Image and Vision Computing 27 (2009) 1695–1701

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Anytime learning for the NoSLLiP tracker

Karel Zimmermann [*], Tomáš Svoboda, Jiří Matas

Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, Center for Machine Perception, Karlovo nam. 13, Prague, Czech Republic

ARTICLE INFO

Article history:
Received 9 January 2008
Received in revised form 12 February 2009
Accepted 11 March 2009

Keywords:
Tracking
Learning
Real time

ABSTRACT

We propose an anytime learning procedure for the Sequence of Learned Linear Predictors (SLLiP) tracker. Since learning might be time-consuming for large problems, we present an anytime learning algorithm which, after a very short initialization period, provides a solution with defined precision. As SLLiP tracking requires only a fraction of the processing power of an ordinary PC, the learning can continue in a parallel background thread continuously delivering improved, i.e. faster, SLLiPs with lower computational complexity and the same precision.

The proposed approach is verified on publicly-available sequences with approximately 12,000 ground-truthed frames. The learning time is shown to be 20 times smaller than standard SLLiP learning based on linear programming, yet its robustness and accuracy is similar. Superiority in the frame-rate and robustness in comparison with the SIFT detector, Lucas–Kanade tracker and Jurie's tracker is also demonstrated.

1. Introduction

Visual tracking is the process of repeated estimation of the state of an object given an image and the state(s) in previous frame(s). The state of an object is a set of parameters determining its pose (e.g. position, scale, rotation) and/or appearance. The most popular tracking approach is the Lucas–Kanade class of trackers [1,7] which minimizes the sum of intensity differences between a template and image data by the Gauss–Newton gradient descent optimization method. The intensity function is locally approximated by the first-order Taylor expansion and motion parameters are estimated as a linear function of image-template differences. The linear function is expressed as the pseudo-inverse of a matrix which is a function of image gradients. Like any other gradient method, the Lucas–Kanade tracker suffers from convergence to a local minimum, an unknown number of required iterations and an unknown basin of convergence.

Cootes et al. [2] noticed that a similar minimization task is solved in each frame and proposed to replace the pseudo-inverse operation with a multiplicative matrix learned on a set of synthetically perturbed examples, see Fig. 1. Cootes' method [2] predicts motion parameters as a linear function of object intensities; we call such methods *learned linear predictors* (LLiP). A LLiP method was adapted by Jurie and Dhome [4] for tracking of rigid objects. Unlike Cootes et al. [2], Jurie's LLiPs are used for prediction of local 2D translations only. This approach has recently attracted interest of the tracking community [3,12] and it has been also generalized to non-linear prediction as demonstrated by Williams et al. [9], who learn the predictor by a Relevance Vector Machine.

Staying in the realm of trained trackers, we proposed [11] a tracker that consists of a Sequence of LLiPs (SLLiP), see Fig. 2. Since the time required for motion prediction by a SLLiP directly corresponds to the number of used pixels, only a small subset of pixels from a template is used. The SLLiP learning is formulated as an optimization problem where time of tracking (computational complexity) is minimized given a predefined precision of motion predictors. In [11], a globally optimal sequence is delivered, the learning might be prohibitively time consuming for large problems.

The main contribution of this paper is a new *anytime* learning approach which, after a very short initialization period, provides a solution with predefined precision. The solution is continuously improved, i.e. the SLLiPs with lower complexity and defined precision allowing for faster tracking are continuously delivered. The anytime learning searches through the space of SLLiPs and successively constructs SLLiPs from LLiPs of different complexities. In order to make the searching process efficient, the branch a bound [5] searching approach is used.

If no constraint on the learning time is imposed, the anytime learning algorithm finds a globally optimal solution with respect to a certain class of predictors. If time consuming learning is not acceptable, the tracking can start immediately after a short initialization period. Since the SLLiP tracking requires only a fraction of processing power of an ordinary PC, the learning can continue in a parallel background thread.

We consider only linear predictors, nevertheless, the method is easily extended to an arbitrary polynomial class by data lifting. For instance, particular monomials can be considered as additional features.

* Corresponding author.
E-mail addresses: zimmerk@cmp.felk.cvut.cz (K. Zimmermann), svoboda@cmp.felk.cvut.cz (T. Svoboda), matas@cmp.felk.cvut.cz (J. Matas).

0262-8856/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2009.03.005

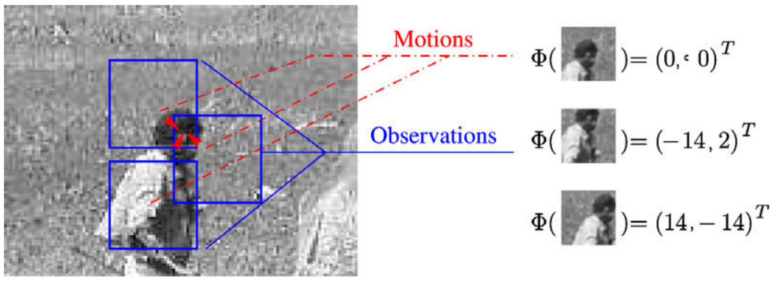


Fig. 1. Learning of the linear mapping between intensities and motions by the LS method from a set of synthetically perturbed examples.

