# Feature Harvesting for Tracking-by-Detection*

Mustafa Özuysal, Vincent Lepetit, François Fleuret, and Pascal Fua

Computer Vision Laboratory
École Polytechnique Fédérale de Lausanne (EPFL) *1015 Lausanne, Switzerland*
{mustafa.oezuysal, vincent.lepetit, francois.fleuret, pascal.fua}@epfl.ch
*http://cvlab.epfl.ch*

**Abstract.** We propose a fast approach to 3–D object detection and pose estimation that owes its robustness to a training phase during which the target object slowly moves with respect to the camera. No additional information is provided to the system, save a very rough initialization in the first frame of the training sequence. It can be used to detect the target object in each video frame independently.
Our approach relies on a Randomized Tree-based approach to wide-baseline feature matching. Unlike previous classification-based approaches to 3–D pose estimation, we do not require an *a priori* 3–D model. Instead, our algorithm learns both geometry and appearance. In the process, it collects, or *harvests*, a list of features that can be reliably recognized even when large motions and aspect changes cause complex variations of feature appearances. This is made possible by the great flexibility of Randomized Trees, which lets us add and remove feature points to our list as needed with a minimum amount of extra computation.

## 1 Introduction

In many 3–D object-detection and pose estimation problems ranging from Augmented Reality to Visual Servoing, run-time performance is of critical importance. However, there usually is time to train the system before actually using it. It has recently been shown [1] that, given a 3–D model, statistical learning techniques [2] can be used during this training phase to achieve robust real-time performance by learning the appearance of features on the target object. As a result, at run-time, it becomes possible to perform wide-baseline matching quickly and robustly, which is then used to detect the object and compute its 3–D pose. Here we show that this approach extends naturally to the case where no *a priori* 3–D model is available, thus removing one of the major limitations of the original method and yielding the behavior depicted by Fig. 1.

The key ingredient of our approach is what we refer to as *feature harvesting*: Assuming that we can first observe the target object moving slowly, we define an ellipsoid that roughly projects at the object's location in the first frame. We extract feature points inside this projection and use the image patches surrounding

* This work was supported in part by the Swiss National Science Foundation.

them to train a first classifier, which is then used to match these initial features in the following frames. As more and more new frames become available, we discard features that cannot be reliably found and add new ones to account for aspect changes. We use new views of the features we retain to refine the classifier and, each time we add or remove a feature, we update it accordingly. Once all the training frames have been processed, we run a bundle-adjustment algorithm on the tracked feature points to also refine the model's geometry. In short, starting from the simple ellipsoid shown in the top row of Fig. 1, we robustly learn both geometry and appearance. An alternative approach to initializing the process would have been to use a fully automated on-line SLAM algorithm [3]. We chose the ellipsoid both for simplicity's sake—successfully implementing a SLAM method is far from trivial—and because it has proved to be sufficient, at least for objects that can be enclosed by one.

The originality of our approach is to use exactly the same tracking and statistical classification techniques, first, to train the system and automatically select the most stable features and, second, to detect them at run-time and compute the pose. In other words, the features we harvest are those that can be effectively tracked by the specific wide-baseline matching algorithm we use. This contrasts with standard classification-based approaches in which classifiers are built beforehand, using a training set manually labeled and that may or may not be optimal for the task at hand. As a result, our system is very easy to train by simply showing it the object slowly moving and, once trained, both very fast and very robust to a wide range of motions and aspect changes, which may cause complex variations of feature appearances.

## 2 Related Work

In recent years, feature-based approaches to object recognition and pose estimation have become increasingly popular for the purpose of 3–D object tracking and detection [4–7], mostly because they are relatively insensitive to partial occlusions and cluttered backgrounds.

These features are often designed to be affine invariant [8]. Once they have been extracted, various local descriptors have been proposed to match them across images. Among these, SIFT [9] has been shown to be one of the most effective [10]. It relies on local orientation histograms and tolerates significant local deformations. In [8], it is applied to rectified affine invariant regions to achieve perspective invariance. In [11], a similar result is obtained by training the system using multiple views of a target object, storing all the SIFT features from these views, and matching against all of them. However, computing such descriptors can be costly. Furthermore, matching is usually performed by nearest-neighbor search, which tends to be computationally expensive, even when using an efficient data structure [12].

