REAL-TIME OBJECT DETECTION AND TRACKING FOR INDUSTRIAL APPLICATIONS

Selim Benhimane[1], Hesam Najafi[2], Matthias Grundmann[3], Ezio Malis[4], Yakup Genc[2], Nassir Navab[1]

[1] *Department of Computer Science, Technical University Munich, Boltzmannstr. 3 , 85748 Garching, Germany*
[2] *Real-Time Vision & Modeling Dept., Siemens Corporate Research, Inc., College Rd E, Princeton, NJ 08540, USA*
[3] *College of Computing, Georgia Institute of Technology, Atlanta, GA 30332, USA*
[4] *I.N.R.I.A. Sophia-Antipolis, France*
selim.benhimane@in.tum.de, hesam.najafi@siemens.com

Abstract: Real-time tracking of complex 3D objects has been shown to be a challenging task for industrial applications where robustness, accuracy and run-time performance are of critical importance. This paper presents a fully automated object tracking system which is capable of overcoming some of the problems faced in industrial environments. This is achieved by combining a real-time tracking system with a fast object detection system for automatic initialization and re-initialization at run-time. This ensures robustness of object detection, and at the same time accuracy and speed of recursive tracking. For the initialization we build a compact representation of the object of interest using statistical learning techniques during an off-line learning phase, in order to achieve speed and reliability at run-time by imposing geometric and photometric consistency constraints. The proposed tracking system is based on a novel template management algorithm which is incorporated into the ESM algorithm. Experimental results demonstrate the robustness and high precision of tracking of complex industrial machines with poor textures under severe illumination conditions.

1 INTRODUCTION

Many applications require tracking of complex 3D objects in real-time. The 3D tracking aims at continuously estimating the 3D displacement of the camera relative to an object of interest, i.e. recovering all six degrees of freedom that define the camera position and orientation relative to an object. The applications include Augmented Reality where real-time registration is essential for visual augmentation of the object.

This paper presents a fully automated 3D object detection and tracking system which is capable of overcoming the problems of robust and high precision tracking of complex objects such as industrial machines with poor textures in real-time. The system proves to be fast and reliable enough for industrial service scenarios, where 3D information is presented to the service technician, e.g. on the head mounted display for hands-free operation. This is achieved by combining a real-time tracking system with a fast object detection system for automatic initialization. For this purpose, we first introduce a template-based tracking system using temporal continuity constraints

for accurate short baseline tracking in real-time. This is achieved by incorporating a template management algorithm into the ESM algorithm (Benhimane and Malis, 2006) which allows an unsupervised selection of the regions of the image belonging to the object and their automatic update.

Imposing temporal continuity constraints across frames increases the accuracy and quality of the tracking results. However, because of the recursive nature of tracking algorithms, they still require an initialization, i.e. providing the system with the initial pose of the object or camera. In fact, once tracking is started, rapid camera or object movements cause image features to undergo large motion between frames which can cause the visual tracking systems to fail. Furthermore, the lighting during a shot can change significantly; reflections and specularities may confuse the tracker. Finally, complete or partial occlusions as well as large amounts of background clutter or simply when the object moves out of the field of view, may result in tracking failure. Once the system loses track, a re-initialization is required to continue track-

ing. In our particular case, namely industrial applications of Augmented Reality, where the user wears a head-mounted camera, this problem becomes very challenging. Due to fast head movements of users head while working in a collaborative industrial environment, frequent and fast initialization is required. Manual initialization or the use of hybrid configurations, for instance instrumenting the camera with inertial sensors or gyroscopes are not a practical solution and in many cases not accepted by the end-users.

We therefore propose a purely vision-based method that can detect the target object and compute its three-dimensional pose from a single image. However, wide baseline matching tends to be both less accurate and more computationally intensive than the short baseline variety. Therefore, in order to overcome the above problems, we combine the object detection system with the tracking system using a management framework. This allows us to get robustness from object detection, and at the same time accuracy from recursive tracking.

2 RELATED WORK

Several papers have addressed the problem of template-based tracking in order to obtain the direct estimation of 3D camera displacement parameters. The authors of (Cobzas and Jagersand, 2004) avoid the explicit computation of the Jacobian that relates the variation of the 3D pose parameters to the appearance in the image by using implicit function derivatives. In that case, the minimization of the image error is done using the inverse compositional algorithm (Baker and Matthews, 2004). However, since the parametric models are the 3D camera displacement parameters, this method is valid only for small displacements around the reference positions, i.e, around the position of the keyframes (Baker et al., 2004). The authors of (Buenaposada and Baumela, 2002) extend the method proposed in (Hager and Belhumeur, 1998) to homographic warpings and make the assumption that the true camera pose can be approximated by the current estimated pose (i.e. the camera displacement is sufficiently small). In addition, the Euclidean constraints are not directly imposed during the tracking, but once a homography has been estimated, the rotation and the translation of the camera are then extracted. The authors of (Sepp and Hirzinger, 2003) go one step further and extend the method by including the constraint that a set of control points on a three-dimensional surface undergo the same camera displacement.

