Robust 3D Object Tracking from Monocular Images using Stable Parts

Alberto Crivellaro, Mahdi Rad, Yannick Verdie, Kwang Moo Yi, Pascal Fua, *Fellow, IEEE*, and Vincent Lepetit

Abstract—We present an algorithm for estimating the pose of a rigid object in real-time under challenging conditions. Our method effectively handles poorly textured objects in cluttered, changing environments, even when their appearance is corrupted by large occlusions, and it relies on grayscale images to handle metallic environments on which depth cameras would fail. As a result, our method is suitable for practical Augmented Reality applications including industrial environments. At the core of our approach is a novel representation for the 3D pose of object parts: We predict the 3D pose of each part in the form of the 2D projections of a few control points. The advantages of this representation is three-fold: We can predict the 3D pose of the object even when only one part is visible; when several parts are visible, we can easily combine them to compute a better pose of the object; the 3D pose we obtain is usually very accurate, even when only few parts are visible. We show how to use this representation in a robust 3D tracking framework. In addition to extensive comparisons with the state-of-the-art, we demonstrate our method on a practical Augmented Reality application for maintenance assistance in the ATLAS particle detector at CERN.

Index Terms—3D Detection, 3D Tracking

1 Introduction

Methods for 3D object detection and tracking have undergone impressive improvements in recent years [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. However, each of the current approaches has its own weaknesses: Many of these approaches [1], [3], [9], [13] rely on a depth sensor, which would fail on metallic objects or outdoor scenes; methods based on feature points [6], [8] expect textured objects; those based on edges [4], [7] are sensitive to cluttered background; most of these methods [3], [2], [5], [11], [15], [10], [12] are not robust to occlusion. We also want a method fast enough for interactive 3D applications.

As Fig. 1 shows, we are interested in scenes exhibiting the conditions of real-world Augmented Reality applications, that is, scenes with poorly textured objects that are possibly visible only through heavy occlusions, drastic light changes, and changing background. A depth sensor is not an option in our setup, as the target objects often have specular surfaces. Feature point-based methods are also prone to fail because of the lack of texture and the ambiguous, repetitive patterns present in the scene.

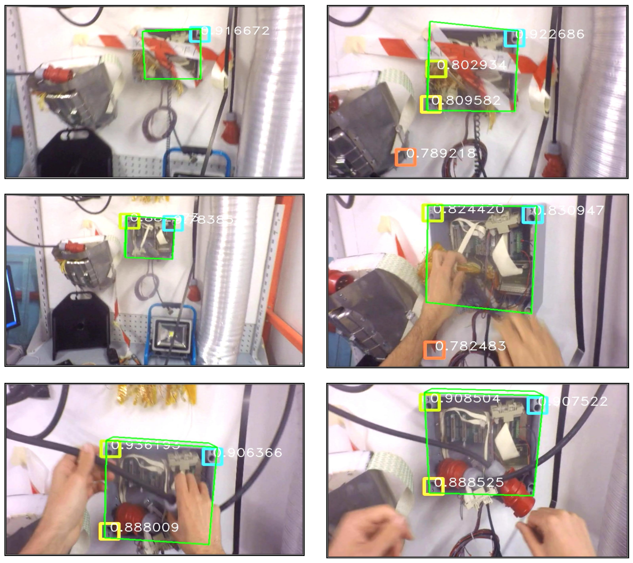


Fig. 1. Our method in action during a demonstrative technical intervention at CERN, Geneva. Detected parts are shown as colored rectangles. The appearance of the scene constantly changes and undergoes heavy occlusions. Despite these difficulties, we accurately estimate the 3D pose of the box, even if only one part is detected or in presence of false detections caused by the cluttered environment.[1]

At the core of our approach is the efficient detection of discriminative parts of the target object. Relying on parts for 3D object detection is not new [16], [2], [17], [12], [10]. The novelty in our approach is a powerful representation of the pose of each part.

Some previous methods use homographies [18], [16],

- *A. Crivellaro is with S&H, Milan, Italy.*
E-mail: a.crivellaro@sehitaly.com
- *K. M. Yi and P. Fua are with the Computer Vision Laboratory, IC Faculty, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne CH-1015, Switzerland.*
E-mail: firstname.lastname@epfl.ch
- *M. Rad and V. Lepetit are with the Institute for Computer Graphics and Vision, Graz University of Technology, Graz 8010, Austria.*
E-mail: lastname@icg.tugraz.at
- *Y. Verdie is with NCam-Tech, Paris, France.*
E-mail: yannick.verdie@ncam-tech.com

[1]Figures are best seen in colors.

[10] to represent a part pose, however this assumes that the object is piece-wise planar, and it is not easy to combine the homographies from several parts together to compute a better pose for the target object. Feature point-based methods simply use the 2D locations of the feature points, which wastes very useful information.

As shown in Fig. 2, we therefore represent the pose of each part by the *2D reprojections of a small set of 3D control points*. The control points are only “virtual”, in the sense they do not have to correspond to specific image features. This representation is invariant to the image location of the part and only depends on its appearance. We show that a Convolutional Neural Network [19] (CNN) can accurately predict the locations of these reprojections, as well as the uncertainty of the location estimates. We analyse in detail the theoretical underpinnings of why this representation is more effective than alternative approaches such as the direct prediction of a rotation and translation, or the prediction of 3D control points themselves; our experimental results confirm our analysis showing a substantial performance gain when employing our part representation.

Given an input image, we run a detector to locate each part on the image. We also use a CNN for this task, but another detection method could be used. We then predict the reprojections of the control points by applying a specific CNN to each hypothesis. This gives us a set of 3D-2D correspondences, from which we can compute the 3D pose of the target object with a simple robust algorithm.

This approach has several advantages:

- We do not need to assume the parts are planar, as was done in some previous work;
- we can predict the 3D pose of the object even when only one part is visible;
- when several parts are visible, we can combine them easily to compute a better pose of the object;
- the 3D pose we obtain is usually very accurate, even when only few parts (possibly a single one) are visible.

Early work on our approach was originally published in [20]. Here, we introduce several contributions, including:

- We discuss a more general algorithm for robust selections of detection candidates exploiting the pose prior;
- we introduce a more sophisticated system for evaluating the pose hypothesis;
- when tracking an object across a video sequence, we make use of an Extended Kalman filter [21] for reducing the jitter and providing smoother trajectories;
- we use a new architecture for the part detector, making use of Local Contrast Normalization for better generalization in presence of heavy light changes;
- we present new experimental results assessing the effectiveness of our method and of the innovations introduced. The dataset originally presented in [20] has been refined with supplementary manual annotations for all the object parts;
- we demonstrate our method on a real Augmented Reality application for maintenance assistance at CERN.

In the remainder of the paper, we first discuss related work in Section 2, we describe our approach in Sections 3 to 7, and we evaluate it in Section 8 on challenging datasets.

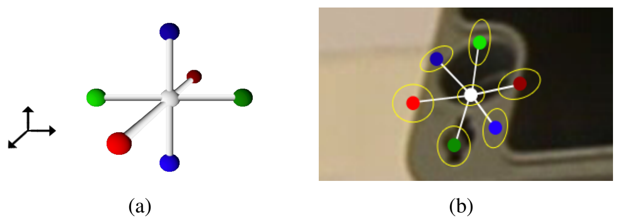


Fig. 2. Our representation of the 3D pose of an object part. (a) We consider seven 3D control points for each part, arranged to span 3 orthogonal directions. (b) Given an image patch of the part, we predict the 2D reprojections of these control points using a regressor, and the uncertainty of the predictions.

2 Related Work

3D object detection has a long history in Computer Vision, and we focus here on representative works. A well-established research direction relies on edges [22], [23], [24], but they are sensitive to large occlusions and clutter. More recently, keypoint-based methods became popular [25], [26], [27] probably because keypoints can be extracted and matched more reliably. Unfortunately, the use of keypoints is limited when the target object is poorly textured. Some works combine keypoints with edges [28], [29] or stereo information [6]. However, extracting and matching edges remains delicate, and requiring a stereo configuration limits the applicability of the 3D tracker.

Besides keypoints, silhouettes and region based methods have also been proposed. In [30], [31], 3D tracking problem is considered as joint 2D segmentation and 3D pose estimation problem, and the method looks for the pose that best segments the target object from the background. Contours and edges are used in [32] with multiple hypotheses to provide robust pose estimation. Partial occlusions, however, are difficult to handle with such approaches.

The development of inexpensive 3D sensors such as the Kinect has recently sparkled different approaches to 3D object detection. [1], [33] use votes from pairs of 3D points and their normals to detect 3D objects. [34] uses a decision tree applied to RGB-D images. [3] uses a template-based representation for dealing with poorly textured objects. Despite its robustness to clutter and light changes, according to our experimental results, this approach is sensitive to occlusions, a key-requirement in our context. The more recent [9], [11] rely on recognition of local patches. However all these methods were designed for RGB-D images, which are not an option in our target applications. Like [35], we address the problem of evaluating a pose hypothesis in a reliable way in presence of clutter and occlusions. Our solution, presented in Section 6.4, provides a fast and reliable solution without relying on depth images.

Like [36], [16], we learn 3D poses. Nonetheless, our part-based approach with our representation for the part poses allow us to be much more robust to occlusions, while such approaches are not straightforwardly generalizable to a part-based framework.

Since our approach is based on object parts, it is also related to works such as [37], [2], [10], [12] that mostly focus on category rather than instance detection. These works were mostly motivated by the success of the Deformable Part Model [38] developed for 2D detection, which was extended successfully to 3D, for example in [2]. [10] also performs 3D tracking through part-based particle filtering by integrating multi-view. [37] uses contours as parts. In [12], 3D shared parts are learned with CAD models and real images for fine pose estimation. However, these works are not robust to occlusions of some of the parts, especially because the 2D location of the part is solely considered to constrain the object pose.

Finally, a very active and related field is SLAM (Simultaneous Localization and Mapping) [39], [40], [41]. On one hand, SLAM does not require prior 3D knowledge, but on the other hand it is limited to estimate a relative pose only, which is not suitable for many Augmented Reality applications. The semi-dense approach proposed by [41] is particularly effective in handling sparsely textured surfaces; on the other hand, as other SLAM approaches, it is prone to fail on dynamic scenes and highly occluded targets.