Another weakness of these descriptors is that they are predefined and do not adapt to the specific images under consideration. [5] addresses this issue by building the set of the image neighborhoods of features tracked over a sequence.


**Fig. 1.** Our approach to 3–D object detection applied to a toy car, a face, and a glass. In each one of the three cases, we show two rows of pictures. The first represents the training sequence, while the second depicts detection results in individual frames that are not part of the training sequence. We overlay the ellipsoid we use as our initial model on the images of the first row. The only required manual intervention is to position it in the very first image. To visualize the results, we attach a 3–D referential to the center of gravity of the ellipsoid and use the estimated 3–D pose to project it into the images. Note that, once trained, our system can handle large aspect, scale, and lighting changes. It can deal with the transparent glass as well as with the hand substantially occluding the car. And when a complete occlusion occurs, such as when the book completely hides the face, it simply returns no answer and recovers when the target object becomes visible again.

Kernel PCA is then performed on this set to compute a descriptor for each feature. This approach, however, remains computationally expensive.

By contrast, [1] proposes a classification-based approach that is both generic and faster. Since the set of possible patches around an image feature under

changing perspective and lighting conditions can be seen as a class, it is possible to train a classifier—made of Randomized Trees (RT) [2]—to recognize feature points by feeding it samples of their possible appearances. In the case of 3–D objects, these samples are synthesized using a textured model of the target object. This is effective because it allows the system to learn potentially complex appearance changes. However, it requires building the 3–D model. This can be cumbersome if the object is either complex or made of a non-Lambertian material that makes the creation of an accurate texture-map non-trivial. If one is willing to invest the effort, it can of course be done but it is time consuming. The approach we introduce here completely does away with this requirement.

## 3 Randomized Trees for Feature Recognition

The approach we use as a starting point [1] relies on matching image features extracted from training images and those extracted from images acquired at run-time under potentially large perspective and scale variations. It formulates wide-baseline matching as a classification problem by treating the set of all possible appearances of each individual *object feature*, typically a 3–D point on the object surface, as a class. During training, given at least one image of the target object, *image features*, are extracted and associated to object features. These features are taken to be extrema of Laplacian extracted from the first few octaves of the images. This simple multi-scale extraction and the classifier work in tandem to recognize the features under large variation of both scale and appearance. Image patches surrounding the image features are then warped to generate numerous synthetic views of their possible appearance under perspective distortion, which are then used to train a set of Randomized Trees (RTs) [2]. These RTs are used at run-time to recognize the object features under perspective and scale variations by deciding to which class, if any, their appearance belongs.

The training procedure outlined above assumes that a fixed number of image features have been extracted *beforehand* and that their number does not change. This is not true in our case because image features can be added or discarded *during* training. Therefore, in the remainder of this section, we first recall the original formulation [1] and then extend it to allow the addition and removal of object features on the fly. RTs appear to be a very good trade-off between the efficiency of the recognition, and these possibilities of manipulations.

### 3.1 Wide Baseline Matching using Randomized Trees

Let us consider a set of 3–D object features $\{\mathbf{M}_i\}$ that lie on the target object and let us assume that we have collected a number of image patches $f_{i,j}$ centered on the projections of $\mathbf{M}_i$ into image $j$, for all available $i$ and $j$. The $\{f_{i,j}\}$ constitute the training set we use to train the classifier $\hat{\mathscr{R}}$ to predict to which $\mathbf{M}_i$, if any, a given image patch $f$ corresponds, in other words, to approximate as well as possible the actual mapping $\mathscr{R}(f) = i$. At run-time, $\hat{\mathscr{R}}$ can then be used to recognize the object features by considering the image patch $f$ around

a detected image feature. Given the 3–D position of the $\mathbf{M}_i$, this is what is required to compute 3–D pose.