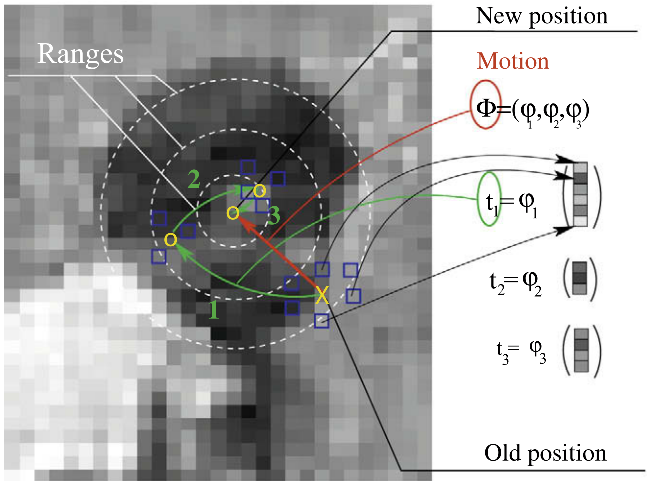


Fig. 2. SLLiP consists of a sequence of linear mappings. Computational complexity of tracking, which directly corresponds to the number of used pixels, is minimized in a learning stage.

The rest of the paper is organized as follows: Section 2 summarizes properties of SLLiPs [11] and introduces notation and definitions. Section 3.1 describes how a training set is constructed from a single image. In Section 3.2 the proposed anytime learning algorithm is presented. Section 4 compares learning/tracking time, robustness and accuracy of the proposed approach with state-of-the-art approaches [4,6,7,11] on ground truthed sequences. Section 5 concludes the paper.

2. Problem formulation

In this section, we introduce formal definitions of LLiP and SLLiP and formulate their learning as a constrained optimization problem. Let us suppose we are given an image I of an object to be tracked. Object motion is robustly determined by Ransac from local motions of some points on the object. These points are called *reference points* and their motion is estimated from their neighbourhoods. For motion predictors, it is not necessary to use all neighbourhood pixels, because sufficient precision is achievable even with smaller number of pixels. Therefore only a selected subset of pixels $X = \{\mathbf{x}_1 \cdots \mathbf{x}_c\}$, called *support set*, is used. LLiP estimates motion of the reference point from the intensities observed on the support set. These intensities are stored in the *observation vector* denoted $\mathbf{I}(X)$.

We denote $(\mathbf{t} \circ X)$ the support set warped by a motion with parameters $\mathbf{t}$. For example, if the considered motion is a 2D translation, then $(\mathbf{t} \circ X) = (X + \mathbf{t}) = \{(\mathbf{x}_1 + \mathbf{t}), \ldots, (\mathbf{x}_c + \mathbf{t})\}$. There is a mapping (rendering) from parameters $\mathbf{t}$ to observations $\mathbf{I}(\mathbf{t} \circ X)$, which is usually not invertible. We therefore search for a mapping approximating a the set of motions $\mathbf{t}$ which could have generated the observation $\mathbf{I}(\mathbf{t} \circ X)$. This mapping assigns a p-vector of motion parameters to a c-vector of observation.

Definition 1. Linear predictor (LLiP) is an ordered pair $\varphi = (\mathtt{H}, X)$, which assigns p-vector of motion parameters $\mathbf{t} = \mathtt{H}\mathbf{I}(X)$ to c-vector of observations $\mathbf{I}(X)$, where $\mathtt{H} \in R^{p \times c}$.

All predictors φ are characterized by the following parameters, see also Fig. 3:

Definition 2. Complexity $c(\varphi) = |X|$ of predictor φ is the cardinality of the predictor's support X.

Definition 3. Range $R(\varphi)$ of the predictor φ is a set of motion parameters.

Definition 4. Error of predictor $\varphi = (\mathtt{H}, X)$ for range $R(\varphi)$ is $\lambda(\varphi) = E(\|\mathbf{t} - \mathtt{H}\mathbf{I}(\mathbf{t} \circ X)\|_2^2),\ \forall \mathbf{t} \in R(\varphi)$, where $E(.)$ denotes the expectation value with respect to $\mathbf{t}$ uniformly distributed on $R(\varphi)$.[1]

Predictor complexity approximately corresponds to the number of multiplications and sums necessary for motion estimation. It is clear that there is no ideal predictor which would simultaneously has (very) low complexity, (very) large range and (very) small error. It is easy to see that the higher the complexity the better the prediction. However, as the complexity increases towards the complete template, the improvements become less and less significant. In general, for large ranges it is very difficult to achieve a good prediction with any complexity. In order to overcome this limitation we develop a *sequential predictor* $\Phi = (\varphi_1 \cdots \varphi_m)$. Since the sequential predictor is provably superior to a single monolithic predictor, it allows lower complexity for higher precision. A vector of motion parameters $\mathbf{t}$ is predicted in m steps as follows:

$$\begin{aligned}
&\mathbf{t}_1 = \mathtt{H}_1(\mathbf{I}(X_1)), \quad \mathbf{t}_2 = \mathtt{H}_2(\mathbf{I}(\mathbf{t}_1 \circ X_2)),\\
&\mathbf{t}_3 = \mathtt{H}_3(\mathbf{I}(\mathbf{t}_2 \circ \mathbf{t}_1 \circ X_3)), \ldots, \mathbf{t}_m = \mathtt{H}_m\left(\mathbf{I}\left(\left(\bigcirc_{i=1}^{m-1} \mathbf{t}_i\right) \circ X_m\right)\right),\\
&\mathbf{t} = \bigcirc_{i=1}^{m} \mathbf{t}_i,
\end{aligned} \tag{1}$$

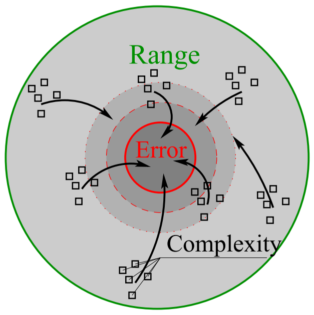
The first vector of motion parameters $\mathbf{t}_1$ is directly predicted from intensities observed *at locations defined by the support set* X_1. The second predictor estimates motion parameters $\mathbf{t}_2$ from intensities $\mathbf{I}(\mathbf{t}_1 \circ X_2)$ observed on the its support set warped by $\mathbf{t}_1$, and so on. The advantage is that each predictor in a sequence is more and more specific, using a smaller range which corresponds to the accuracy of the preceding predictor.

