FAST LEARNABLE OBJECT TRACKING AND DETECTION IN HIGH-RESOLUTION OMNIDIRECTIONAL IMAGES

David Hurych, Karel Zimmermann and Tomáš Svoboda
Center for Machine Perception, Department of Cybernetics
Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic

Keywords: Detection, Tracking, Incremental, Learning, Predictors, Fern, Omnidirectional, High-resolution.

Abstract: This paper addresses object detection and tracking in high-resolution omnidirectional images. The foreseen application is a visual subsystem of a rescue robot equipped with an omnidirectional camera, which demands real time efficiency and robustness against changing viewpoint. Object detectors typically do not guarantee specific frame rate. The detection time may vastly depend on a scene complexity and image resolution. The adapted tracker can often help to overcome the situation, where the appearance of the object is far from the training set. On the other hand, once a tracker is lost, it almost never finds the object again. We propose a combined solution where a very efficient tracker (based on sequential linear predictors) incrementally accommodates varying appearance and speeds up the whole process. We experimentally show that the performance of the combined algorithm, measured by a ratio between false positives and false negatives, outperforms both individual algorithms. The tracker allows to run the expensive detector only sparsely enabling the combined solution to run in real-time on 12 MPx images from a high resolution omnidirectional camera (Ladybug3).

1 INTRODUCTION

This paper focuses on the problem of real-time object detection and tracking in a sequence of high-resolution omnidirectional images. The idea of combining a detector and fast alignment by a tracker has already been used in several approaches (Li et al., 2010; Hinterstoisser et al., 2008). The frame rate of commonly used detectors naturally depends on both the scene complexity and image resolution. For example, the speed of ferns (Özuysal et al., 2010), SURF (Bay et al., 2006) and SIFT (Lowe, 2004) detectors depends on the number of evaluated features, which is generally proportional to the scene complexity (e.g. number of Harris corners) and image resolution. The speed of Waldboost (Šochman and Matas, 2005) (or any cascade detector) depends on the number of computations performed in each evaluated subwindow. In contrast, most of the trackers are independent of both the scene complexity and image resolution. This guarantees stable frame rate however, once the tracker is lost it may never recover the object position again. Adaptive trackers can follow an object which is far from the training set and cannot be detected by the detector. We propose to combine a detector and a tracker to benefit from robustness (ability to find an object) of detectors and locality (efficiency) of trackers.

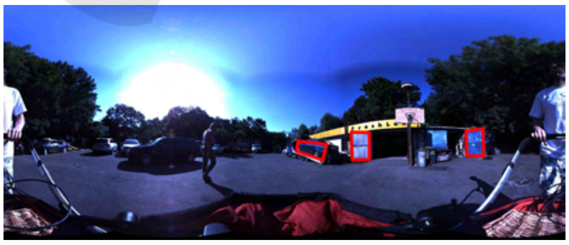
Figure 1: Omnidirectional high resolution image (12 Mpx) captured by Ladybug 3 camera. Three objects are marked.

Ferns-based detector (also used by (Hinterstoisser et al., 2008) for 10 fps tracking-by-detection) is one of the fastest object detectors because of the low number of evaluated binary features on detected Harris corners. The speed makes the ferns detector ideal for the purpose of object detection in large images.

One of the most popular template trackers is the KLT tracker (Shi and Tomasi, 1994), which uses the Lucas-Kanade *gradient descent* algorithm (Lucas and Kanade, 1981). The algorithm has become very popular and has many derivations (Baker and Matthews, 2004). The gradient descent is a fast algorithm yet, it has to compute the image gradient, the Jacobian and inverse Hessian of the modeled warp in every frame. For some simple warps, the Jacobian may

Hurych D., Zimmermann K. and Svoboda T..
FAST LEARNABLE OBJECT TRACKING AND DETECTION IN HIGH-RESOLUTION OMNIDIRECTIONAL IMAGES.
DOI: 10.5220/0003369705210530
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2011), pages 521-530
ISBN: 978-989-8425-47-8
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

be precomputed (Hager and Belhumeur, 1998), (Dellaert and Collins, 1999). One may also get a rid of the inverse Hessian computation by switching the roles of the template and image (Baker and Matthews, 2004). Nevertheless we always need to compute the image gradients and in general case also the Jacobian and inverse Hessian of the warp. An alternative for template tracking are *regression-based* methods (Jurie and Dhome, 2002), (Zimmermann et al., 2009a). They avoid the computation of image gradient, Jacobian and inverse Hessian by learning a regression matrix from training examples. Once learned they estimate the tracking parameters directly from the image intensities. If the regression function is linear, it is called *linear predictor*. The training phase is the biggest disadvantage of linear predictors, because the tracking cannot start immediately. Nevertheless, the regression matrix (function) may be estimated only from one image in a short time (few seconds). The training examples are generated by random warpings of the object template and collecting image intensities. This regression matrix may be updated by additional training examples during tracking (Hinterstoisser et al., 2008).

Recently, it has been shown (Özuysal et al., 2010), (Hinterstoisser et al., 2008), that taking advantage of the learning phase, greatly improves the tracking speed and makes the tracker more robust with respect to large perspective deformations. A learned tracker is able to run with fragment of processing power and estimates object position in complicated or not yet seen poses. However, once the tracker gets lost it may not recover the object position.

To fulfill the real-time requirements, we propose a combination of a robust detector and a very efficient tracker. Both, the detector and the tracker, are trained from image data. The tracker gets updated during the tracking. The tracker performance is extremely fast and as a result of that, faster than real-time tracking allows for multiple object tracking.