These methods work well for small interframe displacements. Their major drawback is their restricted convergence radius since they are all based on a very local minimization where they assume that the image pixel intensities vary linearly with respect to the estimated motion parameters. Consequently, inescapably, the tracking fails when the relative motion between the camera and the tracked objects is fast. Recently, in (Benhimane and Malis, 2006) , the authors propose a *2nd-order optimization algorithm* that considerably increases the convergence domain and rate of standard tracking algorithms while having an equivalent computational complexity. This algorithm works well for planar and simple piecewise planar objects. However, it fails when the tracking needs to take new regions into account or when the appearance of the object changes during the camera motion.

In this paper, we adopt an extension of this algorithm and we propose a template management algorithm that allows an unsupervised selection of regions belonging to the object newly visible in the image and their automatic update. This greatly improves the results of the tracking in real industrial applications and makes it much more scalable and applicable to complex objects and severe illumination conditions.

For the initialization of the tracking system a desirable approach would be to use purely vision-based methods that can detect the target object and compute its three-dimensional pose from a single image. If this can be done fast enough, it can then be used to initialize and re-initialize the system as often as needed. This problem of *automated initialization* is difficult because unlike short baseline tracking methods, a crucial source of information, a strong prior on the pose and with this spatial-temporal adjacency of features, is not available.

Computer vision literature includes many object detection approaches based on representing objects of interests by a set of local 2D features such as corners or edges. Combination of such features provide robustness against partial occlusion and cluttered backgrounds. However, the appearance of the features can be distorted by large geometric and significant illumination changes. Recently, there have been some exciting breakthroughs in addressing the problem of wide-baseline feature matching. For instance, feature detectors and descriptors such as SIFT (Lowe, 2004) and SURF (Bay et al., 2006) and affine covariant feature detectors (Matas et al., 2002; Mikolajczyk and Schmid, 2004; Tuytelaars and Van Gool, 2004) have been shown maturity in real applications. (Mikolajczyk et al., 2005) showed that a characterization of affine covariant regions with the SIFT descriptor can cope with the geometric and photometric deformations between wide baseline images. More recently, (Lepetit and Fua, 2006) introduced a very robust feature matching in real-time. They treat the matching

problem as a classification problem using Randomized Trees.

In this paper, we propose a framework to alleviate some of the limitations of these approaches in order to make them more scalable for large industrial environments. In our applications, both 3D models and several training images are available or can be created easily during an off-line process. The key idea is to use the underlying 3D information to limit the number of hypothesis reducing the effect of the complexity of the 3D model on the run-time matching performance.

3 TRACKING INITIALIZATION

3.1 The approach

The goal is to automatically detect objects and recover their pose for arbitrary images. The proposed object detection approach is based on two stages: A learning (or training) stage which is done off-line and the matching and pose estimation stage at run-time. The entire learning and matching processes are fully automated and unsupervised. In the training phase, a compact appearance and geometric representation of the target object is built. In the second phase, when the initialization is invoked, an image is processed for detecting the target object using the representation built in the training phase, enforcing both photometric and geometric consistency constraints.

3.2 The training phase

Given a set of calibrated training images, in the first step of the learning stage a set of stable feature regions are selected from the object by analyzing a) their detection repeatability and accuracy from different viewpoints and b) their distinctiveness, i.e. how distinguishable the detected object regions are.

3.2.1 Feature extraction and evaluation

First, affine covariant features are extracted from the images. In our experiments we use a set of state-of-the-art affine region detectors (Mikolajczyk et al., 2005). Region detection is performed in two major steps. First interest points in scale-space are extracted, e.g. Harris corners or blob like structures by taking local extrema of a Difference-of-Gaussian pyramid. In the second step for each point an elliptical region is determined in an affine invariant way. See (Mikolajczyk et al., 2005) for more detail and a comparison of affine covariant region detectors. An accelerated version of the detectors can be achieved by using libraries such as Intel's IPP. Next, the intensity pattern within each feature region is described with the SIFT descriptor (Lowe, 2004) representing the local orientation histograms. The SIFT descriptor has been shown to be very effective on a number of measures (Mikolajczyk and Schmid, 2005). Given a sequence of images, the elliptical feature regions are detected independently in each viewpoint. The corresponding image regions cover approximately the same object surface region (*model region*). The, the feature regions are normalized against the geometric and photometric deformations and described by the SIFT descriptor to obtain a viewpoint and illumination invariant description of the intensity patterns within each region. Hence, each model region is represented by a set of descriptors from multiple views (*multiple view descriptors*). However, for a better statistical representation of the variations in the multiple view descriptors each model region needs to be sampled from different viewpoints. Since not all samplings can be covered by the limited number of training images, synthesized views are created from other viewpoints using computer graphics rendering techniques. Having a set of calibrated images and the model of the target object, the viewing space is coarsely sampled at discrete viewpoints and a set of synthesized views are generated.

