Tracking by an Optimal Sequence of Linear Predictors

Karel Zimmermann, Jiří Matas, *Member*, *IEEE*, and Tomáš Svoboda, *Member*, *IEEE*

Abstract—We propose a learning approach to tracking explicitly minimizing the computational complexity of the tracking process subject to user-defined probability of failure (loss-of-lock) and precision. The tracker is formed by a Number of Sequences of Learned Linear Predictors (NoSLLiP). Robustness of NoSLLiP is achieved by modeling the object as a collection of local motion predictors—object motion is estimated by the outlier-tolerant RANSAC algorithm from local predictions. The efficiency of the NoSLLiP tracker stems 1) from the simplicity of the local predictors and 2) from the fact that all design decisions, the number of local predictors used by the tracker, their computational complexity (i.e., the number of observations the prediction is based on), locations as well as the number of RANSAC iterations, are all subject to the optimization (learning) process. All time-consuming operations are performed during the learning stage—tracking is reduced to only a few hundred integer multiplications in each step. On PC with 1xK8 3200+, a predictor evaluation requires about 30 μs. The proposed approach is verified on publicly available sequences with approximately 12,000 frames with ground truth. Experiments demonstrate superiority in frame rates and robustness with respect to the SIFT detector, Lucas-Kanade tracker, and other trackers.

Index Terms—Image processing and computer vision, scene analysis, tracking.

1 Introduction

Visual tracking is the process of repeated estimation of the pose of an object (e.g., position) in an image given its pose(s) in previous frame(s). Tracking has many applications such as surveillance, 3D object modeling, augmented reality, and medical imaging. Since many applications have real-time requirements, very low computational complexity is a highly desirable property. Our primary objective is to find a very fast tracking method with defined precision and robustness.

A natural formulation of tracking is a search for a pose that optimizes a similarity criterion function. For example, Lucas and Kanade [1], [2] use the steepest descent optimization to minimize the sum of square differences between the template and image data (see Fig. 1). Other approaches [3], [4], [5] scan the image by a learned classifier, which evaluates the similarity criterion. Regression-based methods [6], [7], [8], which do not require any criterion function, estimate the object pose directly from the observed intensities by a learned regression mapping. The methods proceed by collecting training examples—pairs of observed intensities and corresponding poses—and use machine learning techniques to learn the regression function. In tracking, the regression method is initialized by the previous pose or, if available, by the pose derived from a dynamic model. A learned regression function estimates actual object pose directly from the intensities observed around the initial location.

The more complex the regression function is, the more achievable the precise pose estimation is. Increasing the complexity, however, often suffers from diminishing returns and very complex functions are prone to overfitting. We follow a simple assumption that it is easier to estimate the actual state if the method is initialized in the close neighborhood of searched pose. Accepting this assumption, it is better to exploit a less complex regression function for coarse state estimation and use the newly obtained state for the initialization of another function. The coarse estimate of the state allows the consecutive regression functions to operate within a smaller range of poses and achieve a higher precision with reasonable complexity. Hence, instead of learning a sophisticated predictor, we use a sequence of simple regression functions concatenated so that each of the functions compensate only errors of its predecessor and thus refines the previous estimations. While a single regression function operates on a fixed set of intensities (features), the sequence of functions allows for higher precision because the set of the intensities is updated successively as the actual pose accuracy increases. We learn the optimal sequence of regression functions.

Since the computational time of tracking (i.e., the overall complexity of the used regression method) is usually an issue, the learning is formulated as a minimization of the complexity subject to a user-predefined accuracy and robustness. Note that a single regression function is a special case of a sequence. Since the globally optimal solution is found, the sequence is superior to a single regression function. Any arbitrary regression function allows concatenating, but we observed that the sequence of linear functions achieves high precision with a low computational cost. Focusing on sequences of linear functions we achieved an algorithm that estimates object pose using only a fraction of processing power of an ordinary computer.

• *The authors are with the Czech Technical University, Faculty of Electrical Engineering, Department of Cybernetics, Karlovo namesti 13, 121 35 Prague 2, Czech Republic.*
E-mail: karel.zimmermann@esat.kuleuven.be, {matas, svoboda}@cmp.felk.cvut.cz.

Manuscript received 19 June 2007; revised 19 Dec. 2007; accepted 30 Apr. 2008; published online 8 May 2008.
Recommended for acceptance by P. Perez.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2007-06-0371.
Digital Object Identifier no. 10.1109/TPAMI.2008.119.

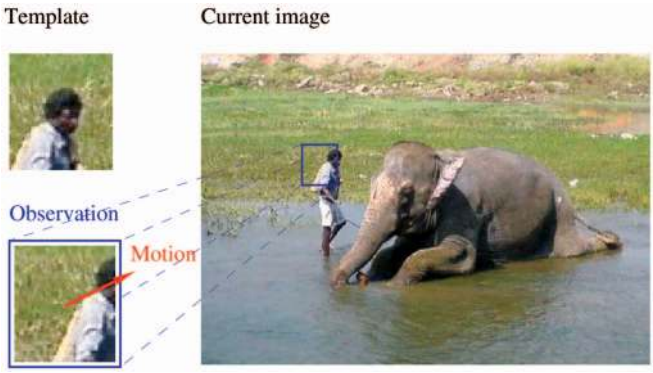

Fig. 1. Tracking: We define visual tracking as the process of repeated estimation of the pose of an object given an image and pose(s) in previous frame(s).

2 The State-of-the-Art

The most common approach to tracking is repeated optimization of some criterion function $f(\mathbf{t};\mathbf{I},\mathbf{t}_0)$ over the space of object poses $\mathbf{t}\in S$, given image $\mathbf{I}$ and previous pose $\mathbf{t}_0$

$$\mathbf{t}^* = \arg\min_{\mathbf{t}\in S} f(\mathbf{t};\mathbf{I},\mathbf{t}_0), \tag{1}$$

where $\mathbf{t}^*$ is the estimate of the current pose of the object. Criterion $f(\mathbf{t};\mathbf{I},\mathbf{t}_0)$ includes some implicit or explicit model of possible object appearances and optionally some relation to $\mathbf{t}_0$. Criterion f could be, e.g., obtained as a similarity function or a classifier or foreground/background probability ratio learned from training examples. We call these methods *optimization-based tracking*.

Optimization-based tracking is an online optimization process solving problem (1). While some approaches [3], [4], [5], [9] exhaustively scan a subset of object poses S with a classifier approximating $f(\mathbf{t};\mathbf{I},\mathbf{t}_0)$, other approaches [1], [2], [10], [11] use a gradient optimization of a criterion approximating $f(\mathbf{t})$.

Unlike optimization-based tracking, *regression-based tracking* methods attempt to model explicitly a relationship between observations and state $\mathbf{t}^*$ without any necessity of defining $f(\mathbf{t};\mathbf{I},\mathbf{t}_0)$. They learn a mapping $\varphi(\mathbf{I},\mathbf{t}_0)$ in a supervised way from synthesized training data [6], [7], [8].

Tracking methods based on exhaustive scanning can operate within a small range of poses or over the whole image. On the other hand, tracking methods based on the gradient optimization or regression estimate object pose only locally within a certain range of poses. We understand these local methods as complementary to the scanning-based methods since every pose in a scanned grid can be optionally preprocessed by such local method.

Tracking based on the gradient optimization does not require any learning procedure; however, it suffers from problems of local optimization: convergence to a local minimum, unknown number of required iterations, and unknown basin of convergence. In the state-of-the-art, we further focus on regression-based tracking.

2.1 Regression-Based Tracking

Regression-based tracking approaches [6], [7], [8] estimate location $\mathbf{t}$ directly from locally observed intensities. Such approach requires a learning stage, where pairs of motions $\mathbf{t}$ and corresponding observed intensities $\mathbf{I}(\mathbf{t}\circ X)$ are collected and a mapping $\varphi:\mathbf{I}\to\mathbf{t}$ minimizing the error on these examples is estimated (see Fig. 2),

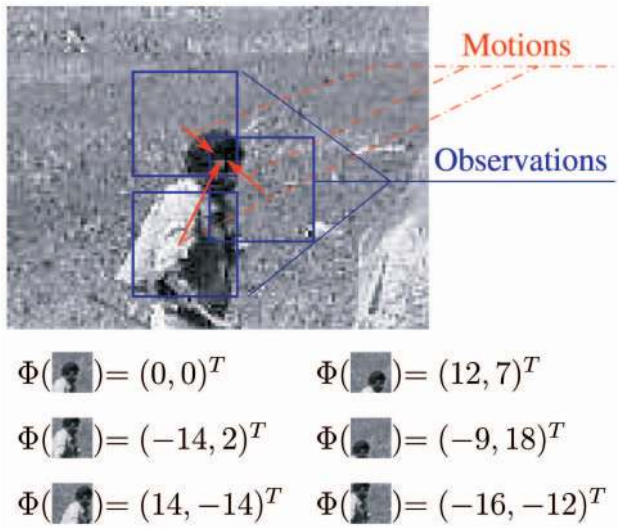

Fig. 2. Learning linear mapping between intensities and motion in advance. The mapping is learned by an LS method from a set of synthetically perturbed examples.

$$\varphi^* = \arg\min_{\varphi} \sum_{\mathbf{t}} \|\varphi(\mathbf{I}(\mathbf{t}\circ X)) - \mathbf{t}\|. \tag{2}$$

In the tracking stage, the learned mapping $\varphi^*(\mathbf{I})$ directly estimates motion parameters without necessity of online optimization of any criterion function.

Noticing that Lucas-Kanade tracker [1] solves a similar optimization task in each frame, one can replace the pseudoinverse operation by matrix $\mathtt{H}$ learned on a set of synthesized examples. Mapping φ then transforms to the linear function between intensities $\mathbf{I}(X\circ\mathbf{t})$ and motion $\mathbf{t}$,

$$\mathbf{t} = \varphi(\mathbf{I}(X)) = \mathtt{H}(\mathbf{I}(X) - \mathbf{J}(X)), \tag{3}$$

where $\mathtt{H}$ is the matrix of some learned coefficients. In the tracking procedure, motion parameters $\mathbf{t}$ are simply computed as a linear function $\mathtt{H}(\mathbf{I}(X)-\mathbf{J}(X)$ of the object intensities. We call such method *learned linear predictor* (LLiP). In the following, the learning of LLiP is described.

Let us suppose we are given an image template $\mathbf{J}=\mathbf{J}(X)$ and collected training pairs $(\mathbf{I}^i,\mathbf{t}^i)$ $(i=1\ldots d)$ of observed intensities $\mathbf{I}^i$ and corresponding motion parameters $\mathbf{t}^i$, which align the object with current frame. Then, the *training set* is an ordered pair $(\mathtt{I},\mathtt{T})$, such that $\mathtt{I}=[\mathbf{I}^1-\mathbf{J},\mathbf{I}^2-\mathbf{J},\ldots\mathbf{I}^d-\mathbf{J}]$ and $\mathtt{T}=[\mathbf{t}^1,\mathbf{t}^2,\ldots\mathbf{t}^d]$. Given the training set, LLiP coefficients minimizing the square of Euclidean error on the training set are found as follows:

First, the learning task is formulated and rewritten to a more convenient form:

$$\begin{aligned}\mathtt{H}^* &= \arg\min_{\mathtt{H}} \sum_{i=1}^{d} \left\|\mathtt{H}(\mathbf{I}^i-\mathbf{J}) - \mathbf{t}^i\right\|_2^2 = \arg\min_{\mathtt{H}} \|\mathtt{HI}-\mathtt{T}\|_F^2\\ &= \arg\min_{\mathtt{H}}\ \mathrm{trace}(\mathtt{HI}-\mathtt{T})(\mathtt{HI}-\mathtt{T})^\top\\ &= \arg\min_{\mathtt{H}}\ \mathrm{trace}(\mathtt{HII}^\top\mathtt{H}^\top - 2\mathtt{HIT}^\top + \mathtt{TT}^\top).\end{aligned}$$

Next, its derivative is set equal to zero:

$$\begin{aligned} 2\mathtt{H}^*\mathtt{I}\mathtt{I}^\top - 2\mathtt{T}\mathtt{I}^\top &= 0, \\ \mathtt{H}^*\mathtt{I}\mathtt{I}^\top &= \mathtt{T}\mathtt{I}^\top, \\ \mathtt{H}^* &= \mathtt{T}\underbrace{\mathtt{I}^\top(\mathtt{I}\mathtt{I}^\top)^{-1}}_{\mathtt{I}^+} = \mathtt{T}\mathtt{I}^+. \end{aligned} \tag{4}$$

Since the method is very fast and simple, it has various applications in tracking approaches. In particular, Cootes et al. [7], [12], [13] estimate the parameters of Active Appearance Model (AAM), i.e., deformable model with the shape and appearance parameters projected into a lower dimensional space by the PCA. They use a linear predictor (3) learned by the LS method (4) to estimate all parameters of the AAM. Since the linearity holds only for a small range of the parameters, the solution is iterated. Iterations are computed with the same matrix, but the length of the optimization step is locally optimized.