1.1 Related Work

We use a similar approach to (Hinterstoisser et al., 2008), who also use a fern object detector and a linear predictor with incremental learning for homography estimation. The detector is used for object localization and also for a rough estimation of patch transformation. The initial transformation is further refined by the linear predictor, which predicts full 2D homography. The precision of the method is validated by inverse warping of the object patch and correlation-based verification with the initial patch. The detector is run in every frame of the sequence of 0.3 Mpx images processing 10 frames per second (fps). This approach however, would not be able to perform in real-time on 12 Mpx images. We use the fern detector to determine tentative correspondences and we run RANSAC on detected points to estimate the affine transformation. After a positive detection we apply the learned predictor in order to track the object for as many frames as possible. (Hinterstoisser et al., 2008) use an iterative version of linear predictor similar to the one proposed by (Jurie and Dhome, 2002), while we use SLLiP version. The SLLiP proved (Zimmermann et al., 2009a) to be faster than the iterative version, while keeping the high precision of the estimation. Our tracker is incrementally updated during tracking (Hinterstoisser et al., 2008; Li et al., 2010). We validate the tracking by the updated tracker itself (see Section 2.2), which is more precise, than correlation-based verification by a single template in case of varying object appearance.

Recently (Holzer et al., 2010) used adaptive linear predictors for real-time tracking. Adaptation is done by growth or reduction of the tracked patch during tracking and update of the regression matrices. However, this approach is not suitable for our task, because of the need to keep in memory the large matrix with training examples, which is needed for computation of the template reduction and growth. This training matrix grows with additional training examples collected for on-line learning, which is undesirable for long-term tracking.

(Li et al., 2010) use linear predictors in the form of locally weighted projection regressors (LWPR) as a part of self-tuning particle filtering (ISPF) framework. They approximate a non-linear regression by a piece-wise linear models. In comparison we use a sequence of learnable linear predictors (SLLiP) similar to (Zimmermann et al., 2009b), which uses the result of previous predictors in sequence as the starting point for another predictor in a row. In (Li et al., 2010) the partial least-squares is used for data dimension reduction. We use a subset of template pixels spread over the object in regular grid, which proved to be sufficient for dimensionality reduction, while keeping the high precision and robustness of tracking.

The rest of this paper is organized as follows. In Section 2 you find the formal descriptions of used ferns detector and sequential predictor tracker and in Section 2.3 the outline of our algorithm. In Section 3 we present the general evaluation of our algorithm. A detailed evaluation of the detector and tracker are given in Sections 3.1 and 3.2. In the last two sections we discuss the computational times of the algorithm and conclude the paper.

2 THEORY

The method combines a fern-based detector and a tracker based on sequential linear predictors. Both the detector and the tracker are trained from the image data. The tracker has its own validation and is incrementally re-learned as the tracking goes. The detector locates the object in case the tracker gets lost.

2.1 Ferns-based Detector

Object is modeled as a spatial constellation of detected harris corners on one representative image. In a nutshell: the fern detector first estimates similarity between harris corners detected in the current frame and harris corners on the model. The thresholded similarity determines tentative correspondences, which are further refined by RANSAC selecting the largest geometrically consistent subset (i.e. set of inliers). In our approach object was modeled as a plane. Since we observed that the estimation of full homography transformation was often ill-conditioned, because of both insufficient number of detected corners and non-planarity of the object, the RANSAC searches for the affine transformation, which showed to be more robust.

Detailed description of the similarity measure is in (Özuysal et al., 2010). In the following, we provide just short description for the sake of completeness. The similarity measures probability $p(\mathbf{V}(\mathbf{v}),\mathbf{w})$ that the observed appearance of the neighbourhood $\mathbf{V}(\mathbf{v})$ of the detected corner $\mathbf{v}$ corresponds to the model corner $\mathbf{w}$. The appearance is represented as a sequence of randomly selected binary tests, i.e. given the corner $\mathbf{v}$ and sequence of n point pairs $\{(\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2),\dots(\mathbf{x}_n,\mathbf{y}_n)\}$, the appearance of the $\mathbf{v}$ is encoded as binary code $V_k(\mathbf{v}) = I(\mathbf{v}+\mathbf{x}_k) > I(\mathbf{v}+\mathbf{y}_k)$, where $I(\mathbf{v}+\mathbf{x}_k)$ is the image intensity.

On one hand, it is insuficient to model probabilities of binary tests independently, i.e. assuming that $p(\mathbf{V}(\mathbf{v}),\mathbf{w}) = \prod_{k=1}^{n} p_k(V_k(\mathbf{v}),\mathbf{w})$. On the other hand, modeling $p(\mathbf{V}(\mathbf{v}),\mathbf{w}) = p(V_1(\mathbf{v}),\dots,V_n(\mathbf{v}),\mathbf{w})$ is ill-conditioned, since we would have to estimate probability in 2^n bins, where n is usually equal to several hundreds. Therefore, we divide the sequence of n binary tests into $N = n/m$ subsequences with length $m \approx 8-11$. Subsequences are selected by N membership functions $I(1)\dots I(N)$ and we denote $h_k = \mathrm{card}(I_k)$, $k = 1\dots N$. Finally, we consider these subsequences to be statistically independent and model the probability as:

$$p(\mathbf{V}(\mathbf{v}),\mathbf{w}) = \prod_{k=1}^{N} p_k(V_{I_k(1)}(\mathbf{v}),\dots,V_{I_k(h_k)}(\mathbf{v}),\mathbf{w}) \quad (1)$$

The proposed detector requires an off-line training phase, within which the subsequent probabilities are estimated. Once the probabilities are pre-computed, we use them on-line to determine the tentative correspondences. In the following both phases are detailed.