Once all the features are extracted, we select the most stable and distinctive feature regions which are characterized by their detection and descriptor performance. The evaluation of the detection and descriptor performance is done as following. The detection performance of a feature region is measured by the detection repeatability and accuracy. The basic measure of accuracy and repeatability is based on the relative amount of overlap between the detected regions in each image and the respective reference regions projected onto that image using the ground truth transformation. The reference regions are determined from the parallel views to the corresponding model region. The performance of the descriptor is measured by the matching criterion, i.e. how well the descriptor represents an object region. This is determined by comparing the number of corresponding regions obtained with the known ground truth and the number of correctly matched regions.

Depending on the 3D structure of the target object a model region may be clearly visible only from certain viewpoints in the scene. Therefore, we create for each model feature a *similarity map* which represents its visibility distribution by comparing it with the corresponding extracted features. As a similarity measure, we use the Mahalanobis distance between the respective feature descriptors. For each model region, a *visibility map* is the set of its appearances in the views from all possible viewpoints. Based on the computed similarity values of each model region, we cluster groups of viewpoints together using the mean-shift algorithm (Comaniciu and Meer, 2002).

The clustered viewpoints for a model region m_j are $W(m_j) = \{v_{j,k} \in \mathbb{R}^3 | 0 < k \leq N_j\}$, where $v_{j,k}$ is a viewpoint of that region.

3.2.2 Learning the statistical feature representation

This section describes a method to incorporate the multiple view descriptors into our statistical model. To minimize the impact of variations of illumination, especially between the real and synthesized images, the descriptor vectors are normalized to unit magnitude. Two different method have been tested for the feature representation: PCA and Randomized Trees. We use the PCA-SIFT descriptor (Ke and Sukthankar, 2004) for a compact representation using the first 32 components. The descriptor vectors $g_{i,j}$ are projected into the feature space to a feature vector $e_{i,j}$. For each region, we take k samples from the respective views so that the distribution of their feature vectors $e_{i,j}$ for $0 < j \leq K$ in the feature space is Gaussian. To ensure the Gaussian distribution of the vectors for each set of viewpoints $W(m_j)$ we apply the χ^2 test for a maximal number of samples. If the χ^2 test fails after a certain number of samplings for a region, the region will be considered as not reliable enough and will be excluded. For each input set of viewpoints, we then learn the covariance matrix and the mean of the distribution.

In the case of Randomized Trees, we use a random set of features to build the decision trees similar to (Lepetit and Fua, 2006). At each internal node, a set of tests involving comparison between two descriptor components are randomly drawn. The tests that result in maximum information gain is chosen as the splitting criteria at the node. This estimate is the conditional distribution over the classes given that a feature reaches that leaf. Furthermore, to reduce the variances of the conditional distribution estimates, multiple Randomized Trees are trained. It has been shown by (Lepetit and Fua, 2006) that the same tree structure could be used, with good performance for a different object, by only updating the leaf nodes. Our experiments confirm this observation.

3.3 The matching and pose estimation phase

At run-time features are first extracted from the test image in the same manner as in the learning stage and their descriptor vectors are computed. Matching is the task of finding groups of corresponding pairs between the regions extracted from test image and the feature database, that are consistent with both appearance and geometric constraints. This *matching problem* can be formulated as a classification problem. The goal is

to construct a classifier so that the misclassification rate is low. We have tested two different classification techniques, PCA-based Bayesian classification and Randomized Trees (RDT).