3 Robust 3D Pose Estimation With Parts

Our goal is to estimate the 3D pose of a known rigid object with respect to a projective camera given a grayscale image of the object. We assume the internal parameters of the camera are known. Additionally, we assume that we are given a non-textured 3D model of the object, such as a triangular mesh, and a set of manually labeled parts on it. A part is simply defined as a discriminative region of the object, which can easily be detected on an input image. The object model is only used for annotating the 3D location of the parts on the object and for computing the silhouette of the object under different views, as described in Section 6.4. This allows us to use very simple models, for example a parallelepiped for an electric box, or a cylinder for a food can. We can thus neglect details that would be difficult or impossible to reconstruct, such as the interior of the electric box depicted in Fig. 1. Ideally, the parts should be spread over the object. No assumption is made about their size: usually, bigger parts are more discriminative, but smaller parts are less likely to be partially occluded. The 3D pose of the object is retrieved exclusively from its parts, while the appearance of the rest of the object can freely vary with occlusions, clutter, etc., without affecting the final result. A very small number of parts is required by our framework—in all our tests we employed at most 4 parts for an object, and, in general, our objects of interest have very few discriminative regions, so we select the parts by hand. For training our algorithm, we make use of a set of registered training images, showing the parts under different poses and lighting conditions.

symbol	meaning
i	index of a training image
p	index of a part
k	index of a control point or its projection
l	index of a detection candidate on a testing image
$\mathbf{C}_p$	3D center of the p-th part
I_i	$i-$th training image
$\mathbf{c}_{ip}$	projection of $\mathbf{C}_p$ in the i-th training image
$\mathbf{v}_{ipk}$	projection of $\mathbf{V}_{pk}$ in the i-th training image
$\hat{\mathbf{c}}_{pl}$	l-th detection candidate for the projection of $\mathbf{C}_p$ in an input image
s_{pl}	score for this detection
$\mathbf{V}_{pk}$	k-th 3D control point of the p-th part
$\hat{\mathbf{v}}_{pk}$	prediction for the projection of $\mathbf{V}_{pk}$ (no outlier)
S_{pk}	covariance for prediction for the projection of $\mathbf{V}_{pk}$ (no outlier)
$\hat{\mathbf{v}}_{pkl}$	l-th prediction for the projection of $\mathbf{V}_{pk}$ in an input image
$\mathbf{q}$	an image patch
S_q	Size of image patch $\mathbf{q}$
I	incoming image at test time
M	number of components of the Mixture-of-Gaussians pose prior
$(\overline{\mathbf{p}}_m, \mathsf{S}_m)$	average and covariance of the m-th component of the pose prior
$\hat{\mathbf{p}}^{(m)}$	pose estimated starting from the m-th component of the prior
$\hat{\mathbf{p}}$	final estimation of the pose

TABLE 1

Main notations.

After detecting several candidates for each of the parts of the target object as described in Section 4, we select the most likely candidates given a prior on the pose as explained in Section 6.2. For each selected candidate, we estimate the 3D pose of the target part (Section 5) and, if more than one part are visible, we combine the 3D poses computed for each part for estimating the pose of the target object (Section 6). Since several priors can be used at the same time, we assign a score to each of the computed poses. This score depends on several cues, and is also learned using linear regression (Sections 6.3 and 6.4). Finally, we select the pose with the best score as our final estimation. When tracking frames across a video sequence, we employ the Extended Kalman filter described in Section 7 in order to reduce the jitter and provide smoother trajectories.

4 Part Detection

The first step of our pipeline is the detection of the visible object parts on the image. Different methods could be employed for this step. Motivated by the recent success of the Convolutional Neural Networks for object detection [42], [43], [44], we use a CNN for predicting the parts locations on the image, which appears to work also well for this task.

In order to learn to detect the parts, we exploit a set of registered training images as the one shown in Fig. 3(a). We denote our training data as [2]:

$$\mathcal{T} = \left\{ \left(I_i, \left\{ \mathbf{c}_{ip} \right\}_p, \left\{ \mathbf{v}_{ipk} \right\}_{pk} \right) \right\}_i \ , \qquad (1)$$

where I_i is the i-th training image, $\mathbf{c}_{ip}$ the projection of the center $\mathbf{C}_p$ of the p-th part on I_i, and $\mathbf{v}_{ipk}$ the projection of the k-th control point of the p-th part on the image.

During an offline stage, we train a CNN with a standard multi-class architecture shown in Fig. 4 to detect the parts. The input to this CNN is a 32×32 image patch $\mathbf{q}$, its output consists of the likelihoods of the patch to correspond to

[2]The main notations are summarized in Table 1.

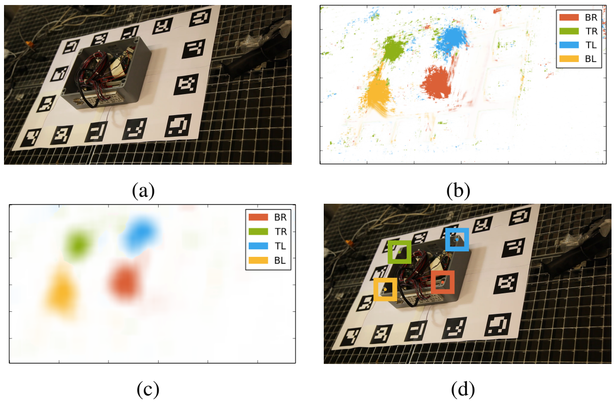


Fig. 3. Detecting the parts. (a) An input image of the box. (b) The output of the $\text{CNN}^{\text{part-det}}$ for each image location. Each color corresponds to a different part. (c) The output after Gaussian smoothing. (d) The detected parts, corresponding to the local maximums in (c).

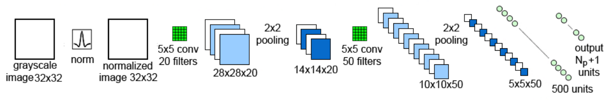


Fig. 4. Architecture of $\text{CNN}^{\text{part-det}}$ for part detection. The last layer outputs the likelihoods of the patch to correspond to each part or to the background.

one of the $N_P + 1$ parts. We train the CNN with patches randomly extracted around the centers $\mathbf{c}_{ip}$ of the parts in the training images I_i and patches extracted from the background, and by optimizing the negative log-likelihood over the parameters w of the CNN:

$$\widehat{w} = \arg\min \sum_{p=0}^{N_P} \sum_{\mathbf{q}\in\mathcal{T}_p} - \log \text{softmax}(\text{CNN}_w^{\text{part-det}}(\mathbf{q}))[p] \quad , \tag{2}$$

where $\mathcal{T}_j$ is a training set made of image patches centered on part p and $\mathcal{T}_0$ is a training set made of image patches from the background, $\text{CNN}_w^{\text{part-det}}(\mathbf{q})$ is the N_P+1-vector output by the CNN when applied to patch $\mathbf{q}$, and $\text{softmax}(\text{CNN}_w^{\text{part-det}}(\mathbf{q}))[p]$ is the p-th coordinate of vector $\text{softmax}(\text{CNN}_w^{\text{part-det}}(\mathbf{q}))$.

At run time, we apply this CNN to each 32×32 patch in the input images captured by the camera. This can be done very efficiently as the convolutions performed by the CNN can be shared between the patches [45]. As shown in Fig. 3, we typically obtain clusters of large values for the likelihood of each part around the centers of the parts. We therefore apply a smoothing Gaussian filter on the output of the CNN, and retain only the local maximums of these values as candidates for the locations of the parts.

The result of this step is, for each part p, a set $\mathcal{S}_p = \{(\hat{\mathbf{c}}_{pl}, s_{pl})\}_l$ of 2D location candidates $\hat{\mathbf{c}}_{pl}$ for the part along with a score s_{pl} that is the value of the local maxima returned by the CNN. We will exploit this score in our pose estimation algorithm described in Section 6. We typically get up to 4 detections for each part on an input image.

For better robustness to illumination changes, we normalize the images with a Difference-of-Gaussians:

$$\mathbf{q} = (G_{\sigma_2} - G_{\sigma_1}) * \mathbf{q}' \tag{3}$$

where $\mathbf{q}'$ is the original grayscale input patch before normalization, G_{σ_1} and G_{σ_2} are 2D Gaussian kernels of manually selected standard deviations σ_1 and σ_2 respectively, and $*$ the symbol for the product of convolution. We experimentally found this method to perform better than Local Contrast Normalization, which is often the normalization method used with Convolutional Neural Networks.

5 Part Pose Estimation

5.1 Representation of the Part Pose

The second step of our pipeline consists in predicting the pose of each part, starting from information about its local appearance, i.e. an image patch $\mathbf{q}$ extracted on an image I around the projection of the part center $\mathbf{c}$. More exactly, we seek a function:

$$\mathcal{P}: \ \mathbf{Q} \times \mathbb{R}^2 \longrightarrow SE(3) \tag{4}$$

that, given $\mathbf{q}$ and $\mathbf{c}$, computes the pose of the part $\mathbf{p} = \mathcal{P}(\mathbf{q}, \mathbf{c})$ on image I; $\mathbf{Q}$ and $SE(3)$ are, respectively, the space of the image patches of size $S_q \times S_q$, and the space of the 3D rigid transforms.

For a given $\mathbf{c}$, $\mathcal{P}(\cdot, \mathbf{c})$ should be insensitive to imaging changes due to noise, light conditions, etc., and it has no clear analytical form. Moreover, in order to simplify the problem, we seek for a pose representation $\mathcal{P}$ of the form:

$$\mathcal{P}(\mathbf{q}, \mathbf{c}) = \mathcal{R}(\mathcal{Q}(\mathbf{q}), \mathbf{c}), \tag{5}$$

where $\mathcal{Q}(\mathbf{q})$ is some representation of the pose of the patch *that does not depend on the position of the patch on the image*, and $\mathcal{R}$ is a function that does not depend on the patch appearance, but on the pose representation computed by $\mathcal{Q}$. To allow $\mathcal{Q}$ to account for all the appearance changes of the patch, we approximate it with a non-linear regressor.