In principle any kind of classifier could have been used. RTs, however, are particularly well adapted because they naturally handle multi-class problems, while being both robust and fast. Multiple trees are grown so that each one yields a different partition of the space of image patches. The tree leaves contain an estimate of the posterior distribution over the classes, which is learned from training data. A patch $f$ is classified by dropping it down each tree and performing an elementary test at each node, which sends it to one side or the other, and considering the sum of the probabilities stored in the leaves it reaches. We write

$$\hat{\mathscr{R}}(f) = \underset{i}{\text{argmax}} \sum_{T \in \mathcal{T}} \hat{P}_{L(T,f)}(\mathscr{R}(f) = i) \ , \tag{1}$$

where $i$ is a label, the $\hat{P}_{L(T,f)}(\mathscr{R}(f) = i)$ are the posterior probabilities stored in the leaf $L(T, f)$ of tree $T$ reached by $f$, and $\mathcal{T}$ is the set of Randomized Trees. Such probabilities are evaluated during training as the ratio of the number $n_i^L$ of patches of class $i$ in the training set that reach $L$ and the total number $n_i$ of patches of class $i$ that is used in the training. This yields

$$\hat{P}_L(\mathscr{R}(f) = i) \simeq \frac{n_i^L / n_i}{S_L} \ , \tag{2}$$

where $S_L = \sum_j \frac{n_j^L}{n_j}$ is a normalization term that enforces $\sum_i \hat{P}_L(\mathscr{R}(f) = i) = 1$. We normalize by the number of patches because the real prior on the class is expected to be uniform, while this is not true in our training population. Although any kind of test could be performed at the nodes, simple binary tests based on the difference of intensities of two pixels have proved sufficient. Given two pixels $\mathbf{m_1}$ and $\mathbf{m_2}$ in $f$ and their gray levels $I(f, \mathbf{m_1})$ and $I(f, \mathbf{m_2})$ after some Gaussian smoothing, these tests are of the form

$$\begin{array}{ll} \text{If } I(f, \mathbf{m_1}) \leq I(f, \mathbf{m_2}) & \text{go to left child,} \\ \text{otherwise} & \text{go to right child.} \end{array} \tag{3}$$

This test is very simple and requires only pixel intensity comparisons. In practice, classifying a patch involves only a few hundreds of intensity comparisons and additions per patch, and is therefore very fast.

### 3.2 Randomized Trees and On-line Training

The approach described above assumes that the complete training set is available from the beginning, which is not true in our case as object features may be added or removed while the classifier is being trained. Here we show how to overcome this limitation by modifying the tree-building algorithm in two significant ways.

First, in [1], the node tests are chosen so as to minimize leaf entropy, which is estimated according to the training set. Without the complete training set, this cannot be meaningfully done. Instead, we build the tree by randomly selecting

the tests, that is to say the $\mathbf{m_1}$ and $\mathbf{m_2}$ locations of Eq. 3. The training data is only used to evaluate the $\hat{P}_L$ posterior probabilities in the leaves of these randomly generated trees. Surprisingly, this much simplified procedure, which is going to allow us to iteratively estimate the $\hat{P}_L$ values, results in virtually no loss of classification performance [13]. Interestingly, a similar result has also been reported in the context of 2–D object recognition [14].

Second, we introduce a mechanism for updating the tree when new views of an existing object feature are introduced or when an object feature is either added or removed, which the RT approach lets us do very elegantly as follows.

- **Incorporating New Views of Object Features.** Recall that, during the initial training phase, patches are dropped down the tree and the number of patches reaching leaf $L$ is plugged into Eq. 2 to derive $\hat{P}_L$ for each class at leaf $L$. Given a new view, we want to use it to refine these probability estimates. To this end, we invert the previous step and compute the number of patches reaching leaf $L$ as

$$n_i^L = \hat{P}_L(\mathscr{R}(f) = i) \times n_i \times S_L.$$

  This only requires storing the normalization terms $S_L$ at each leaf $L$ and keeping the $n_i$ counters for each class. We then use newly detected patches to increment $n_i^L$ and $n_i$. When all the new patches have been processed, we again use Eq. 2 to obtain the refined values of $\hat{P}_L$. Note that we do not store the image patches themselves, which could cost a lot of memory for long training sequences.
- **Adding and Removing Object Features.** The flexible procedure outlined above can also be used to add, remove or replace the classes corresponding to specific object features during training. Removing class $i$ and the corresponding object feature merely requires setting

$$n_i^L = n_i = 0.$$

  We can then replace the $i^{th}$ feature by a new one by simply changing the $\mathbf{M}_i$ 3–D coordinates introduced at the beginning of Section 3.1 to be those of the new object feature and using patches centered around the new projections of $\mathbf{M}_i$ to estimate $\hat{P}_L$.