Offline Training Phase. First n binary tests are randomly selected and divided into N subsequences. The model is estimated from one sample image, where Harris corners are detected within delineated object border. Appearance of each corner's neighbourhood is modeled by N h_k-dimensional binary hypercubes, with 2^{h_k} bins, representing joint probability $p_k(V_{I_k(1)}(\mathbf{v}),\dots,V_{I_k(h_k)}(\mathbf{v}),\mathbf{w})$. To estimate values of the probability, each corners neighbourhood is L-times perturbated within the range of local deformations we want to cope with. For each perturbed training sample and each subsequence, binary tests are evaluated and correspoding bin is increased by $1/L$. Note that different Harris corners are modeled via different probabilities but the same binary tests, which allows significant improvement in the online running phase, since the computational complexity of the similarity computation is almost independent of the number of Harris corners on the model.

Online Running Phase. Given an input image, Harris corners are detected. For each corner $\mathbf{v}$, binary tests are evaluated and similarity to each model corner is computed via Equation 1. Similarities higher than a chosen threshold determine tentative correspondeces. Eventually, RANSAC estimates affine transformation between model and the given image. Confidence of the detection is equal to the number of inliers.

2.2 Sequential Linear Predictors

We extend the anytime learning of the Sequential Learnable Linear Predictors (SLLiP) by (Zimmermann et al., 2009b) in order to predict not only translation but also the full homography transformation. Next extension is the incremental learning of new object appearances also used by (Hinterstoisser et al., 2008). The predictor essentially estimates deformation parameters directly from image intensities. It requires a short offline learning stage before the tracking starts. The learning stage consists of generating exemplars and estimation of regression functions. We use a simple cascade of 2 SLLiPs - first for 2D motion estimation (2 parameters) and second for homography estimation (8 parameters). The homography is parameterized by position of 4 patch corners. Knowing the corners position and having the reference coordinates, we compute the homography transformation for the whole patch. We have experimentally verified, that this 2-SLLiP configuration is more stable than using just one SLLiP to predict homography. First the translation is roughly estimated by first SLLiP and than a precise homography refinement is done. Because of speed, we opted for least squares learning of SLLiPs similarly, as suggested by (Zimmermann et al., 2009b).

Lets denote the translation parameters vector $\mathbf{t}^t = [\Delta x, \Delta y]^T$, estimated by the first SLLiP, and the homography parameters vector $\mathbf{t}^a = [\Delta x_1, \Delta y_1, \ldots, \Delta x_4, \Delta y_4]^T$, estimated by the second SLLiP which represents the motion of 4 object corners $\mathbf{c}_i = [x_i, y_i]^T, i = 1, \ldots, 4$. The object point $\mathbf{x} = [x, y]^T$ from previous image is transformed to corresponding point $\mathbf{x}'$ in current image accordingly

$$\mathbf{p} = \mathtt{A}\left(\begin{bmatrix}\mathbf{x}\\1\end{bmatrix} + \begin{bmatrix}\mathbf{t}^t\\0\end{bmatrix}\right) \quad (2)$$

$$\mathbf{x}' = [p_x/p_z, p_y/p_z]^T, \quad (3)$$

where $\mathbf{p}$ are homogeneous coordinates. The homography matrix $\mathtt{A}$ is computed from 4-point correspondences, between shifted object corners $\mathbf{c}_i + \mathbf{t}^t$ from previous image and current corners positions $\mathbf{c}_i + \mathbf{t}^t + \left[\mathbf{t}^a_{2i-1}, \mathbf{t}^a_{2i}\right]^T, i = 1, \ldots, 4$ estimated by the 2-SLLiP tracker.

Estimation of parameters vectors $\mathbf{t}^t$ and $\mathbf{t}^a$, *learning* and *incremental learning* will be explained for a single SLLiP with general parameters vector $\mathbf{t}$. Equations are valid for both SLLiPs, which we use. SLLiP is simply a sequence of linear predictors. Predictors in this sequence estimate the parameters successively (4), thus each improving the result of previous predictor estimation and lowering the error of estimation. SLLiP tracks according to

$$\begin{aligned}\mathbf{t}_1 &= \mathtt{H}_1 I(X_1) \quad (4)\\ \mathbf{t}_2 &= \mathtt{H}_2 I(\mathbf{t}_1 \circ X_2)\\ \mathbf{t}_3 &= \mathtt{H}_3 I(\mathbf{t}_2 \circ X_3)\\ &\vdots\\ \mathbf{t} &= \bigcirc_{(i=1,\ldots,k)} \mathbf{t}_i,\end{aligned}$$

where I is current image and X is a set of 2D coordinates (called *support set*) spread over the object position from previous image. $I(X)$ is a vector of image intensities collected at image coordinates X. Operation $\circ$ stands for transformation of support set points using (2) and (3), i.e. aligning the support set to fit the object using parameters estimated by the previous predictor in the sequence. Final result of the prediction is a vector $\mathbf{t}$ which combines results of all predictions in the sequence. The model θ_s for SLLiP is formed by the sequence of predictors $\theta_s = |\{\mathtt{H}_1, X_1\}, \{\mathtt{H}_2, X_2\}, \ldots, \{\mathtt{H}_k, X_k\}|$. Matrices $\mathtt{H}_1, \mathtt{H}_2, \ldots, \mathtt{H}_k$ are linear regression matrices which are learned from training data.

