A Novel Representation of Parts for Accurate 3D Object Detection and Tracking in Monocular Images

Alberto Crivellaro[1] Mahdi Rad[2] Yannick Verdie[1] Kwang Moo Yi[1]
Pascal Fua[1] Vincent Lepetit[2]

[1]Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[2]Institute for Computer Graphics and Vision, Graz University of Technology, Austria
{alberto.crivellaro, yannick.verdie, kwang.yi, pascal.fua}@epfl.ch
{rad, lepetit}@icg.tugraz.at

Abstract

We present a method that estimates in real-time and under challenging conditions the 3D pose of a known object. Our method relies only on grayscale images since depth cameras fail on metallic objects; it can handle poorly textured objects, and cluttered, changing environments; the pose it predicts degrades gracefully in presence of large occlusions. As a result, by contrast with the state-of-the-art, our method is suitable for practical Augmented Reality applications even in industrial environments. To be robust to occlusions, we first learn to detect some parts of the target object. Our key idea is to then 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 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 are visible.

1. Introduction

3D object detection and tracking methods have undergone impressive improvements in recent years [5, 27, 13, 4, 30, 25, 39, 17, 1, 45, 38, 20, 35, 44]. However, each of the current approaches has its own weaknesses: Many of these approaches [5, 13, 1, 35] rely on a depth sensor, which would fail on metallic objects or outdoor scenes; methods based on feature points [25, 17] expect textured objects; those based on edges [4, 39] are sensitive to cluttered background; most of these methods [13, 27, 30, 38, 11, 45, 20] are not robust to occlusion. We also want a method fast enough for interactive 3D applications.

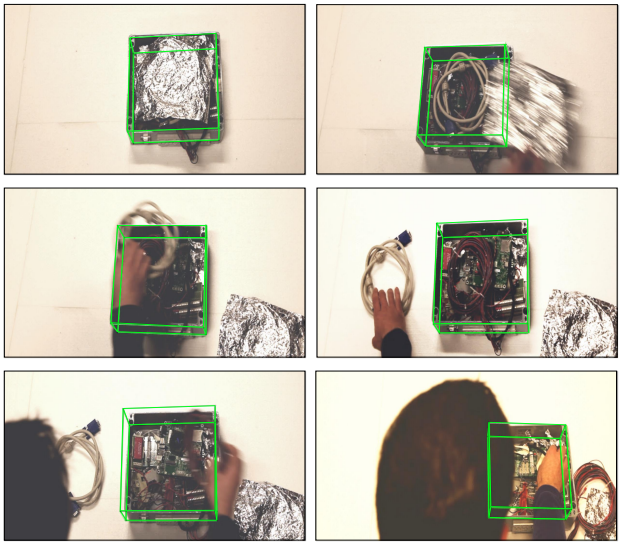
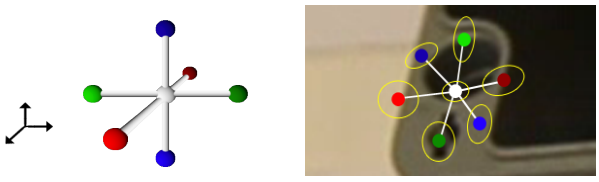


Figure 1. Detecting a box in 3D with a regular camera under challenging conditions. On this dataset, the user removes objects from the box and leaves them on the table, often occluding large portions of the box. Despite these difficulties, we can accurately estimate the 3D pose of the box, in each image independently.

As Fig. 1 shows [1], we are interested in scenes with poorly textured objects, possibly visible only under 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 also fail because of the lack of texture. These are typical conditions of many Augmented Reality applications.

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 [12, 27, 33, 20, 45]. The

[1]All the figures of this work are best seen in colors.



(a) (b)

Figure 2. Our representation of the 3D pose of an object part. (a) We consider seven 3D control points for each part, arranged to span into 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.

novelty in our approach is a powerful representation of the pose of each part.

Some previous methods used homographies [16, 12, 45] 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 propose to represent the pose of each part by the *2D reprojections of a small set of 3D control points*. The control points are only "virtual", and do not have to correspond to specific image features. This representation is invariant to the part's image location and only depends on its appearance. We show that a Convolutional Neural Network [19] (CNN) can predict the locations of these reprojections very accurately, and can also be used to predict the uncertainty of these location estimates.

Given an input image, we run a detector to obtain a few hypotheses on the image locations of each part. 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, some of which may be erroneous, but 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 (or a single one) are visible.

In the remainder of the paper, we first discuss related work in Section 2, we describe our approach in sections 3 and 4, and we evaluate it in Section 5 on challenging datasets.