This approach was later adapted by Jurie and Dhome [6] for tracking of rigid objects. Unlike Cootes et al. [7], Jurie's linear predictors estimate local 2D translations only. The global motion is estimated from local motions by the RANSAC algorithm, showing the method to be very efficient and robust. Williams et al. [8] extended the approach to the nonlinear translation predictors learned by Relevance Vector Machine (RVM) [14]. Agarwal and Triggs [15] used RVM to learn the linear and nonlinear mapping for tracking of 3D human poses from silhouettes.

Drucker et al. [16] search for the regression function that has at most ϵ deviation from the actually obtained poses $\mathbf{t}^i$. Their method, which is called Support Vector Regression Machine, is similar to the Support Vector Machine [17] and allows also the extension for nonlinear kernels. Detailed description may be found in [18].

Zhou et al. [19] proposed greedy learning for additive regression function:

$$\varphi(\mathbf{I}(X)) = \sum_{i=1}^{c} \varphi_i(\mathbf{I}_i(X)), \tag{5}$$

where $\mathbf{I}(X)$ are some image features. The learning consists of c-steps, within each of them *weak regressor* $\varphi_i(\mathbf{I}(X)$ minimizing the training error is estimated.

Zhou et al. [19] use the weak regressor formed by a linear combination of binary functions. They constrained the coefficients of the linear combination to have the same absolute values. Such constraint allows to find a closed-form solution in each of c learning greedy steps. Bissacco et al. [20] extended the learning technique for the L-nary regression trees and showed that it outperforms [19].

3 Contribution

We contribute to the regression-based methods. Rather than proposing a special learning procedure for a special type of the regression function, we present an optimal way to concatenate different regression functions into a sequence. Our main idea follows the fact that the intensities (features) of pixels located close to the searched pose are usually more convenient for precise pose estimation than some other intensities.

Let us suppose we are given a class of regression functions. Each of the functions operates within a different range of poses and has different precisions and computational complexities. Given predefined range and precision, we want to design a regression-based tracking method. The simplest thing one can do is to select a function with sufficient range and precision. Of course, such a function need not even exists and, if so, it could have a very high computational complexity. The other possibility is to select a sequence of functions such that the first function provides a coarse estimate of the pose. The following function is consequently allowed to operate within a smaller range of poses. If it is true, that the intensities of pixels located close to the searched pose are more convenient for the pose estimation, such function naturally achieves a higher precision with a reasonable complexity. Similarly, another ancestor again refines from the precision of previously estimated poses.

In continuation of that, we define learning as a search for a sequence with the lowest computational complexity subject to predefined precision and range. We learn the optimal sequence of regression functions, which is, in general, superior to a single function. Since LLiPs are easy to operate and allow for good precision on low computational complexity, we demonstrate the method on the Sequences of LLiPs (SLLiPs). Note that the linear predictor can be naturally extended to an arbitrary linear combination of nonlinear mappings by data lifting; therefore, the linearity is not too much restricting condition.

We further extend the method for tracking of the objects modeled by a set of sequential predictors. While each predictor estimates local motion independently, object motion is determined by RANSAC from these local motions. We optimize the ratio between the number of RANSAC iterations and the number of used predictors subject to a user-predefined frame rate. Since we do not make any assumptions about the object pose, visibility, and suitability of the predictors, the set of used predictors must be optimized online. Therefore, we learn a set of predictors equally distributed on the object and select an *active* subset that optimizes trade-off between coverage and quality in each frame separately.

4 Method Overview

Because of robustness, the object is locally represented as a set of compact regions. Position of each compact region is determined by its *reference point*, e.g., the geometrical mean of pixels in the region. Since we do not make any a priori assumptions which positions are the most suitable for the motion estimation, we learn the SLLiPs for evenly distributed reference points on the object. During the learning stage, which is outlined in Algorithm 2, the globally optimal SLLiPs are estimated for all reference points.

Sections 5, 6, and 7 describe learning of the individual optimal SLLiP. Section 5 introduces definitions. Section 6 formulates the learning task as an optimization problem. In this section, we also show that an optimal SLLiP can be created exclusively from LLiPs learned by a minimax method. Hence, the learning is compound: First, a set of LLiPs is learned by minimax optimization (Section 6.1), and then a sequence of LLiPs creating an optimal SLLiP is selected (Section 6.2). An efficient heuristic for support set selection that minimizes error on training data is described in Section 7.

Algorithm 1.

1) Select a set of reference points.
2) For each reference point on the object:
   a) For some discretized values of parameters (ranges and complexities):
      - Generate examples of (observation, motion) pairs.
      - Learn a set of LLiPs by the minimax method (Section VI-A).
   b) Select LLiPs creating an optimal SLLiP, (Section VI-B).
3) Compute the optimal balance between a number of SLLiPs and RANSAC iterations (Section VIII-B).

NoSLLiP tracker, which is summarized in Algorithm 2, first selects a set of SLLiPs considering the trade-off between the quality and coverage of a visible part of the object (Section 8.1). These SLLiPs are used for local motion estimation in the particular frame. The global motion is determined by RANSAC, given the set of local motions. The trade-off between time spent with the local and global motion estimations is also considered and optimized in Section 8.2. The proposed method is experimentally verified on synthetic and real data with ground truth in Section 9.

Algorithm 2.

1) Select a set of active SLLiPs (Section VIII-A).
2) Estimate local motions by the selected SLLiPs.
3) Estimate object motion from the local motions by RANSAC (Section VIII-B).
4) Capture the next frame and goto 1.

5 Predictors, Properties, and Terminology

In this section, we define predictor and sequential predictor and show their fundamental properties, which are further used for learning. Let us suppose that the object state is given by object pose parameters (e.g., position).[1] In each frame, we update the object state by current motion parameters estimated by the predictor from a subset of object pixels. The subset of the object pixels is called the *support set* $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_c\}$. The intensities observed on the support set X are collected in the *observation vector* $\mathbf{I}(X)$.

Ideally, a predictor would use a support set minimizing the prediction error. However, the problem has combinatorial complexity and we discuss it later in Section 7; let us assume for now that a support set has been selected.

We denote $(\mathbf{t} \circ X)$ as the support set transformed 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 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 set of motions $\mathbf{t}$ that could have generated the observation $\mathbf{I}(\mathbf{t} \circ X)$. This mapping, which is called a *regressor*, assigns a p-vector of motion parameters to a c-vector of observation.

1. In general, object could be represented by more than one predictor. Such representation allows for robust object pose estimation by RANSAC and we discuss this extension in Section 8. For now, let us suppose that only one predictor is associated with the object.

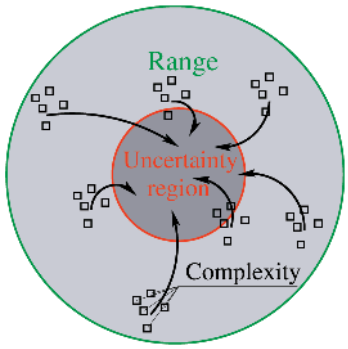

Fig. 3. Definitions: Range, accuracy, and complexity. An example with 2D translations.

Regressors $\hat{\varphi}$ are completely characterized by their complexity, range, and uncertainty region.

Definition 1. Complexity $c(\hat{\varphi})$ *of regressor* $\hat{\varphi}$ *is a value proportional to the computational cost of the regressor. It is equal to the size of a support set for linear regressor.*

Definition 2. Range $R(\hat{\varphi})$ *of the regressor* $\hat{\varphi}$ *is a set of motion parameters.*[2]

Definition 3. *The* uncertainty region *of the regressor* $\hat{\varphi}$ *is the smallest region*

$$\Lambda(\hat{\varphi}) = \{\Delta\mathbf{t} \mid \Delta\mathbf{t} = \mathbf{t} \circ \hat{\varphi}(\mathbf{I}(\mathbf{t} \circ X)),\ \forall \mathbf{t} \in R(\hat{\varphi})\}. \quad (6)$$

The uncertainty region is the smallest region within which all the prediction errors from the range $R(\hat{\varphi})$ *lie (see Fig. 3).*

In order to simplify the learning procedure, we select only a class (e.g., circles or squares) $\{\Lambda_\lambda\}_{\lambda \in \mathbb{R}}$ parameterizable by one scalar parameter $\lambda \in \mathbb{R}$ such that

$$\forall\, \lambda_1, \lambda_2 \in \mathbb{R} : \lambda_1 < \lambda_2 \Rightarrow \Lambda_{\lambda_1} \subset \Lambda_{\lambda_2}. \quad (7)$$

Ranges R_r are selected from the same class of regions and parameterized by the same parameter $r \in \mathbb{R}$. According to (7), parameter λ (and r) are proportional to the area of the region; therefore, we sometimes refer to it as an area of the region and use notation $\lambda(\hat{\varphi})$ (and $r(\hat{\varphi})$) to denote corresponding values of $\Lambda(\hat{\varphi})$ and $R(\hat{\varphi})$, respectively. An extension to regions parameterizable by more than one parameter is discussed later.

5.1 Predictors

Definition 4. Predictor $\varphi(c, r, \lambda)$ *is an ordered 4-tuple* $(\hat{\varphi}, X, R_r, \Lambda_\lambda)$, *where* X *is the support set,* $c \approx |X|$ *is complexity (for linear case* $|X| = c$*),* $\hat{\varphi}$ *is the regressor,* R_r *is range, and* Λ_λ *is the uncertainty region.*

Even though we defined the predictor as 4-tuple, we parameterize all predictors by the three parameters: complexity c, range r, and uncertainty region λ, the regressor $\hat{\varphi}$ is omitted. It actually says that two predictors with the same (c, r, λ) and different regressors $\hat{\varphi}_1, \hat{\varphi}_2$ are equivalent.

In order to assure that increase in complexity does not reduce the prediction abilities, we further restrict ourselves to the class of support sets satisfying that every support set

2. Note that this is not the range in the usual mathematical meaning.

contains all support sets with lower complexity. This is assured by the successive support set construction algorithm described in Section 7. This is, however, still an insufficient assumption. It is also required that regressors must be able to ignore values of some input pixels. In the following definition, we define the class of such regressors.

