from the sets P_{new} and P_{old}.

n_{torepl} is calculated with the goal of filling up the number of positively classified interest points (TP) in every frame to the number of initial interest points (TP_{init}). This is done by using $n_{torepl} = \frac{TP_{init} - TP}{2}$. If TP is greater than TP_{init}, n_{torepl} is simply set to zero. Factor two is included to learn more conservatively by replacing fewer points.

P_{new}, which is the set of candidates to replace existing interest points, is a specific subset of $S_{unmatched}$, namely all those points that are within the rectangle R_t and which are, in feature space, more than g_{thresh} away from the closest background interest point.

P_{old} are points in S_{fg} which have a weight lower than w_{min}.

Candidates for S_{bg} are added to the set, then set items are removed using roulette wheel selection according to their weights until $|S_{bg}| = n_{bg,max}$ is reached again.

In figure 1 we show an overview of the algorithm workflow.

III. Results

Collins et al. [10] define an evaluation framework including simple **metrics** with overlapping bounding boxes, which is basically all we need. Based on their metrics we define the following: The tracking rate is the number of frames in which the object has been correctly tracked, divided by the number of frames N in the sequence. Correctness is defined in terms of overlapping bounding boxes of the ground truth bounding box $R_{GT,i}$ and the tracker bounding box R_i.

$$TR = \frac{\sum_{i=0}^{N-1} x_i}{N}$$

$$x_i = \begin{cases} 1, & \text{when } R_{GT,i} \cap R_i > t_{ov} \max(R_{GT,i}, R_i) \\ 0, & \text{otherwise} \end{cases}$$

The maximum function was chosen instead of the union to keep the penalty of a deviation of the bounding box smaller. For the **evaluation** we use the videos provided by David

Ross et al. [11][1], since they show a single object and one is recorded with a non-stationary camera. The videos are also used in [8]. Ground Truth was created using Viper GT[2]. Figure 2 shows samples from the evaluation sequences.

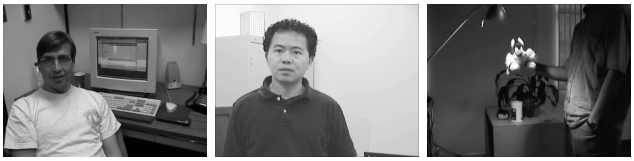


Figure 2. Samples from Dudek, Ming-Hsuan and Sylvester video datasets

Our IPTracker method is compared with our implementation of the Online Boosting method [8] and with the mean shift tracker [12]. We are doing the comparison with 3 sequences, namely Dudek, which shows a moving camera and therefore moving background, as well as Plush Toy and Ming-Hsuan.

[%]	Dudek	Ming	Sylvester
IP	99.83	98.95	100.00
OB	91.40	*99.08	92.56
MeanShift	89.05	*83.09	83.93

Table I
RESULTS OF TRACKER COMPARISON, TRACKING RATE IN %

Table I shows the results of this evaluation step. The reason for not reaching 100 % in the Ming Sequence is that in the ground truth the object is still marked while it is leaving the scene and therefore just partly visible, but the trackers lose the object when parts of the bounding box are out of the image area.

The **Online Boosting** Tracker is based on [8], but uses Haar features only. For the OnlineBoosting and MeanShift algorithm with the Ming video the result had to be adjusted (marked with a * in the table): The tracker has no handling for object loss, it keeps tracking even when the object has left the scene. Therefore we did not count the frames where no object was present.

Now the performance is compared using **low frame rate** sequences. For this purpose the input videos were subsampled to 5 FPS.

Table II shows that the IPTracker does not perform well with the LFR videos due to the keypoint extraction method. While with the non-LFR videos the minimum of a 20-frame average is at least 11, in LFR situations it is 8.75 for the Dudek Sequence, 9.35 for the Ming sequence and only 2.4 for the Sylvester sequence, which explains the tracking loss

[1]Videos are available at http://www.cs.toronto.edu/~dross/ivt/, last visited 6th May 2010

[2]see http://viper-toolkit.sourceforge.net/docs/gt/, last visited 6th May 2010

[%]	Dudek LFR	Ming LFR	Sylvester LFR
IP	82.29	81.06	40.63
OB	75.52	*99.60	97.32
MeanShift	73.96	*84.13	82.59

Table II
RESULTS OF TRACKER LFR COMPARISON, TRACKING RATE IN %

after 40 % of the video. In this sequence there are 18 frames without a match at all. In other words, the object appearance changes too much so that the keypoint method cannot find matches anymore even from one frame to the next. The correlation between tracking quality and matching performance is evident. The tracker cannot compensate for the problems of the underlying keypoint extractor. The results of this analysis can be seen in table III.