In our algorithm, the 2 SLLiPs are learned from one image only and they are incrementally learned during tracking. A few thousands of training examples are artificially generated from the training image using random perturbations of parameters in vector $\mathbf{t}$, warping the support set accordingly and collecting the image intensities. The column vectors of collected image intensities $I(X)$ are stored in matrix $\mathtt{D}_i$ and perturbed parameters in matrix $\mathtt{T}_i$ columnwise. Each regression matrix in SLLiP is trained using the least squares method $\mathtt{H}_i = \mathtt{T}_i\mathtt{D}_i^T\left(\mathtt{D}_i\mathtt{D}_i^T\right)^{-1}$.

Incremental learning corresponds to an on-line update of regression matrices $\mathtt{H}_i, i = 1, \ldots, k$. An efficient way of updating regression matrices was proposed by (Hinterstoisser et al., 2008). Each regression matrix $\mathtt{H}_i$ can be decomposed as follows

$$\mathtt{H}_i = \mathtt{Y}_i\mathtt{Z}_i, \quad (5)$$

where $\mathtt{Y}_i = \mathtt{T}_i\mathtt{D}_i^T$ and $\mathtt{Z}_i = \left(\mathtt{D}_i\mathtt{D}_i^T\right)^{-1}$. New training example $\mathbf{d} = I(X)$ with parameters $\mathbf{t}$ is incorporated into the predictor as follows

$$\mathtt{Y}_i^{j+1} = \mathtt{Y}_i^j + \mathbf{t}\mathbf{d}^T \quad (6)$$

$$\mathtt{Z}_i^{j+1} = \mathtt{Z}_i^j - \frac{\mathtt{Z}_i^j\mathbf{d}\mathbf{d}^T\mathtt{Z}_i^j}{1 + \mathbf{d}^T\mathtt{Z}_i^j\mathbf{d}}, \quad (7)$$

where the upper index j stands for the number of training examples. After updating matrices $\mathtt{Y}_i$ and $\mathtt{Z}_i$ we update the regression matrices $\mathtt{H}_i$ using (5). For more details about incremental learning see (Hinterstoisser et al., 2008).

The tracking procedure needs to be validated in order to detect the loss-of-track. When the loss-of-track occurs, the object detector is started instead of tracking. To *validate* the tracking we use the first SLLiP, which estimates 2D motion of the object. We utilize the fact that the predictor is trained to point to the center of learned object when initialized in a close neighborhood. On the contrary, when initialized on the background, the estimation of 2D motion is expected to be random. We initialize the predictor several times on a regular grid (validation grid - depicted by red crosses in Fig. 2) in the close neighborhood of current position of the tracker. The close neighborhood is defined as 2D motion range (of the same size as the maximal parameters perturbation used for learning), for which the predictor was trained. In our case the range is $\pm(patch\_width/4)$ and $\pm(patch\_height/4)$. We let the SLLiP vote for the object center from each position of the validation grid and observe the 2D vectors, which should point to the center of the object, in the case, when the tracker is well aligned on the object. When all (or sufficient number of) the vectors point to the same pixel, which is also the current tracker position, we consider the tracker to be on its track. Otherwise, when the vectors point to some random directions, we say that the track is lost, see Fig. 2. The same approach for tracking validation was suggested in (Hurych and Svoboda, 2010). The next section describes in detail the algorithm used in our system, which combines the ferns detector and 2-SLLiP tracker.

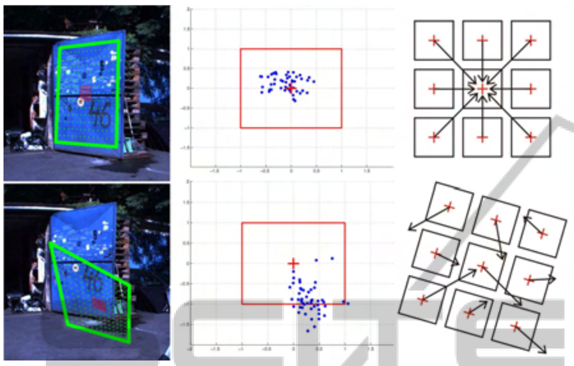
Figure 2: Validation procedure demonstrated in two situations. The first row shows successful validation of tracked *blue_door*, the second row shows loss of track caused by a bad tracker initialization. First column shows the tracker position marked by green. The third column depicts the idea of validation - i.e. a few initializations of the tracker (marked by red crosses) around its current position and the collection of votes for object center. When the votes point to one pixel, which is also the current tracker position (or close enough to the center), the tracker is considered to be well aligned on the object. When the votes for center are random and far from current position the loss-of-track is detected. In the second column we see the collected votes (blue dots), the object center (red cross) and the motion range (red rectangle) normalized to $< -1, 1 >$, for which was the SLLiP trained.

2.3 The Algorithm

Our algorithm combines the ferns detector and 2-SLLiP tracker together. In order to achieve real-time performance, we need to run the detector only when absolutely necessary. The detection runs when the object is not present in the image or the tracker loses its track. As soon as the object is detected, the algorithm starts tracking and follows the target as long as possible. Since tracking requires only fragment of computational power, computational time is spared for other tasks. The on-line incremental update of the regressors helps to keep longer tracks. When the validator decides that the track is lost, the detector is started again until next positive detection is achieved. To lower the number of false detections to minimum, we run the validation after each positive response of the detector. The pseudo-code shown in algorithm 1 should clarify the whole process.