2. Related Work

3D object detection has a long history, and we focus here on representative works. A well-established research direction relies on edges [10, 21, 14], but they are sensitive to large occlusions and clutter. More recently, keypoint-based methods became popular [34, 41, 42] 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 [31, 3] or stereo information [25]. 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 [29, 28], 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 [2] 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. [5, 32] use votes from pairs of 3D points and their normals to detect 3D objects. [18] uses a decision tree applied to RGB-D images. [13] uses a template-based representation for dealing with poorly textured objects. The more recent [1, 38] 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 [37] and [12], we learn 3D poses. Nonetheless, our part-based approach allows 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 [26, 27, 45, 20] that mostly focus on category rather than instance detection. These works were mostly motivated by the success of the Deformable Part Model [7] developed for 2D detection, which was extended successfully to 3D, e.g. in [27]. [45] also performs 3D tracking through part-based particle filtering by integrating multi-view. [26] uses contours as parts. In [20], 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) [15, 24, 6]. 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. Moreover SLAM is prone to fail on dynamic scenes.

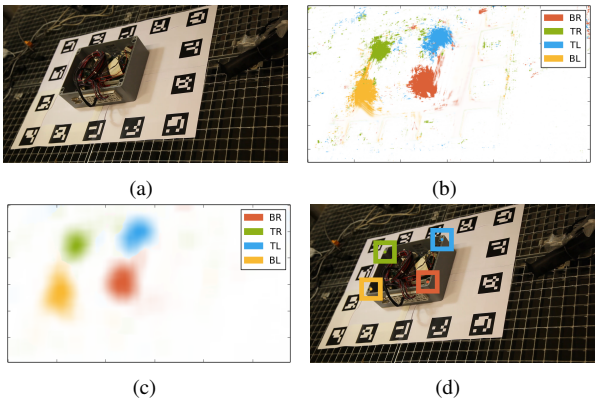


Figure 3. Detecting the parts. (a) An input image of the box. (b) The output of the $\text{CNN}^{\text{part-det}}$ for each image location. Each color corresponds to a different part. (c) The output after Gaussian smoothing. (d) The detected parts, corresponding to the local maximums in (c).

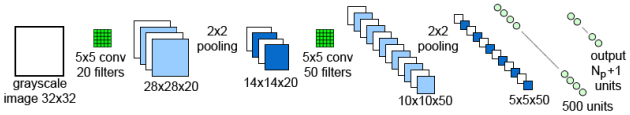


Figure 4. Architecture of $\text{CNN}^{\text{part-det}}$ for part detection. The last layer outputs the likelihoods of the patch to correspond to each part or to the background.

3. Part Pose Representation and Estimation

Given an input grayscale image[2], we want to estimate the 3D pose $\mathbf{p}$ of a calibrated projective camera with respect to a known rigid object. We assume that we are given a 3D model of the object, for example in the form of a triangular mesh, and a set of manually labelled *parts* on the object. A very small number of parts is required by our framework; in all our tests we employed at most 4 parts for an object. We currently select the parts by hand (automatic selection is left to future work). The parts should ideally be easy to detect in images, and spread over the object.

In this section, we justify our choice for the representation of the pose parts, and we explain how we detect the parts and predict their poses. The next section describes our algorithm to compute the pose of the camera based on the predicted pose parts. The main notations are resumed in Table 3.1.

3.1. Representing the Part Poses

One can think of different ways to represent the 3D poses of parts of objects. For example, it is possible to use homographies [16, 12, 45]. However, this assumes that the part surface is planar, and makes it difficult to merge the contributions of the different parts.

Another possibility we considered is to predict from the appearance of the part, a 3D rotation matrix and the depth value of its center. Assuming an orthogonal projection, it is possible to retrieve the 3D translation as well, from the patch center image location and the predicted depth. However, this representation is not translation invariant in a full perspective model. Also it is not clear how to merge rotations for estimating the pose of the whole target object. Finally, it is difficult to predict the depth accurately from the image patch, as our results will demonstrate.

Since our final solution is based on 3D control points, as already mentioned, we could also directly predict the 3D locations of the 3D Control Points in the camera reference system: This makes combining the poses simpler, as this only involves computing the rigid motion between two sets of 3D points [40]. Unfortunately, this representation is not translation invariant. Moreover, as for the previous option, it requires to directly predict the depths of the points, which is far from accurate in our experiments.