These update mechanisms are the basic tools we use to recursively estimate the RTs while harvesting features, as discussed in the next section.

## 4 From Harvesting to Detection

In this section, we show that standard frame-to-frame tracking and independent 3–D detection in each individual frame can be formalized similarly and, therefore, combined seamlessly as opportunity dictates. This combination is what we refer to as tracking-by-detection. The originality of our approach is to use exactly the same image feature recognition technique at all stages of the process, first, to

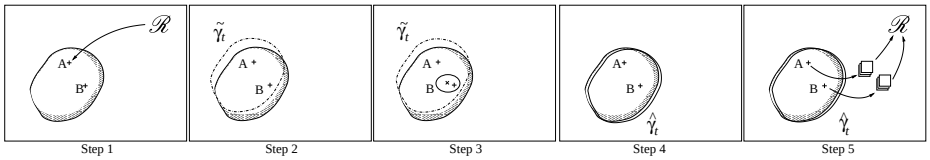


**Fig. 2.** The five steps of feature harvesting introduced at the beginning of Section 4.1.

train the system and automatically select the most stable features and, second, to detect them at run-time.

We first give an overview of our method. We then explain how the tracking is performed *without* updating the classifier, and conclude with the complete "feature harvesting" framework.

### 4.1 Overview

As shown in the top row of Fig. 1, to initialize the training process, we position the ellipsoid that we use as an initial 3–D model so that it projects on the target object in the first frame. We then extract a number of image features from this first image and back-project them to the ellipsoid, thus creating an initial set of the $\{\mathbf{M}_i\}$ object features of Section 3.1. By affine warping lightly the image patches surrounding the image features, we create the $f_{i,j}$ image patches that let us instantiate a first set of randomized trees.

During training, new features detected on the object are integrated into the classifier. Because the number of such features can become prohibitively large when dealing with long training sequences, it is desirable to keep the ones that are successfully detected and recognized by the classifier most often, and remove the other ones. More precisely, given the set of trees trained using the first frame or more generally all frames up to frame $t-1$, we handle frame $t$ using the five-step feature-harvesting procedure described below and illustrated by Fig. 2:

1. We extract image features from frame $t$ and use the classifier to match them, which, in general, will only be successful for a subset of these features.
2. We derive a first estimate $\widetilde{\gamma}_t$ of the camera pose from these correspondences using a robust estimator that lets us reject erroneous correspondences.
3. We use $\widetilde{\gamma}_t$ to project unmatched image features from frame $t-1$ into frame $t$ and match them by looking for the image features closest to their projections.
4. Using these additional correspondences, we derive a refined estimate $\widehat{\gamma}_t$.
5. We use small affine warping of the patches around image features matched in frame $t$ to update the classifier as discussed in Section 3.2. Features that have not been recognized often are removed to be replaced by new ones.

At run-time, we use the exact same procedure, with one single change: We stop updating the classifier, which simply amounts to skipping the fifth step.

### 4.2 3–D Tracking by Detection

Let us first assume that the classifier $\mathscr{R}$ has already been trained. Both tracking and detection can then be formalized as the estimation of the camera pose $\Gamma_t$ from image features extracted from all previous images that we denote $I_{s\leq t}$. In other words, we seek to estimate the conditional density $p(\Gamma_t \,|\, I_{s\leq t})$.

A camera motion model —appearing as the term $p(\Gamma_t \,|\, \Gamma_{t-1})$ in the following derivations— should be chosen. It often assumes either constant velocity or constant acceleration. This is fine to regularize the recovered motion but can also lead to complete failure. This tends to occur after an abrupt motion or if $\Gamma_{t-1}$ is incorrectly estimated, for example due to a complete occlusion. $\Gamma_t$ can then have any value no matter what the estimate of $\Gamma_{t-1}$ is. In such a case, we should consider the density of $\Gamma_t$ as uniform and write $p(\Gamma_t \,|\, \Gamma_{t-1}) \propto \lambda$, which amounts to treating each frame completely independently. In our implementation, we use a mixture of these two approaches and take the distribution to be

$$p(\Gamma_t \,|\, \Gamma_{t-1}) \propto m(\Gamma_{t-1}, \Gamma_t) = \exp\left(-\left(\Gamma_t - \Gamma_{tq-1}\right)^\top \mathbf{\Sigma}^{-1} \left(\Gamma_t - \Gamma_{t-1}\right)\right) + \lambda\,. \quad (4)$$