Definition 5. *Let $\mathcal{F}^c$ denote some class of regressors $\hat{\varphi} : \mathbb{R}^c \to \mathbb{R}^p$ with the same support set X. Let*

$$\mathcal{F} = \{\mathcal{F}^1 \cup \mathcal{F}^2 \cup \cdots \cup \mathcal{F}^c \cdots\}.$$

$\mathcal{F}$ is called domain-independent class *if*

$$\forall\, c\, \forall\, \hat{\varphi}_1 \in \mathcal{F}^c\ \exists \hat{\varphi}_2 \in \mathcal{F}^{c+1} \text{ such that}$$
$$\forall\, \mathbf{I} \in \mathbb{R}^c\ \forall\, u \in \mathbb{R}\ \ \hat{\varphi}_2([\mathbf{I}, u]) = \hat{\varphi}_1(\mathbf{I}).$$

This is, for example, satisfied for the following class of regressors:

$$\mathcal{F}^1 = \{a_1 \cdot x_1 \,|\, a_1 \in \mathbb{R}\},$$
$$\mathcal{F}^2 = \{a_1 \cdot x_1 + a_2 \cdot x_2 \,|\, a_1, a_2 \in \mathbb{R}\},$$
$$\mathcal{F}^c = \left\{\sum_{i=1}^{c} a_i \cdot x_i \,|\, a_i \in \mathbb{R}, i = 1 \ldots c\right\},$$

parameterized by coefficients $a_i \in \mathbb{R}$, because it can ignore an arbitrary input x_i by setting the corresponding coefficient to zero. On the contrary, the following class is not a domain-independent class:

$$\mathcal{F}^1 = \{a \cdot x_1 \,|\, a \in \mathbb{R}\},$$
$$\mathcal{F}^2 = \{a \cdot (x_1 + x_2) \,|\, a \in \mathbb{R}\},$$
$$\mathcal{F}^c = \left\{a \cdot \sum_{i=1}^{c} x_i \,|\, a \in \mathbb{R}\right\},$$

parameterized only by one coefficient $a \in \mathbb{R}$. In general, the class of polynomials of an arbitrary order parameterized by all of their coefficients is an example of the domain-independent class.

Note that not all good properties of the predictors are simultaneously achievable. It is clear that there is no ideal predictor that would simultaneously have (very) low complexity, (very) large range, and (very) small error. We denote the achievable subset of predictors in (c, r, λ) space by ω (see Fig. 4 for an example). Predictors lying on the border of ω are very important because it will be shown later that optimal sequential predictors are exclusively formed from these predictors.

Definition 6. λ-minimal predictors $\varphi^+(c, r)$ *are predictors having the minimal achievable λ for a given range r and complexity c:*

$$\varphi^+(c, r) \in \arg\min_{\varphi}\{\lambda \mid \varphi(c, r, \lambda) \in \omega\}. \tag{8}$$

Note that λ-minimal predictors are the predictors lying on the boundary of ω (see Fig. 4c).

A simple consequence of Definition 5 is that more complex predictors can do everything that the simpler ones can. This is shown in the two following propositions, which summarize the properties of λ-minimal predictors. The propositions are not crucial for the understanding of the learning procedure; however, we later use them to prove that the learning algorithm can be simplified.

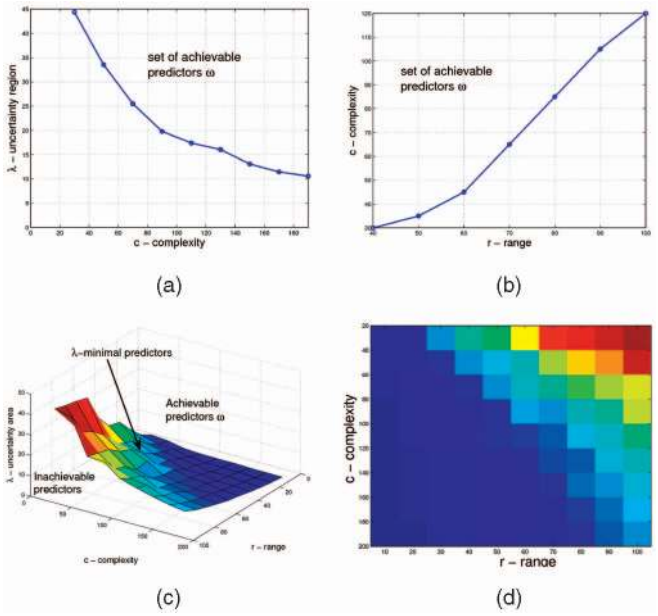

Fig. 4. Set of achievable predictors. Color codes the size of uncertainty region. (a) λ-lower bound as a function of complexity. (b) Complexity as a function of range. (c) Set of achievable predictors in (c, r, λ)-space. (d) Rolled-out λ-lower bound.

Proposition 1. *The uncertainty region of a λ-minimal* predictor *is a nonincreasing function of the complexity c.*

Proof. We prove that the uncertainty region cannot increase with complexity (see, for example, Fig. 4a). Let us suppose we are given two λ-minimal predictors with regressors $\hat{\varphi}_1^+ \in \mathcal{F}^c, \hat{\varphi}_2^+ \in \mathcal{F}^{c+1}$. Since λ-minimal predictors are predictors with minimum uncertainty region, their regressors have to satisfy

$$\hat{\varphi}_1^+ \in \arg\min_{\hat{\varphi}_1 \in \mathcal{F}^c} \lambda(\hat{\varphi}_1), \tag{9}$$

$$\hat{\varphi}_2^+ \in \arg\min_{\hat{\varphi}_2 \in \mathcal{F}^{c+1}} \lambda(\hat{\varphi}_2). \tag{10}$$

We prove that a regressor with higher complexity $\hat{\varphi}_2^+$ has the uncertainty region smaller or equal to the uncertainty region of a regressor with smaller complexity $\hat{\varphi}_1^+$, i.e., $\lambda(\hat{\varphi}_1^+) \geq \lambda(\hat{\varphi}_2^+)$. This fact is shown by contradiction; therefore, we assume that

$$\lambda\big(\hat{\varphi}_1^+\big) < \lambda\big(\hat{\varphi}_2^+\big). \tag{11}$$

Since we know that $\hat{\varphi}_1^+ \in \mathcal{F}^c$, then, according to Definition 5, there exists some $\hat{\varphi}_3^+ \in \mathcal{F}^{c+1}$ such that

$$\forall\, \mathbf{I} \in \mathbb{R}^c\ \forall\, u \in \mathbb{R}\ \ \hat{\varphi}_3^+(\mathbf{I}) = \hat{\varphi}_1^*([\mathbf{I}, u]).$$

It also implies that $\lambda(\hat{\varphi}_3^+) = \lambda(\hat{\varphi}_1^+)$. Hence, according to the assumed inequality (11),

$$\lambda\big(\hat{\varphi}_1^+\big) = \lambda\big(\hat{\varphi}_3^+\big) < \lambda\big(\hat{\varphi}_2^+\big).$$

This leads us to the contradiction, because there exists regressor $\hat{\varphi}_3^+$, which has smaller uncertainty region than $\hat{\varphi}_2^+$, and therefore $\hat{\varphi}_2^+$ could not be the optimal

solution of problem (10) and, consequently, $\hat{\varphi}_2^+$ could not be the regressor of any λ-minimal predictor with complexity $c+1$. □

Note that Proposition 1 is valid for arbitrary predictors that are optimal with respect to some criterion. For example, if we had been dealing with predictors minimizing mean euclidean prediction error, say e, then the minimal e would have been a nonincreasing function of the complexity as well.

Proposition 2. *Uncertainty region of λ-minimal predictor is a nondecreasing function of the range.*

Proof. Given two λ-minimal predictors

$$\varphi_1^+ = \varphi^+(c, r_1) = \arg\min_{\varphi}\{\lambda \mid \varphi(c, r_1, \lambda) \in \omega\},$$
$$\varphi_2^+ = \varphi^+(c, r_2) = \arg\min_{\varphi}\{\lambda \mid \varphi(c, r_2, \lambda) \in \omega\},$$

such that $r_2 > r_1$, we prove that the predictor with larger range r_2 has larger or at most the same uncertainty region as a predictor with smaller range r_1, i.e., $r_2 > r_1 \Rightarrow \lambda(\varphi_2^+) \geq \lambda(\varphi_1^+)$.

The implication is proved by contradiction. We assume $r_2 > r_1$ and $\lambda(\varphi_2^+) < \lambda(\varphi_1^+)$. Since $R_{r_1} \subset R_{r_2}$, the predictor φ_2 can also predict every motion from range r_1. Consequently, we can define a new predictor $\varphi_1' = (\hat{\varphi}_2^+, X, R_{r_1}, \Lambda_{\lambda_2})$ operating on range r_1 with

$$\lambda(\varphi_1') = \lambda(\varphi_2^+) < \lambda(\varphi_1^+). \tag{12}$$

This is in contradiction to the fact that φ_1^+ is a λ-minimal predictor because we have just found another predictor φ_1', which has a smaller uncertainty region. □

5.2 Sequential Predictor

It directly follows from Proposition 1 that the higher the complexity is, the better the prediction is. However, increasing the complexity has diminishing returns (see, for example, Fig. 4). For large ranges, it is usually very difficult to achieve a good prediction even with the complexity corresponding to the cardinality of the complete template. In order to overcome this limitation, we develop a *sequential predictor* $\Phi = (\varphi_1 \dots \varphi_m)$ (see Fig. 5), which estimates vector of motion parameter $\mathbf{t}$ in m steps as follows:

$$\begin{aligned}
\mathbf{t}_1 &= \hat{\varphi}_1(\mathbf{I}(X_1)),\\
\mathbf{t}_2 &= \hat{\varphi}_2(\mathbf{I}(\mathbf{t}_1 \circ X_2)),\\
\mathbf{t}_3 &= \hat{\varphi}_3(\mathbf{I}(\mathbf{t}_2 \circ \mathbf{t}_1 \circ X_3)),\\
&\vdots\\
\mathbf{t}_m &= \hat{\varphi}_m\left(\mathbf{I}\left(\left(\bigcirc_{i=1}^{m-1} \mathbf{t}_i\right) \circ X_m\right)\right),\\
\mathbf{t} &= \bigcirc_{i=1}^{m} \mathbf{t}_i.
\end{aligned}$$

The first vector of motion parameters $\mathbf{t}_1$ is estimated directly by predictor φ_1 from the intensities observed in support set X_1. This predictor has a known uncertainty region λ_1 within which all its predictions lie. Therefore, the successive predictor φ_2 is learned only on the range $r_2 \approx \lambda_1$ corresponding to this uncertainty region, which is usually significantly smaller than range r_1 of the first predictor. The smaller range yields the smaller uncertainty region. The advantage is that the predictors in the sequence are more and more specific, which consequently allows the prediction to be very accurate for reasonably textured regions. It is experimentally shown that the sequential predictor, which is superior to a single predictor, yields significantly lower complexity and a higher precision.

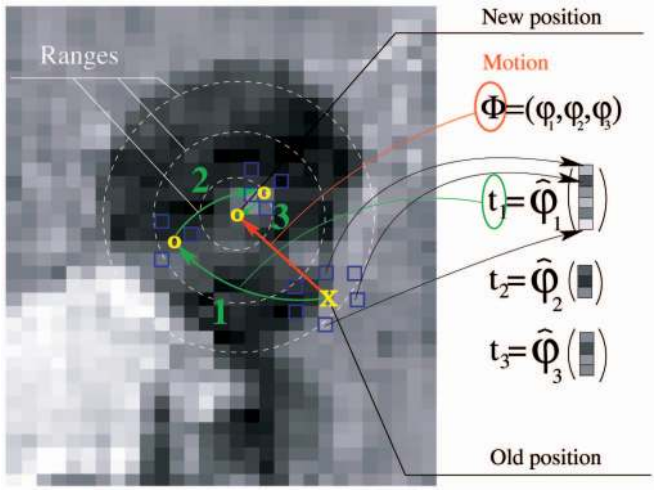

Fig. 5. *Sequential predictor* $\Phi = (\varphi_1 \dots \varphi_m)$ estimates vector of motion parameters $\mathbf{t}$ (denoted by red arrow) in m steps by m different predictors $\varphi_1 \dots \varphi_m$. Particular predictors and the number of steps are the subject of learning.