Algorithm 1: Detection and Tracking.

```
Select object
$model\_fern \leftarrow$ learn fern detector
$model\_tracker \leftarrow$ learn 2 – SLLiP tracker
$lost \leftarrow$ true
$i \leftarrow 0$
while next image is available do
    get next image
    $i \leftarrow i + 1$
    if $lost$ then
        $detected \leftarrow$ detect object
        if $detected$ then
            initialize tracker
            estimate homography
            $valid \leftarrow$ validate position
            if $valid$ then
                $lost \leftarrow$ false
                continue
            end if
        end if
    else
        track object
        if $i$ mod $5 == 0$ then
            $valid \leftarrow$ validate position
            if $valid$ then
                $model\_tracker \leftarrow$ update tracker
            else
                $lost \leftarrow true$
                continue
            end if
        end if
    end if
end while
```

3 EXPERIMENTAL RESULTS

The foreseen scenario for the use of our method is a visual part of mobile rescue robot navigation system. The operator selects one or more objects in the scene and the robot (carying a digital camera) should navigate itself through some space, by avoiding tracked obstacles to localized object of interest. The experiments simulate the foreseen use. Several objects were selected in one frame of particular sequence and from this starting frame they were tracked and detected.

Three types of experiments were performed. First we run the ferns detector itself in every frame without tracking. Second we run the 2-SLLiP tracker with validation without the recovery by detector. And finally, we run the combination of both. In all experiments were both the detector and the tracker trained from a

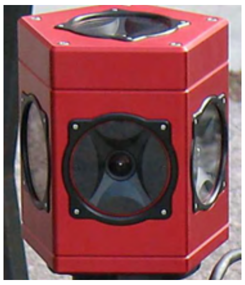
Figure 3: Ladybug 3 with 5 cameras placed horizontaly in circle and one camera looking upwards for capturing omnidirectional images.

single image. The detector and the tracker perform best on planar objects, because of the modeled 2D homography transformation. We tested our algorithm also on non-planar objects (*lying human*, *crashed car*) to see the performance limits and robustnes of our solution, see Section 3.3. Algorithm was tested on 8 objects in 4 videosequences. The ladybug camera provides 8 fps of panoramic images captured from 6 cameras simultaneously. Five cameras are set horizontaly in a circle and the sixth camera looks upwards, see Fig. 3. The panoramic image is a composition of these 6 images and has resolution of 5400×2248 pixels (12 Mpx). Fig. 1 and Fig. 4 show examples of the composed scenes and tested objects. Appearance changes for few selected objects are depicted in Fig. 5. Notice the amount of non-linear distorsion caused by the cylindrical projection. The objects of interest are relatively small in comparison to the image size. In average the object size was 400×300 pixels. The ground-truth homography for each object was manually labeled in each frame. For evaluation of the detection/tracking performance we provide ROC curves for each tested object. The ROC curve illustrates *false positive rate* versus *false negative rate*.

- *False positive (FP)* is a positive vote for an object presence in some position, but the object was not there.
- *False negative (FN)* is a negative vote for an object presence in some position, where the object actually was present.

In ROC diagrams we want to get as close to the point $(0,0)$ as possible. Each point in the curve of ROC diagram is evaluated for one particular *confidence threshold* c. In our system the *confidence* r for one detection is given by the number of affine RANSAC inliers after positive detection. The tracker keeps the confidence from last detection until the loss-of-track. With growing confidence we get less false positives, but also more false negatives (we may miss some positive detections). For one particular c we compute the

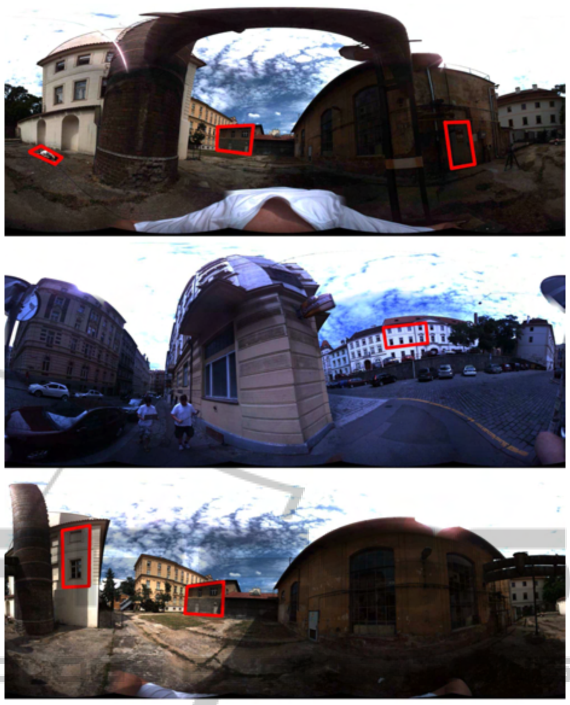
Figure 4: Example images with tracked objects marked by red rectangles.