Definition 5. Sequential predictor (SLLiP) is an m-tuple $\Phi = (\varphi_1, \ldots, \varphi_m)$ of predictors $\varphi_i \in \omega,\ i = 1 \cdots m$, where ω is a set of predictors.

[1] In practice, the error is the mean value of square Euclidean error of all predictions from the range.

Abbreviation	Meaning
LK	Lucase-Kanade tracker [7]
LS	Least Squares
MM	Minimax
LLiP	Learned Linear Predictor
SLLiP	Sequential LLiP
NoSLLiP	Number of SLLiPs
MM SLLiP	SLLiP learned by [11]
LS SLLiP	SLLiP anytime learning

(a) Table of used abbreviations.



(b) Definitions

Fig. 3. (a) Table of used abbreviations. (b) Definitions: The range, complexity and prediction error of a learned linear predictor.

The set of predictors ω can include all possible predictors, or its convenient subset. Because of computational complexity of the learning process, only the predictors with $\mathtt{H}$ minimizing their prediction error for a given support set and training set, will be further considered.

Definition 6. *Optimal sequential predictor* is a sequential predictor

$$\Phi^* = \operatorname*{arg\,min}_{\Phi\in\omega^m, m} \left\{ \sum_{i=1}^{m} c(\varphi_i) | \lambda(\varphi_m) \leqslant \lambda^* \right\}, \tag{2}$$

where λ^* is predefined prediction error, c is predictor complexity, ω is a set of predictors and $\omega^m = \omega \times \omega \times \cdots \times \omega$ is a set of sequential predictors of length m.

3. Anytime learning of SLLiP

We define learning as a search for the optimal SLLiP subject to a predefined prediction error (Definition 6). In general, the learning procedure consists of two steps: support set selection and SLLiP optimization. The support set selection is a combinatorial problem, the solution of which might be time consuming [8,10]. Since we are interested in applications where the learning time is an issue, a randomly selected support set is used instead. The SLLiP optimization is also simplified by restricting ω to be a class of LLiPs with the minimal prediction error (Definition 4). Nevertheless, the proposed learning algorithm can be used to find the globally optimal solution with respect to arbitrary ω. For example, ω could be a set of LLiPs learned by the minimax method, then the result of learning would be the same as of the algorithm proposed in [11].

Note that the globally optimal solution found with respect to the restricted ω is not guaranteed to provide globally optimal solution with respect to the set of *all* possible LLiPs. In Section 3.1, training set construction from a single image is described. Section 3.2 then presents SLLiP learning.

3.1. Training set construction

Given a predefined range of motions within which the tracker is assumed to operate, we perturb the support set by motion with parameters $\mathbf{q}^i$ randomly (uniformly) generated inside the range. Each motion $\mathbf{q}^i$ warps the support set X to a set X^i, where a vector of intensities $\mathbf{I}^i$ is observed, see Fig. 4. Given the observed intensities, we search for a mapping assigning motion $\mathbf{t}^i = (\mathbf{q}^i)^{-1}$, which warps X^i as close as possible to the original support set X. These examples are stored in matrices $\mathtt{I} = [\mathbf{I}^1 \cdots \mathbf{I}^d]$ and $\mathtt{T} = [\mathbf{t}^1 \cdots \mathbf{t}^d]$. The ordered triple $(\mathtt{I}, \mathtt{T}, \mathtt{X})$ of such matrices and ordered d-tuple of support sets $X = \{X^1 \cdots X^d\}$ composes a *training set*.

3.2. Learning algorithm

In this section, we describe the method searching for the optimal sequential predictor given a training set $(\mathtt{I}, \mathtt{T}, X)$ generated on image I. Since we restricted the set of considered LLiPs ω to the set of LLiPs minimizing prediction error λ, the LLiP learned from the training set is $\varphi = (\mathtt{H}^*, X)$, where

$$\mathtt{H}^* = \operatorname*{arg\,min}_{\mathtt{H}\in R^{p\times c}} \|\mathtt{HI} - \mathtt{T}\|_F^2 = \mathtt{TI}^+ \tag{3}$$

and X is the support set aligned with the object.

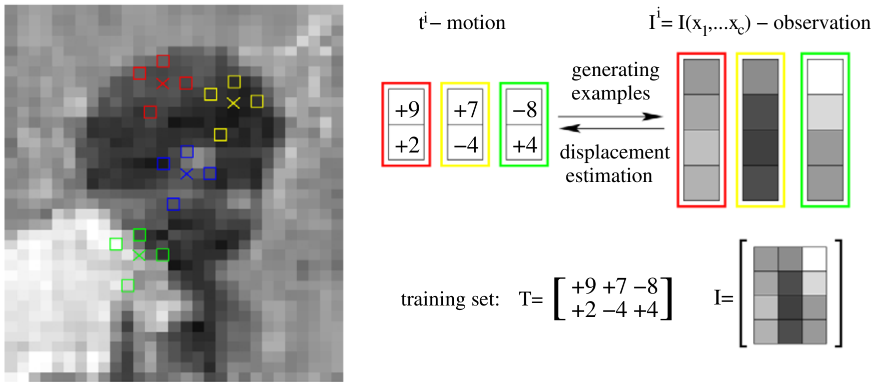


Fig. 4. An image patch is perturbed by the motion parameters within a predefined range in order to create a set of synthesized examples of observed intensities $\mathbf{I}^i$ and motions $\mathbf{t}^i$.