In the first method, the descriptors are projected into feature space using PCA. We then use the Bayesian classifier to decide whether a test descriptor belongs to a view set class or not. Let $C = \{C_1, ..., C_N\}$ be the set of all classes representing the view sets and let F denote the set of 2D-features $F = \{f_1, ..., f_K\}$ extracted from the test image. Using the Bayesian rule the *a posteriori probability* $P(C_i|f_j)$ for a test feature f_j that it belongs to the class C_i is calculated as

$$P(C_i|f_j) = \frac{p(f_j|C_i)P(C_i)}{\sum_{k=1}^{N} p(f_j|C_k)P(C_k)}. \qquad (1)$$

For each test descriptor the a posteriori probability of all classes is computed and the best candidate matches are selected using thresholding. Let $m(f_j)$ be the respective set of most probable potential matches $m(f_j) = \{C_i|P(C_i|f_j) \geq T\}$. The purpose of this threshold is only to accelerate the runtime matching and not to consider matching candidates with low probability. However this threshold is not crucial for the results of pose estimation.

In the second method using RDTs the feature descriptor is dropped down each tree independently. The average distribution amongst those stored in all reached leaf nodes is used to classify the input feature, utilizing maximum a posteriori estimation.

Once initial matches are established the respective visibility maps are used as geometric consistency constrains to find subsets of $N \geq 3$ candidate matches for pose estimation as described in (Najafi et al., 2006). This way the search space is further constrained compared to a plain or "exhaustive" of RANSAC where all pairs of candidate matches are examined. Furthermore, contextual information, such as flexible feature templates (Li et al., 2005) or shape context (Belongie et al., 2002), can be used for further local geometric constraints.

In the case of using RDTs for classification, the estimated pose can further be used to update the decision Trees to the actual viewing conditions at runtime (Boffy et al., 2006) . Our experiments showed that this can significantly improve the matching performance and makes it more robust to illumination changes, such as cast shadows or reflections.

In our experiments the RDT-based classification has been shown to have slightly better classification rate than the PCA-based one. Our non-optimized implementation needs about 200 ms for the PCA based and 120 for RDT-based approach. To evaluate the
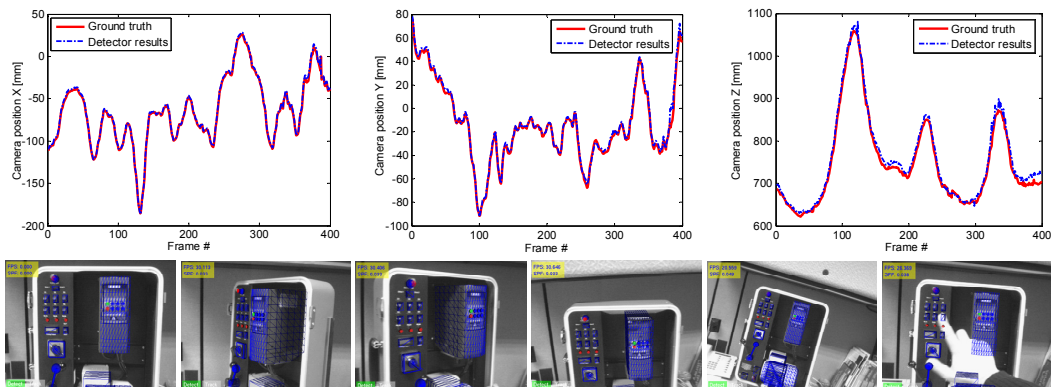
Figure 1: Comparison of pose estimation results against ground truth. The solid red line in each plot represents the true value of the three coordinates of the camera position and the dashed blue line depicts the estimated one by our detection algorithm for each frame of a long sequence which includes strong viewpoint and scale changes as well as occlusions (bottom row).

robustness and accuracy of our detection system we used long sequences taken from a control panel. Each frame was then used separately as input image to detect and estimate the pose of the panel. Fig. 1 depicts the pose estimation results compared to the ground truth data created by tracking the object using RealViz^{tm} software. The median error in camera position is about 2.0 cm and about 10 degrees in camera rotation.

4 VISUAL TRACKING

4.1 Theoretical Background

In general, the CAD models used in the industry are stored as polygon meshes that consist in sets of vertices and polygons defining the 3D shape of the objects. These meshes are made of simple convex polygons and, in most of the industrial 3D models, they are made of triangles. Consequently, it is legitimate to consider that the image contains the projection of a piecewise-planar environment since every triangular face defines a plane. For each plane in the scene, there exists a ($3{\times}3$) homography matrix $\mathbf{G}$ that links the coordinates $\mathbf{p}$ of a certain point in a reference image $\mathscr{I}^*$ to its corresponding point $\mathbf{q}$ in the current image $\mathscr{I}$. We suppose that the "image constancy assumption" is verified, i.e., after an image intensity normalization (that makes the images insensitive to linear illumination changes), we have: $\mathscr{I}^*(\mathbf{p}) = \mathscr{I}(\mathbf{q})$. The homography matrix is defined up to a scale factor. In order to fix the scale, we choose $\det(\mathbf{G}) = 1$. Let $\mathbf{w}(\mathbf{G}) : \mathbb{R}^2 \to \mathbb{R}^2$ be the coordinate transformation (a warping) such that: $\mathbf{q} = \mathbf{w}(\mathbf{G})(\mathbf{p})$. The camera displacement between two views can be represented by a ($4{\times}4$) matrix $\mathbf{T} \in \mathbb{SE}(3)$ (the Special Euclidean group): $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$, where $\mathbf{R}$ is