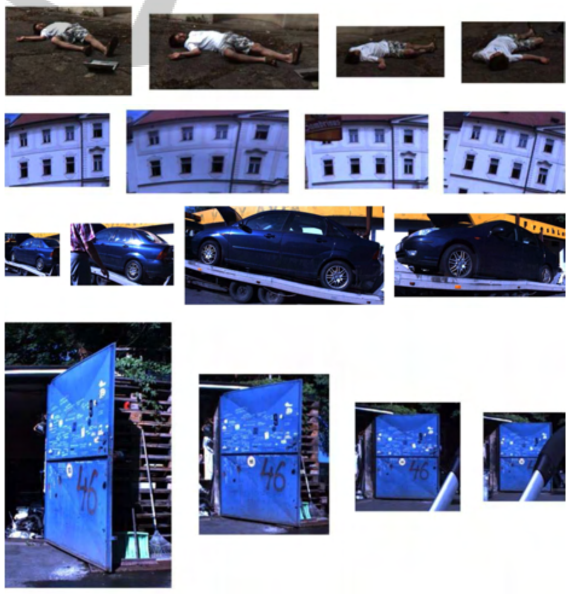
Figure 5: Four of eight tested objects under different view angles and appearances.

diagram coordinates as follows:

$$\mathrm{FP}(c) = \sum_{j=1}^{n} (\mathrm{FP}, \text{where } r_j > c)/n \qquad (8)$$

$$\mathrm{FN}(c) = \sum_{j=1}^{n} (\mathrm{FN}, \text{where } r_j > c)/n, \qquad (9)$$

where *n* is a number of frames in sequence. To draw the whole ROC curve we compute the coordinates for a discrete number of confidences from interval $< 0,1 >$ and use linear interpolation for rest of the values.

In Fig. 6 we show three different ROC curves. Each curve corresponds to one method used to search for the object position in sequences. In order to make the evaluation less dependent on a particular object, we computed mean ROC curves over all tested objects for different methods. The green curve depicts the performance of the tracker itself, run on every object from the first frame until the loss-of-track without the recovery by the detector. The blue curve shows results obtained by the fern detector itself run on every frame of all sequences. And finally the red curve shows results, when our algorithm was used. We may observe, that our algorithm performance is better (curve is the closest to point $(0,0)$) than both individual methods. The separate ROC curves for individual objects may be seen in Fig. 7 and Fig. 9. The experiments are organised as follows. The ferns detector is evaluated in Section 3.1, the performance of tracker is examined in Section 3.2. The algorithm 1, which combines both is evaluated in Section 3.3. And finally in Section 3.4 we provide computation times of all main parts of the algorithm.

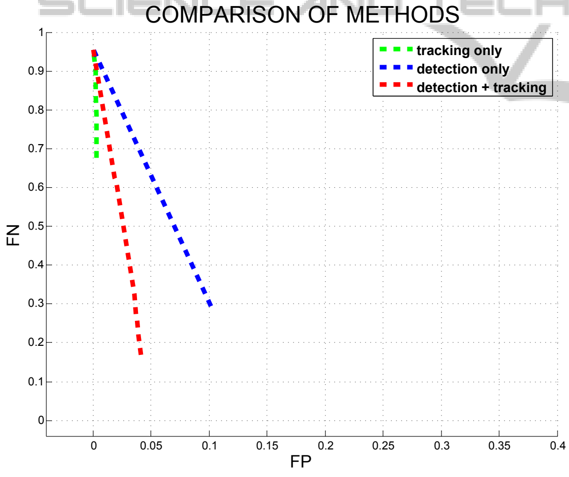

Figure 6: Each curve corresponds to results of one method computed as mean ROC curve over all objects.

3.1 Detector Evaluation

Using only the detector (tracking by detection) would be too slow for desired real-time performance in sequence of large images. Nevertheless we evaluate the performance of the detector itself to see how the adition of SLLiP tracker lowers the false positive and false negative rate (see Section 3.3).

In this experiment the detector was run with slightly different set of parameters than in the experiment which combines it with the tracker. This was necessary in order to achieve the best detection performance. For example here it was not possible to aditionally validate the positive detection by the validator. So we needed to increase the threshold for number of RANSAC inliers necessary for positive detection to lower the number of false positives.

It was also necessary to adjust the detector parameters according to expected object scale and rotation changes. In average the detector was searching for the object in 3 scales and it was able to detect objects under ± 20 degrees rotation. In Fig. 7, the ROC curves are depicted for detector run in every frame for different objects. The results show that

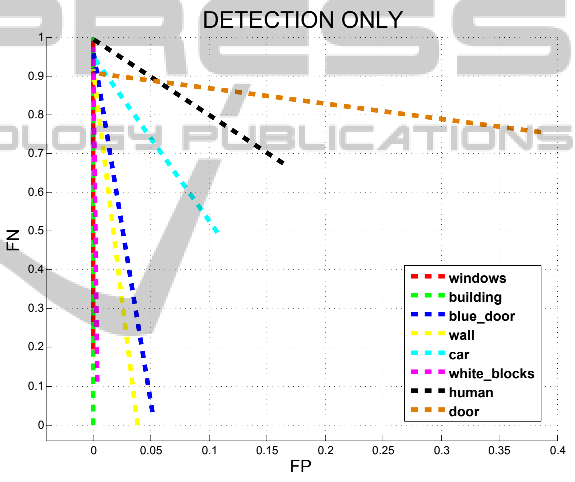

Figure 7: Each curve corresponds to detection results for one object.

some objects were detected in almost all cases correctly, while some other objects, like the *door*, with poor results. *Door* was the most complicated object for harris corners-based detector, since only 21 keypoints were detected over the object, which were spread mostly in the central part of the door. That is why there was almost always a low number of inliers comming out of the RANSAC algorithm. This object was lately successfuly tracked by the tracker. Another complicated object was the *car*, due to its reflective surface and vast visual angle changes. Finally, the *human* lying on the floor was also a challenging object due to its non-planarity. As you will see in Section 3.3, the integration of tracking to the algorithm lowers the number of FP and FN and significantly speeds up the algorithm, see Section 3.4.