A crucial point to address is how to define $\mathcal{Q}(\cdot)$, that is, how to choose the most suitable representation for the pose of each part. $\mathcal{Q}$ should satisfy the following constraints:

- Combining the poses of an arbitrary number of parts must be easy and efficient;
- the pose representation should be translation invariant, that is, $\mathcal{Q}(\mathbf{q})$ should not depend on the position of the patch on the image;
- since we approximate $\mathcal{Q}$ with a regressor, the pose representation should be should be tied-in with the regressor's capabilities. For example, as our experimental results show, it is very hard for a regressor to accurately estimate the scale or the depth of a part from a patch.

A priori, we can imagine several ways to represent the 3D poses of the parts:

- **Homography**: it is possible to use homographies for representing the pose of each part [18], [16], [10]. However, this assumes that the part surface is planar,

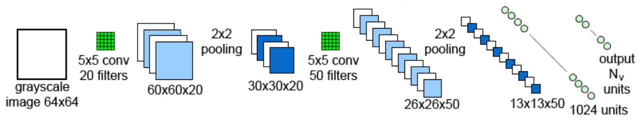


Fig. 5. Architecture of the CNN $\text{CNN}^{\text{cp-pred-}p}$ predicting the projections of the control points.

and makes it difficult to merge the individual pose estimations from the different parts.

- **3D Pose**: Another possibility is having the output of $\mathcal{Q}(\mathbf{q})$ consists of a 3D rotation and the depth value for the patch center. It is then possible to retrieve the 3D translation as well, from the location of the patch on the image and the predicted depth. However, this representation is not translation invariant in a fully perspective model. Also it is not easy to merge rotations for estimating the pose of the whole target object. Finally, this choice requires to predict the depth accurately from a single image patch, which appears to be very difficult to do accurately in our experiments.
- **3D Control points**: Since our final solution is based on 3D control points, as already mentioned, we could set the output of $\mathcal{Q}(\mathbf{q})$ to be the 3D locations of the control points in the camera reference system. In this case, estimating the pose becomes simple, since it only involves computing the rigid motion between two sets of 3D points [46]. Moreover, also combining the poses of the parts becomes a trivial task, as we simply need to compute the rigid motion between multiple sets of 3D points. However, as it will be explained in Section 8.4, this pose representation is not translation invariant under a fully perspective model, and more importantly in practice, accurately predicting the 3D points is difficult.
- **2D reprojections of 3D control points**: This is the representation we propose. The part poses are represented as the 2D reprojections of a set of 3D control points. This representation is fully translation invariant. It is straightforward to combine the poses of an arbitrary number of parts, by grouping all the 2D reprojections together and solving a PnP problem. We don't have to predict the depths or the 3D locations of the control points, which, as noted above, is very difficult to do accurately. These advantages entail a significant accuracy gain, as shown by our results in Section 8.4. The control points are purely virtual and do not correspond to any physical feature of the parts, therefore we can freely set their configuration. We evaluate different configuration in Section 8.5.

5.2 Prediction of the Reprojections of the Control Points

Once the parts are detected, we apply a regressor to the patches centered on the candidates $\hat{\mathbf{c}}_{pl}$ to predict the projections of the control points for these candidates. We also implemented this regressor as a CNN; each part has its specific CNN. As shown in Fig. 5, these networks take as input a patch of size of 64×64. The output layer is made of $2N_V$ neurons, with N_V the number of control points of the part, which predicts the 2D locations of the control points. We train each of these CNNs during an offline stage by simply minimizing over the parameters w of the CNN the squared loss of the predictions:

$$\widehat{w} = \arg\min \sum_{(\mathbf{q},\mathbf{w})\in\mathcal{V}_p} ||\mathbf{w} - \text{CNN}_w^{\text{cp-pred-}p}(\mathbf{q})||^2 \ , \qquad (6)$$

where $\mathcal{V}_p$ is a training set of image patches $\mathbf{q}$ centered on part p and the corresponding 2D locations of the control points concatenated in a $(2N_V)$-vector $\mathbf{w}$, and $\text{CNN}_w^{\text{cp-pred-}p}(\mathbf{q})$ is the prediction for these locations made by the CNN specific for part p, given patch $\mathbf{q}$ as input. At run-time, we obtain for each $\hat{\mathbf{c}}_{pl}$ candidate, several predictions $\{\hat{\mathbf{v}}_{pkl}\}$ for the control points projections. In addition, we can estimate the 2D uncertainty for the predictions, by propagating the image noise through the CNN that predicts the control point projections [21]. Let us consider the matrix:

$$\mathsf{S}_V = \mathsf{J}_{\hat{\mathbf{c}}}(\sigma \mathsf{Id})\mathsf{J}_{\hat{\mathbf{c}}}^\top = \sigma \mathsf{J}_{\hat{\mathbf{c}}}\mathsf{J}_{\hat{\mathbf{c}}}^\top \ , \qquad (7)$$

where σ is the standard deviation of the image noise (assumed to be Gaussian and to affect each image pixel independently), Id the $64^2 \times 64^2$ Identity matrix, and $\mathsf{J}_{\hat{\mathbf{c}}}$ the Jacobian of the function computed by the CNN, evaluated at the patch centered on the candidate $\hat{\mathbf{c}}$. Such a Jacobian matrix can be computed easily with a Deep Learning framework such as Theano [47] thanks to the Chain Rule, by multiplying the Jacobians of the successive layers of the network together. By neglecting the correlation between the different control points, we can compute the 2×2 uncertainty matrix S_{pk} for each control point k efficiently of part p, without having to compute the entire, and very large, product in Eq. (7):

$$\mathsf{S}_{pk} = \sigma \mathsf{J}_{\hat{\mathbf{c}}}^{pk} \mathsf{J}_{\hat{\mathbf{c}}}^{pk\top} \ , \qquad (8)$$

where $\mathsf{J}_{\hat{\mathbf{c}}}^{pk}$ is made of the two columns of $\mathsf{J}_{\hat{\mathbf{c}}}$ that correspond to the reprojection of the control point k. An example of predicted control points is shown in Fig. 2(b).

6 Object Pose Estimation

In this Section, we detail how we use the predicted reprojections to robustly estimate the object pose.

As in previous work [48], we assume that we are given a prior on the pose $\mathbf{p}$, in the form of a Mixture-of-Gaussians $\{(\overline{\mathbf{p}}_m, \mathsf{S}_m)\}$. This prior is very general, and allows us to define the normal action range of the camera. For example, the camera is unlikely to be a few centimetres from the object, or more than tens of meters away, or facing the object upside-down. Moreover, the pose computed for the previous frames can be easily incorporated within this framework to exploit temporal consistency.

In the following, we will first assume that this prior is defined as a single Gaussian distribution of mean and covariance $(\overline{\mathbf{p}}_0, \mathsf{S}_0)$. We will extend our approach to the Mixture-of-Gaussians in Section 6.3.

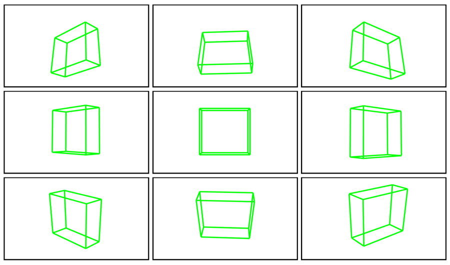


Fig. 6. Pose prior for an electric box: Projections of the box by each of the 9 Gaussians centers $\overline{\mathbf{p}}_m$.

6.1 Using a single Gaussian Pose Prior

Let us first assume there is no outlier returned by the part detection process or by the control point prediction, and that all the parts are visible. Then, the object pose $\hat{\mathbf{p}}$ can be estimated as the minimizer of $F(\mathbf{p})$, with $F(\mathbf{p}) =$

$$\frac{1}{N_P}\sum_{p,k} \text{dist}^2(\mathsf{S}_{pk}, \Gamma_{\mathbf{p}}(\mathbf{V}_{pk}), \hat{\mathbf{v}}_{pk}) + (\mathbf{p}-\overline{\mathbf{p}}_0)^\top \mathsf{S}_0^{-1}(\mathbf{p}-\overline{\mathbf{p}}_0) \quad , \tag{9}$$

where the sum is extended over all the control points of all the parts, and $\Gamma_{\mathbf{p}}(\mathbf{V})$ is the 2D projection of $\mathbf{V}$ under pose $\mathbf{p}$. $\hat{\mathbf{v}}_{pk}$ is the projection of control point $\mathbf{V}_{pk}$ and S_{pk} its uncertainty estimated as in Eq. (8). Since we assume here that there is no outlier, we dropped here the l index corresponding to the multiple detections. $\text{dist}(.)$ is the Mahalanobis distance:

$$\text{dist}^2(\mathsf{S}, \mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1-\mathbf{v}_2)^\top \mathsf{S}^{-1}(\mathbf{v}_1-\mathbf{v}_2) \quad . \tag{10}$$

$F(\mathbf{p})$ is minimized using the Gauss-Newton algorithm initialized with $\overline{\mathbf{p}}_0$. At each iteration, we update the estimated covariance of the computed pose using the Extended Kalman Filter update formula [21] when optimizing Eq. (9).

6.2 Outlier rejection for the detected parts

In practice, for the location of the p-th part, the detection procedure described in Section 4 returns a set of hypotheses $\mathcal{S}_p = \{\hat{\mathbf{c}}_{pl}\}_l$. To reject outliers in detection, we use the fact that at most one detection is correct for each part, and keep the hypotheses that are most in agreement with the pose prior: After ranking the candidates according to their score s_{pl}, we examine the best three candidates for each part and we form all the possible sets $\mathcal{C} = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_p, \ldots\}$ of detections containing at most one candidate for each part. Given the pose prior $\overline{\mathbf{p}}_0$, we evaluate the set of candidates $\mathcal{C}$ with the following steps:

1) Select two random candidates $\hat{\mathbf{c}}_{p_1}$, $\hat{\mathbf{c}}_{p_2} \in \mathcal{C}$, and translate the pose prior $\overline{\mathbf{p}}_0$ to obtain a new prior $\overline{\mathbf{p}}_0^{TS}$ that best fits $\hat{\mathbf{c}}_{p_1}$, $\hat{\mathbf{c}}_{p_2}$. More exactly, we adjust the in-plane translation such that:

$$\Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_1}) + \Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_2}) = \hat{\mathbf{c}}_{p_1} + \hat{\mathbf{c}}_{p_2} \tag{11}$$

and the off-plane component such that:

$$||\Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_1}) - \Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_{p_2})|| = ||\hat{\mathbf{c}}_{p_1} - \hat{\mathbf{c}}_{p_2}||. \tag{12}$$

2) We keep considering $\mathcal{C}$ only if all the detections it contains are consistent with the new prior. This test can be formalized as:

$$\begin{array}{ll} \forall \hat{\mathbf{c}}_p \in \mathcal{C}: & \rho_p < T^2 \\ \text{with} & \rho_p = \text{dist}^2(\hat{\mathsf{S}}_0(\mathbf{C}_p), \Gamma_{\overline{\mathbf{p}}_0^{TC}}(\mathbf{C}_p), \hat{\mathbf{c}}_p) \end{array} \tag{13}$$

where $\hat{\mathsf{S}}_0(\mathbf{C}_p) = \mathsf{J}\ \mathsf{S}_0 \mathsf{J}^\top$, with J the jacobian of $\Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_p)$, is the covariance of the projected control point $\Gamma_{\overline{\mathbf{p}}_0^{TS}}(\mathbf{C}_p)$; we set the threshold $T = 40$ pixels in all our experiments.