the rotation matrix and $\mathbf{t}$ is the translation vector of the camera between the two views. The 2D projective transformation $\mathbf{G}$ is similar to a matrix $\mathbf{H}$ that depends on $\mathbf{T}$: $\mathbf{G}(\mathbf{T}) = \mathbf{KH}(\mathbf{T})\mathbf{K}^{-1}$, where $\mathbf{K}$ is the upper triangular matrix containing the camera intrinsic parameters. The matrix $\mathbf{H}(\mathbf{T})$ can be written as: $\mathbf{H}(\mathbf{T}) = (\mathbf{R} + \mathbf{tn}^{*\top}) / (\sqrt[3]{1 + \mathbf{t}^\top \mathbf{Rn}^*})$, where $\mathbf{n}^*$ is the normal vector to the plane and $\|\mathbf{n}^*\| = d^*$ where d^* is the distance of the plane to the origin of the reference frame. Now, let $\mathbf{A}_i$, with $i \in \{1, ..., 6\}$, be a basis of the Lie algebra $\mathfrak{se}(3)$ (the Lie algebra associated to the Lie Group $\mathbb{SE}(3)$). Any matrix $\mathbf{A} \in \mathfrak{se}(3)$ can be written as a linear combination of the matrices $\mathbf{A}_i$: $\mathbf{A}(\mathbf{x}) = \sum_{i=1}^{6} x_i \mathbf{A}_i$, where $\mathbf{x} = [x_1, ..., x_6]^\top$ and x_i is the i-th element of the base field. The exponential map links the Lie algebra to the Lie Group: $\exp : \mathfrak{se}(3) \to \mathbb{SE}(3)$. Consequently, a homography matrix $\mathbf{H}$ is a function of $\mathbf{T}$ that can be locally parameterized as: $\mathbf{H}(\mathbf{T}(\mathbf{x})) = \mathbf{H}(\exp(\mathbf{A}(\mathbf{x})))$.

4.2 Problem statement

Let $q = nm$ be the total number of pixels in some image region corresponding to the projection of a planar patch of the scene in the image $\mathscr{I}^*$. This image region is called the reference template. To track the template in the current image $\mathscr{I}$ is to find the transformation $\overline{\mathbf{T}} \in \mathbb{SE}(3)$ that warps a pixel of that region in the image $\mathscr{I}^*$ into its correspondent in the image $\mathscr{I}$, i.e. find $\overline{\mathbf{T}}$ such that $\forall i$, we have: $\mathscr{I}\left(\mathbf{w}\left(\mathbf{G}(\overline{\mathbf{T}})\right)(\mathbf{p}_i)\right) = \mathscr{I}^*(\mathbf{p}_i)$. For each plane in the scene, we consider its corresponding template and its homography. Here, for the sake of simplicity, we describe the computations for a single plane. Knowing an approximation $\widehat{\mathbf{T}}$ of the transformation $\overline{\mathbf{T}}$, the problem is to find the incremental transformation $\mathbf{T}(\mathbf{x})$ such that the dif-

ference between the current image region $\mathscr{I}$ warped with $\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$ and the reference image $\mathscr{I}^*$ is null, i.e. $\forall i \in \{1, ..., q\}$: $y_i(\mathbf{x}) = \mathscr{I}\left(\mathbf{w}(\mathbf{G}(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})))(\mathbf{p}_i)\right) - \mathscr{I}^*(\mathbf{p}_i) = 0$. Let $\mathbf{y}(\mathbf{x})$ be the $(q \times 1)$ vector containing the image differences: $\mathbf{y}(\mathbf{x}) = [y_1(\mathbf{x}), ..., y_q(\mathbf{x})]^\top$. Then, the problem consists of estimating $\mathbf{x} = \widetilde{\mathbf{x}}$ verifying the system:

$$\mathbf{y}(\widetilde{\mathbf{x}}) = \mathbf{0} \qquad (2)$$

The system is then non linear and needs to be solved iteratively.