Obviously, we consider only those sequential predictors that satisfy $R(\hat{\varphi}_{i+1}) \supseteq \Lambda(\hat{\varphi}_i)$, $i = 1 \dots m-1$. The range of each particular predictor must accommodate the uncertainty region of its predecessor at least. The uncertainty region of the sequential predictor is understood as the uncertainty region of the last predictor and its range as the range of the first predictor.

Definition 7. *The* sequential predictor *of order m is an m-tuple $\Phi = (\varphi_1(c_1, r_1, \lambda_1), \dots, \varphi_m(c_m, r_m, \lambda_m)$ of predictors $\varphi_i \in \omega$ such that $R(r_{i+1}) \supseteq \Lambda(\lambda_i)$, $i = 1 \dots m-1$. The uncertainty region of the sequential predictor Φ is λ_m and its range is r_1.*

6 Learning Optimal Sequential Predictors

In the previous section, we defined the predictor and the sequential predictor. In this section, we first define the optimal sequential predictor and show that it can be created exclusively from the λ-minimal predictors (Definition 6). Section 6.1 describes learning of the λ-minimal predictor, given a training set. In Section 6.2, a set of λ-minimal predictors with different complexities and ranges is learned; selection of an optimal sequence of the predictors from the set is formulated as a search for the cheapest path in a graph.

Definition 8. *The* optimal sequential predictor *is*

$$\Phi^* = \arg\min_{\Phi \in \Omega, m \in \mathbb{N}^+} \left\{ \sum_{i=1}^{m} c_i \mid r_1 \geq r_0, \lambda_m \leq \lambda_0 \right\}, \tag{13}$$

where Ω is the set of all sequential predictors, r_0 is the predefined range, λ_0 is the predefined uncertainty region, and $\mathbb{N}^+$ is the set of positive integral numbers.

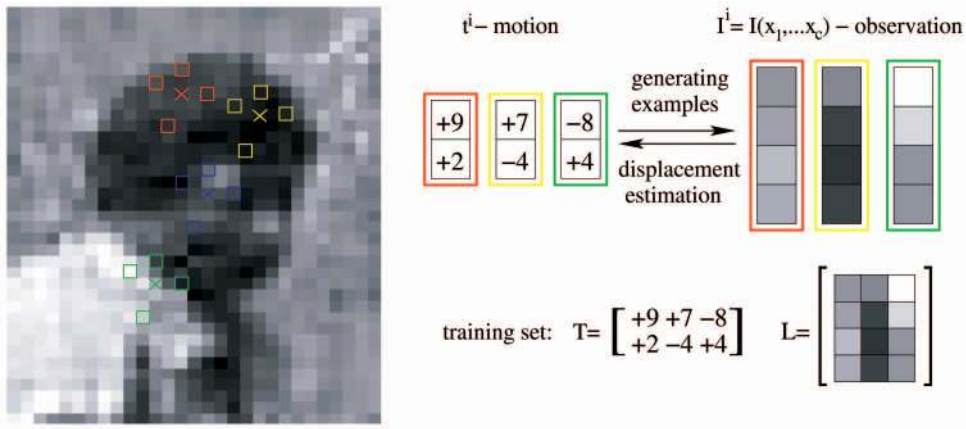

Fig. 6. The image is perturbed by the motion parameters included in the range r, creating the set of synthesized examples of observed intensities $\mathbf{I}^i$ and motions $\mathbf{t}^i$.

Proposition 3. *There is at least one optimal sequential predictor created exclusively from the λ-minimal predictors.*

Proof. The proposition is proven by showing that any non-λ-minimal predictor can be replaced by a λ-minimal predictor of the same complexity. It is then clear that, for every non-λ-minimal predictor φ_i from the optimal sequence, there exists a λ-minimal predictor φ_i^+ with the same complexity, such that the following holds:

$$\Lambda(\varphi_i^+) \subset \Lambda(\varphi_i) \subset R(\varphi_i) \subset R(\varphi_i^+).$$

Therefore, φ_i^+ can replace φ_i. □

Therefore, we consider only predictors with the smallest uncertainty region λ, i.e., predictors lying on the λ-lower bound defined by (8). In that way, the λ-lower bound, 2D manifold in (c, r, λ) space (Fig. 4c), is rolled out to the (c, r) space (Fig. 4d). Task (13) reduces to

$$\Phi^* = \underset{\Phi\in\Omega^+, m\in\mathbb{N}^+}{\arg\min} \left\{ \sum_{i=1}^{m} c_i \mid r_1 \geq r_0, \lambda_m \leq \lambda_0 \right\}, \qquad (14)$$

where Ω^+ is the set of sequential predictors created only by the λ-minimal predictors (8).

The procedure of linear λ-minimal predictor learning is carried out by linear programming in Section 6.1. In Section 6.2, a sequence of the λ-minimal predictors creating the optimal sequential predictor Φ^* (14) is selected from a set of learned λ-minimal predictors. The problem is formulated as searching of the cheapest path in a graph.

6.1 Learning Linear λ-Minimal Predictor $\hat{\varphi}^+$

6.1.1 Linear Predictor

In order to estimate a predictor satisfying (8), the regressor $\hat{\varphi}$ needs to be specified in detail. We restrict ourselves to LLiPs, i.e., predictors with linear regressor.

Linear regressor $\hat{\varphi}_L$ is a linear mapping defined as

$$\mathbf{t} = \hat{\varphi}_L(\mathbf{I}) = \mathtt{H}\mathbf{I}, \qquad (15)$$

where $\mathtt{H}$ is a $2 \times c$ matrix. Similarly, sequential linear predictor is the SLLiP. Note that the time required for motion estimation by LLiP is determined by the size of its support set; therefore, $c = |X|$.

Although we will further work with linear predictors, the method allows a natural extension to an arbitrary class of functions formed by a linear combination of kernel functions by *data lifting*. Polynomial mapping of a given order is an example. In that case, all monomials are considered as further observed intensities. It allows the learning procedure to deal with nonlinear mappings via linear mappings with higher dimension.

Training set construction. Let us suppose we are given a reference point, a support set, and a predefined range of motion within which the regressor is assumed to operate. We perturb the support set by the 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. 6). Given the observed intensity, we search for a mapping assigning motion $\mathbf{t}^i = -(\mathbf{q}^i)$, which warps the support set X^i back to the original support set X. These examples are stored in matrices $\mathtt{I} = [\mathbf{I}^1 \dots \mathbf{I}^d]$ and $\mathtt{T} = [\mathbf{t}^1 \dots \mathbf{t}^d]$. The ordered triple $(\mathtt{I}, \mathtt{T}, \mathcal{X})$ of such matrices and ordered d-tuple of support sets $\mathcal{X} = \{X^1 \dots X^d\}$ is called a *training set*.

Learning linear regressor given a training set. Let us suppose we are given a training set $(\mathtt{I}, \mathtt{T}, \mathtt{X})$. While Jurie and Dhome [6] obtain $\mathtt{H}$ by the least squares method $\mathtt{H} = \mathtt{T}\mathtt{I}^+ = \mathtt{T}\mathtt{I}^\top(\mathtt{I}\mathtt{I}^\top)^{-1}$, we search for λ-minimal predictor (Definition 6), i.e., the predictor with the smallest uncertainty region (8). As we mentioned before, the uncertainty region is assumed to be from a class parameterizable by one scalar parameter λ. In the following, we show how to find λ-minimal predictor for the class of squares and rectangles. See the Appendix, which can be found on the Computer Society Digital Library at http:// doi.ieeecomputersociety.org/10.1109/TPAMI.2008.119, for other uncertainty region classes (e.g., circles or ellipses).

Restricting to the square-shaped uncertainty regions centered in the origin of the coordinate system (see Fig. 7a) and parameterized by parameter λ (8) defining the λ-minimal predictor, simplifies as follows:

$$\begin{aligned}\mathtt{H}^* &= \arg\min_{\mathtt{H}}\left(\max_i \|\mathtt{H}\mathbf{I}^i - \mathbf{t}^i\|_\infty\right)\\ &= \arg\min_{\mathtt{H},\lambda}\{\lambda \mid \forall_i |\mathtt{H}\mathbf{I}^i - \mathbf{t}^i| < \mathbf{1}\lambda\}\\ &= \arg\min_{\mathtt{H},\lambda}\ \lambda\\ &\quad \text{subject to}: -\lambda \le (\mathtt{H}\mathbf{I}^i)_k - \mathbf{t}_k^i \le \lambda,\\ &\qquad\qquad i = 1\ldots d, k = 1,2.\end{aligned} \tag{16}$$

We reformulate problem (16) as a linear program

$$\min_{\mathbf{x}}\{\mathbf{c}^\top\mathbf{x} \mid \mathtt{A}\mathbf{x} \le \mathbf{b}\}, \tag{17}$$

where

$$\mathbf{x} = \begin{bmatrix}\mathbf{h}_1\\ \vdots\\ \mathbf{h}_p\\ \lambda\end{bmatrix},\ \mathbf{c} = \begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix},\ \mathtt{A} = \begin{bmatrix}\mathtt{I}^\top & \mathtt{0} & \ldots & \mathtt{0} & -\mathbf{1}\\ \mathtt{0} & \mathtt{I}^\top & \ldots & \mathtt{0} & -\mathbf{1}\\ \vdots & & \ddots & & \vdots\\ \mathtt{0} & \ldots & \mathtt{0} & \mathtt{I}^\top & -\mathbf{1}\\ -\mathtt{I}^\top & \mathtt{0} & \ldots & \mathtt{0} & -\mathbf{1}\\ \mathtt{0} & -\mathtt{I}^\top & \ldots & \mathtt{0} & -\mathbf{1}\\ \vdots & & \ddots & & \vdots\\ \mathtt{0} & \ldots & \mathtt{0} & -\mathtt{I}^\top & -\mathbf{1}\end{bmatrix},$$

$$\mathbf{b} = \begin{bmatrix}\mathbf{t}_1\\ \vdots\\ \mathbf{t}_p\\ -\mathbf{t}_1\\ \vdots\\ -\mathbf{t}_p\end{bmatrix},$$

with $\mathbf{h}_i$ as the column vector corresponding to the ith row of matrix $\mathtt{H}$.

Since the computation of each component of the predicted parameters can be considered as an independent task, estimation of each row of $\mathtt{H}$ is solved separately.[3] Hence, the task splits into p independent linear problems, where each of them determines one row $\mathbf{h}_j^{T*}$ of matrix $\mathtt{H}$. The problem is solved as follows:

$$\begin{aligned}\mathbf{h}_j^{T*} &= \arg\min_{\mathbf{h}_j}\max_i\left\{\left\|\mathbf{h}_j^\top\mathbf{I}^i - t_j^i\right\|_\infty\right\}\\ &= \arg\min_{\mathbf{h}_j^\top,\lambda_j}\left\{\lambda_j \mid \forall_i \left|\mathbf{h}_j^\top\mathbf{I}^i - t_j^i\right| < \lambda_j\right\}.\end{aligned}$$

Denoting

$$\mathbf{x}_j = \begin{bmatrix}\mathbf{h}_j\\ \lambda_j\end{bmatrix},\ \mathbf{c} = \begin{bmatrix}0\\ \vdots\\ 0\\ 1\end{bmatrix},\ \mathtt{A} = \begin{bmatrix}\mathtt{I}^\top & \mathbf{0} & -\mathbf{1}\\ -\mathtt{I}^\top & -\mathbf{1} & \mathbf{0}\end{bmatrix},\ \mathbf{b}_j = [\mathbf{t}_j],$$

linear programming problem (17) is obtained. The shape of such uncertainty region is a rectangle, which is in 2D space parameterized by two parameters—length of its sides (see Fig. 7c). Since we want to work with uncertainty regions parameterizable by one parameter, it could be considered as a square with the side equal to the longer rectangle side. The result is the same as if the square shape is assumed in advance and the learning is significantly faster.