3.2 Tracker Evaluation

This experiment shows performance of the tracker without the recovery by the detector. The tracker is composed of 2 SLLiPs (for translation and homography). Each SLLiP has 3 predictors in sequence with support set sizes $|X_1| = 225$, $|X_2| = 324$ and $|X_3| = 441$. The support set coordinates were spread over the object in regular grid. The tracker was incrementaly learned and validated during tracking until it lost its track or until the end of sequence. The tracking was manually initialized always in the first image of sequence (different form training image), where the object appeared. Some objects were tracked through the whole sequence. Some objects were lost after few frames, when there was fast motion right in the beginning. In Fig. 8 you may see the lengths of successful tracking until the first loss-of-track. In case of partial

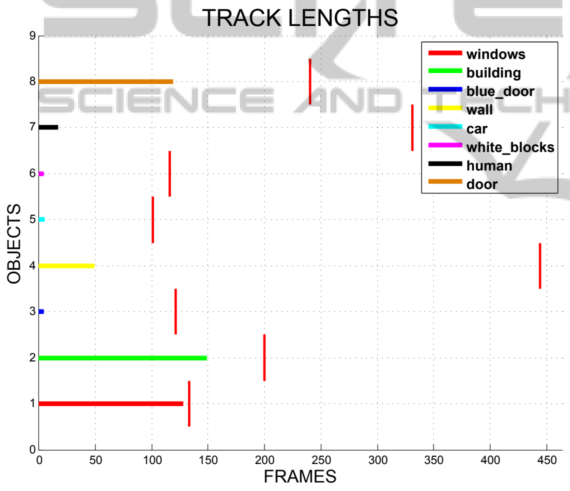

Figure 8: Each horizontal line depicts the length of track for one object until the first loss-of-track. The red vertical lines show the last frame of particular subsequence, where the object was fully or partially visible.

occlusion the tracker sometimes jitters or even fails. Nevertheless, when it is incrementally learned, it is able to handle the occlusion as a new object appearance. Incremental learning itself is very helpful for increasing the robustness of the tracker (Hinterstoisser et al., 2008), (Hurych and Svoboda, 2010). The estimation of homography is very precise for planar objects.

Tracked objects appear in images as patches in resolutions varying from 211×157 (33 127 pixels) to 253×919 (232 507 pixels). Both SLLiPs work only with the subset of all patch pixels (same subset size for all objects). When tracking, each SLLiP needs to read only 990 intensity values, which is given by the sum of support set sizes of predictors in sequence. This brings another significant speed-up for the learning and tracking process.

3.3 Detector + Tracker Evaluation

Final experiment evaluates the performance of algorithm described in Section 2.3. The combination of the detector and the tracker improves the performance of the algorithm (lowers FP and FN), as may be seen in Fig. 9 and Fig. 6. This is caused by their complementarity in failure cases. Tracker is very robust even under extreme perspective deformations, while the detector is not able to recognize these difficult object poses. On the other hand the detector is robust to partial occlusion, where the tracker usually fails and needs to be recovered and re-learned. In com-

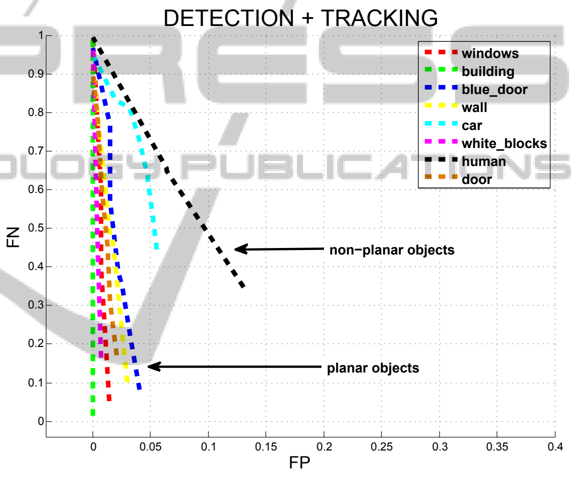

Figure 9: Each curve corresponds to results of one object detection and tracking. The ROC curves fit more to the left side of the diagram. This is caused by the high confidence of detections and tracking. The high confidence is actually valid, because of the very low number of false positives, as we may observe.

parison with the detector (see Fig. 7), our algorithm in average significantly improves the results. Only few objects, which were perfectly detected by the detector (e.g. *white blocks* and *blue door*) have a little worse results with our algorithm. This was caused by the tracking validation, which was running not every frame, but only every 5 frames, which means, that the tracker was lost a few frames just before loss-of-track detection by validation and received a few FPs and FNs. This small error could be eliminated by running the validation in every frame. The extreme efficiency of sequential predictors allows tracking much faster than real-time, which provides enough computational time for validation and incremental learning of the tracker. Running validation after each positive detection allows us to make the ferns detector more sensitive. We lower the threshold which specifies the number of neccessary inliers, which allows more true positive, but also more false positive detections. After each detection, which has small number of inliers, we initialize the tracker in detected pose, let the tracker vote for homography and run the validation. Validation eliminates possible false positive detections and let pass the true positives.

The most difficult object for our algorithm was the *crashed car*, the appearance of which was changing signifficantly during the sequence, due to its reflective surface, non-planar shape and vast changes in visual angle. Detection and tracking of *lying human* was successful in high percentage of detected occurences and low FP and FN. But the precision of homography estimation was quite poor as expected, because of its non-planar geometry. Nevertheless the incremental learning kept the tracker from loosing its track too often. The robust detector and incremental learning of the tracker allows for tracking of more complex (non-planar) objects, but high precision homography estimation can not be expected. Planar or semi-planar objects were detected and tracked with high accuracy.