4.3 The ESM algorithm

Let $\mathbf{J}(\mathbf{x})$ be the $(q \times 6)$ Jacobian matrix describing the variation of the vector $\mathbf{y}(\mathbf{x})$ with respect to the Cartesian transformation parameters $\mathbf{x}$. Let $\mathbf{M}(\mathbf{x}_1, \mathbf{x}_2)$ be the $(q \times 6)$ matrix that verifies $\forall (\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^6 \times \mathbb{R}^6$: $\mathbf{M}(\mathbf{x}_1, \mathbf{x}_2) = \nabla_{\mathbf{x}_1} (\mathbf{J}(\mathbf{x}_1)\mathbf{x}_2)$. The Taylor series of $\mathbf{y}(\mathbf{x})$ about $\mathbf{x} = \widetilde{\mathbf{x}}$ can be written:

$$\mathbf{y}(\widetilde{\mathbf{x}}) \approx \mathbf{y}(\mathbf{0}) + \left(\mathbf{J}(\mathbf{0}) + \frac{1}{2}\mathbf{M}(\mathbf{0}, \widetilde{\mathbf{x}})\right)\widetilde{\mathbf{x}} = \mathbf{0} \qquad (3)$$

Although the optimization problem can be solved using 1st-order methods, we use an efficient 2nd-order minimization algorithm (Benhimane and Malis, 2006). Indeed, with little extra computation, it is possible to perform a stable and local quadratic convergence when the 2nd-order approximation is verified: for $\mathbf{x} = \widetilde{\mathbf{x}}$, the system (2) can then be written: $\mathbf{y}(\widetilde{\mathbf{x}}) \approx \mathbf{y}(\mathbf{0}) + \frac{1}{2}(\mathbf{J}(\widetilde{\mathbf{x}}) + \mathbf{J}(\mathbf{0}))\widetilde{\mathbf{x}} = \mathbf{0}$. The update $\widetilde{\mathbf{x}}$ of the 2nd-order minimization algorithm can then be computed as follows: $\widetilde{\mathbf{x}} = -2(\mathbf{J}(\mathbf{0}) + \mathbf{J}(\widetilde{\mathbf{x}}))^+\mathbf{y}(\mathbf{0})$. Thanks to the simultaneous use of the reference and the current image derivatives, and thanks to the Lie algebra parameterization, the ESM algorithm improves the convergence frequency and the convergence rate of the standard 1st-order algorithms (see (Benhimane and Malis, 2006) for more details).

4.4 Tracking multiple faces and speeding up the system

In the formulation of the ESM algorithm, it is implicitly assumed that all reference faces come from the same image and they were extracted in the same coordinate system. However, this is not true in practice since two faces can be seen for the first time under different poses. This problem is solved by introducing the pose $\widetilde{\mathbf{T}}_j$ under which the face j was extracted the first time into the algorithm. This leads to a different system from the one described in equation (2) and it becomes: $\sum_j \mathbf{y}_j(\widetilde{\mathbf{x}}) = \mathbf{0}$ where $\mathbf{y}_j(\mathbf{x})$ is the $(q_j \times 1)$ vector containing the image differences of the face j: $\mathbf{y}_j(\mathbf{x}) = [y_{j1}(\mathbf{x}), ..., y_{jq_j}(\mathbf{x})]^\top$ and $\forall i \in [1, q_j]$:

$y_{ji} = \mathscr{I}\left(\mathbf{w}\left(\mathbf{G}\left(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\widetilde{\mathbf{T}}_j^{-1}\right)\right)(\mathbf{p}_i)\right) - \mathscr{I}^*(\mathbf{p}_i)$. This modification was proposed in (Ladikos et al., 2007).

In order to speed up the tracking process, the templates are reduced a few times in size by a factor of two to create a stack of templates at different scales. The optimization starts at the highest scale level (lowest resolution) and the cost function is minimized on this level until convergence is achieved. This is continued until the lowest scale level (highest resolution) is reached.

5 TEMPLATE MANAGEMENT

5.1 Simplifying the 3D model

The ESM algorithm can be used to track any industrial object since every model can be described by a set of triangular faces (each face defining a plane). However, the number of the faces grows very quickly with the model complexity. For instance, the model shown in Figure 2(a) is described by a mesh of more than $20,000$ faces while the number of planes that can be used for the tracking process is much smaller. In addition, given the camera position, the visibility check of the faces can not be processed in real-time.

For these reasons, we transform the original CAD model in order to simplify the triangulated mesh, so that the largest connected faces are obtained as polygons. This procedure reduces the amount of faces to work on by around 75% in our examples.