3) If several sets $\mathcal{C}$ pass this test, we retain the one with the largest number of detected parts. If several retained sets have the same number of points, we keep the one with the smallest average error $\overline{\rho} = \frac{1}{|\mathcal{C}|}\sum_p \rho_p$ of its points.

4) Finally, we run the Gauss-Newton optimization of Eq. (9) using the detections in the retained set to obtain a pose estimate.

If the object of interest has a single part, we simply select the detection candidate with the highest score.

6.3 Using a Mixture-of-Gaussians for the Pose Prior

In practice, the prior for the pose is in the form of a Mixture-of-Gaussians $\{(\overline{\mathbf{p}}_m, \mathsf{S}_m)\}_m$ with $M = 9$ components. The prior we use for the BOX dataset is shown in Fig. 6. We apply the method described above to each component, and obtain M possible pose estimates: $\hat{\mathbf{p}}^{(1)}, \ldots, \hat{\mathbf{p}}^{(M)}$.

6.4 Identifying the Best Pose Estimate

To finally identify the best pose estimate $\hat{\mathbf{p}}$ among the different estimates obtained with the Mixture-of-Gaussians prior, we evaluate each $\hat{\mathbf{p}}^{(m)}$ using several cues. As it is difficult to combine cues of different natures, our key idea here is to train a linear regressor to weight the contributions of the different cues and predict a penalty.

More exactly, we use the angle α and the scale difference δ_{scale} between the quaternions for $\hat{\mathbf{p}}^{(m)}$ and the corresponding component of the prior, the final value of the objective function $F(\hat{\mathbf{p}}^{(m)})$ defined in Eq. (9), and a score $\xi(\hat{\mathbf{p}}^{(m)})$ measuring the correlation between the edges in the image and the object contours after projection by $\hat{\mathbf{p}}^{(m)}$. $\xi(\hat{\mathbf{p}}^{(m)})$ is computed as:

$$\xi(\hat{\mathbf{p}}^{(n)}) = \sum_{\mathbf{x}} \left(\mathbf{n}(\mathbf{x}) \cdot [I_u(\mathbf{x}), I_v(\mathbf{x})]^\top\right) \quad , \tag{14}$$

where $\mathbf{n}(\mathbf{x})$ is the unit normal of the projected object contour at pixel $\mathbf{x}$, $I_u(\mathbf{x})$ and $I_v(\mathbf{x})$ are the partial derivatives of the incoming image I at pixel $\mathbf{x}$, and the sum is over the pixels $\mathbf{x}$ lying on the re-projected contours of the object.

Offline, we create a training set generated from the training sequence by adding noise to the ground truth poses, and computing the values of our different cues. For each sample, we compute a penalty that is the sum of the euclidean norms of the rotation and translation components of the absolute pose error [49] introduced by the noise. We can then train a linear regressor to predict this penalty given our different cues. At run-time, we simply have to use the linear regressor to predict the penalties of the pose estimates, and keep the one with the smallest penalty.

7 Tracking Frames across a Video Sequence and Pose Filtering

When tracking an object across a video sequence, if a pose is estimated for a given frame, we add it as a component of the pose prior for the next frame. This allows us to easily exploit temporal constraints. [3] Moreover, we use a Kalman Filter for reducing jitter and provide smoother trajectories.

7.1 Extended Kalman Filter for 3D Tracking

In visual tracking, Kalman Filters typically treat images as observations. However, this requires the linearisation of the imaging process with respect to the 3D pose, which can result in a poor approximation. Therefore, we chose to consider our pose estimation method as a "black box", and we treat the poses it predicts as *observations* for the filter, alleviating the need for linearisation.

7.1.1 State Vector

We model the camera motion as a first order, discrete-time dynamic system, and the state vector at time t is provided by the $(12)-$vector:

$$\mathbf{s}_t = [\mathbf{t}_t^\top,\ \mathbf{r}_t^\top,\ \mathbf{v}_t^\top,\ \omega_t^\top]^\top \ , \tag{15}$$

where $\mathbf{t}_t$ is the translation component of the camera pose, $\mathbf{r}_t$ is the exponential map representation of the rotation component of the pose, $\mathbf{v}_t$ is the linear velocity and ω_t the angular velocity. At each time step, our estimation of the system state is updated according to the available observations with the *predictor-corrector* scheme of Kalman Filters. First, the state estimate $\tilde{\mathbf{s}}_{t-1}$ and its covariance $\tilde{\mathsf{S}}_{t-1}$ are updated with a motion model to predict the state at current time $\tilde{\mathbf{s}}_{t-1}^t$ and the covariance $\tilde{\mathsf{S}}_{t-1}^t$. Then, the observation of the current state is employed for correcting the initial prediction and obtain the final state estimation $\tilde{\mathbf{s}}_t$.

7.1.2 Notations

For sake of clarity, we summarize here the notation convention of this section. For a given quantity $\mathbf{x}$, then:

- $\tilde{\mathbf{x}}_{t-1}$ is the estimate of $\mathbf{x}$ at the end of step $t-1$;
- $\tilde{\mathbf{x}}_{t-1}^t$ is the estimate of $\mathbf{x}$ at time t obtained by updating $\tilde{\mathbf{x}}_{t-1}$ according to the dynamic model;
- $\hat{\mathbf{x}}_t$ is the observed value of $\mathbf{x}$ at step t, typically the camera pose predicted by the method described above.
- $\tilde{\mathbf{x}}_t$ is the final estimate of $\mathbf{x}$ at time t, obtained correcting $\tilde{\mathbf{x}}_{t-1}^t$ according to the observation $\hat{\mathbf{x}}_t$.

[3]The covariance of the new component can be computed as explained in Section 6.1. We empirically found that this lead to very small values of the covariances of the computed poses, so the covariance of the new prior component is set as $\frac{10^{-3}}{|\tilde{\mathcal{C}}|}\ \mathsf{Id}$, where $|\tilde{\mathcal{C}}|$ is the number of parts employed for computing the pose and Id is the $(6\times 6)-$identity matrix.

7.1.3 Predictor: State Update

The state at each time step is predicted from the estimate of the state at the previous time step using the following motion model:

$$\begin{aligned}\tilde{\mathbf{t}}_{t-1}^t &= \tilde{\mathbf{t}}_{t-1} + \delta t\ \tilde{\mathbf{v}}_{t-1}\\ \tilde{\mathbf{r}}_{t-1}^t &= \log(\delta\mathbf{q}(\tilde{\omega}_{t-1})\ \cdot\ \mathbf{q}(\tilde{\mathbf{r}}_{t-1}))\\ \tilde{\mathbf{v}}_{t-1}^t &= \tilde{\mathbf{v}}_{t-1}\\ \tilde{\omega}_{t-1}^t &= \tilde{\omega}_{t-1}\ \ ,\end{aligned} \tag{16}$$

where δt is the time difference between 2 subsequent time steps, $\delta\mathbf{q}(\cdot)$ and $\mathbf{q}(\cdot)$ are unit quaternions obtained from exponential map representations with the exponential mapping, $\cdot$ denotes the quaternion multiplication and $\log(\cdot)$ maps the quaternion back to the exponential map representation. Without loss of generality, we will take $\delta t = 1$.

The covariance of the state is updated using:

$$\tilde{\mathsf{S}}_{t-1}^t = \mathsf{J}_{update}\tilde{\mathsf{S}}_{t-1}\mathsf{J}_{update}^\top + \mathsf{A}\ \ , \tag{17}$$

where J_{update} is the $(12\times 12)-$jacobian matrix of the update (16), and A is given by :

$$\mathsf{A} = \begin{bmatrix}\frac{1}{3}a\mathsf{Id} & 0 & \frac{1}{2}a\mathsf{Id} & 0\\ 0 & \frac{1}{3}b\mathsf{Id} & 0 & \frac{1}{2}b\mathsf{Id}\\ \frac{1}{2}a\mathsf{Id} & 0 & a\mathsf{Id} & 0\\ 0 & \frac{1}{2}b\mathsf{Id} & 0 & b\mathsf{Id}\end{bmatrix}, \tag{18}$$

where Id is the $(3\times 3)-$identity matrix, and a and b are 2 parameters corresponding to the incertitude about the temporal derivatives of the velocities. We empirically set $a = b = 100$ in all our experiments. Interested readers can refer to [50] and its references for further details about the derivation of matrix A.