This is why we propose to represent the part pose 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 simply grouping all the 2D reprojections together and solving a PnP problem; we do not need to predict the depth of the 3D points, which is difficult to do accurately. These advantages entail a tremendous accuracy gain, as showed by our results in Section 5.2. In our experiments, we used 7 control points for each part, spanning 3 orthogonal directions, as shown in Fig. 2(a), however other configurations could probably be used.

symbol	meaning
i	index of a training image
j	index of a part
k	index of a control point or its projection
l	index of a detection
$\mathbf{C}_j$	3D center of the j-th part
$\mathbf{c}_{ij}$	projection of $\mathbf{C}_j$ in the i-th image
$\hat{\mathbf{c}}_{jl}$	l-th detection for the projection of $\mathbf{C}_j$ in an input image
s_{jl}	score for this detection
$\mathbf{V}_{jk}$	k-th 3D control point of the j-th part
$\mathbf{v}_{ijk}$	projection of $\mathbf{V}_{jk}$ in the i-th image
$\hat{\mathbf{v}}_{jk}$	prediction for the projection of $\mathbf{V}_{jk}$ (no outlier)
S_{jk}	covariance for prediction for the projection of $\mathbf{V}_{jk}$ (no outlier)
$\hat{\mathbf{v}}_{jkl}$	l-th prediction for the projection of $\mathbf{V}_{jk}$ in an input image
$\mathbf{q}$	an image patch

Table 1. Main notations.

3.2. Detecting the Parts

We use a set of registered training images of the target object under different poses and lighting (as the one shown in Fig. 3(a)) to learn to detect the parts and predict their control points. We will denote our training data as:

$$\mathcal{T} = \left\{ \left(I_i, \{\mathbf{c}_{ij}\}_j, \{\mathbf{v}_{ijk}\}_{jk} \right) \right\}_i , \quad (1)$$

[2] All experiments of this work were performed with VGA images

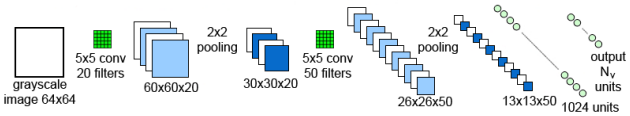


Figure 5. Architecture of a CNN $\text{CNN}^{\text{cp-pred-}j}$ for predicting the projections of the control points.

where I_i denotes the i-th training image, $\mathbf{c}_{ij}$ the projection of the center $\mathbf{C}_j$ of the j-th part on I_i, and $\mathbf{v}_{ijk}$ the projection of the k-th control point of the j-th part in this image.

During an offline stage, we train a first 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 $P(J = j \mid \mathbf{q})$ of the patch to correspond to one of the N_P parts. We train the CNN with patches randomly extracted around the centers $\mathbf{c}_{ij}$ of the parts in images I_i and patches extracted from the background, and by optimizing the negative log-likelihood over the parameters w of the CNN:

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

where $\mathcal{T}_j$ is a training set made of image patches centered on part j 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 $\text{softmax}(\text{CNN}_w^{\text{part-det}}(\mathbf{q}))[j]$ is the j-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 [9]. 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 j, a set $\mathcal{S}_j = \{(\hat{\mathbf{c}}_{jl}, s_{jl})\}_l$ of 2D location candidate $\hat{\mathbf{c}}_{jl}$ for the part together with a score s_{jl} 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 4. We typically get up to 4 detections for each part in a given input image.

3.3. Predicting the Reprojections of the Control Points and their Uncertainty

Once the parts are detected, we apply a second CNN to the patches centered on the candidates $\hat{\mathbf{c}}_{jl}$ to predict the projections of the control points for these candidates. 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:

$$\widehat{w} = \arg\min \sum_{(\mathbf{q},\mathbf{w}) \in \mathcal{V}_j} ||\mathbf{w} - \text{CNN}_w^{\text{cp-pred-}j}(\mathbf{q})||^2 \quad , \tag{3}$$

where $\mathcal{V}_j$ is a training set of image patches $\mathbf{q}$ centered on part j and the corresponding 2D locations of the control points concatenated in a $(2N_V)$-vector $\mathbf{w}$, and $\text{CNN}_w^{\text{cp-pred-}j}(\mathbf{q})$ is the prediction for these locations made by the CNN specific for part j, given patch $\mathbf{q}$ as input.

At run-time, we obtain for each $\hat{\mathbf{c}}_{jl}$ candidate, predictions $\{\hat{\mathbf{v}}_{jkl}\}$ for the control points projections. In addition, we estimate the 2D uncertainty for the predictions, by propagating the image noise through the CNN that predicts the control point projections [43]. Let us consider the matrix:

$$\mathsf{S}_V = \mathsf{J}_{\hat{\mathbf{c}}}(\sigma \mathsf{I}) \mathsf{J}_{\hat{\mathbf{c}}}^\top \quad , \tag{4}$$

where σ is the standard deviation of the image noise assumed to be Gaussian and affect each image pixel independently, I 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, by composing the Jacobians of the successive layers of the network. We neglect the correlation between the different control points to finally extract from the block diagonal of S_V the 2×2 uncertainty matrix noted S_{jkl} below for each control point. An example of predicted control points and their uncertainties is shown in Fig. 2(b). Note that we can easily compute the S_{jkl} matrices without having to compute the entire, and very large, product in Eq. (4).