5.2 Building the BSP Trees

Using the simplified model, we construct a 3D Binary Space Partioning tree (BSP tree) that allows to determine the visible surfaces of the considered object given the camera position. The BSP tree is static, image size invariant and it depends uniquely on the 3D model so it can be processed off-line once and for all. The processing time for building a BSP tree is fast since it is just $O(N)$, where N is the number of faces in the model. For more details about BSP trees, refer to (Foley et al., 1990; Schneider and Eberly, 2002). From any relative position between the camera and the object, the visible faces can be automatically determined. In fact, the BSP tree is used to determine the order of all faces of the model with respect to the optical axis toward the camera. Backfaces and too small faces are discarded because they do not provide usable information to the tracking process. From the z-ordered sequence of faces, we obtain a list of those, which are not occluded by others. Once a face is determined to be occluded, it has never to be checked again, so even if the worst case of this occlusion culling is $O(N^2)$, the amortized cost are supposed to be around $O(N)$, although this was not proved.

To deal with partial occlusions a face is considered to be occluded, if more than 5% of its area is occluded by other faces. This calculation is done by 2D polygon intersection test. This task is done using 2D BSP trees, which makes it possible to determine the intersection in just $O(E1 + E2)$, where $E1$ and $E2$ are the number of edges of the two faces to intersect.

5.3 Template update and database management

The automatic extraction of the *trackable* faces leads to the question, whether this can be done during the tracking procedure to update the faces and increase the number of the images that can be tracked from the reference images without any update. The naive approach to re-extract the faces leads to error integration and severe drifts. To prevent this, we used a two-step approach. First, when a face j_1 is seen for the first time, it will be stored in a Database. For each face exists two templates. The one in the Database (called reference template), referring to the first time of extraction and the current one, referring to the last time of extraction. The face is first tracked with the current template which leads to a camera pose parameters $\mathbf{x}_1$. Then, it is tracked with the reference template starting at $\mathbf{x}_1$ which leads to camera pose parameters $\mathbf{x}_2$. The second step is to examine the difference in camera motion $\|\mathbf{x}_1 - \mathbf{x}_2\|$. If it is below a given threshold, the reference template in the Database is updated by the current one (see (Matthews et al., 2003) for more details). The procedure of determining visible faces and updating the templates are done every 0.5 second. Their processing time is about 40ms. Therefore, one image has to be dropped (if the acquisition framerate is 25 fps).

6 EXPERIMENTAL RESULTS

In this paragraph, we describe experimental results showing the result of the selected detection and tracking algorithms. The experiments are done in a real industrial scenario (see Figure 2). Given the CAD model of an industrial machine (see Figure 2(a)) and few calibrated images (see Figure 2(b)), the objective is to estimate the pose of a HMD camera worn by a worker and observing the scene in real-time. This makes it possible to overlay virtual objects containing useful information into the scene in order to help the training and the maintenance processes for the workers. In this experiment, we overlay a colored 3D model of the object (see Figure 2(c)). The acquisition framerate is at 25 frames per second, and the image dimensions are (640×480). The object used is poor in texture and composed of a material with non Lam-

bertian surfaces (and therefore highly sensitive to the lightening conditions). It shows the benefit of the selected approaches and the intrinsic robustness of the algorithms to the illumination changes.

(a) The 3D model   (b) The keyframe   (c) The overlay

Figure 2: The industrial machine. See text.

The method proposed in this paper allows to handle big interframe displacements and permits to overlay more accurately the CAD model of the machine on the acquired images along the whole sequence. In the Figure 3, despite the lack of texture on the considered object, we can see that the camera pose has been accurately estimated, even when very few parts are visible. Thanks to the fast convergence rate and to the high convergence frequency of the 2nd-order minimization algorithm and to thanks to template management algorithm, the augmentation task is correctly performed. Thanks to the tracking initialization that was invoked each time the object was out of the camera field of view, the pose estimation given by the complete system was possible each time that the object is in the image and it was precise enough to nicely augment the observed scene with virtual information (see the video provided as additional material).