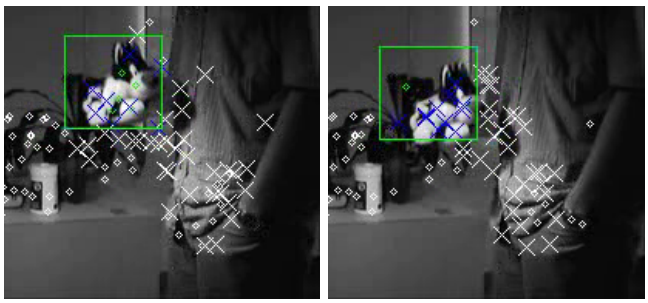


Figure 3. IPTracker LFR tracking problem with Sylvester sequence

Figure 3 shows frames 90 and 91 of the Sylvester sequence, 5 frames before the tracking loss occurs. While there are many new keypoints added to the set S_{fg} in the first frame (marked with blue crosses within the marked bounding box) there is only one keypoint in the latter frame which was classified positively (marked with a green circle within the bounding box).

	Dudek	Ming	Sylvester
Normal Average	24.88	23.54	19.64
Normal Minimum	11.10	15.75	11.55
LFR Average	17.96	18.11	7.91
LFR Minimum	8.75	9.35	2.40

Table III
MATCHING PERFORMANCE

For the OnlineBoosting and MeanShift algorithm with the Ming video the result was once again adjusted (marked with a * in the table). The LFR situation does not make much of a difference for the MeanShift method, only with the Dudek sequence the object is lost for the last 30 frames. The reason for not being influenced by LFR is that the histogram features used with MeanShift seem to not be that sensitive to appearance changes; tracking is still possible because of the smaller search space due to the mean shift procedure.

The **long term stability** of the algorithm is evaluated by starting an image sequence in a loop, playing the video every second time in reverse to avoid jumps. We did this evaluation with both the Dudek and the Ming sequence. In the Dudek sequence the tracking was still stable after processing more than 37000 frames. The Ming sequence showed the same behavior, the object was also correctly tracked for more than 37000 frames.

We created an artificial **occlusion** of 50 % of the ground truth bounding box by overlaying a black bar over a period of 20 frames, furthermore we used the sequences Occluded Face and Occluded Face 2[3] (see figure 4). The occlusions show no influence on tracking quality. This is caused by the interest point based concept, the only requirement is that there are enough remaining interest points (at least one in each frame). Invisible object parts do not cause problems in locating the object.

Figure 4. Sample screenshots with occlusions

To bring up the speed issue, the IPTracker reaches a **frame rate** of 9 frames per second for the 320×240 videos running on a single CPU core (with an Intel Core2 Quad with 4 × 2.6 GHz).

IV. Conclusion and Future Work

We have presented an interest point based tracking algorithm. The object appearance is learned using one-shot learning, the representation is in the form of sets of point descriptors which are updated online at every step. In our evaluation we have demonstrated the method's capabilities as a general-purpose tracking algorithm, as it is not specialized on any specific object type, there is also no need for learning, the only input required is a bounding box in the first frame. We have shown the long-term stability of our tracker with a video loop as well as stability under partial occlusions. The evaluation has also shown the limitations of the technique, which originate from the results of the interest point extractor. For future research we plan to do an evaluation with various interest point detectors and feature extractors to find out if there are methods which provide a significantly better matching performance. We have not experimented with deformable objects like pedestrians yet, though we expect correct tracking because of our experience with occlusions. Furthermore, we plan to compare our method to Tracking with Multiple Instance Learning, for which the authors provide both the code and their results on publicly available datasets. Applying Multiple Instance Learning or a Boosting based method to determine the interest point weights might improve our point selection process.

[3]Videos are available at http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml, last visited 6th May 2010

References

[1] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade, "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different lifespans," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[2] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[3] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: a survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.

[4] L. Szumilas, "Scale and rotation invariant shape matching," Ph.D. dissertation, Vienna University of Technology, Institute of Computer Aided Automation, Pattern Recognition and Image Processing Group, Vienna, Austria, 2008.

[5] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, vol. 3951/2006. Springer Berlin / Heidelberg, 2006, pp. 404–417.

[6] D.-N. Ta, W.-C. Chen, N. Gelfand, and K. Pulli, "Surftrac: Efficient tracking and continuous object recognition using local feature descriptors," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, 2009, pp. 2937–2944.

[7] W. He, T. Yamashita, H. Lu, and S. Lao, "Surf tracking," in *International Conference on Computer Vision*, 2009, pp. 1586–1592.

[8] H. Grabner, M. Grabner, and H. Bischof, "Real-time tracking via on-line boosting," in *British Machine Vision Conference*, vol. 1, 2006, pp. 47–56.

[9] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 0, 2009, pp. 983–990.

[10] R. Collins, X. Zhou, and S. Teh, "An open source tracking testbed and evaluation web site," in *IEEE Int. Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2005.

[11] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang, "Incremental learning for robust visual tracking," in *International Journal of Computer Vision*, vol. 77. Springer, 2008, pp. 125–141.

[12] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE Computer Society, 2002, pp. 603–619.