4. Estimating the Object Pose

Thanks to our representation for the part poses, estimating the object pose is straightforward, since each control point provides a 3D-2D correspondence. We describe here the method we use, other methods are probably possible.

We assume that we are given a prior on the pose $\mathbf{p}$, in the form of a Mixture-of-Gaussians $\{(\overline{\mathbf{p}}_i, \mathsf{S}_i)\}$, as was done e.g. in [23]. This prior is very general, and allows us to define the normal action range of the camera. 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 4.3.

4.1. 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

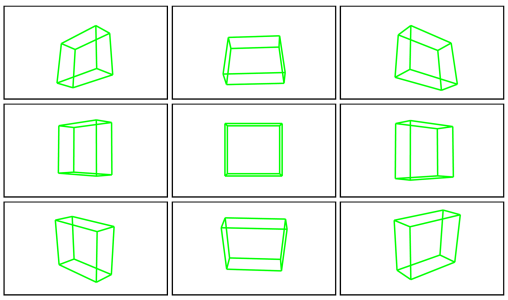


Figure 6. Visualisation of the pose prior for the BOX sequences: Projections of the bounding box of the box by each of the 9 Gaussians centers $\overline{\mathbf{p}}_i$.

all the parts are visible. Then, the object pose $\hat{\mathbf{p}}$, or equivalently the camera pose, can be estimated as the minimizer of $F(\mathbf{p})$, with $F(\mathbf{p}) =$

$$\frac{1}{N_P}\sum_{j,k}\text{dist}^2(\mathsf{S}_{jk},\Gamma_{\mathbf{p}}(\mathbf{V}_{jk}),\hat{\mathbf{v}}_{jk})+ (\mathbf{p}-\overline{\mathbf{p}}_0)^\top\mathsf{S}_0^{-1}(\mathbf{p}-\overline{\mathbf{p}}_0) \quad , \tag{5}$$

where the sum is over all the control points of all the parts, and $\Gamma_{\mathbf{p}}(\mathbf{V})$ is the 2D projection of $\mathbf{V}$ under pose $\mathbf{p}$. $\hat{\mathbf{v}}_{jk}$ is the projection of control point $\mathbf{V}_{jk}$ and S_{jk} its uncertainty estimated as explained in Section 3.3—since we assume there is no outlier, we dropped here the l index corresponding to the multiple detections. dist(.) is the Mahalanobis distance:

$$\text{dist}^2(\mathsf{S},\mathbf{v}_1,\mathbf{v}_2)=(\mathbf{v}_1-\mathbf{v}_2)^\top\mathsf{S}^{-1}(\mathbf{v}_1-\mathbf{v}_2) \quad . \tag{6}$$

$F(\mathbf{p})$ is minimized using the Gauss-Newton algorithm initialized with $\mathbf{p}_0$.

4.2. Robust detection of parts

In practice, for the location of the j-th part, the detection procedure described in Section 3.2 can get a set of hypotheses $\mathcal{S}_j$, and at most one is correct.

Checking all the possible combinations would be time consuming, so we rank the candidates according to their score s_{jl}, keep the best four candidates for each part and greedily examine the possible sets $\mathcal{C}$ of correspondences between a part and the candidate detections.

Similarly to [23], we exploit the pose prior for first quickly evaluating if the correspondences in $\mathcal{C}$ can yield a good pose estimate.

We only consider a set $\mathcal{C}$ if

$$\begin{aligned}&\forall j\in\mathcal{C}: \quad \tilde{\rho}_j < T^2\\ &\text{with} \quad \tilde{\rho}_j=\rho_j-\rho_{\hat{j}}\\ &\qquad\quad \rho_j=\text{dist}^2(\hat{\mathsf{S}}_0(\mathbf{C}_j),\Gamma_{\mathbf{p}_0}(\mathbf{C}_j),\hat{\mathbf{c}}_{jl})\end{aligned} \tag{7}$$

where $T=40$, and where $\hat{\mathsf{S}}_0(\mathbf{C}_j)=\mathsf{J}\,\mathsf{S}_0\mathsf{J}^\top$, with J the jacobian of $\Gamma_{\mathbf{p}_0}(\mathbf{C}_j)$, is the covariance of the projection $\Gamma_{\mathbf{p}_0}(\mathbf{C}_j)$ of $\mathbf{C}_j$, and $\rho_{\hat{j}}$ is a random candidate of the set (since it is reasonable to suppose that at least one candidate of a part has been reliably detected).