Ideally, the predictor learned according to Eq. (3) would transform intensities $\mathtt{I}$ to motions $\mathtt{T}$. However, such predictor usually does not exist. Therefore the observed intensities are transformed into motion parameters $\mathtt{T}^{(1)}$ which are as close as possible to the desired motions $\mathtt{T}$. We warp each support set $X^i \in X$ by motion parameters $\mathtt{T}^{i,(1)}$ obtaining the new support set $X^{i,(1)} = \mathtt{T}^{i,(1)} \circ X^i$. Denoting $\mathtt{I}^{i,(1)}$ the intensities observed on these newly obtained support sets $X^{i,(1)}$, we form the new training set $(\mathtt{I}^1, \mathtt{T}, X^1)$. We refer to it as to training set *invoked by predictor* φ_1 and denote it $T(\varphi_1, c)$, where c denotes the size of support set used in the training set. Similarly, we define training set *invoked by sequential predictor* as $T(\Phi, c)$.

As already mentioned, the size of the support set influences the prediction error. Removing some pixels from the support set necessarily results in error increase.[2] Given a training set $T(\Phi, c_1)$ we can simply generate a training set $T(\Phi, c_2),\ c_2 < c_1$ for the predictor with a lower complexity c_2 by removing corresponding number of pixels from X and corresponding number of rows from matrix $\mathtt{I}$. We refer to this process as training set *restriction*.[3]

In order to simplify the problem, we further work with a discretized set of complexities C. The optimal sequence of predictors is found by searching through the set of all SLLiPs, which involve $\sum_{i=1}^{m} |C|^i$ elements, where $|C|$ denotes the size of C and m is maximum length of SLLiP. In order to make the searching process efficient, the branch and bound [5] searching approach is used. Sequential predictors are successively constructed from the LLiPs of different complexities. In the first level, we learn LLiPs for all complexities in C according to Eq. (3). They correspond to the SLLiPs of the length equal to one. One of these SLLiPs, Φ, is expanded in the next iteration. The expansion means that Φ is successively extended by LLiPs with different complexities learned on training set invoked by itself $T^i(\Phi, c)$. This process creates $|C|$ new SLLiPs, which could be expanded in further iterations. Once a SLLiP with a sufficiently small prediction error (feasible solution) is found, all other partially constructed SLLiPs with a higher complexity are terminated, i.e. they will never be expanded. The smallest complexity c^* of the feasible solution is saved and once any SLLiP reaches a higher complexity it is automatically terminated.

The learning process is summarized in Algorithm 1; see also Fig. 5, which demonstrates six iterations of the algorithm on a toy example with $C = \{20, 300\}$, range equal to 40% of the object size and the predefined error set to 10% of the object size. In the first iteration, two LLiPs with complexities 20 and 300 are learned, denoted φ_1 and φ_2. Obviously, the LLiP with the higher complexity achieves lower prediction error $\lambda(\varphi_2) = 0.15$. Since no solution has been found, φ_2 is expanded in the second iteration, i.e. we learn two further LLiPs, denoted φ_{21} and φ_{22}, on the training set invoked by φ_2. Since both newly constructed SLLiPs $\Phi_1 = (\varphi_2, \varphi_{21})$ and $\Phi_2 = (\varphi_2, \varphi_{22})$ achieve sufficiently low prediction error, i.e. smaller than $\lambda^* = 0.1$, the one with the lower complexity, i.e. $c(\Phi_1) = 300 + 20 = 320$, is selected and the other one, Φ_2, is terminated. Φ_1 could be immediately used for tracking, while the learning can continue: in the third iteration, φ_1 is expanded. Since $c(\varphi_1 \varphi_{12}) = 300 + 300 = 600 > c(\Phi_2) = 320$, this SLLiP is terminated. In the remaining iterations the not terminated SLLiP is further expanded till the solution, SLLiP Φ_3 consisting of 5 LLiPs, is reached. Since the complexity $c(\Phi_3) = 5 \times 20 = 100$ is smaller than $c(\Phi_1) = 320$, Φ_1 is replaced by Φ_3. And since there are no more SLLiPs to expand, Φ_3 is accepted as the final solution.

Of course, it is likely that better solution exist, consisting from the LLiPs with complexities not constrained to $C = \{20, 300\}$, but this is just a toy example demonstrating the learning process. In practice, we work with $|C| \in \{10 \cdots 15\}$.

Algorithm (anytime learning of SLLiP)

```
Input:
• Range $R$ within which SLLiP is expected to operate.
• Set of considered complexities $C$.
• Predefined accuracy $\lambda^*$.
• Training image $I$.
• Support set $X$.
  (1) Set:
      $c^* = \infty$ (complexity of the simplest admissible SLLiP found) and
      $i = 0$ (number of current iteration).
  (2) Generate training sets $T^0(c),\ \forall c \in C$ on the predefined range $R$.[4]
  (3) Initialize set $\Omega$ of learned active SLLiPs as a set of LLiPs learned on
      $T^0(c),\ \forall c \in C$ according Eq. (3).
  (4) $\Phi = S(\Omega),\ \Omega = \Omega \setminus \Phi$ (Select and remove $\Phi$ according to a strategy $S$.)
  (5) For each $c \in C$: (expand $\Phi$)
      (a) Generate training set $T^i(\Phi, c)$ invoked by $\Phi$.
      (b) Learn LLiP $\varphi$ for $T^i(\Phi, c)$ according to Eq. (3).
      (c) $\Phi' = (\Phi, \varphi),\ \Omega = \Omega \cup \Phi'$ (add the new SLLiP to $\Omega$)
      (d) If $c(\Phi') < c^*$ then $\Phi^* = \Phi'$ and $c^* = c(\Phi')$ (replace solution)
      (e) For $\forall \Phi'' \in \Omega$ with $c(\Phi'') > c^*$, do $\Omega = \Omega \setminus \Phi''$ (terminate the SLLiPs
          with higher complexity)
      end

  (6) If $\Omega = \emptyset$ stop otherwise $i = i + 1$ and goto 4.
Output:
• Optimal SLLiP $\Phi^*$
```