If L_∞ in problem (16) is replaced by L_1, the uncertainty region is L_2 hypercube (square in 2D) rotated by $45°$ and the problem is solved alike (see Fig. 7b). The combination of L_∞ and L_1 norms also allows to work with L_2 circle approximation (see Fig. 7d). Note that we can also work with elliptic regions as shown in Figs. 7d, 7e, 7f, and 7g. In order to adjust a trade-off between the robustness of the minimax solution and the accuracy of the LS solution, it is also possible to formulate the criterion as a weighted sum of LS error and minimax error, which can be shown to be a semidefinite problem (see Fig. 7h). A detailed description of such uncertainty region extensions can be found in [21].

6.2 Learning Optimal Sequential Predictor Φ^*

In this section, we describe the selection of the optimal sequence of predictors from a set of learned λ-minimal predictors. We assume that we are able to estimate the λ-minimal predictors (8), e.g., the linear predictors as shown in the previous section. The set of λ-minimal predictors $\varphi^+(c,r)$ for some discretized values of complexities $c \in C$ and ranges $r \in R$ is denoted by ω^+. Note that ω^+ is actually a subset of the set of all possible λ-minimal predictors; however, for the sake of simplicity, we use the same notation. Fig. 8a shows uncertainty region $\lambda(c,r)$ (size coded by color) of the λ-minimal predictors as a function of complexity $c \in C$ (vertical axis) and range $r \in R$ (horizontal axis).

Given the set ω^+, predefined range r_0, and uncertainty region λ_0, we search for an ordered subset of ω^+ that forms the optimal sequential predictor Φ^*, which minimizes the complexity. Since the predefined range r_0 of the sequential predictor is the range $r_1 = r_0$ of the first predictor in the sequence, the first predictor must lie in the corresponding (usually the most right) column. For this range, the predictors with different complexities are available in that column. The higher the complexity is, the smaller the uncertainty region (see Fig. 8a), where the size of the uncertainty region decreases with the complexity for each particular range. Selection of a particular complexity c_1 determines the first λ-minimal predictor $\varphi^+(c_1,r_1)$ in the sequence. The size of the corresponding uncertainty region $\lambda(\varphi^+(c_1,r_1))$ determines an admissible range r_2 of the following predictor, which has to be at least as large as the uncertainty region according to its definition, i.e., $r_2 \ge \lambda(c_1,r_1)$. The following proposition shows that it is sufficient to consider only the smallest possible range.

Proposition 4. *Range r_i of a λ-minimal predictor $\varphi^+(c_i,r_i)$ in an optimal sequence of λ-minimal predictors has to be as tight as possible to the uncertainty region λ_{i-1} of its predecessor, i.e., asymptotically, in a continuous case $r_i = \lambda_{i-1}$.*

Proof. The uncertainty region is a nonincreasing function of complexity, according to Proposition 1:

$$c_2 > c_1 \Rightarrow \lambda(\varphi^+(c_2,r)) \le \lambda(\varphi^+(c_1,r)).$$

However, a λ-minimal predictor whose complexity can be decreased without uncertainty region increase cannot be part of an optimal sequence. We therefore consider only a λ-minimal predictor whose uncertainty region is a

3. This is significantly faster than the computation of one larger problem.

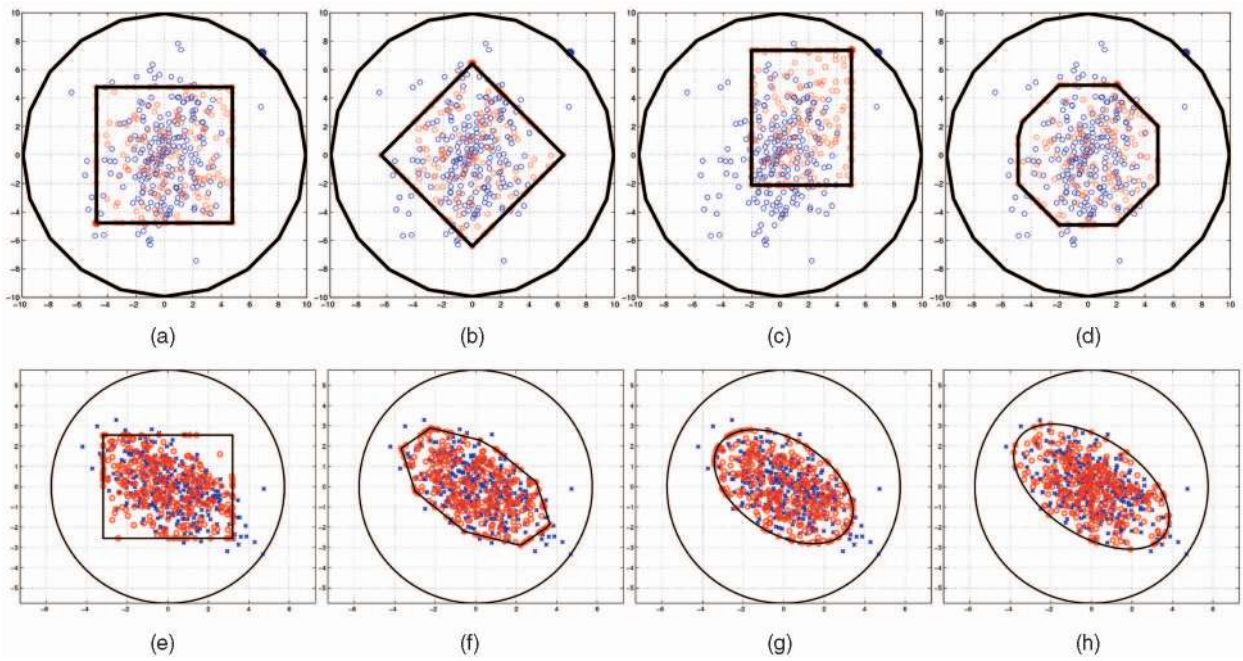

Fig. 7. **Different classes of uncertainty regions:** Points correspond to the prediction errors Δt of 2D translation on a training set. Errors of the predictor learned by the LS method are in blue and by the minimax method in red. Uncertainty regions and ranges are black. Only (a), (b), and (e) are described in this paper, see [21] for a detailed description of the other classes. (a) L_∞-circle. (b) L_1-circle. (c) Biased rectangle. (d) LP circle approx. (e) LP rectangle. (f) LP ellipse approx. (g) SDP ellipse. (h) SDP LS ellipse.

decreasing function of complexity, i.e., for which the following holds:

$$c_2 > c_1 \Rightarrow \lambda(\varphi^+(c_2, r)) < \lambda(\varphi^+(c_1, r)),$$

and, consequently,

$$\lambda(\varphi^+(c_2, r)) \geq \lambda(\varphi^+(c_1, r)) \Rightarrow c_2 \leq c_1. \tag{18}$$

Hence, the uncertainty region is strictly a decreasing function of the complexity. Putting this together with Proposition 2, which claims that the uncertainty region is a nondecreasing function of range, we prove that the complexity is a nondecreasing function of the range for every fixed $\lambda_0 = \lambda(\varphi^+(c_1, r_1)) = \lambda(\varphi^+(c_2, r_2))$ because

$$r_2 > r_1 \underset{\text{Prop.2}}{\Rightarrow} \begin{array}{l} \lambda(\varphi^+(c_1, r_2)) \geq \lambda(\varphi^+(c_1, r_1)) \\ \lambda(\varphi^+(c_2, r_2)) \geq \lambda(\varphi^+(c_2, r_1)) \\ \lambda(\varphi^+(c_1, r_1)) = \lambda(\varphi^+(c_2, r_2)) \end{array} \Rightarrow$$

$$\Rightarrow \begin{array}{l} \lambda(\varphi^+(c_1, r_2)) \geq \lambda(\varphi^+(c_2, r_2)) \\ \lambda(\varphi^+(c_1, r_1)) \geq \lambda(\varphi^+(c_2, r_1)) \end{array} \underset{\text{Eq. (18)}}{\Rightarrow} c_2 \leq c_1.$$

Since the complexity is a nondecreasing function of the range, considering a larger range $r_i > \lambda_{i-1}$ than necessary leads only to the increase of the complexity c_i. Taking into account that this would necessarily increase the complexity of the resulting sequential predictor, the smallest possible range $r_i = \lambda_{i-1}$ must be used. □

Note that if only the smallest possible ranges are considered, then the constructed graph has at most $|C| \cdot |R|$ edges. On the contrary, without Proposition 4, the constructed graph would have $|C| \cdot |R|^2$ edges.

The arrows in Fig. 8a show the smallest possible ranges for the predictors with different complexities. A sequence with the last predictor with uncertainty region λ_m smaller than λ_0 can be constructed, see, for example, the two sequences in Fig. 8b. Furthermore, we search for the sequence consisting of predictors converging to the sufficiently small uncertainty regions with the lowest complexity.

We formulate the previous problem as the search for the cheapest path in the graph $\mathcal{G} = (V \equiv R, E \subseteq R \times R, \alpha : E \to C)$, where R is the set of considered ranges, C is the set of considered complexities, and operator α assigns a cost to each edge (see Fig. 8). It means that each range is associated with a vertex and a set of edges starting from this range, which stand for predictors

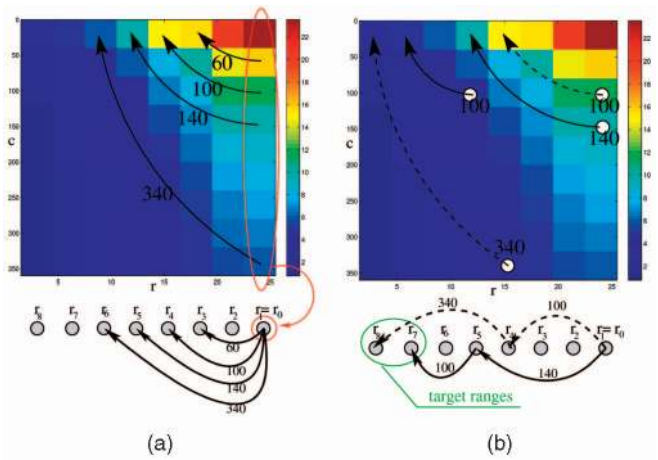

Fig. 8. (a) Construction of a graph $\mathcal{G}$ from a set of λ-minimal predictors ω^+. Different complexities of the first predictor lead to different uncertainty regions and therefore different ranges of the second predictor. Edges from the range r_0 of the first predictor, depicted by black arrows, show the ranges of the second predictor corresponding to the complexities of the first predictor. The cost of edges corresponds to the complexities. (b) Two paths to the target ranges (solid line denotes the optimal path).

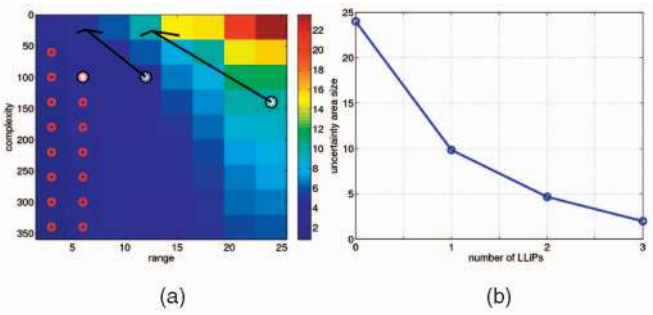

Fig. 9. (a) Size of uncertainty regions (coded by colors) as a function of complexity c (vertical axis) and range r (horizontal axis) and the optimal path from the initial r_0 to a predictor with sufficiently small uncertainty region (red circles). (b) Size of uncertainty region after each iteration ($\text{number of LLiPs} = 0$ corresponds to the range $r_0 = 25$).

with different complexities. Edge cost is equal to its complexity. We construct the graph by adding forward edges for each particular range.