If $\mathcal{C}$ passes this test, we compute the average distance $\overline{\rho}=\frac{1}{|\mathcal{C}|}\sum_j\tilde{\rho}_j$ of its points. We keep the $N_{\mathcal{C}}$ sets with the lowest average distance (in practice, we set $N_{\mathcal{C}}=4$ for all our experiments); we run the Gauss-Newton optimization of Eq. (5) using each $\mathcal{C}$ to obtain a pose estimate, and evaluate it as explained in Section 4.3.

4.3. 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 $\{(\overline{\mathbf{p}}_m,\Sigma_m)\}_m$ with $M=9$ components (the prior employed for the **BOX** dataset is shown in Fig.6). We apply the method described above to each component, and obtain $MN_{\mathcal{C}}$ possible pose estimates: $\hat{\mathbf{p}}^{(1)},\ldots,\hat{\mathbf{p}}^{(MN_{\mathcal{C}})}$.

To finally identify the best pose estimate, we evaluate each $\hat{\mathbf{p}}^{(n)}$, employing a weighted sum of several cues: the angle between the quaternions for $\hat{\mathbf{p}}^{(n)}$ and the corresponding $\mathbf{p}_i$ prior; the average reprojection error of the set of control points $\mathcal{C}$ according to $\hat{\mathbf{p}}^{(n)}$; the correlation between the object contours after projection by $\hat{\mathbf{p}}^{(n)}$ and the edges detected in the image. For setting the weights, we train a simple linear regressor on the training video sequences to predict the Euclidean distance between $\hat{\mathbf{p}}^{(n)}$ and the groundtruth. At testing time, we use the linear regressor to evaluate the quality of the computed pose, i.e we keep the pose that gives the smallest predicted distance.

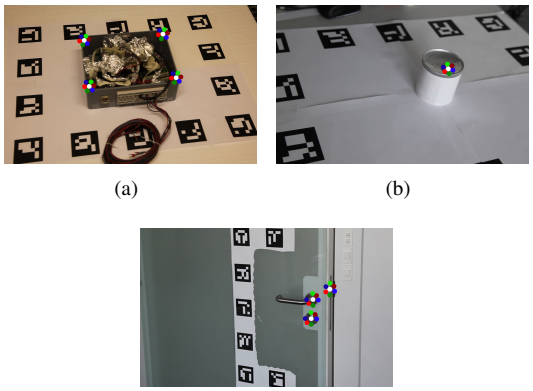
If the optimization of Eq. (5) converges, we add to the initial prior the estimated pose and its covariance as part of the pose prior for the next frame. This helps enforcing temporal consistency. The pose covariance is obtained using the Extended Kalman Filter update formula [43] when optimizing Eq. (5).

5. Experimental Results

In this section, after describing the datasets we used for evaluating our method, we present and discuss the results of our evaluation. In Section 5.2 we validate the choice of reprojections of control points for representing the pose of each part. Then, in Section 5.3 and 5.4 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.

5.1. Datasets

There is currently no standard dataset for benchmarking 3D object detection and tracking methods in presence of heavy occlusions and cluttered, dynamic background. We therefore introduce several datasets for extensive evaluation of 3D object tracking, consisting of both learning data and



(a) (b) (c)

Figure 7. Training images and control points we used for the BOX, the CAN and the DOOR datasets.

testing video sequences, a CAD model without texture and the groundtruth pose for all the sequences. For each dataset, the training set is made of 3000 frames. 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 maintenance intervention by a technician. 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 the 4 corners of the box as parts, as shown in Fig. 3.
- **CAN Dataset:** The target object of this dataset is a food can. In the test videos, 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. 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 particularly 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.9, the knob, the keyhole and the lock of the door. A CAD model of the door is provided as well.

The images of the training and testing videos of the datasets were registered using the ARUCO marker tracking tool [8]. As showed in Fig. 7, we used a full grid of markers for the learning sequences, while a small set of markers has been placed far from the objects for the testing sequences. We then cropped them so that they could not influence detection and tracking performance when testing the methods.

5.2. Validation of the Part Pose Representation

To validate the part pose representation introduced in Section 3.1, 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:

- **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 [22].
- **3D Control Points:** The output of the CNN are the coordinates of the projected control points shown in Fig. 2 on the patch and a depth for each control point. So, the 3D coordinates of the control points in the camera reference system can be computed; the poses of the parts are then combined by computing the 3D rigid transform aligning the points in the camera and in the world reference system in a least-square sense.
- **2D Control Points and PnP:** The output of the CNN is given by the coordinates of the reprojections of the control points, as described in Section 3.1. 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. 8. The last choice entails a tremendous accuracy gain over the previous ones.