3.4 Computation Times

The algorithm was run on standard PC with 64 bit, 2.66 GHz CPU. The object detector was implemented in language C and run in the system as MEX. The RANSAC, 2-SLLiP tracker and the rest of the algorithm were implemented in Matlab. When we pretermit the high resolution images, the PC memory requirements were minimal. The computation times for 12 Mpx image sequences are following:

(implementation in language C)

- detector learning: 2 sec for 200 classes, i.e. 10 ms per class (50 ferns with depth 11 and 500 training samples per class).
- detection: 0.13 ms for evaluation of 1 harris point with 50 ferns and 200 classes. The computational time changes linearly with the number of classes. For one image with 5350 harrises, which passed the quality threshold, it took 0.7 sec. Usually we run the detector in 3 scales.

(implementation in Matlab)

- learning SLLiP trackers: 6 sec for the translation SLLiP with 1500 training samples and 9 sec for the homography SLLiP with 3500 training samples.
- tracking: 4 ms per image. This computational time is summed for both SLLiP trackers.
- validation: 72 ms per one validation. In our experiments, the validation was run every 5 frames during tracking.
- incremental learning: 470 ms together for 10 samples for the translation SLLiP and 10 samples for the homography SLLiP. Incremental learning was triggered every 5 frames after successful validation.

Average amount of harris points in one image was around 50000, from which around 5300 passed the harris quality threshold (Shi and Tomasi, 1994) and were evaluated by ferns detector. The use of object detector is neccessary, but its runtime needs to be reduced to a minimum because of the high computational time. The tracker runs very fast, which allows for multiple object tracking, incremental learning and tracking validation.

4 CONCLUSIONS AND FUTURE WORK

In this work we combined ferns-based object detector and 2-SLLiP tracker in an efficient algorithm suitable for real-time processing of high resolution images. The amount of streamed data is huge and we need to avoid running the detector too often. That is why we focused on updating the 2-SLLiP model during tracking, which helped to keep the track even when the object appeared under serious slope angles and with changing appearance. In comparison with the detector run on every frame, our algorithm runs not only much faster, but also lowers the number of false positives and false negatives.

In our future work we want to focus on incremental learning of both the detector and the tracker. The detector is robust to partial occlusion, since it works with harris corners (Harris and Stephen, 1988) sparsely placed around the object unlike the patch-based tracker. On the other hand the tracker is more robust to object appearance changes and keeps tracking even the signifficantly distorted objects, which the detector fails to detect. This gives the opportunity to deliver the training examples for the detector in cases where it fails, while the tracker holds and vice-versa. We would like to develop a suitable strategy for mutual incremental learning.

ACKNOWLEDGEMENTS

The 1st author was supported by Czech Science Foundation Project P103/10/1585. The 2nd author was supported by Czech Science Foundation Project P103/11/P700. The 3rd author was supported by EC project FP7-ICT-247870 NIFTi. Any opinions expressed in this paper do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

REFERENCES

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. In *International Journal of Computer Vision*, volume 56, pages 221–255.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2006). Speeded-up robust features. In *Proceedings of IEEE European Conference on Computer Vision*, pages 404–417.

Dellaert, F. and Collins, R. (1999). Fast image-based tracking by selective pixel integration. In *Proceedings of the International Conference on Computer Vision: Workshop of Frame-Rate Vision*, pages 1–22.

Hager, G. D. and Belhumeur, P. N. (1998). Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039.

Harris, C. and Stephen, M. (1988). A combined corner and edge detection. In Matthews, M. M., editor, *Proceedings of the 4th ALVEY vision conference*, pages 147–151, University of Manchaster, England. on-line copies available on the web.

Hinterstoisser, S., Benhimane, S., Navab, N., Fua, P., and Lepetit, V. (2008). Online learning of patch perspective rectification for efficient object detection. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Holzer, S., Ilic, S., and Navab, N. (2010). Adaptive linear predictors for real-time tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR), 2010 IEEE*, pages 1807–1814.

Hurych, D. and Svoboda, T. (2010). Incremental learning and validation of sequential predictors in video browsing application. In *VISIGRAPP 2010: International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, volume 1, pages 467–474.

Jurie, F. and Dhome, M. (2002). Hyperplane approximation for template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:996–1000.

Li, M., Chen, W., Huang, K., and Tan, T. (2010). Visual tracking via incremental self-tuning particle filtering on the affine group. In *Conference on Computer Vision and Pattern Recognition (CVPR), 2010 IEEE*, pages 1315–1322.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal on Computer Vision*, 60(2):91–110.

Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application in stereo vision. In *Proceedings of the 7th International Conference on Artificial Intelligence*, pages 674–679.

Özuysal, M., Calonder, M., Lepetit, V., and Fua, P. (2010). Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461.

Shi, J. and Tomasi, C. (1994). Good features to track. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 593–600.

Šochman, J. and Matas, J. (2005). Waldboost - learning for time constrained sequential detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 150–157.

Zimmermann, K., Matas, J., and Svoboda, T. (2009a). Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):677–692.

Zimmermann, K., Svoboda, T., and Matas, J. (2009b). Anytime learning for the NoSLLiP tracker. *Image and Vision Computing, Special Issue: Perception Action Learning*, 27(11):1695–1701.