The Dijkstra algorithm [22] searches for the cheapest path to the ranges with predictors with sufficiently small uncertainty regions, depicted by red circles in Fig. 9a. These predictors are called *target* predictors and their ranges are called *target* ranges. The solution is a sequence of predictors associated with edges on the cheapest path to a target range plus its cheapest target predictor. If more than one target range exists, then there are more possible solutions and the cheapest solution is selected. The solution is the optimal sequential predictor (14). The method is summarized in Algorithm 3.

Algorithm 3.

1) Estimate set of λ-minimal predictors $\omega^+ = \{\varphi^+(c,r) \mid c \in C,\ r \in R\}$.
2) Construct graph $\mathcal{G} = (V \equiv R, E \subseteq R \times R, \alpha : E \to C)$:
   for each $r \in R$ and each $c \in C$,
   a) Find the smallest possible following range v^* achievable by $\varphi^+(c,r)$:
   $v^* = \text{argmin}_{v \in R} \{v \mid \lambda(c,r) < v\}$
   b) $E = E \cup (r, v^*)$ and $\alpha(r, v^*) = c$
   end
3) Dijkstra($\mathcal{G}$, r_0) $\Rightarrow$ Compute the cheapest paths from r_0 to each range $r \in R$ by Dijkstra algorithm.
4) $\rho(r)$ denotes complexity of the cheapest path to range r. Consequently, $\varphi_t^+(c_t, r_t) = \text{argmin}_{\varphi^+(c,r) \in \omega_T} \{\rho(r) + c\}$ is the last predictor.
5) The optimal sequential predictor is created from the sequence of predictors associated with the edges of the cheapest path to r_t and the last predictor $\varphi_t^+(c_t, r_t)$.

The optimal path is depicted in Fig. 9a. For instance, the optimal sequence for the example in Fig. 9a, where $r_0 = 100$, $\lambda_0 = 2$, is created as $\Phi^* = (\varphi^+(140,25), \varphi^+(100,12), \varphi^+(100,5))$ and the corresponding uncertainty regions are (10, 4.5, 2).

Note that, due to simplicity, we focus on the one-variable parameterized uncertainty regions. Extension of the proposed method to the more than one-variable parameterized uncertainty regions is straightforward. For w-dimensional parameterization of the uncertainty region, ω^+ is $w+1$-dimensional space.

7 Selection of Support Set for Efficient Tracking

Until now, we assumed that the support set was given. Since the support set selection, which minimizes an error on a training set, has combinatorial complexity, we propose a heuristic method. The only condition on the proposed heuristic is that every selected support set of complexity c also contains support set of complexity $c-1$, which consequently assures monotonicity of the λ-bound, as shown in Section 5.

Let us suppose we are given training set $(\mathtt{I}, \mathtt{T}, \mathcal{X})$ with support set covering the whole object. We define a support set *selection* vector $\mathbf{u} \in \{0,1\}^b$, which determines the support set selected from a b-pixel template; used pixels marked by ones, unused pixels marked by zeros, respectively. The prediction error of a predictor operating on the support set selected by $\mathbf{u}$ is

$$e(\mathbf{u}) = \left\| \mathtt{T} - \mathtt{T}(\mathtt{I}(\mathbf{u},:))^+ \mathtt{I}(\mathbf{u},:) \right\|_F^2, \tag{19}$$

where $\mathtt{I}(\mathbf{u},:)$ is a submatrix of $\mathtt{I}$ with rows selected by $\mathbf{u}$. Given a desired complexity c, the optimal solution of the problem

$$\mathbf{u}^* = \underset{\substack{\mathbf{u} \in \{0,1\}^b, \\ \|\mathbf{u}\|_1 = c}}{\arg\min}\, e(\mathbf{u}) \tag{20}$$

is usually intractable because the problem has combinatorial complexity. Therefore, we propose the following greedy LS algorithm for the support set selection problem, which searches for a solution convenient for efficient tracking.

Algorithm 4.

1) Let $\mathbf{u} = \mathbf{0}$ is the selection and $\mathtt{L}, \mathtt{T}$ are given training examples.
2) Repeat c-times:

$$j^* = \underset{j=1\ldots b}{\text{argmin}} \{e(\mathbf{u} + \delta_j)\},$$

$$\mathbf{u}(j^*) = 1$$

where δ_j is b-vector of zeros with "1" at the position j.

Recently, an extension to LK tracker was published by Benhimane et al. [23], where the most convenient (with respect to gradient optimization method) subset of pixels is selected during a training stage. According to the published experimental data, such improvement decreases error rate of no more than 20 percent. While Benhimane et al. optimize only the subset of pixels and preserves the gradient-based tracking, we optimize both the set of pixels and the motion estimation method.

8 Tracking Objects with a Known Geometrical Model

If the object is represented only by a single SLLiP, the robustness to partial occlusions/noise and the dimensionality of predicted motions are limited. Therefore, we represent the object by Number of SLLiPs (NoSLLiP tracker). Such a representation requires a geometrical model of the object. Since the geometrical model estimation is beyond the scope of this work, we mainly work with planar or piecewise

planar objects, the accurate geometrical model of which could be manually estimated with negligible effort.

Besides the geometrical model estimation, many other questions must be answered: in particular, how many SLLiPs should be used, where it should be attached to the model, and how particular motion contributions should be combined. In our approach, we follow the most common way of robust motion estimation based on RANSAC. Since we do not make any assumptions about the object pose, we learn SLLiPs equally distributed on the object. During the tracking stage, a set of active SLLiPs, maximizing a trade-off between coverage and quality, is automatically selected and used for motion estimation. This method is introduced in Section 8.1. It is also not clear how many SLLiPs should be used and how many RANSAC iterations should be computed. Section 8.2 describes the method estimating the ratio between NoSLLiP and number of RANSAC iterations, which maximize a probability of successful tracking.

8.1 Online Selection of Active Predictor Set

Let us suppose that a set of SLLiPs evenly distributed on the object is available. In the following, we describe how to select a subset of SLLiPs, which assures both a reasonable coverage of the object and quality of SLLiPs. It is not possible to find the set of regions suitable for object tracking independently on the object position because, if the object changes its pose, some points can disappear and the global motion estimation can easily become ill-conditioned. In this section, we present an online method that automatically selects a subset of n predictors, called *active predictor set*, from all visible predictors.

To optimize the distribution of SLLiPs across the surface, we define *coverage measure* $r(Z)$ and *quality measure* $q(Z)$ of the set of SLLiP's reference points Z. Note that we have no theoretical justification for these definitions and we do not claim that this is the only right way to define it. We provide only one possible definition that might not be convenient for some applications.

Definition 9. Coverage measure *is*

$$r(Z) = \sum_{\mathbf{z}\in Z} d(\mathbf{z}, Z \setminus \mathbf{z}), \tag{21}$$

where distance between point $\mathbf{z}$ *and set* Z *is defined as the distance from the closest element of the set*

$$d(\mathbf{z}, Z) = \min_{\mathbf{y}\in Z} \|\mathbf{z} - \mathbf{y}\|. \tag{22}$$

Ideally, for optimal robustness to occlusion, the coverage measure would be maximized. In practice, particular SLLiPs differ by their complexities. Complexity corresponds to the suitability of SLLiP neighborhood for motion estimation. We have experimentally shown that the lower the complexity, the higher the robustness. Therefore, we derive the quality measure from complexity $c(\mathbf{z})$.

Definition 10. Quality measure *is*

$$q(\mathbf{z}) = \left| c(\mathbf{z}) - \max_{\mathbf{y}\in Z} c(\mathbf{y}) \right|. \tag{23}$$

To find a suitable subset Z of predictors from all visible predictors $\widetilde{Z}$, we seek to optimize the weighted sum of the coverage r and quality q:

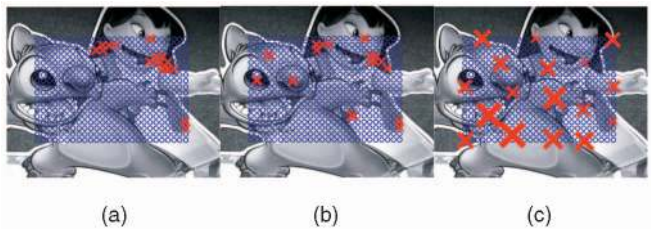

Fig. 10. Object coverage by predictors for different weightings. Blue circles correspond to all learned predictors and red crosses to the selected predictors. Size of crosses corresponds proportionally to the complexity. (a) $w = 0$. (b) $w = 0.1$. (c) $w = 1$.

$$f(Z) = w\frac{r(Z)}{r(\widetilde{Z})} + (1-w)\frac{q(Z)}{q(\widetilde{Z})}, \tag{24}$$

where $w \in [0; 1]$ is the coverage weight. Algorithm 5 selects a set of active SLLiPs, given predefined number of SLLiPs n.

Algorithm 5.

1) Let $\widetilde{Z}$ be the set of visible predictors and $Z = \emptyset$ a subset of selected reference points.
2) Select $\mathbf{z}^* = \arg\max_{\mathbf{z}\in\widetilde{Z}\setminus Z} f(\mathbf{z} \cup Z)$
3) $Z = \mathbf{z}^* \cup Z$ and $\widetilde{Z} = \widetilde{Z} \setminus \mathbf{z}^*$
4) if $|Z| = n$ end, else goto 2

Fig. 10 shows the results obtained for $w = \{0, 0.1, 0.5, 1\}$. If $w = 0$, n predictors with the highest quality are selected and SLLiPs are stacked in one corner. Conversely, $w = 1$ causes that SLLiPs are equally spread across the object.

8.2 Object Motion Estimation

Objects are modeled as a spatial constellation of optimal SLLiPs (henceforward just predictors), which estimates 2D translation. Object motion is estimated from these local translations by RANSAC algorithm. We understand tracking as a time-limited task, where the object pose needs to be estimated from a camera image before the next image comes. There is a trade-off between the time spent with the local motion estimation and the global motion estimation. While there are n local motions estimated by n predictors, the global motion is estimated by h iterations of RANSAC. The longer the time spent with each particular step, the higher the probability of successful tracking. We address the following question: Given the frame rate and the computational costs of different operations at a specific computer, how many predictors should be used and how many RANSAC iterations should be performed in order to maximize the probability of successful tracking?