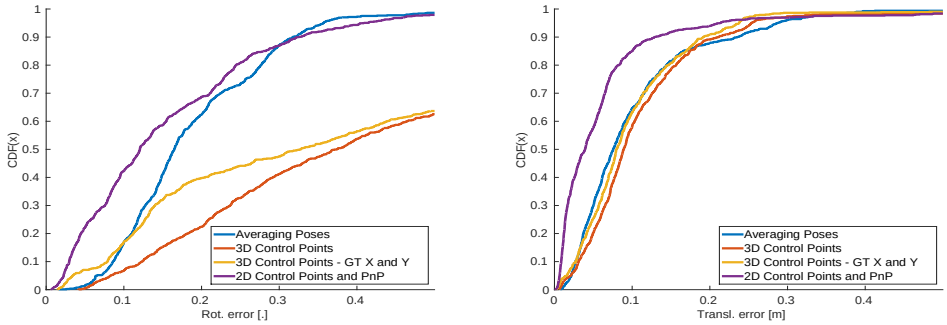
We also performed two other experiments:

- 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**);
- we replaced the predicted depths by the 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 Control Points and PnP** (for sake of clarity, the **3D Control Points - GT Depth** curve is not shown in Fig. 8). This shows that predicting the depths is a difficult task, while predicting the 2D locations is much easier.

5.3. Comparison Framework

We compared our approach with three state-of the art methods, LINE-2D [13], PWP3D [28] and LSD-SLAM [6]. LINE-2D is one of the best methods for rigid object recognition and it 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



(a) Rotation error CDF: BOX dataset-video #1 (b) Translation error CDF: BOX dataset-video #1

Figure 8. Pose estimation results for the BOX dataset - video #1 for different parametrizations of the part poses.

Experiment	BOX dataset		CAN Dataset		DOOR Dataset
	Video #1	Video #2	Video #1	Video #2	Video #1
nb. of frames	892	500	450	314	564
LSD-SLAM	0.37 - 0.39*	0.42 - 0.56	0.17 - 0.27	0.38 - 0.58	0.50 - 0.37
PWP3D	0.10 - 0.20*	0.21 - 0.49	0.12 - 0.56	0.13 - 0.51	0 - 0
LINE-2D	0.31 - 0.38	0.34 - 0.45	0.10 - 0.53	0.11 - 0.46	0.12 - 0.06
Our method	**0.73 - 0.84**	**0.60 - 0.85**	**0.37 - 0.86**	**0.47 - 0.74**	**0.78 - 0.66**

Table 2. Experimental results. We report the AUC scores for the rotation and the translation errors for the five video sequences of our datasets. A * after the scores indicates that the method was re-initialized with the groundtruth for frame 500.

as us. Nevertheless, we believe the comparison highlights the strengths and weaknesses of the compared methods.

For every testing video, we compare the poses computed by each method for all frames. Following the evaluation framework in [36], we align each of the trajectories with respect to the same reference system. Then, we compute the absolute pose error for each frame, the empirical cumulative distribution functions for the rotation and translation components (as shown in Fig.10 for one of the videos), and report the Area Under Curve (AUC) scores for each of them in Table 2. The rotation error is computed as the distance between the exponential maps of the poses.

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.

5.4. 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 tracking.

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, on the other hand, 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.

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. Finally, PWP3D immediately looses tracking, while 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.

5.5. Runtimes

Our current implementation on an Intel Core i7-4820K desktop with GeForce GTX 780 Ti takes 22 ms for the part detection, plus 30 ms to predict the control points for each detected part. The pose estimation takes about 150 ms. Many optimizations are possible; for example, the control point predictions for each part could be run in parallel.