Note that the selection strategy S which selects a SLLiP from Ω (step 4), may influence the learning behavior. However, if Algorithm 1 satisfies condition $\Omega = \emptyset$ in step 6, Φ^* is an optimal SLLiP with respect to the set of considered LLiPs ω. In our implementation, we first use the strategy which expands the SLLiP with the highest complexity. This strategy usually finds a solution Φ^* in a few iterations. This solution is of a high complexity, but the prediction error is guaranteed and the tracking can start. Then the strategy is switched and the SLLiPs with the average complexity are preferably expanded. Once a solution is reached, it can be immediately used for tracking with a lower performance. If the learning continues, the SLLiP can be in future replaced by better solutions.

The stopping condition (step 6) could be also optionally replaced for example by a maximum number of iterations, maximum running time, maximum depth of the constructed graph or an arbitrary intersection of these conditions. However, such replacement might influence the optimality of the found Φ^*.

4. Experiments

The proposed method is verified on real sequences with planar objects. The object is represented as a Number of SLLiPs (NoSLLiP), which estimates local translations at a few points on the object. Object motion, i.e. a homography, is determined from these local translations by the RANSAC. Note that although we work with planar objects in order to avoid problems of 3D reconstruction, the proposed trackers could be attached to a 3D model with a reasonable texture, as was shown in [11].

The quantitative evaluation of robustness and accuracy of SLLiPs is conducted on sequences with three different objects (MOUSEPAD-MP, TOWEL and PHONE), where ground truth positions of the object corners in total number 11963 frames were manually labeled.[5] Accuracy is measured in each corner as a percentage; the displacement error is related to the current size of the object upper edge. Robustness is

[2] Proof of this claim is detailed in [11].

[3] Since the support set is selected randomly its restriction is random as well. If, for instance, the greedy construction [8,10,11] had been used, than the order of the selection would have provided the importance measure of the support pixels and the lastly selected pixels would have been removed firstly in the restriction.

[4] We generate only $T^0(c_{max})$ where $c_{max} = \max C$ is maximum complexity of C, the other training sets with the lower complexity are constructed by its restriction.

[5] These sequences in conjunction with the ground truth are available at ftp://cmp.felk.cvut.cz/pub/cmp/data/lintrack/index.html.