The probability of a successful pose estimation in h-iterations of the RANSAC method is

$$P_R(k, h) = 1 - \left(1 - \left(\frac{k}{n}\right)^v\right)^h, \tag{25}$$

where n is the number of tracked points, k is the number of successfully tracked points, and v is the minimal number of points needed for the pose estimation. Note that $\frac{k}{n}$ is the percentage of the successfully tracked points (inliers). The number of successfully tracked points k is not known in advance, it is a random quantity with binomial distribution

$$P_k(k) = P_{\text{bin}}(n, k) = \binom{n}{k} p^k (1-p)^{n-k}, \tag{26}$$

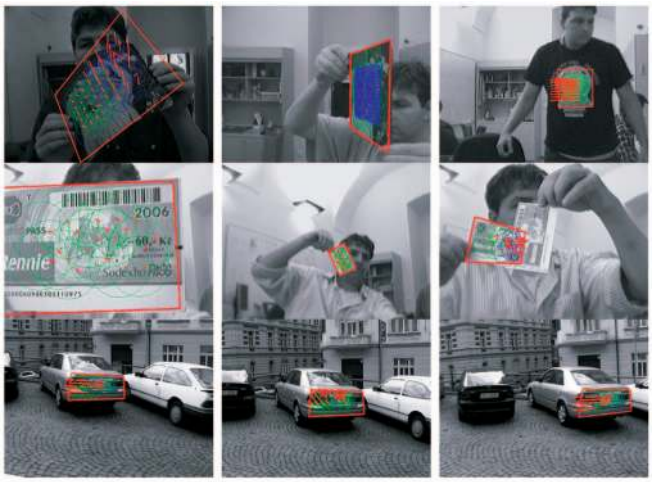

Fig. 11. **Robustness to partial occlusions and fast motion:** Green/blue circles outline inliers/outliers and red arrows show local motion estimated by SLLiPs. Support set is outlined by blue points.

where p is the probability of successful tracking of each particular reference point. Hence, the probability of successful tracking is

$$P_{\text{success}}(n,p,h) = \sum_{k=1}^{n} P_R(k,h)P_k(k) = \sum_{k=1}^{n} P_R(k,h)P_{\text{bin}}(n,k)$$
$$= \sum_{k=1}^{n}\left[1-\left(1-\left(\frac{k}{n}\right)^m\right)^h\right]\binom{n}{k}p^k(1-p)^{n-k}.$$

In the rest, we assume that p is a constant value that has been estimated, e.g., online as a mean number of inliers or measured on training data. $P_{\text{success}}(n,p,h)$ is therefore replaced by $\hat{P}_{\text{success}}(n,h)$. The case where p is not fixed is discussed later. Given

- the maximum time t we are allowed to spend in pose estimation,
- time t_0 of one RANSAC iteration, and
- times $t_1,\ldots,t_n$ required for local motion estimation or reference points $1,\ldots n$,

we formulate the following constrained optimization task:

$$(n^*,h^*) = \left\{\arg\max_{n,h}\hat{P}_{\text{success}}(n,h) \mid ht_0 + \sum_{i=1}^{n} t_i \le t\right\}. \quad (27)$$

Since, the probability $\hat{P}_{\text{success}}(n,h)$ is a monotonously increasing function in all variables, the maximum has to be located on the boundary $\{[n,h] \mid ht_0 + \sum_{i=1}^{n} t_i = t\}$ of the constrained set. Consequently, problem (27) can be rewritten as the unconstrained 1D problem as follows:

$$n^* = \arg\max_n \hat{P}_{\text{success}}\left(n, \frac{t-\sum_{i=1}^{n} t_i}{t_0}\right) = \arg\max_n \overline{P}_{\text{success}}(n). \quad (28)$$

We are not able to prove analytically the concavity of this function, but it is experimentally shown that $\overline{P}_{\text{success}}(n)$ is a concave function. If interested in a real-time application, Golden mean optimization is a natural choice. The probability evaluation is very simple and the computational time can be practically neglected.

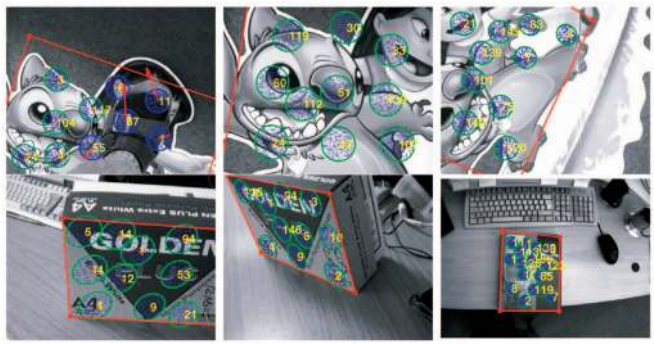

Fig. 12. **Tracking with a variable set of active predictors:** Yellow numbers denote IDs of particular SLLiPs. Blue points represent support set, green circles highlight inliers, and red arrows outline local motion estimated by SLLiPs.

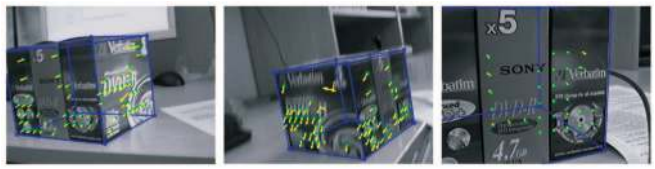

Fig. 13. **Three-dimensional tracking:** Variable set of active predictors and motion blur.

9 Experiments

Properties of SLLiP tracking and learning algorithms are experimentally verified. In Section 9.1, some preliminary results on challenging sequences are demonstrated. In Section 9.2, robustness and accuracy are evaluated on ground-truthed sequences. In particular, Section 9.2.1 describes ground-truthed data and Section 9.2.2 compares SLLiP to the state-of-the-art approaches. In Section 9.3, additional properties such as relation between robustness and speed relation between predefined and achieved accuracy are summarized.

9.1 Preliminary Qualitative Evaluation

In the first experiment, the NoSLLiP tracker is qualitatively evaluated on real sequences with planar and 3D rigid objects, which exhibit oblique views, motion blur, partial occlusions, and significant scale changes.[4] Tracking of various objects with partial occlusions and motion blur is shown in Fig. 11. Green/blue circles outline inliers/outliers, and red arrows show local motions estimated by SLLiPs. In some images also, the support set is outlined by blue points. Tracking of objects with variable set of active predictors is demonstrated in Figs. 12 and 13. The active set of visible SLLiPs is estimated by Algorithm 4. Yellow numbers denote IDs of particular SLLiPs. Although we mainly work with planar objects in order to avoid problems arising from inaccurate 3D reconstruction, SLLiPs are attachable to an arbitrary 3D model (see, for example, Fig. 13).

9.2 Quantitative Evaluation of Robustness and Accuracy

9.2.1 Ground-Truthed Data

The quantitative evaluation of robustness and accuracy of SLLiPs is conducted on sequences with three different

4. We encourage the reader to look also at video sequences available at http://cmp.felk.cvut.cz/demos/Tracking/linTrack.

Fig. 14. **Ground-truthed sequences.** 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. A blue rectangle delineates the object. Percentage values in corners are current corner speeds related to the current size of the object upper edge.

objects: MOUSEPAD (MP), TOWEL, and PHONE, where ground truth positions of the object corners in a total number of 11,963 frames were manually labeled[5] (see Fig. 14, for some examples). Accuracy is measured by average error in object corners. Error is expressed in percentage and normalized by the actual size of the object upper edge, in order to make the measure independent to the actual scale. Robustness is measured by the number of loss-of-locks, defined as the cases where the error was higher than 25 percent in at least one of the corners. In loss-of-lock frames, the tracker was reinitialized from the ground truth and the accuracy did not contribute to the total accuracy statistics.

9.2.2 Comparison of SLLiPs to the State-of-the-Art

Table 1 compares the NoSLLiP tracker to the state-of-the-art Lowe's SIFT detector [24] (method: SIFT),[6] Lucas-Kanade tracker [1] (method: LK tracker), and Jurie's LLiP tracker learned by the Least Squares method [6] (method: LLiP LS). All of these local motion estimators were combined with RANSAC to keep test conditions as similar as possible. SIFT mainly fails in frames with strong motion blur or in frames where the object was very far from the camera. LK tracker, which 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 and the tracking failed for fast motions.

According to the detailed speed comparison published in [2], six-parameter optimization by Inverse Compositional (IC) algorithm implemented in MATLAB runs approximately 10 times faster than the optimization by Forward Additive algorithm used in LK tracker. However, the basin of attraction and sensitivity to noise are the same. Since SLLiP tracker is also implemented in MATLAB, the achieved frame rates of LK, IC, and SLLiP are comparable. Concerning the computational complexity of LK, IC, and SLLiP: The computational complexity of one iteration computed on n-pixel template and p-vector of pose parameters by LK is $\mathcal{O}(p^2n + p^3)$ and by IC is $\mathcal{O}(pn + p^3)$. SLLiPs exploit only a small subset of pixels. Since we experimentally verified that approximately $\sqrt{n}$ pixels is used from n-pixel template, the computational complexity of one iteration of SLLiP (i.e., LLiP) is $\mathcal{O}(\sqrt{n}p)$.

Jurie's tracker is an 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 interframe motions.

TABLE 1
Comparison of Robustness and Accuracy of SLLiP, LK, LLiP Trackers (MATLAB Implementation), and SIFT Detector (C++ Implementation) on MP Sequence

Method	Object	Frame-rate [fps]	Loss-of-locks [-/-]	Error [%]
NoSLLiP	MP	18.9	13/6935	1.5
SIFT [24]	MP	0.5	281/6935	1.4
LK (IC) tracker [1]	MP	2.6 (25)	398/6935	2.4
LLiP LS [6]	MP	24.4	1083/6935	6.3
LLiP LS [6] 1/2	MP	24.2	93/6935	3.0
NoSLLiP	TOWEL	21.8	5/3229	2.1
NoSLLiP	PHONE	16.8	20/1799	1.8

The frame rate of the IC algorithm is estimated based on the comparison published in [2].

9.3 Additional Experiments

9.3.1 Robustness Analysis

We defined SLLiP as a sequence of LLiPs satisfying that the range of every predictor is at least as large as the uncertainty region of its predecessor, i.e., $\forall\, r_{i+1} \geq \lambda_i$, $i = 1 \ldots m-1$. We showed in Proposition 4 that the complexity minimization in the learning stage results in equality $\forall\, r_{i+1} = \lambda_i$, $i = 1 \ldots m-1$. As a result, whenever the testing data are corrupted by noise, the prediction might not be within the range of the following predictor, which might consequently cause a divergence of the SLLiP. For practical applications, a margin assuring robustness to the noise is required. We require $\forall\, r_{i+1} \geq \lambda_i(1+\gamma)$, $i = 1 \ldots m-1$ for a nonnegative number γ. We claim that the higher is the margin γ, the higher is the robustness against noise but simultaneously also the higher the complexity of the optimal SLLiPs.

Quantitative robustness evaluation is performed by computing the average number of loss-of-locks as a function of the margin (Fig. 15a), average complexity of SLLiPs as a function of the margin (Fig. 15b), and average frame rate as a function of the margin (Fig. 15c). A test sequence based on the ground-truthed sequence was intentionally made more challenging. The experiment is conducted on a selected mouse pad subsequence (frames 3,500-6,000), where only each second frame is processed in order to increase interframe motion and, consequently, to achieve a statistically important number of loss-of-locks. The sequence is processed with SLLiPs learned for six

5. These ground-truthed sequences are available at ftp://cmp.felk.cvut.cz/pub/cmp/data/lintrack.

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

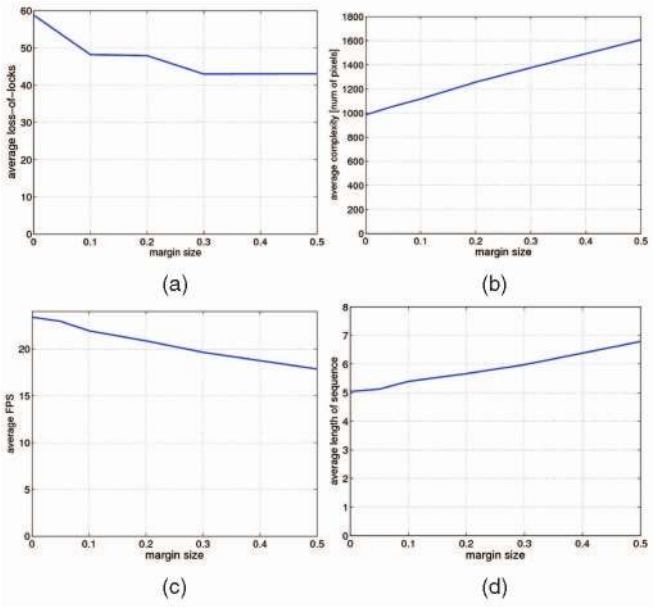

Fig. 15. Robustness analysis: The higher the margin, the higher the robustness to noise but also the higher the complexity of SLLiPs. (a) Loss-of-locks. (b) Complexity of SLLiPs. (c) Frame-rate. (d) Number of LLiPs in SLLiPs.