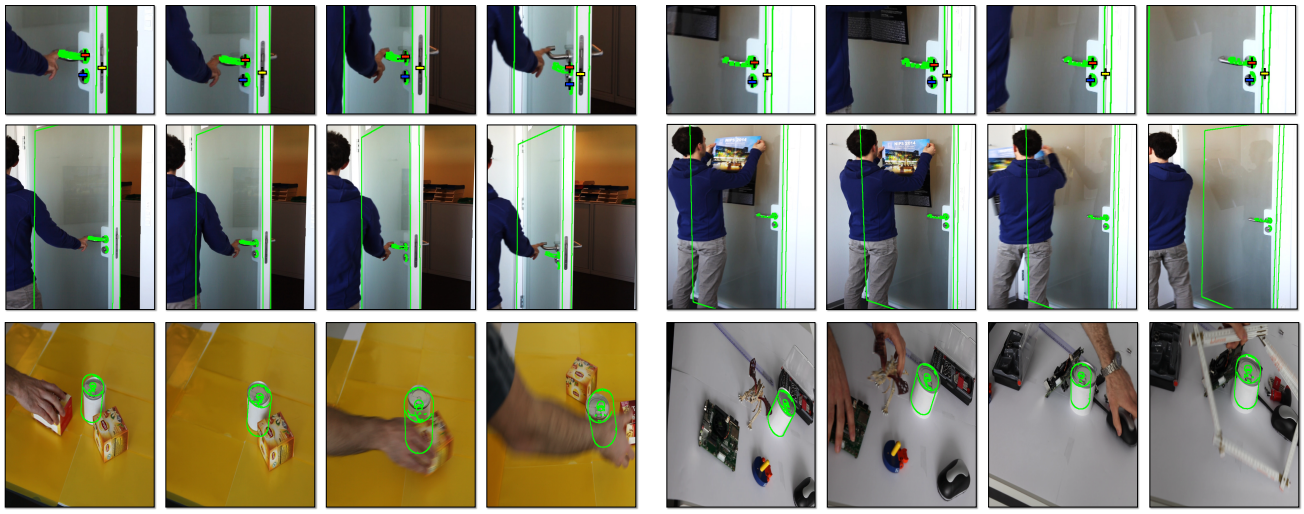
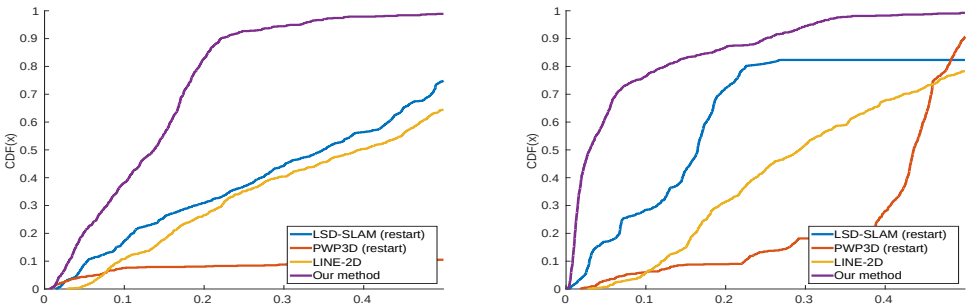


Figure 9. Results for the Door and the Can dataset. **Top-Middle:** Our method detects the door knob, the keyhole and the lock and successfully retrieves the correct pose of the door. **Bottom:** Our method correctly estimates the 3D pose of the can using the can tab only. The video sequences are provided as the supplementary material.



(a) Rotation error CDF: BOX dataset-video #1 (b) Translation error CDF: BOX dataset-video #1

Figure 10. The rotation and translation error Cumulative Distribution Functions (CDF) on the BOX dataset - Video #1: LSD-SLAM and PWP3D were both re-initialized with the groundtruth at frame 500.

6. Conclusion

We introduced a method for detecting an object and estimating its 3D pose in the challenging conditions that occur in practical applications. The core of our contribution is in the representation of the 3D pose of discriminative parts of the target objects. The parameters of this representation—the projections of the control points—can be inferred from images using statistical methods; each part provides enough information to estimate the object pose, and when several parts are visible, they can be easily combined to obtain a better estimate than a single part alone.

We believe that this representation, simple and powerful, could be useful not only for object instance detection, but also for the 3D pose estimation of categories of objects, where the current approaches drastically suffer from partial occlusions.

Acknowledgement: This work was supported in part by the EU projects EDUSAFE and MAGELLAN. The authors thank the anonymous reviewers for their helpful and constructive advice.

References

[1] 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. 1, 2

[2] G. Chliveros, M. Pateraki, and P. Trahanias. Robust Multi-Hypothesis 3D Object Pose Tracking. In *ICCV*, 2013. 2

[3] C. Choi, A. Trevor, and H. Christensen. RGB-D Edge Detection and Edge-Based Registration. In *IROS*, 2013. 2

[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. 1

[5] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model Globally, Match Locally: Efficient and Robust 3D Object Recognition. In *CVPR*, 2010. 1, 2

[6] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *ECCV*, 2014. 2, 6

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object Detection with Discriminatively Trained Part Based Models. *PAMI*, 2010. 2

[8] 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*, 47(6):2280–2292, 2014. 6