This lets us both enforce temporal consistency constraints and to recover from tracking failures by relying on single-frame detection results. In our implementation, the respective values of $\mathbf{\Sigma}$ and $\lambda$ were chosen manually.

Unfortunately, introducing the term $\lambda$ process precludes the use of standard particle filtering techniques. Our camera pose space has six dimensions, and the required number of particles, which grows exponentially with the number of dimensions, would be too large to make particle filters tractable. Therefore we have to restrict ourself to the estimation of the mode $\widehat{\gamma_t}$ of this density:

$$\widehat{\gamma_t} = \underset{\gamma}{\operatorname{argmax}}\ P(\Gamma_t = \gamma \,|\, I_{s\leq t})\,,$$

in which the expression of $P(\Gamma_t = \gamma \,|\, I_{s\leq t})$ can be found using the standard Bayesian tracking relation:

$$P(\Gamma_t = \gamma \,|\, I_{s\leq t}) \propto P(I_t \,|\, \Gamma_t = \gamma) P(\Gamma_t = \gamma | \Gamma_{t-1} = \widehat{\gamma_{t-1}}) P(\Gamma_{t-1} = \widehat{\gamma_{t-1}} | I_{s<t})\,. \quad (5)$$

As described in the overview, we apply a RANSAC based approach on the set $\widetilde{n_t}$ of correspondences obtained using the classifier to derive a first estimate $\widetilde{\gamma_t}$ for the camera pose. A new set $\widehat{n_t}$ is then made of the inliers of $\widetilde{n_t}$, and completed by projecting the unmatched object features with $\widetilde{\gamma_t}$ and matched each of them with the closest image feature. A numerical optimization is then performed to find $\widehat{\gamma_t}$ by minimizing the log-likelihood of $m(\widehat{\gamma_{t-1}}, \gamma) P(I_t \,|\, \Gamma_t = \gamma)$:

$$\widehat{\gamma_t} = \underset{\gamma}{\operatorname{argmin}} \sum_{n\in\widehat{n_t}} \|\mathbf{P}(\gamma)\mathbf{M}(n) - \mathbf{m}(n)\|^2 + \rho\left(\left(\gamma - \widehat{\gamma_{t-1}}\right)^\top \mathbf{\Sigma}^{-1} \left(\gamma - \widehat{\gamma_{t-1}}\right)\right) \quad (6)$$

where $\mathbf{P}(\gamma)$ is the projection matrix for the camera pose $\gamma$, $\rho$ is the Tukey robust estimator that approximates the logarithm of (4), and $\mathbf{M}(n)$ and $\mathbf{m}(n)$ are respectively the object feature and the image feature for correspondence $n$.

The advantage of the classifier is that there is no need for the previous pose. However, this procedure can result in some jittering on the estimated pose over a sequence. To enforce temporal consistency and reduce the effect, when $\widehat{\gamma_{t-1}}$ is valid, we also consider transient object features which projections can be matched across $I_{t-1}$ and $I_t$ using standard cross-correlation. Their 3–D positions can be estimated from the rough model and $\widehat{\gamma_{t-1}}$ by back-projection. According to our experience, over two consecutive frames, this position is accurate enough to improve the recovered displacement. These additional correspondences are integrated in Eq. (6) for pose estimation exactly in the same way as the correspondences established with the classifier. Note that these correspondences are not required by our method, but they are useful to reduce the jittering effect.