7.1.4 Corrector: Taking into Account Observations

After computing a prediction of the current state and its covariance, we correct it taking into account our observation, the pose $\hat{\mathbf{p}}_t$. Since we cannot observe the velocities directly, their estimations would stay indefinitely stuck in the initial state if we only use the motion model of Eq. (16). To avoid this problem, we compute the velocities as:

$$\hat{\mathbf{v}}_t = (\hat{\mathbf{t}}_t - \tilde{\mathbf{t}}_{t-1}) \quad \text{and} \quad \hat{\omega}_t = \omega(\tilde{\mathbf{r}}_{t-1}, \hat{\mathbf{r}}_t)\ \ , \tag{19}$$

where the angular velocity $\omega(\mathbf{r}_1,\mathbf{r}_2)$ between 2 consecutive rotations $\mathbf{r}_1,\ \mathbf{r}_2$ is estimated as follows:

$$\begin{aligned}&\mathsf{R}_1 = \mathsf{R}(\mathbf{r}_1)\ , \qquad \mathsf{R}_2 = \mathsf{R}(\mathbf{r}_2)\ , \qquad \delta\mathsf{R} = \mathsf{R}_2\mathsf{R}_1^\top\ \ ,\\ &\theta = \operatorname{acos}\left(\frac{\operatorname{trace}(\delta\mathsf{R}) - 1}{2}\right)\ ,\Omega = \frac{\theta}{2\sin(\theta)}(\delta\mathsf{R} - \delta\mathsf{R}^\top)\ \ ,\\ &\omega(\mathbf{r}_1,\mathbf{r}_2) = [\Omega_{21}, \Omega_{02}, \Omega_{10}]^\top,\end{aligned}$$

where $\mathsf{R}(\mathbf{r})$ is the $(3\times 3)-$rotation matrix corresponding to the rotation vector $\mathbf{r}$, and $\omega = [0,\ 0,\ 0]^\top$ if $||\theta||$ is smaller than a threshold for preventing division by 0.

For the covariance of the observed state $\hat{\mathsf{S}}_t$, we employ a constant, diagonal covariance matrix. Finally, we simply have to apply standard Kalman update equations for correcting the pose estimate.

7.1.5 Initialization - Outlier Rejection

For the first frame of the video sequence, we initialize the state vector estimate with $\hat{\mathbf{p}}_t$ and null velocities. Special care must be taken in order to detect and reject outliers in the observed poses. In practice, we use the following tests:

- if an observed pose $\hat{\mathbf{p}}_t$ is not close to the last estimation $\tilde{\mathbf{p}}_{t-1}$, then it is probably an outlier and should not be taken into account;
- if 2 consecutive observed poses $\hat{\mathbf{p}}_{t-1}$ and $\hat{\mathbf{p}}_t$ are close to each other, then they are probably not outliers, even if they are far from the last pose estimate.

If the observed pose $\hat{\mathbf{p}}_t$ is detected an outlier according to these tests, we then set $\tilde{\mathbf{s}}_t := \tilde{\mathbf{s}}_{t-1}^t$. If outlier observed poses are observed for more than 3 frames in a row, we assume that tracking is lost. Tracking is then automatically re-initialized with the observed pose as soon as 2 consecutive poses are observed, sufficiently close to each other.

8 Experimental Results

In this section, after describing the datasets we use for evaluating our method in Section 8.2, we present and discuss the results of our evaluation. In Section 8.3 we assess the effectiveness of our part detection method, as well as that of the Diffference-of-Gaussians (DoG) Normalization introduced in Section 4. In Sections 8.4 and 8.5 we validate the choice of reprojections of control points for representing the pose of each part, and investigate the different possible configurations. Then, in Section 8.6 and 8.7 we present the results of an extensive comparison with other methods, showing that our approach achieves state-of-the-art performances on our challenging sequences. In Section 8.8 we describe our experiments on a recent dataset, the T-less dataset, investigating the influence of the number and shape of the object parts on the pose detection results. Finally, in Section 8.9 we present a real use case application of our method, an Augmented Reality application for maintenance assistance at CERN.

8.1 Evaluation protocol

In order to quantitatively evaluate the performances of a method on a video sequence, we compute the rotation and translation components of the absolute pose error [49] for each frame, and then trace their Cumulative Distribution Functions (CDF), as shown for example in Figures 11. The Area Under Curve (AUC) score, defined as the integral of the CDF curve between 0 and some maximum error threshold E, is reported for facilitating comparisons between methods, for example in Table 2. We select $E = 0.5$ for all our experiments and divide all the AUC scores by E for normalizing them, so that a perfect method would have a score of 1.

8.2 Datasets

We run our extensive evaluations on datasets originally introduced in [20] and on the recent T-less dataset [51] , consisting of both learning data and testing video sequences representing several non-textured, highly occluded objects.

8.2.1 Occlusion Datasets

The datasets introduced in [20] consist in non-textured CAD models, training and testing sequences for 3 objects. All the images are in the VGA resolution (640×480). For each dataset, we randomly select 3000 frames from the training images as training set. We test our approach on the following datasets:

- **BOX Dataset:** The target object for this dataset is an electric box. In the test videos, it is manipulated by a user, filled and emptied with objects, simulating, for example, a technical intervention. The training images show the box on a uniform background, with different objects inside and outside it. A CAD model is made by a simple parallelepiped. We use 4 corners of the box as parts, as shown in Fig. 8(a).
- **CAN Dataset:** The target object of this dataset is a food can. The label is completely blank, and the top of the can is specular. Distractor objects are present in the scene and large occlusions occur. Only the can lid breaks the the cylindrical symmetry of the object, making the pose estimation almost ambiguous. We use the top of the can as a single part, Fig. 8(b). A CAD model of the can is provided.
- **DOOR Dataset:** This datasets consists of one video showing a daily set-up where a non-textured door is opened and closed by a user. Despite the apparent triviality of the sequence, our tests show that it is very challenging to track the pose of the door along the full video, when it moves on a cluttered background. For this dataset, we track the 3 parts shown in Fig. 8(c), the knob, the keyhole and the lock of the door. A simple CAD model of the door is available as well.

The images of the training and testing videos of the datasets were registered using the ARUCO marker tracking tool [52]. The markers on the test sequences have been cropped or masked, so that they could not influence detection and tracking performance when testing the methods.

We also manually labelled the ground-truth locations of the detected parts for all the test video sequences of the original dataset presented in [20], so that more accurate experiments for evaluating the detector can be performed, such as those presented in Section 8.3. The manually labelled parts have also been employed for refining the ground-truth poses. Because of this, some of the experimental results presented in this work may be numerically slightly different from the ones reported in [20], although no substantial difference in the results has been detected. All the refined datasets are publicly available at `http://cvlab.epfl.ch/data/3d_object_tracking`.

8.2.2 T-less Dataset

This challenging dataset [51] is made of a collection of 21 non-textured objects, mainly plugs, plastic boxes and electric components. Non-textured CAD models, training and testing sequences are given for each object. For our tests, we employed monocular images captured with the Canon IXUS 950 IS camera. At the best of our knowledge,

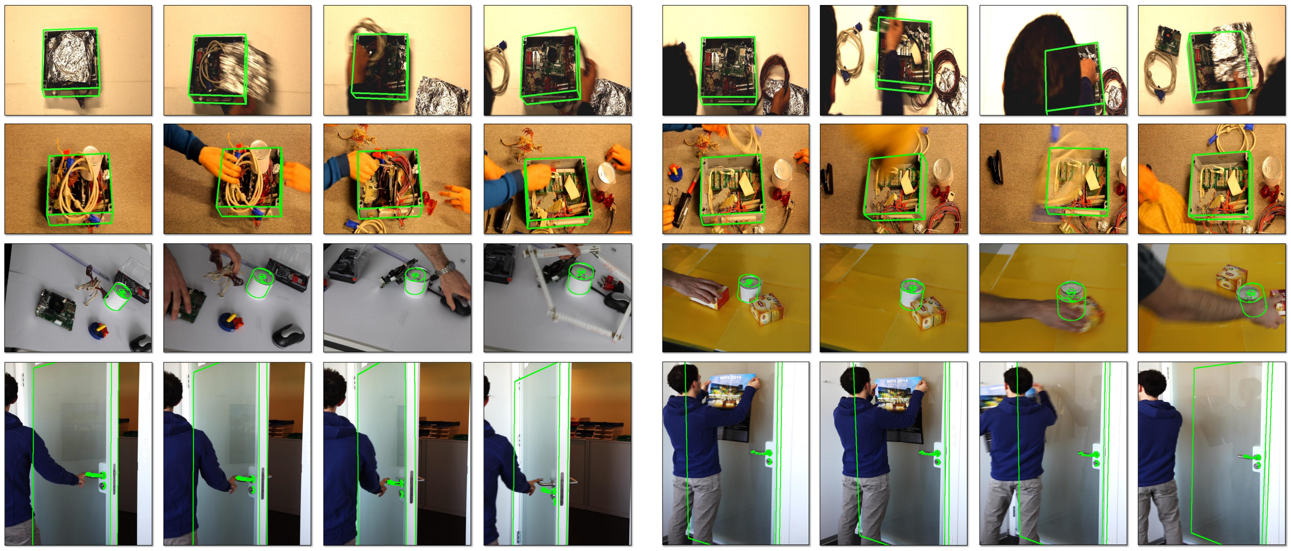


Fig. 7. Qualitative results for our challenging datasets. **Top:** We track the box despite large changes in the background and in the lighting conditions on both sequences of the BOX dataset. **Middle:** Our method correctly estimates the 3D pose of the can using the can tab only. **Bottom:** The pose of the door is retrieved starting from the door knob, the keyhole and the lock. The video sequences are provided as the supplementary material.

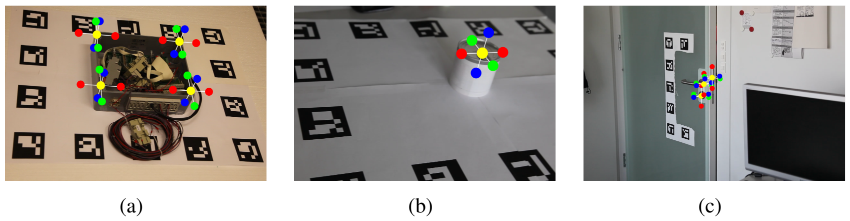


Fig. 8. Training images and control points we used for the BOX, the CAN and the DOOR datasets. The center of each part is shown in yellow. Control points are zoomed for better visualization.

this is the first work presenting results on this recent and challenging dataset. Even the simplest testing sequences, where the objects undergo under moderate occlusions over a uniform background, are challenging, because of ambiguities, lack of texture, symmetrical items, repetitive structures, and shape similarities among the objects. We tested our method on Sequence 3, which exhibits 5 non-textured objects from a uniformly sampled view hemisphere. Since the objects do not move within the scene, we employed the 4 objects shown in Figure 9,(a)-(d) as parts in our framework, considering the whole scene as target object and the fifth object as an occlusion.