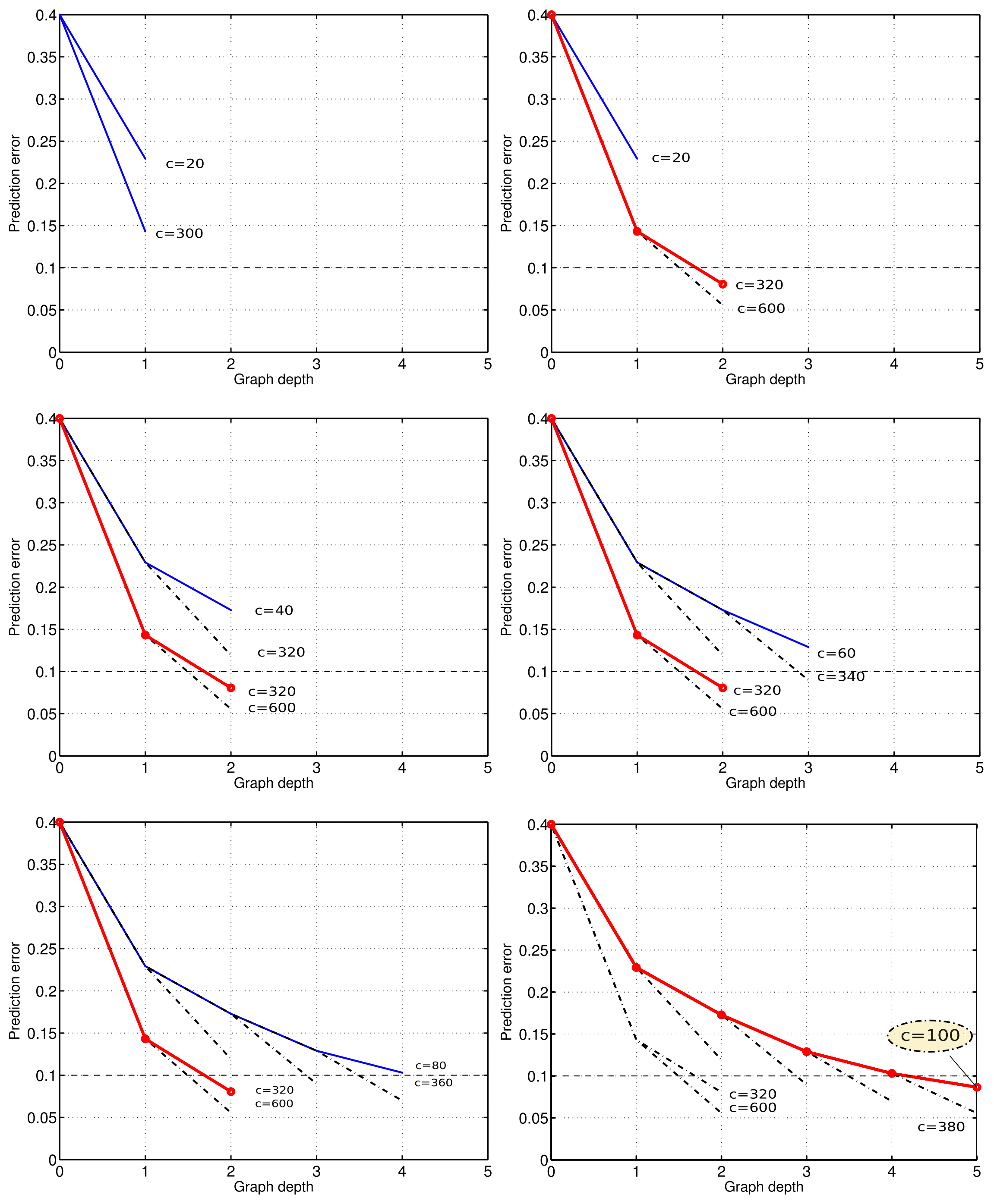


Fig. 5. Demonstration of progress of Algorithm 1 for a toy example with $C = \{20, 300\}$, $|C| = 2$, $R = 0.4$ and $\lambda^* = 0.1$. Blue denotes a set of current SLLiPs Ω, black denotes terminated SLLiPs and red delineates solution Φ^* with the lowest complexity so far.

measured by the number of loss-of-locks, defined as the cases where the accuracy was worse than 25%. In loss-of-lock frames, the tracker was reinitialized from the ground truth and the accuracy did not contribute to the total accuracy statistic. Some of the successfully tracked frames, which include oblique views, motion blur and significant scale changes, are presented in Fig. 6. The results are summarized in Table 1.

In the first row, results of NoSLLiP tracker with SLLiPs learned by Algorithm 1 with no time constraint (method LS) are presented. Second row contains results for SLLiPs learned by minimax [11] (method MM). The minimax learning minimize the size of a compact region within which all predictions lie (uncertainty region) instead of the square of Euclidean error, therefore higher robustness is achieved, but the learning is 10–20 times longer. Tracking accuracy of MM and LS methods varies with data; the accuracy is similar for MP and TOWEL sequences, but PHONE object contains similar repetitive structure (buttons) which make the LS accuracy significantly worse.

Robustness of the predictor is given by the shape of the error distribution, because the higher the probability of large errors the higher the probability of the predictor failure in the next frame due to its initialization out of its range. Shape of MM SLLiP distribution (blue solid line) and LS SLLiP distribution (red solid line) are shown in Fig. 7. We observe that LS SLLiPs are more likely to have higher errors in difficult cases however, the accuracy in easier cases is higher. In addition to this we can also compare predefined uncertainty region λ_0 of MM SLLiP, predefined prediction error ϵ_0 of

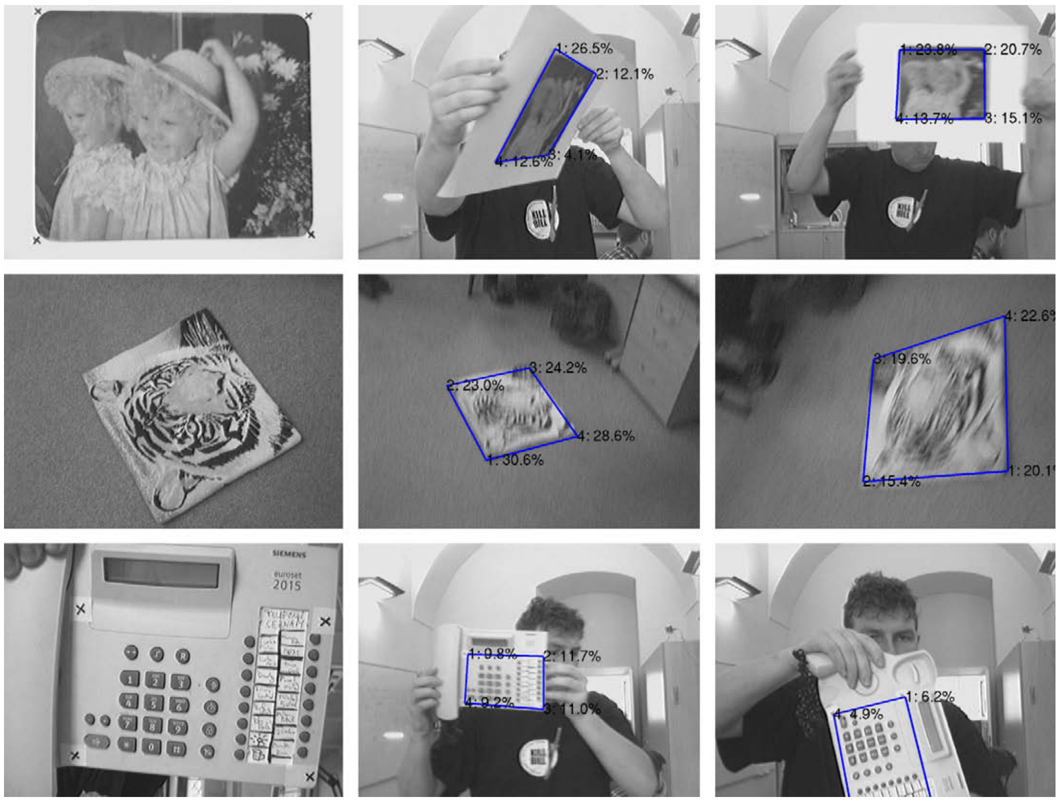


Fig. 6. The left column shows images used for training. The middle and right columns demonstrate some successfully tracked frames with strong motion blur from the testing sequences. Blue rectangle delineates the object. Percentage values in corners are current corner speeds related to the current size of the object upper edge.