### 4.3 Feature Harvesting

During training we use the same process but now the classifier is not initially available and we want to create it incrementally by "feature harvesting." This implies keeping or discarding object features such as those shown in Fig. 3. Let us first denote by $r_t^*$ the best classifier obtained with the images $I_{s\leq t}$ and the feature correspondences computed using the poses $\gamma_{s\leq t}$:

$$r_t^* = \underset{r}{\operatorname{argmax}}\ P(\mathscr{R} = r \,|\, \Gamma_{s\leq t} = \gamma_{s\leq t},\, I_{s\leq t})\ .$$

Here we show that $r_t^*$ can be used to compute $\widehat{\gamma_{t+1}}$ under reasonable assumptions. We have:

$$\begin{aligned}
&P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t}) \\
&= \sum_r P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t}, \mathscr{R} = r) = \sum_r P(\Gamma_t = \gamma_t, I_t, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}, \mathscr{R} = r) \\
&= \sum_r P(\Gamma_t = \gamma_t, I_t \,|\, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}, \mathscr{R} = r) P(\mathscr{R} = r \,|\, \Gamma_{s<t} = \gamma_{s<t}, I_{s<t}) \times \\
&\quad P(\Gamma_{s<t} = \gamma_{s<t}, I_{s<t})
\end{aligned}$$

All the classifiers have a negligible probability $P(\mathscr{R} = r \,|\, P_{s<t} = \gamma_{s<t},\, I_{s<t})$ except for those concentrated around $r = r_{t-1}^*$. Otherwise, that would mean that other classifiers than $r_{t-1}^*$ constructed with $\gamma_{s<t}$, $I_{s<t}$ would be as good as $r_{t-1}^*$, which is not realistic since $r_{t-1}^*$ has been built from these poses and images. Let us continue the derivation:

$$\begin{aligned}
&\simeq P(\Gamma_t = \gamma_t,\, I_t \,|\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t}) \\
&= P(I_t \,|\, \Gamma_t = \gamma_t,\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) \times \\
&\quad P(\Gamma_t = \gamma_t \,|\, \Gamma_{s<t} = \gamma_{s<t},\, I_{s<t},\, \mathscr{R} = r_{t-1}^*) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t}) \\
&\simeq P(I_t \,|\, \Gamma_t = \gamma_t,\, \mathscr{R} = r_{t-1}^*) P(\Gamma_t = \gamma_t \,|\, \Gamma_{s<t} = \gamma_{s<t}) P(\Gamma_{s<t} = \gamma_{s<t},\, I_{s<t})
\end{aligned}$$

because the incoming image does not depend on the poses except on the current one, and the current pose does not depend on the previous images neither on


**Fig. 3.** The harvest. (a) Three sample patches for three distinct features on the glass. Note that the foreground is relatively constant while the background changes drastically. (b) Three sample patches for three distinct face features, obtained under changing light and orientation. (c) Patches corresponding to object features found to be unreliable and discarded during training.

the classifier, which is reasonable. By applying the Bayes' theorem on the terms $P(\Gamma_{s\leq t} = \gamma_{s\leq t}, I_{s\leq t})$ and $P(\Gamma_{s<t} = \gamma_{s<t}, I_{s<t})$, we get:

$$P(\Gamma_{s\leq t} = \gamma_{s\leq t} \,|\, I_{s\leq t}) \simeq$$
$$\frac{P(I_{s<t})}{P(I_{s\leq t})} P(I_t \,|\, P_t = \gamma_t, \mathscr{R} = r^*_{t-1}) P(\Gamma_t = \gamma_t \,|\, \Gamma_{t-1 s<t} = \gamma_{s<t}) P(\Gamma_{s<t} = \gamma_{s<t} \,|\, I_{s<t})$$

And under standard probabilistic tracking hypotheses, we finally obtain:

$$P(\Gamma_t = \gamma_t \,|\, I_{s\leq t}) \propto$$
$$P(I_t \,|\, P_t = \gamma_t, \mathscr{R} = r^*_{t-1}) P(\Gamma_t = \gamma_t \,|\, \Gamma_{t-1} = \gamma_t) P(\Gamma_{t-1} = \gamma_{t-1} \,|\, I_{s<t})$$

which is the same expression as Eq. 5 used for tracking, except that the classifier $r^*_{t-1}$ appears in the observation model. That means that the same method as in Section 4.2 can be used to estimate $\widehat{\gamma_t}$. Once this pose is found, $r^*_{t-1}$ is updated using correspondences between object features and image features to give $r^*_t$ as explained in Section 3.2.