8.3 Part Detection

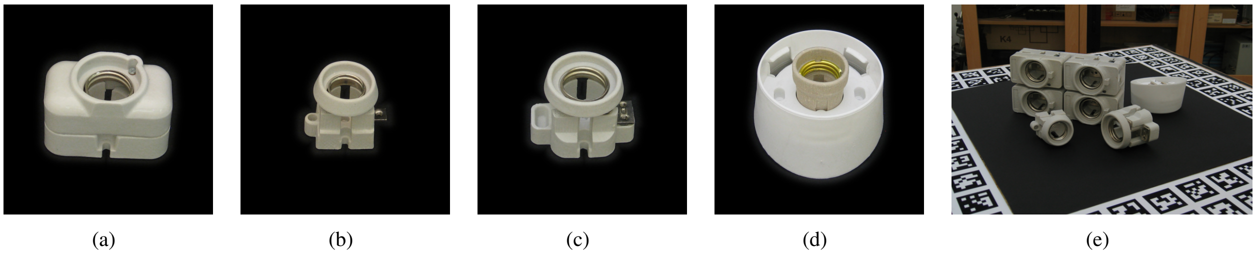
Our pipeline does not depend on a particular choice of a detector for localizing the object parts on the image. Nonetheless, the detector described in Section 4 provides an excellent trade-off between speed and accuracy: We assess here our choice by comparing it with a state-of-the-art detector, LINE-2D [3].[4] In this case, we trained an instance of LINE-2D for each part, starting from 32×32 RGB patches surrounding the part of interest. The amount of learning data was the same as for our CNN-based detector.

[4]For all the tests presented in this paper, we employed the LINE-2D implementation provided by OpenCV-2.4.12. Implementation of the authors was used for other methods employed in Section 8.6, LSD-SLAM and PWP3D.

At test time, we kept the best 4 candidates in each image for each detector and computed the detection error as the euclidean norm between the ground-truth position of the part on the image and the closest detection candidate. The CDF curves for the BOX dataset are shown in Fig. 10. We also assessed the importance of the DoG normalization introduced in Section 4. For all parts, our detector consistently outperforms LINE-2D, and the DoG normalization further increases performances in most of the cases.

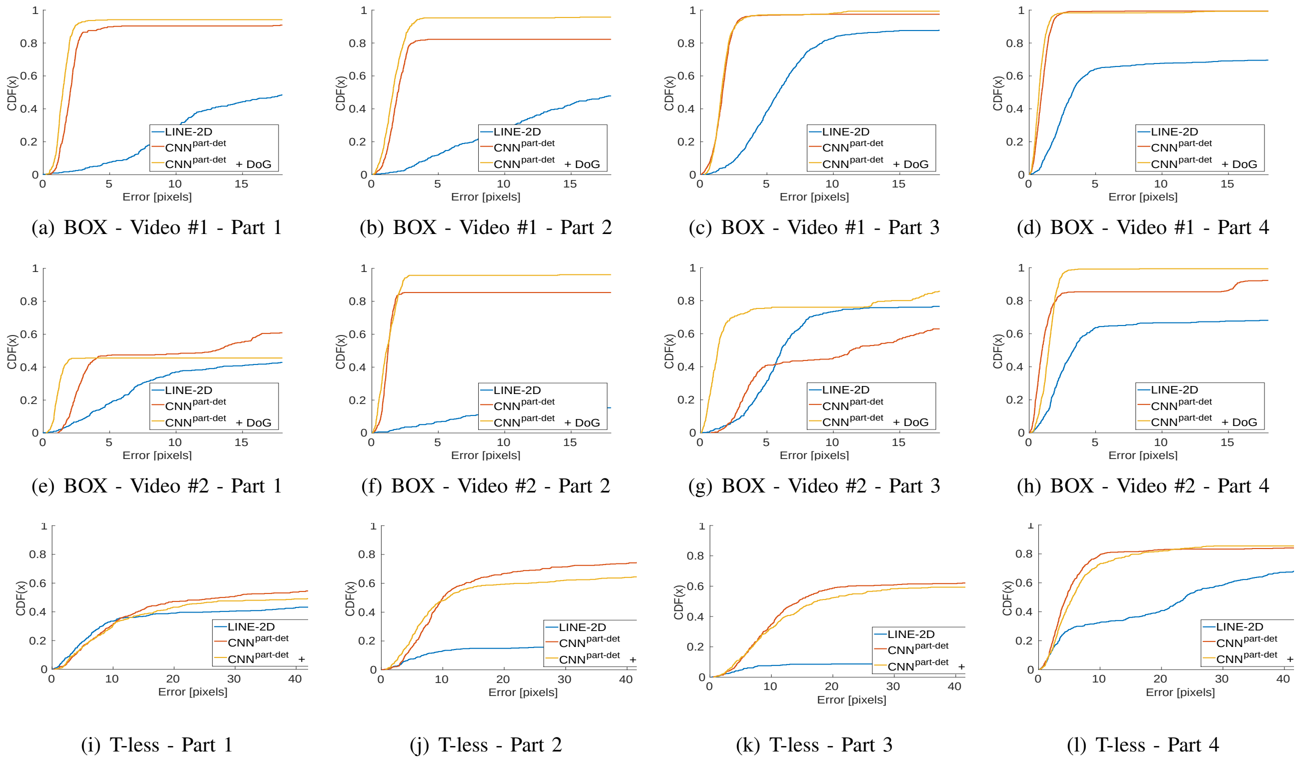
In both videos, LINE-2D performs reasonably well on the upper corners of the box—parts #3 and #4—while the accuracy for the two other corners is much lower. This is probably because in our test dataset, the edges of the upper corners are visible against a bright background and their shapes are easily recognizable. We also observed that DoG normalization is particularly effective for the Video #2, where the lighting and the background are completely different from the training videos, as opposed to the Video #1. Finally, the scores of all detectors for the Video #2, for the bottom-left corner (Part #1) is significantly lower. This is probably due to the fact that at about half of the sequence a distractor object is very close to the part, altering its appearance, and the shadow patterns change frequently around this part. Still, we can accurately predict the pose of the object because the other parts are reliably detected.

Moreover, we tested our detector on the objects of Scene 3 of the T-less dataset shown in Figure 9. Results are



(a) (b) (c) (d) (e)

Fig. 9. Parts and test sequence from the T-less dataset. (a)-(d): four items employed as parts of the scene. (e): Testing sequence. The fifth object in the scene is not employed, acting as a supplementary occlusion.



(a) BOX - Video #1 - Part 1 (b) BOX - Video #1 - Part 2 (c) BOX - Video #1 - Part 3 (d) BOX - Video #1 - Part 4

(e) BOX - Video #2 - Part 1 (f) BOX - Video #2 - Part 2 (g) BOX - Video #2 - Part 3 (h) BOX - Video #2 - Part 4

(i) T-less - Part 1 (j) T-less - Part 2 (k) T-less - Part 3 (l) T-less - Part 4

Fig. 10. Results of the experiment described in Section 8.3: detection error Cumulative Distribution Functions (CDF) for the BOX and the T-Less datasets for different detectors. Top row: BOX - Video #1. Middle row: BOX - Video #2. Bottom row: T-less dataset.

reported on the bottom row of Figure 10. Ambiguities badly affect LINE-2D results. Because of shape similarities between different objects of the dataset, we trained our detector hard-sample mining. Part 1 is particularly difficult to detect because of the presence of the occlusion object, whose parts closely resemble to the other objects. On this dataset, DoG normalization does not enhance the performances of the detector, possibly because the lighting conditions of the training and testing sequences are closer than those of the Occlusion datasets.

8.4 Validation of the Part Pose Representation

To validate our part pose representation based on the 2D reprojections of 3D control points introduced in Section 5, we trained several regressor CNNs for predicting the object pose of all the frames of the first video of the BOX Dataset. Each CNN was trained to predict a different part pose representation, which yields to different strategies to combine the contributions of the different parts:

- **Averaging Poses:** The output of the CNN is a 3D rotation and a depth for each part. The in-plane components of the translation are retrieved from the position of the patch on the image. The full object pose is then obtained by averaging the parts poses. Rotations were averaged as proposed in [53].
- **3D Control Points:** The predicted representation is made by the coordinates of the 3D control points shown in Fig. 2 in the camera reference system. Since the 3D coordinates of the control points in the camera system depend on the position of the patch on the image, we employ the following indirect estimation: The output of the CNN consists in a depth value for the center of the patch, and a set of offsets for all the other control points $\{(\delta x/\delta z, \delta y/\delta z, \delta z)\}_k$. The 3D locations of all the control points can be straightforwardly retrieved. The poses of the parts are then estimated and combined by computing the 3D rigid transform aligning the points in the camera and in

the world reference system in a least-square sense [54].

- **2D Reprojections of 3D Control Points:** The output of the CNN is given by the coordinates of the reprojections of the control points, as described in Section 5. The pose is computed by solving the PnP problem after gathering all the 3D-2D correspondences given by all the parts.

The results are shown in Fig. 11. The last choice entails a significant accuracy gain over the previous ones.

To obtain more insight, we also performed two other experiments:

- we replaced the predicted 2D reprojections in the case of the **3D Control Points** experiment by the ground truth (**3D Control Points - GT X and Y**);
- instead of replacing the 2D reprojections by the ground truth, we replaced the depth by its ground truth (**3D Control Points - GT Depth**).

In the first case, the results did not improve much. In the second case, the results are equivalent to the ones of **2D Reprojections of 3D Control Points** (for sake of clarity, the **3D Control Points - GT Depth** curve is not shown in Fig. 11). This shows that predicting the depths is a much more difficult task, than predicting the 2D locations.

8.5 Virtual Points Configuration

In order to assess the influence of the number and configurations of control points on the accuracy of our method, we tested the configurations shown in Fig. 13 on the CAN dataset. We created different configurations with an increasing number of virtual points, and disposed them regularly around the part center. The comparison is performed on the CAN dataset, probably the most challenging one, because the object of interest is tracked using a single part, so we expect the pose estimation results to be particularly sensitive to the disposition and number of control points. We trained one regressor for each of the configurations shown in Fig. 13 from the same learning data, and run the pose estimation for each configuration, starting from the same detection candidates for the can lid. Results are shown in Fig. 12. In general, we observed that:

- configurations spanning the 3 orthogonal directions perform better than planar configurations;
- increasing the number of control points improves results up to 7 points, while no noticeable improvement is obtained by using configurations with more points.