[9] 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. 4

[10] C. Harris and C. Stennett. RAPID-a Video Rate Object Tracker. In *BMVC*, pages 151–156, 1990. 2

[11] K. He, L. Sigal, and S. Sclaroff. Parameterizing Object Detectors in the Continuous Pose Space. In *ECCV*, 2014. 1

[12] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online Learning of Patch Perspective Rectification for Efficient Object Detection. In *CVPR*, 2008. 1, 2, 3

[13] 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*, 34(5):876–888, May 2012. 1, 2, 6

[14] G. Klein and D. Murray. Full-3D Edge Tracking with a Particle Filter. In *BMVC*, pages 1119–1128, 2006. 2

[15] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *ISMAR*, 2007. 2

[16] K. Koser and R. Koch. Perspectively Invariant Normal Features. In *ICCV*, 2007. 2, 3

[17] N. Kyriazis and A. Argyros. Scalable 3D Tracking of Multiple Interacting Objects. In *CVPR*, 2014. 1

[18] K. Lai, L. Bo, X. Ren, and D. Fox. A Scalable Tree-Based Approach for Joint Object and Pose Recognition. In *AAAI*, 2011. 2

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *IEEE*, 1998. 2

[20] J. Lim, A. Khosla, and A. Torralba. FPM: Fine Pose Parts-Based Model with 3D CAD Models. In *ECCV*, 2014. 1, 2

[21] D. G. Lowe. Fitting Parameterized Three-Dimensional Models to Images. *PAMI*, 13(5):441–450, June 1991. 2

[22] F. Markley, Y. Cheng, J. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(4):1193–1197, 2007. 6

[23] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose Priors for Simultaneously Solving Alignment and Correspondence. In *ECCV*, October 2008. 4, 5

[24] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense Tracking and Mapping in Real-Time. In *ICCV*, 2011. 2

[25] 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. 1, 2

[26] N. Payet and S. Todorovic. From Contours to 3D Object Detection and Pose Estimation. In *ICCV*, 2011. 2

[27] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D Geometry to Deformable Part Models. In *CVPR*, 2012. 1, 2

[28] V. Prisacariu and I. Reid. PWP3D: Real-Time Segmentation and Tracking of 3D Objects. *IJCV*, 98:335–354, 2012. 2, 6

[29] V. Prisacariu, A. Segal, and I. Reid. Simultaneous Monocular 2D Segmentation, 3D Pose Recovery and 3D Reconstruction. In *ACCV*, 2012. 2

[30] R. Rios-cabrera and T. Tuytelaars. Discriminatively Trained Templates for 3D Object Detection: A Real Time Scalable Approach. In *ICCV*, 2013. 1

[31] E. Rosten and T. Drummond. Fusing Points and Lines for High Performance Tracking. In *ICCV*, October 2005. 2

[32] 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. 2

[33] A. Shrivastava and A. Gupta. Building Part-Based Object Detectors via 3D Geometry. In *ICCV*, 2013. 1

[34] I. Skrypnyk and D. G. Lowe. Scene Modelling, Recognition and Tracking with Invariant Image Features. In *ISMAR*, pages 110–119, November 2004. 2

[35] S. Song and J. Xiao. Sliding Shapes for 3D Object Detection in Depth Images. In *ECCV*, 2014. 1

[36] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IROS*, 2012. 7

[37] D. Tan and S. Ilic. Multi-Forest Tracker: A Chameleon in Tracking. In *CVPR*, pages 1202–1209, 2014. 2

[38] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim. Latent-Class Hough Forests for 3D Object Detection and Pose Estimation. In *ECCV*, 2014. 1, 2

[39] F. Tombari, A. Franchi, and L. D. Stefano. BOLD Deatures to Detect Texture-Less Objects. In *ICCV*, 2013. 1

[40] S. Umeyama. Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *PAMI*, 13(4), 1991. 3

[41] L. Vacchetti, V. Lepetit, and P. Fua. Stable Real-Time 3D Tracking Using Online and Offline Information. *PAMI*, 26(10):1385–1391, October 2004. 2

[42] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose Tracking from Natural Features on Mobile Phones. In *ISMAR*, September 2008. 2

[43] G. Welch and G. Bishop. An Introduction to Kalman Filter. Technical report, Department of Computer Science, University of North Carolina, 1995. 4, 5

[44] P. Wohlhart and V. Lepetit. Learning Descriptors for Object Recognition and 3D Pose Estimation. In *CVPR*, 2015. 1

[45] Y. Xiang, C. Song, R. Mottaghi, and S. Savarese. Monocular Multiview Object Tracking with 3D Aspect Parts. In *ECCV*, 2014. 1, 2, 3