To validate this training procedure, we performed the experiment depicted by Fig. 4, which clearly shows that the recovered camera trajectory does not drift.

## 5 Results

In this section we demonstrate the effectiveness and generality of our approach using three very different objects, a toy car, a face, and a partially-textured transparent glass. In all three cases, we follow the same procedure: We show the system the training sequence depicted by the top rows of Fig. 1, which is used to harvest features as discussed in Section 4.3. When all the training frames have been processed, we freeze the set of RTs we have built and proceed with the tracking-by-detection approach of Section 4.2. Our non-optimized implementation runs at 5Hz during tracking, and 1Hz during training. About 20% of the

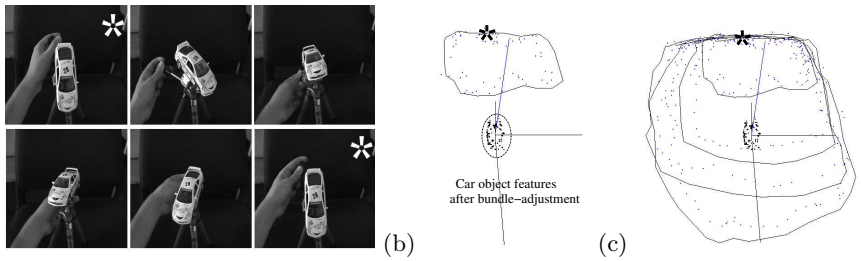


**Fig. 4.** (a) Sample frames from a training sequence where the toy car is fixed to a tripod and rotated four times. The frames marked with a star show the reference position which is reached in all four loops. (b) Recovered relative camera motion with respect to the toy car after the first loop. The trajectory is shown in the referential of the ellipsoid. The dots represent the trajectory before bundle-adjustment, the plain curve after. (c) Camera motion for all four loops. As can be seen, there is no drift. Note that all four loops go through the star.

time is devoted to extracting and recognizing the features, and the remaining 80% by the pose estimation procedure. This could be considerably sped-up by using more efficient strategies [15].

Figs 5, 6, and 7 show a number of frames extracted from test sequences of several hundreds frames—the toy sequence is made of about 1500 frames—in which our target objects translate and rotate. Because the object is re-detected in every frame, the algorithm is robust to abrupt motion and complete occlusion. For example, after the third frame of Fig. 5, the car falls on the ground and has to be picked up. As soon as it becomes visible again, the system reacquires it. The same happens in the example of Fig. 6 after the subject hides his face behind the book. These examples highlight some of the strengths of our algorithm:

- **Robustness to cluttered background.** Once trained, the classifier is feature-specific enough so that it does not get confused by cluttered background as shown in Fig. 5.
- **Insensitivity to scale changes.** Thanks to the multi-scale approach to feature detection described at the beginning of Section 3, the algorithm can handle a very broad range of scales, including scales that were not part of the training sequence. As shown in several of the examples of Figs 5 and 6, the system keeps on successfully detecting even though the target object moves both much closer and much further.
- **Robustness to complex illumination effects.** In the case of the face, we deliberately changed the lighting when acquiring the training sequence of Fig. 1 to build lighting invariance into the classifier. As can be seen in the bottom rows of Fig. 6, this was successful and gives the system robustness to very marked lighting changes. While it was not necessary for the toy car because it has a simple shape, it experimentally appeared that a training sequence with such variations greatly improve the results.
- **Handling transparencies.** Finally, we can also handle the partially-textured transparent glass of Fig. 6 by using a suitable training sequence with a com-

**Fig. 5.** Detecting the car in a sequence that involves abrupt motions, large scale and lighting changes, and very substantial occlusions. To visualize the results, we attach a 3–D referential to the center of gravity of the initial ellipsoid and use the estimated 3–D pose to project it into the images. We also overlay the projections of the harvested object feature points. The toy car is successfully detected in all frames except those where it is almost entirely occluded. And, because the object is re-detected in every frame, the system easily recovers after such a failure.

plex background. It lets the classifier learn that the parts of the patches surrounding feature points that overlap the transparent parts or the background are not relevant for classification purposes. Our algorithm can automatically reject feature points on transparent parts. At run-time features can thus be successfully recognized even if the background has changed.