8.6 Comparison Framework

We compared our approach with three state-of-the-art methods, LINE-2D [3], PWP3D [31] and LSD-SLAM [41]. LINE-2D proceeds using very fast template matching. PWP3D is an accurate and robust model-based 3D tracking method based on segmentation. LSD-SLAM is a recent, powerful and reliable SLAM system: amongst other things, it does not require prior 3D knowledge, while we know the 3D locations of the control points and their appearances. The comparison should therefore be taken with caution, as this method does not aim to achieve exactly the same task as us. Nevertheless, we believe the comparison highlights the strengths and weaknesses of the compared methods. For every test video, we compare the poses computed by each method for all frames. Following the evaluation framework in [49], we align each trajectory with respect to the same reference system. In each test, the templates for LINE-2D were extracted by the same 3000 images we employed for training our method. PWP3D was manually initialized using the ground-truth pose data, while LINE-2D, LSD-SLAM and our method do not require any initial pose.

8.7 Results

Quantitative results of our tests are shown in Table 2. LINE-2D, LSD-SLAM, and PWP3D actually fail very frequently on our sequences, drifting or loosing track.

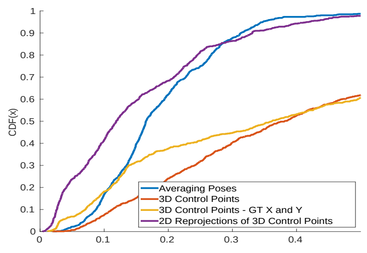
In the BOX dataset, on the longest of our video sequences, we also re-initialized LSD-SLAM and PWP3D using the ground-truth pose at roughly half of the video, but their accuracy over the whole sequence remains outperformed by our method. LINE-2D often fails matching the templates not only when the contours of the box are occluded, but also because its appearance is constantly changed by objects put inside and outside it.

For the CAN dataset, we use a single part to track the full object. In the first video the silhouette of the can is seldom occluded: LINE-2D and PWP3D achieve similar performances, while the lack of texture and the distractor objects make LSD diverge. In the second video, where occlusions occur more often but the background color is different from the one of the can, LSD-SLAM performs better. On both videos, our method consistently outperforms all other methods. Notice that all methods have a quite bad score in retrieving the rotation on this dataset, probably because of the symmetric shape of the object.

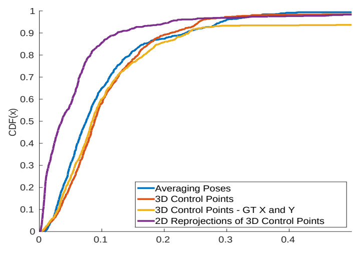
In the DOOR dataset test, LSD-SLAM fails as soon as the door starts to move. LINE-2D fails very often because of the ambiguous contours present in the scene. PWP3D immediately looses tracking. Our method manages to track frames across the whole video. This result is somehow surprising, since PWP-3D exploits the appearance of the whole door, while our method just exploits a minimal part of its structure. We only use the CAD model for predicting contours and evaluating the computed poses, as explained in Section 6.4. On all datasets, both the Kalman Filtering and the DoG normalization entail a significant improvement of the performances. Qualitative results of our method are reported in Figure 7.

8.8 Results on the T-Less dataset

In order to assess the effectiveness of the prediction of the pose when varying the number and shape of parts, we tested our method on Sequence 3 of the T-Less dataset, employing the 4 differently shaped objects shown in Figure 9,(a)-(d) as parts in our framework, considering the whole scene as the target object and the fifth, largest object as an occlusion.

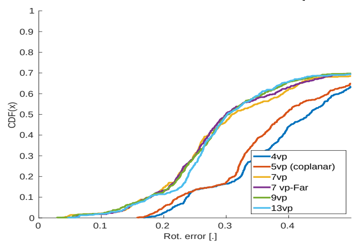


(a) Rotation error CDF: BOX - Video #1

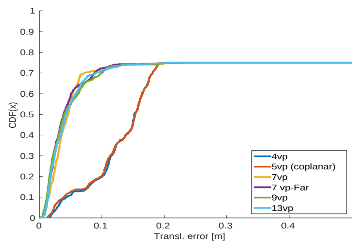


(b) Translation error CDF: BOX - Video #1

Fig. 11. Rotation and translation error CDF graphs for the BOX dataset - Video #1 and for the pose parametrizations presented in Section 8.4. Our pose representation entails a substantial performance gain.



(a) Rotation error CDF: CAN - Video #1



(b) Translation error CDF: CAN - Video #1

Fig. 12. Rotation and translation error CDF graphs for the configurations of virtual points shown in Figure 13 for the CAN dataset-Video #1. Best results are obtained by the configurations spanning the 3 orthogonal axes. Increasing the number of virtual points does not improve results above 7 virtual points.

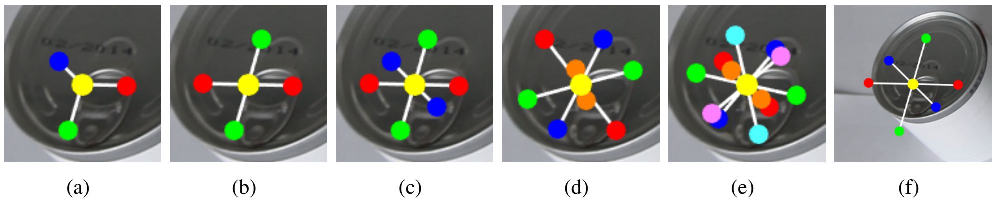


Fig. 13. Different configurations of control points tested on the CAN dataset, with (a) 4 control points spanning the 3 axes; (b) 5 co-planar control points; (c) 7 control points spanning the 3D axes; (d) 9 control points disposed in the center and on the corners of a cube; (e) 13 control points disposed in the center and on the corners of an icosahedron; (f) 7 control points spanning the 3 axes, with a larger spacing.

We run a first series of experiments predicting the scene pose using different numbers of parts and using the ground-truth detections for each part, so that the effect of the shape and the number of parts employed for the pose estimation is better highlighted. Moreover, we also tested our full method, as originally proposed in [20], as well as with the Kalman filter and the patch normalization. Results are reported in Table 3. Unsurprisingly, results confirm that adding parts is always beneficial to the tracking. The DoG normalization does not improve the accuracy of the pose prediction on this dataset, while the Kalman filter still does.

8.9 A Real Use Case: Augmented Reality at CERN

We implemented our pipeline within an Augmented Reality application in a real use case scenario, providing assistance for technical and maintenance interventions in extreme environments in the ATLAS particle detector at CERN. Technical interventions are made particularly challenging by factors as radioactivity, difficulty of access, exposition to hazardous gases, etc. Augmented Reality is being investigated as a way of reducing the time of each intervention and the stress of the operators, by providing them instructions and environmental data in visual form through an head-mounted display (HMD). We built a prototype where images captured by a camera mounted on the operator's helmet are streamed to a server. There, the pose of the object of interest—the electric box of our BOX dataset—is computed and transmitted back to the user's HMD, where authoring content is rendered on the head-mounted display. The setup is shown in Fig. 14, while an example of the outcome of our pipeline is shown in Fig. 1.

8.10 Runtimes

Our current implementation on an Intel i7-6700K laptop with Nvidia NVIDIA Gtx 1080 takes between 100ms (one part) and 140ms (4 parts). Detailed computational times are given in the supplementary material. Many optimizations are possible. For example, the control point predictions for each part could be run in parallel.

9 Conclusion

We showed the advantages of predicting the reprojections of virtual points as a way to estimate the 3D pose of object parts. We believe that this representation, simple and powerful, could be useful not only for object instance

Experiment	BOX dataset Video #1	BOX dataset Video #2	CAN dataset Video #1	CAN dataset Video #2	DOOR dataset Video #1
nb. of frames	892	500	450	314	564
LSD-SLAM	0.37 - 0.61*	0.48- 0.63	0.17 - 0.29	0.38 - 0.48	0.50 - 0.38
PWP3D	0.10 - 0.20*	0.16 - 0.52	0.13 - 0.64	0.13 - 0.51	0 - 0
LINE-2D	0.34 - 0.41	0.34 - 0.44	0.20 - 0.62	0.29 - 0.65	0.13 - 0.14
Our method	0.75 - 0.85	0.57 - 0.85	0.35 - 0.85	0.51 - 0.70	0.72 - 0.61
Our method - KAL	0.78 - 0.86	0.65 - 0.88	0.36 - 0.86	0.51 - 0.70	**0.79 - 0.66**
Our method - DoG	0.76 - 0.85	0.80 - 0.88	0.42 - 0.92	0.52 - 0.74	0.76 - 0.69
Our method - KAL+DoG	**0.78 - 0.86**	**0.82 - 0.90**	**0.42 - 0.93**	**0.55 - 0.75**	**0.76 - 0.70**

TABLE 2

Experimental results. We report the AUC scores for the rotation and the translation errors for the five video sequences of the Occlusion datasets. A star (*) after the scores indicates that the method was re-initialized with the groundtruth for frame 500. We report results of our method as implemented in [20], as well as with the contributions of this work, the Kalman filter (KAL) and the patch normalization (DoG). Both innovations sensibly enhance performances. We report all the corresponding CDF graphs in the supplementary material.