Table 1
Comparison of robustness and accuracy of NoSLLiP learned by anytime algorithm (LS) and minimax algorithm (MM) proposed in [11].

Object	SLLiP learning	Learning time* (s)	Processing (fps)	Loss-of-locks	Mean-error (%)
MP	LS	11	27.6	17/6935	[1.4, 1.3, 1.1, 1.1]
MP	MM [11]	310	18.9	13/6935	[1.3, 1.8, 1.5, 1.6]
TOWEL	LS	16	33.3	2/3229	[1.6, 1.8, 1.1, 1.5]
TOWEL	MM [11]	310	21.8	5/3229	[3.0, 2.2, 1.4, 1.9]
PHONE	LS	21	25.6	55/1799	[7.3, 7.1, 10.6, 6.5]
PHONE	MM [11]	310	16.8	20/1799	[1.2, 1.8, 2.6, 1.9]

* Learing time is an average time required per one SLLiP.

LS SLLiP and true error distribution on ground truthed data (MOUSEPAD sequence). In this experiment we learned 35 SLLiPs with different ranges covering the mousepad. MM SLLiPs are learned to achieve uncertainty region $\lambda_0 = 5\%$ (blue dot-dashed line), LS SLLiPs are learned for prediction error $\epsilon_0 = 3\%$ (red dot-dashed line). Both the uncertainty region and the prediction error are relative to SLLiPs range. λ_0, ϵ_0 were chosen experimentally in order achieve the best performance of SLLiPs. Lower values result in higher complexity and consequent over-fitting. The error is evaluated on those frames, where inter-frame motion is smaller than the learning range of SLLiPs.

Table 2 compares the NoSLLiP tracker to the state-of-the-art Lowe's SIFT detector [6] (method: SIFT),[6] Lucas–Kanade tracker [7] (method: LK tracker) and Jurie's LS LLiP tracker [4] (method: LLiP LS). All these local motion estimators were combined with the RANSAC, to keep test conditions as similar as possible. SIFT tracking mainly fails in frames with strong motion blur or in frames where the object was very far from the camera. LK tracker, which

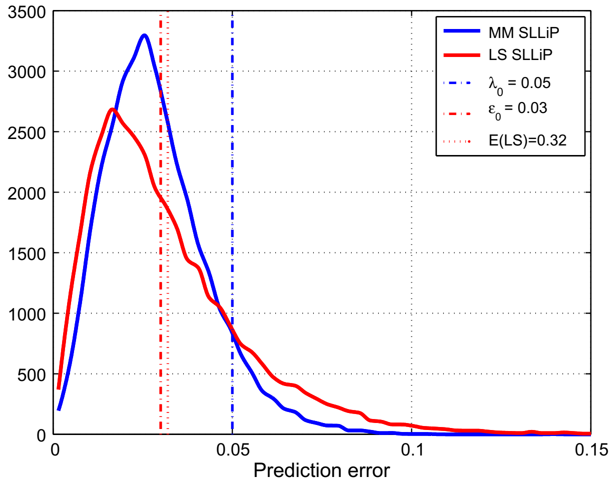
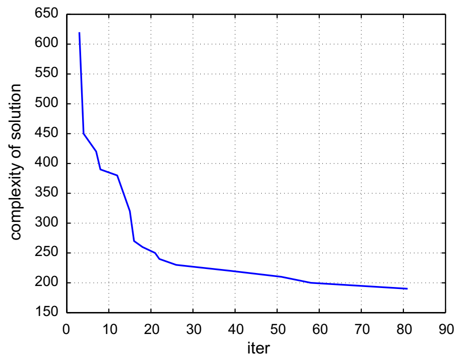


Fig. 7. Accuracy analysis: Comparison of predefined uncertainty region λ_0 of MM SLLiPs (blue dot-dashed line), predefined prediction error ϵ_0 of LS SLLiPs (red dot-dashed line) and true error distribution (blue and red solid lines) on MOUSEPAD sequence.

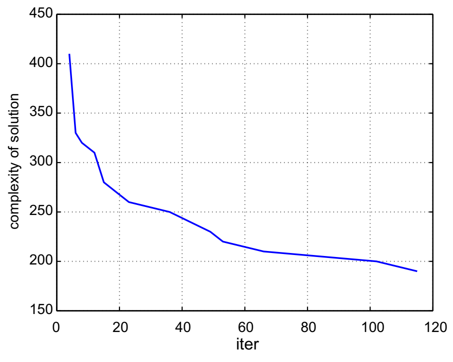
Table 2
Comparison of robustness and accuracy of SLLiP, LK, SIFT and LLiP tracker on MP sequence.

Method	Processing (fps)	Loss-of-locks	Mean-error (%)
SLLiP LS	27.6	17/6935	[1.4, 1.3, 1.1, 1.1]
SLLiP MM [11]	18.9	13/6935	[1.3, 1.8, 1.5, 1.6]
SIFT [6]	0.5	281/6935	[1.6, 1.2, 1.5, 1.4]
LK tracker [7]	2.6	398/6935	[2.3, 2.2, 2.5, 2.5]
LLiP LS [4]	24.4	1083/6935	[5.9, 6.0, 6.7, 6.7]
LLiP LS [4] half-range	24.2	93/6935	[3.1, 2.3, 2.7, 4.0]

[6] We use implementation of the SIFT detector downloaded from http://www.cs.ubc.ca/lowe/keypoints/.



(a) $|C| = 3$



(b) $C = |4|$

Fig. 8. Average SLLiP complexity as a function of iterations of Algorithm 1. (a) and (b) differ by the size of the set of considered complexities $|C|$. The higher the value of $|C|$, the larger the space of considered LLiPs.

estimates the local motion at Harris corners, provided quite good results on the frames where the object was far from the camera, but its basin of attraction was in many frames insufficient for correct motion estimation, failing for fast motions.