## 6 Conclusion

Feature-based approaches to 3–D object detection that take advantage of *a priori* knowledge of the object's shape have consistently shown to be among the most effective. Their drawback is that building an accurate model that includes both 3–D and texture information, while usually possible, tends to be cumbersome. The approach proposed here exhibits the same reliability but completely does


**Fig. 6.** Face results. Note that by contrast with previous face detection approaches, the face pose can be retrieved under (a) large rotations, (b) scale and lighting changes, and (c) different facial expressions. (d) After the occlusion by the book, the algorithm automatically recovers.

**Fig. 7.** Detecting a transparent object with partial texture. The squares in the first three images outline the patches around the features detected at three different scales in a test frame. The straight line segments connect the feature with the corresponding one in a frame of the training sequence. Since during training the system learned which parts of the patches are meaningful as shown in Fig. 3, the image features can be recognized even if the patch overlaps the background or the transparent parts. As shown in the fourth frame, the glass is successfully detected.

away with *a priori* 3–D model building. Instead, during an automated training phase, the system learns both geometry and appearance of object feature points that have been harvested because they can be reliably recognized.

In a more global context, learning based on the consistency between two unknown stochastic variables, in our case between the appearance and the pose and between two poses close in time, is known to tremendously reduce the required amount of expert knowledge. This paradigm has proved its power in speech processing with the Baum-Welch algorithm [16], and as our results demonstrate, is also suitable for object recognition and tracking.

We believe this to be an important step towards developing applications that can handle a hundreds or thousands of objects. Indeed, this will only be possible if only minimal amounts of manual intervention are required, which may preclude the building of 3–D models for all target objects.

## References


1. Lepetit, V., Lagger, P., Fua, P.: Randomized Trees for Real-Time Keypoint Recognition. In: Conference on Computer Vision and Pattern Recognition, San Diego, CA (2005)
2. Amit, Y., Geman, D.: Shape Quantization and Recognition with Randomized Trees. Neural Computation **9** (1997) 1545–1588
3. Davison, A.: Real-Time Simultaneous Localisation and Mapping with a Single Camera. In: International Conference on Computer Vision. (2003) 1403–1410
4. S. Se and D. G. Lowe and J. Little: Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. International Journal of Robotics Research **22** (2002) 735–758
5. Meltzer, J., Yang, M.H., Gupta, R., Soatto, S.: Multiple View Feature Descriptors from Image Sequences via Kernel Principal Component Analysis. In: European Conference on Computer Vision. (2004) 215–227
6. Skrypnyk, I., Lowe, D.G.: Scene modelling, recognition and tracking with invariant image features. In: International Symposium on Mixed and Augmented Reality, Arlington, VA (2004) 110–119
7. Lepetit, V., Fua, P.: Monocular model-based 3d tracking of rigid objects: A survey. Foundations and Trends in Computer Graphics and Vision **1** (2005) 1–89
8. Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., Gool, L.V.: A comparison of affine region detectors. Accepted to International Journal of Computer Vision (2005)
9. Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision **20** (2004) 91–110
10. Mikolajczyk, K., Schmid, C.: A Performance Evaluation of Local Descriptors. In: Conference on Computer Vision and Pattern Recognition. (2003) 257–263
11. Pritchard, D., Heidrich, W.: Cloth motion capture. In: Eurographics. Volume 22. (2003) 263–271
12. Beis, J., Lowe, D.: Shape Indexing using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In: Conference on Computer Vision and Pattern Recognition, Puerto Rico (1997) 1000–1006
13. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. IEEE Transactions on Pattern Analysis and Machine Intelligence (2006) Accepted for publication.
14. Marée, R., Geurts, P., Piater, J., Wehenkel, L.: Random subwindows for robust image classification. In: Conference on Computer Vision and Pattern Recognition. (2005)

15. Chum, O., Matas, J.: Matching with PROSAC - Progressive Sample Consensus. In: Conference on Computer Vision and Pattern Recognition, San Diego, CA (2005) 220–226
16. Rabiner, L., Juang, B.H.: Fundamentals of Speech Recognition. Prentice Hall, Englewood Cliffs, NJ, USA (1993)