Experiment	T-less dataset - Sequence 3
GT-Det - All Parts	0.87 - 0.75
GT-Det - Parts 1 2 3	0.66 - 0.57
GT-Det - Parts 1 2 4	0.83 - 0.72
GT-Det - Parts 1 3 4	0.79 - 0.63
GT-Det - Parts 2 3 4	0.82 - 0.70
GT-Det - Parts 1 2	0.19 - 0.21
GT-Det - Parts 1 3	0.16 - 0.29
GT-Det - Parts 1 4	0.59 - 0.42
GT-Det - Parts 2 3	0.32 - 0.22
GT-Det - Parts 2 4	0.71 - 0.59
GT-Det - Parts 3 4	0.70 - 0.47

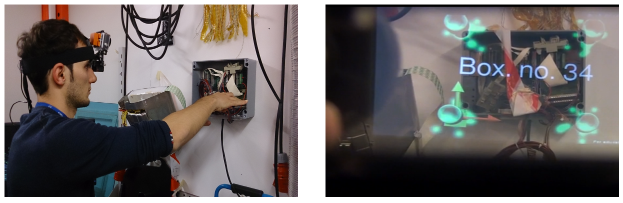
(a)

Experiment	T-less dataset - Sequence 3
Our method	0.50 - 0.37
Our method - KAL	**0.53 - 0.37**
Our method - DoG	0.45 - 0.34
Our method - KAL+DoG	0.49 - 0.37

(b)

TABLE 3

Pose estimation results on the T-less dataset - Sequence 3. (a): the "GT-Det" experiments are run using ground-truth detections for the selected parts; for these experiments we only consider the frames where the relevant parts are visible. (b): results using our full pipeline as well as with the Kalman filter and the normalization.



(a) (b)

Fig. 14. The AR system developed at CERN for assisting technical interventions. (a): a camera over the user's head streams images to a server for pose estimation; the pose is sent back to an head-mounted see-through display (HMD) for rendering. (b): an example of augmented content seen through the HMD.

detection, but also for the 3D pose estimation of categories of objects, where current approaches drastically suffer from partial occlusions.

Acknowledgement: This work was supported in part by the CDL Semantic for 3D Computer Vision and in part by the EU projects EDUSAFE and MAGELLAN.

REFERENCES

[1] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model Globally, Match Locally: Efficient and Robust 3D Object Recognition," in *CVPR*, 2010.

[2] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Teaching 3D Geometry to Deformable Part Models," in *CVPR*, 2012.

[3] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient Response Maps for Real-Time Detection of Textureless Objects," *PAMI*, vol. 34, no. 5, pp. 876–888, 2012.

[4] D. Damen, P. Bunnun, A. Calway, and W. Mayol-cuevas, "Real-Time Learning and Detection of 3D Texture-Less Objects: A Scalable Approach," in *BMVC*, 2012.

[5] R. Rios-cabrera and T. Tuytelaars, "Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach," in *ICCV*, 2013.

[6] K. Pauwels, L. Rubio, J. Diaz, and E. Ros, "Real-Time Model-Based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues," in *CVPR*, 2013.

[7] F. Tombari, A. Franchi, and L. D. Stefano, "BOLD Deatures to Detect Texture-Less Objects," in *ICCV*, 2013.

[8] N. Kyriazis and A. Argyros, "Scalable 3D Tracking of Multiple Interacting Objects," in *CVPR*, 2014.

[9] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D Object Pose Estimation Using 3D Object Coordinates," in *ECCV*, 2014.

[10] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese, "Monocular Multiview Object Tracking with 3D Aspect Parts," in *ECCV*, 2014.

[11] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-Class Hough Forests for 3D Object Detection and Pose Estimation," in *ECCV*, 2014.

[12] J. Lim, A. Khosla, and A. Torralba, "FPM: Fine Pose Parts-Based Model with 3D CAD Models," in *ECCV*, 2014.

[13] S. Song and J. Xiao, "Sliding Shapes for 3D Object Detection in Depth Images," in *ECCV*, 2014.

[14] P. Wohlhart and V. Lepetit, "Learning Descriptors for Object Recognition and 3D Pose Estimation," in *CVPR*, 2015.

[15] K. He, L. Sigal, and S. Sclaroff, "Parameterizing Object Detectors in the Continuous Pose Space," in *ECCV*, 2014.

[16] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit, "Online Learning of Patch Perspective Rectification for Efficient Object Detection," in *CVPR*, 2008.
[17] A. Shrivastava and A. Gupta, "Building Part-Based Object Detectors via 3D Geometry," in *ICCV*, 2013.
[18] K. Koser and R. Koch, "Perspectively Invariant Normal Features," in *ICCV*, 2007.
[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *IEEE*, 1998.
[20] A. Crivellaro, M. Rad, Y. Verdie, K. Yi, P. Fua, and V. Lepetit, "A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images," in *ICCV*, 2015.
[21] G. Welch and G. Bishop, "An Introduction to Kalman Filter," Department of Computer Science, University of North Carolina, Technical Report, 1995.
[22] C. Harris and C. Stennett, "RAPID-a Video Rate Object Tracker," in *BMVC*, 1990.
[23] D. G. Lowe, "Fitting Parameterized Three-Dimensional Models to Images," *PAMI*, vol. 13, no. 5, pp. 441–450, June 1991.
[24] G. Klein and D. Murray, "Full-3D Edge Tracking with a Particle Filter," in *BMVC*, 2006.
[25] I. Skrypnyk and D. G. Lowe, "Scene Modelling, Recognition and Tracking with Invariant Image Features," in *ISMAR*, November 2004.
[26] L. Vacchetti, V. Lepetit, and P. Fua, "Stable Real-Time 3D Tracking Using Online and Offline Information," *PAMI*, vol. 26, no. 10, October 2004.
[27] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg, "Pose Tracking from Natural Features on Mobile Phones," in *ISMAR*, September 2008.
[28] E. Rosten and T. Drummond, "Fusing Points and Lines for High Performance Tracking," in *ICCV*, 2005.
[29] C. Choi, A. Trevor, and H. Christensen, "RGB-D Edge Detection and Edge-Based Registration," in *IROS*, 2013.
[30] V. Prisacariu, A. Segal, and I. Reid, "Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction," in *ACCV*, 2012.
[31] V. Prisacariu and I. Reid, "PWP3D: Real-Time Segmentation and Tracking of 3D Objects," *IJCV*, vol. 98, pp. 335–354, 2012.
[32] G. Chliveros, M. Pateraki, and P. Trahanias, "Robust Multi-Hypothesis 3D Object Pose Tracking," in *ICCV*, 2013.
[33] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects," in *CVPR*, 2013.
[34] K. Lai, L. Bo, X. Ren, and D. Fox, "A Scalable Tree-Based Approach for Joint Object and Pose Recognition," in *AAAI*, 2011.
[35] A. Krull, E. Brachmann, F. Michel, M. Y. Yang, S. Gumhold, and C. Rother, "Learning Analysis-By-Synthesis for 6D Pose Estimation in RGB-D Images," in *ICCV*, 2015.
[36] D. Tan and S. Ilic, "Multi-Forest Tracker: A Chameleon in Tracking," in *CVPR*, 2014.
[37] N. Payet and S. Todorovic, "From Contours to 3D Object Detection and Pose Estimation," in *ICCV*, 2011.
[38] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part Based Models," *PAMI*, vol. 32, no. 9, pp. 1627–1645, 2010.
[39] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *ISMAR*, 2007.
[40] R. Newcombe, S. Lovegrove, and A. Davison, "DTAM: Dense Tracking and Mapping in Real-Time," in *ICCV*, 2011.
[41] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *ECCV*, 2014.
[42] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.
[43] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks," in *International Conference on Learning Representations*, 2014.
[44] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *ICLR*, 2015.
[45] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast Image Scanning with Deep Max-Pooling Convolutional Neural Networks," in *ICIP*, 2013.
[46] S. Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns," *PAMI*, vol. 13, no. 4, 1991.
[47] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow., A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: New Features and Speed Improvements," in *NIPS*, 2012.
[48] F. Moreno-noguer, V. Lepetit, and P. Fua, "Pose Priors for Simultaneously Solving Alignment and Correspondence," in *ECCV*, 2008.
[49] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *IROS*, 2012.
[50] A. Ude, "Filtering in a unit quaternion space for model-based object tracking," *Robotics and Autonomous Systems*, vol. 28, no. 23, 1999.
[51] T. Hodaň, J. Matas, and S. Obdržálek, "On evaluation of 6d object pose estimation," in *ECCV Workshops*, 2016.
[52] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic Generation and Detection of Highly Reliable Fiducial Markers Under Occlusion," *PR*, vol. 47, no. 6, pp. 2280–2292, 2014.
[53] F. Markley, Y. Cheng, J. Crassidis, and Y. Oshman, "Averaging quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007.
[54] D. Eggert, A. Lorusso, and R. Fisher, "Estimating 3D Rigid Body Transformations: A Comparison of Four Major Algorithms," *Machine Vision and Applications*, vol. 9, no. 5-6, pp. 272–290, 1997.

Alberto Crivellaro received his MSc in Math. Engineering from Politecnico di Milano, Italy, his Diplôme d'Ingénieur from the EC Lyon, France, in 2011, and his a Ph.D. at CVLab, EPFL, Switzerland, in 2016, under the supervision of prof. P. Fua and prof. V. Lepetit . His research interests include Augmented Reality, visual tracking, and machine learning.

Mahdi Rad received his B.S. and M.S. degrees in Computer Science from EPFL, Switzerland, in 2012 and 2014. Currently, he is a PhD student in the Computer Vision and Graphics Laboratory at Graz University of Technology. His research interests include Augmented Reality, deep learning, and 3D object tracking.

Yannick Verdie received his B.S. from Virginia-Tech, USA, and his Ph.D. degrees from INRIA, France in 2010 and 2013. After a two-years postdoc position at the CVLab, EPFL, Switzerland, he currently works as R&D Engineer at NCam-Tech, Paris, working on vision-based systems for real-time augmented reality in movies and TV-shows.

Kwang Moo Yi received his B.S. and Ph.D. degrees from the Department of Electrical Engineering and Computer Science of Seoul National University, Korea, in 2007 and 2014. Currently, he is a postdoc researcher in the CVLab at EPFL. His research interests include Augmented Reality, keypoint learning, deep learning, visual tracking, and motion detection.

Pascal Fua is a Professor of Computer Science at EPFL, Switzerland. His research interests include shape and motion reconstruction from images, analysis of microscopy images, and Augmented Reality. He is an IEEE Fellow and has been an Associate Editor of the IEEE journal Transactions for Pattern Analysis and Machine Intelligence.

Vincent Lepetit is a Professor at the Institute for Computer Graphics and Vision, TU Graz. Before that, he was a Research and Teaching Associate at EPFL. His research interests include vision-based Augmented Reality, 3D camera tracking, Machine Learning, object recognition, and 3D reconstruction. He is an editor for the International Journal of Computer Vision (IJCV) and the Computer Vision and Image Understanding (CVIU) journal.