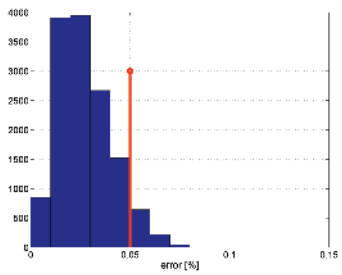

Fig. 16. Comparison of desired and real accuracies.

different margins [0, 0.05, 0.1, 0.2, 0.3, 0.5]. The number of loss-of-locks (Fig. 15a) and the frame rate (Fig. 15c) are evaluated as an average over 20 processing of the sequence with different starting frames. Complexity (Fig. 15b) and length of LLiP sequence (Fig. 15d) are computed as an average over a set of 48 learned SLLiPs.

9.3.2 Accuracy Analysis of Minimax Learning

In this experiment, we compare uncertainty region λ_0 required in learning and real distribution of SLLiP errors. We learned 48 SLLiPs covering the mouse pad with desired accuracy of 5 percent of the range size. The accuracy is evaluated on those frames, where interframe motion is smaller than the learning range of SLLiPs. Fig. 16 shows the histogram of displacement errors with λ_0 denoted by the red line at 0.05. In approximately 10 percent of the cases, the errors are higher, which is presumably caused mainly by the limited ground truth accuracy and partly by the image noise. Note that the optimal SLLiP is guaranteed to converge into λ_0 for all training examples.

9.3.3 Support Set Selection by Greedy LS Algorithm Evaluation

We compare the mean square error (MSE) achieved by the greedy LS support set selection algorithm (Algorithm 4) and the MSE achievable by a random support set selection. Fig. 17a shows the MSE histogram of predictors operating on a randomly selected support set with a size of 20 pixels. The 99 percent left quantile of the histogram is depicted by an empty green circle. It shows that usage of a randomized sampling instead of Algorithm 4 would require a prohibitively high number of iterations to achieve at least comparable results with the proposed greedy LS algorithm.

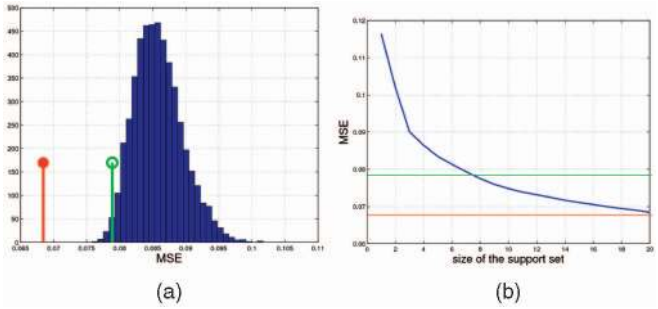

Fig. 17. Histogram of MSEs of predictors with the randomly constructed support sets. (a) $C = 20$, $E_{99} = 0.0789$ (green empty circle denotes 99 percent quantile), $\hat{E} = 0.0684$ (red filled circle denotes MSE of the proposed method). (b) MSE as a function of complexity during the incremental construction of the support set.

Fig. 17b shows MSE as a function of complexity during the incremental construction of the support set. It demonstrates that the 99 percent left quantile of randomly selected support sets is achieved with less than one-half of the support set size. The mean is achievable with less than one-quarter of the support set size.

10 Discussion

10.1 Tracking for Detection

Due to the very high performance of the proposed predictor, there is a possibility coupling it with a detector. Detection performed in a pose grid [3], [4], [5] could be replaced by prediction followed by detection performed in a sparser pose grid. The efficiency of the method depends on the ratio of detectability and predictability radii and times needed for the detection and prediction, respectively.

10.2 Tracking in Feature Space

Instead of the intensities, an arbitrary set of features can be used. There is a large set of linear features, i.e., the features computed as a linear combination of the observed intensities. Since the linear prediction from linear features would be only a linear combination of linear combinations that is again the linear combination, there is almost no reason of using the linear features. In other words, if any linear combination of the intensities is necessary, it is automatically included in the regressor coefficients during the learning stage in an optimal manner. Particular counterexamples are Haar features [25], which allow for faster direct computation from the integral image.

10.3 Nonlinear Regression

If the linear mapping is insufficient, the method allows a natural extension to an arbitrary class of mappings formed as a linear combination of kernel functions by *data lifting*. For example, in the polynomial mapping, particular monomials are considered as further observed intensities.

It allows the learning procedure to deal with higher dimensional linear mappings instead of the nonlinear ones. Since the prediction by a nonlinear predictor is, in general, computationally more complex than by a linear predictor and we have experimentally shown that no substantial improvement is achievable with the nonlinear one, we use only the linear predictor.

10.4 Confidence Measure

We do not propose any confidence measure for tracking with a single sequential predictor. In general, every standard confidence measure can be used, e.g., SSD or a learned classifier. If an object is modeled by a set of predictors, RANSAC determines the number of outliers as a side product of the pose estimation. The number of outliers provides a measure of confidence of the estimated pose. Since we did not investigate this issue in detail, the tracker failure is reported if 50 percent of outliers is reached.

11 Conclusions

We have proposed a learning approach to tracking that explicitly minimizes computational complexity of the tracking process subject to user-defined probability of failure (loss-of-lock) and precision. In our approach, the object is modeled by a set of local motion predictors estimating translations. Object motion is estimated from these translations by RANSAC. Local motion predictors, their locations, and number as well as the number of RANSAC iterations are subject of the optimization. Since the tracker is formed by NoSLLiP, we refer to it as the NoSLLiP tracker.

In experiments, the NoSLLiP tracker was tested on approximately 12,000 frames with a labeled ground truth, showing that the NoSLLiP tracker achieves a significantly smaller number of loss-of-locks than the SIFT detector, the LK tracker, or Jurie's tracker. Since all of the time-consuming computations are performed in the offline stage, the NoSLLiP tracking requires only a few hundred multiplications, yielding extremely efficient motion estimation. Note that a nonoptimized C++ implementation of an average sequential predictor takes only 30 μs.

We encourage the reader to download a MATLAB implementation of the proposed methods of learning and tracking and additional material from http://cmp.felk.cvut.cz/demos/Tracking/linTrack/.

Acknowledgments

Karel Zimmermann was supported by the Czech Academy of Sciences under Project 1ET101210407. Tomáš Svoboda was supported by the Czech Ministry of Education under Project 1M0567 and in part by EC Project FP6-IST-027787 DIRAC. Jiří Matas was supported by the Grant Agency of Czech Republic under Project 102/07/1317. This work was done while Karel Zimmermann was with the Czech Technical University.

References

[1] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Int'l J. Computer Vision and Artificial Intelligence,* pp. 674-679, 1981.

[2] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework," *Int'l J. Computer Vision,* vol. 56, no. 3, pp. 221-255, 2004.

[3] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua, "Feature Harvesting for Tracking-by-Detection," *Proc. Ninth European Conf. Computer Vision,* pp. 592-605, 2006.

[4] S. Avidan, "Support Vector Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 8, pp. 1064-1072, Aug. 2004.

[5] H. Grabner and H. Bischof, "On-Line Boosting and Vision," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 260-267, 2006.

[6] F. Jurie and M. Dhome, "Real Time Robust Template Matching," *Proc. British Machine Vision Conf.,* pp. 123-131, 2002.

[7] T. Cootes, G. Edwards, and C. Taylor, "Active Appearance Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 6, pp. 681-685, June 2001.

[8] O. Williams, A. Blake, and R. Cipolla, "Sparse Bayesian Learning for Efficient Visual Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 8, pp. 1292-1304, Aug. 2005.

[9] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded Up Robust Features," *Proc. Ninth European Conf. Computer Vision,* May 2006.

[10] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking Using Online and Offline Information," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 10, pp. 1385-1391, Oct. 2004.

[11] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 593-600, 1994.

[12] T. Cootes, G. Edwards, and C. Taylor, "Active Appearance Models," *Proc. Fifth European Conf. Computer Vision,* vol. 2, pp. 484-498, 1998.

[13] D. Cristinacce and T. Cootes, "Feature Detection and Tracking with Constrained Local Models," *Proc. 17th British Machine Vision Conf.,* pp. 929-938, 2006.

[14] M.E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Machine Learning Research,* vol. 1, pp. 211-244, 2001.

[15] A. Agarwal and B. Triggs, "Recovering 3D Human Pose from Monocular Images," *IEEE Trans. Pattern Analysis Machine Intelligence,* vol. 28, no. 1, pp. 44-58, Jan. 2006.

[16] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support Vector Regression Machines," *Advances in Neural Information Processing Systems,* vol. 9, pp. 156-161, 1996.

[17] V.N. Vapnik, *The Nature of Statistical Learning Theory (Information Science and Statistics).* Springer-Verlag, Nov. 1999.

[18] A. Smola and B. Schölkopf, "A Tutorial on Support Vector Regression," technical report, ESPRIT Working Group in Neural and Computational Learning II, 1998.

[19] S.K. Zhou, B. Georgescu, X.S. Zhou, and D. Comaniciu, "Image Based Regression Using Boosting Method," *Proc. 10th IEEE Int'l Conf. Computer Vision,* vol. 1, pp. 541-548, 2005.

[20] A. Bissacco, M. Yang, and S. Soatto, "Fast Human Pose Estimation Using Appearance and Motion via Multi-Dimensional Boosting Regression," *Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

[21] K. Zimmermann, "Fast Learnable Methods for Object Tracking," PhD dissertation, Czech Technical Univ., 2008.

[22] B. Bollobás, *Modern Graph Theory.* Springer, 1998.

[23] S. Benhimane, A. Ladikos, V. Lepetit, and N. Navab, "Linear and Quadratic Subsets for Template-Based Tracking," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[24] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[25] A. Haar, "Zur Theorie der Orthogonalen Funktionensysteme," *Annals of Math.,* vol. 69, pp. 331-371, 1910.

Karel Zimmermann received the PhD degree in cybernetics from the Czech Technical University, Prague, Czech Republic, in 2008. He was also with the Technological Education Institute of Crete (2001), with the Technical University of Delft (2002), and with the University of Surrey (2006). Since 2008 he has been a postdoctoral researcher at the Katholieke Universiteit Leuven. He serves as a reviewer for major conferences such as CVPR and ICCV. His current research interests include learnable methods for tracking, detection and recognition.

Jiří Matas received the MSc degree (with honors) in cybernetics from the Czech Technical University, Prague, Czech Republic, in 1987 and the PhD degree from the University of Surrey, Guildford, United Kingdom, in 1995. From 1991 to 1997, he was a research fellow in the Centre for Vision, Speech, and Signal Processing at the University of Surrey. In 1997, he joined the Center for Machine Perception at the Czech Technical University. Since 1997, he has held various positions at these two institutions. He has published more than 100 papers in refereed journals and conference proceedings. His publications have more than 800 citations in the Science Citation Index. He received the best paper prize at the British Machine Vision Conferences in 2002 and 2005 and at the Asian Conference on Computer Vision in 2007. He has served in various roles at major international conferences (e.g., IEEE International Conference on Computer Vision (ICCV), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), International Conference on Pattern Recognition (ICPR), and Neural Information Processing Symposium (NIPS)), cochairing the European Conference on Computer Vision (ECCV) 2004 and CVPR 2007. He is an associate editor-in-chief of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and a member of the editorial board of the *International Journal of Computer Vision*. He is a member of the IEEE.

Tomáš Svoboda received the PhD degree in artificial intelligence and biocybernetics from the Czech Technical University, Prague, Czech Republic, in 2000. He spent three years as a postdoctoral fellow in the Computer Vision Group at the Swiss Federal Institute of Technology (ETH), Zurich. He is currently a senior research fellow at the Czech Technical University. He has published on multicamera systems, omnidirectional cameras, image-based retrieval, and learnable tracking methods. His current research interests include multicamera systems for scene understanding and telepresence, tracking, and video analysis. He is regular member of the program committee of most important computer vision conferences and also serves as a reviewer for major journals like the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* and the *International Journal of Computer Vision (IJCV)*. His publications have more than 100 citations in the Science Citation Index. He is a member of the IEEE.