Since we work with a non-optimized implementation of the LK tracker, the presented frame-rate in this experiment could not serve for a speed comparison. Nonetheless the SLLiP computational complexity is clearly smaller than the complexity of the LK tracker. Jurie's tracker is a LLiP tracker with the support set equal to the whole template learned by LS method for the same reference points and ranges as optimal SLLiPs. Since a single LLiP tracker does not allow sufficient accuracy on the same range, very high loss-of-lock ratio and low accuracy are reported. If the half-range is used, the higher accuracy is achieved, but the number of loss-of-locks is still significantly higher than with NoSLLiP tracker, mainly due to long inter-frame motions.

4.1. Time-constrained learning procedure

The learning procedure proposed in Algorithm 1 might be time consuming if the set of considered LLiPs is too large. Since a long learning time might not be acceptable for some types of applications, either the set of considered LLiPs or the maximum number of iterations have to be restricted. However, the constraint on maximum number of iterations affects the optimality of the found SLLiP, and therefore we show average complexity of the best found solution as a function of iterations in Algorithm 1. Fig. 8 presents this function for different sets of considered LLiPs. One can see that the learning time could be 2–3 times decreased without significant increase of the solution complexity.

Note that there is also another option besides premature interruption of the learning procedure. The anytime learning algorithm, after a short initialization procedure, provides a solution – SLLiP with higher complexity but predefined precision. Having this SLLiP the tracking can immediately start. Since the tracking requires only a fraction of the processing power of an ordinary PC, the learning need not to be necessarily terminated and it might be allowed to run in a parallel background thread continuously providing better and better SLLiPs. This principle theoretically allows to start the tracking procedure immediately without any learning using for example Lucas–Kanade tracker and collect training examples automatically. Once a training set is constructed the learning procedure can run in a parallel thread providing the SLLiPs which continuously replace worse LK trackers. Similar idea based in simple LLiPs was demonstrated in [3].

5. Conclusions

We proposed a fast learning algorithm for the SLLiP learnable tracker [11]. Unlike the original learning procedure, the new algorithm has the anytime property and outputs progressively faster SLLiPs satisfying a user defined accuracy and range. The learning process very quickly returns a SLLiP which is slow, but satisfies the user-defined conditions on accuracy and range. During tracking, the learning is run in a background thread and gradually improves the SLLiP tracker.

The method was quantitatively evaluated on approximately 12,000 labeled frames with three different planar objects. The performance and robustness superiority of the SLLiP tracker in comparison with Lucas–Kanade tracker [7], SIFT detector [6] and Jurie's LLiP tracker [4] was demonstrated. We encourage the reader to download sequences, ground-truth data and a MATLAB implementation which is available at http://cmp.felk.cvut.cz/demos/Tracking/linTrack.

Acknowledgements

K. Zimmermann was supported by Czech Academy of Sciences project 1ET101210407, T. Svoboda by Czech Ministry of Education project 1M0567, J. Matas by EC projects ICT-215078 DIPLECS and FP6-IST-004176 COSPAL.

References

[1] S. Baker, I. Matthews, Lucas–Kanade 20 years on: a unifying framework, International Journal of Computer Vision 56 (3) (2004) 221–255.
[2] T. Cootes, G. Edwards, C. Taylor, Active appearance models, IEEE Transaction on Pattern Analysis and Machine Intelligence 23 (6) (2001) 681–685.
[3] L. Ellis, N. Dowson, J. Matas, R. Bowden, Linear predictors for fast simultaneous modelling and tracking, in: Proceedings of 11th IEEE International Conference on Computer Vision, Workshop on Non-rigid Registration and Tracking Through Learning, Rio de Janeiro, Brazil, IEEE Computer Society, October 2007.
[4] F. Jurie, M. Dhome, Real time robust template matching, in: British Machine Vision Conference, 2002, pp. 123–131.
[5] A. Land, A. Doig, An automatic method for solving discrete programming problems, Econometrica 28 (1960) 497–520.
[6] D. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[7] B. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, International Journal of Computer Vision and Artificial Intelligence (1981) 674–679.
[8] J. Matas, K. Zimmermann, T. Svoboda, A. Hilton, Learning efficient linear predictors for motion estimation, in: S.B. Rangachar Kasturi (Ed.), Proceedings of the 5th Indian Conference on Computer Vision, Graphics and Image Processing, LNCS 4338, Berlin, Germany, Thiagarajar College of Engineering, Springer, Berlin, December 2006, pp. 445–456.
[9] O. Williams, A. Blake, R. Cipolla, Sparse bayesian learning for efficient visual tracking, Pattern Analysis Machine Intelligence 27 (8) (2005) 1292–1304.
[10] S.K. Zhou, B. Georgescu, X.S. Zhou, D. Comaniciu, Image based regression using boosting method, in: Proceedings of the 10th IEEE International Conference on Computer Vision, vol. 1, Washington, DC, USA, IEEE Computer Society, 2005, pp. 541–548.
[11] K. Zimmermann, J. Matas, T. Svoboda, Tracking by an optimal sequence of linear predictors, Transaction on Pattern Analysis Machine Intelligence 31 (4) (2009).
[12] K. Zimmermann, T. Svoboda, J. Matas, Adaptive parameter optimization for real-time tracking, in: Proceedings of the 11th IEEE International Conference on Computer Vision, Workshop on Non-rigid Registration and Tracking Through Learning, Rio de Janeiro, Brazil, IEEE Computer Society, October 2007.