7 CONCLUSION

Real-time 3D tracking of complex machines in industrial environments has been shown to be a challenging task. While frame-based tracking systems can achieve high precision using temporal continuity constraints they need an initialization and usually suffer from robustness against occlusions, motion blur and abrupt camera motions. On the other side object detection systems using wide baseline matching techniques tend to be robust but not accurate and fast enough for real-time applications. This paper presents a fully automated system to overcome these problems by combining a real-time tracking system with a fast object detection system for automatic initialization. This allows us to get robustness from object detection, and at the same time accuracy from recursive tracking. For the automatic initialization, we build a scalable and compact representation of the object of interest during a training phase based on statistical learning techniques, in order to achieve speed and reliability at run-time by imposing both geometric and photo-
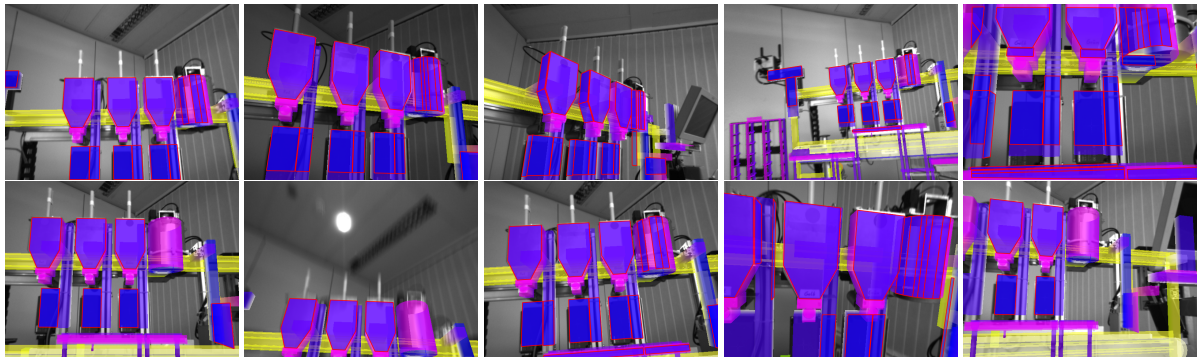
Figure 3: Excerpts from the test sequence showing the result of the tracking system proposed and the quality of the augmentation achieved. The trackable regions selected by the template management algorithm are the ones with red borders.

metric consistency constraints. Furthermore, we propose a tracking approach along with a novel template management algorithm which makes it more scalable and applicable to complex objects under severe illumination conditions. Experimental results showed that the system is able to successfully detect and accurately track industrial machines in real-time.

REFERENCES

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: a unifying framework. *IJCV*, 56(3):221–255.

Baker, S., Patil, R., Cheung, K., and Matthews, I. (2004). Lucas-kanade 20 years on: Part 5. Technical Report CMU-RI-TR-04-64, Robotics Institute, CMU.

Bay, H., Tuytelaars, T., and Van Gool, L. J. (2006). Surf: Speeded up robust features. In *ECCV*, pages 404–417.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522.

Benhimane, S. and Malis, E. (2006). Integration of euclidean constraints in template-based visual tracking of piecewise-planar scenes. In *IROS*, pages 1218–1223.

Boffy, A., Tsin, Y., and Genc, Y. (2006). Real-time feature matching using adaptive and spatially distributed classification trees. In *BMVC*.

Buenaposada, J. and Baumela, L. (2002). Real-time tracking and estimation of planar pose. In *ICPR*.

Cobzas, D. and Jagersand, M. (2004). 3D SSD tracking from uncalibrated video. In *Proc. of Spatial Coherence for Visual Motion Analysis, in conjunction with ECCV*.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *PAMI*, 24(5):603–619.

Foley, J. D., van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer graphics: principles and practice*. Addison-Wesley Longman Publishing Co., Inc.

Hager, G. and Belhumeur, P. (1998). Efficient region tracking with parametric models of geometry and illumination. *PAMI*, 20(10):1025–1039.

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. In *CVPR*, pages 506–513.

Ladikos, A., Benhimane, S., and Navab, N. (2007). A real-time tracking system combining template-based and feature-based approaches. In *VISAPP*.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *PAMI*, 28(9):1465–1479.

Li, Y., Tsin, Y., Genc, Y., and Kanade, T. (2005). Object detection using 2d spatial ordering constraints. In *CVPR*.

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *BMVC*.

Matthews, I., Ishikawa, T., and Baker, S. (2003). The template update problem. In *BMVC*.

Mikolajczyk, K. and Schmid, C. (2004). Scale and affine invariant interest point detectors. *IJCV*, 60(1):63–86.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *PAMI*, 27(10).

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A. C., Matas, J., Schaffalitzky, F., Kadir, T., and Van Gool, L. (2005). A comparison of affine region detectors. *IJCV*, 65(1-2):43–72.

Najafi, H., Genc, Y., and Navab, N. (2006). Fusion of 3D and appearance models for fast object detection and pose estimation. In *ACCV*, pages 415–426.

Schneider, P. J. and Eberly, D. (2002). *Geometric Tools for Computer Graphics*. Elsevier Science Inc.

Sepp, W. and Hirzinger, G. (2003). Real-time texture-based 3-d tracking. In *DAGM Symposium*, pages 330–337.

Tuytelaars, T. and Van Gool, L. (2004). Matching widely separated views based on affine invariant regions. *IJCV*, 59(1):61–85.