Learning 6D Object Pose Estimation Using 3D Object Coordinates

Eric Brachmann[1], Alexander Krull[1], Frank Michel[1], Stefan Gumhold[1],
Jamie Shotton[2], and Carsten Rother[1]

[1] TU Dresden, Dresden, Germany
[2] Microsoft Research, Cambridge, UK

Abstract. This work addresses the problem of estimating the 6D Pose of specific objects from a single RGB-D image. We present a flexible approach that can deal with generic objects, both textured and texture-less. The key new concept is a learned, intermediate representation in form of a dense 3D object coordinate labelling paired with a dense class labelling. We are able to show that for a common dataset with texture-less objects, where template-based techniques are suitable and state of the art, our approach is slightly superior in terms of accuracy. We also demonstrate the benefits of our approach, compared to template-based techniques, in terms of robustness with respect to varying lighting conditions. Towards this end, we contribute a new ground truth dataset with 10k images of 20 objects captured each under three different lighting conditions. We demonstrate that our approach scales well with the number of objects and has capabilities to run fast.

1 Introduction

The tasks of object instance detection and pose estimation are well-studied problems in computer vision. In this work we consider a specific scenario where the input is a single RGB-D image. Given the extra depth channel it becomes feasible to extract the full 6D pose (3D rotation and 3D translation) of rigid object instances present in the scene. The ultimate goal is to design a system that is fast, scalable, robust and highly accurate and works well for generic objects (both textured and texture-less) present in challenging real-world settings, such as cluttered environments and with variable lighting conditions.

For many years the main focus in the field of detection and 2D/6D pose estimation of rigid objects has been limited to objects with sufficient amount of texture. Based on the pioneering work of [11,15], practical, robust solutions have been designed which scale to large number of object instances [18,20]. For textured objects the key to success, for most systems, is the use of a sparse representation of local features, either hand crafted, e.g. SIFT features, or trained from data. These systems run typically a two-stage pipeline: a) putative sparse feature matching, b) geometric verification of the matched features.

Recently, people have started to consider the task of object instance detection for texture-less or texture-poor rigid objects, e.g. [7,8,21]. For this particular challenge it has been shown that template-based techniques are superior. The main

D. Fleet et al. (Eds.): ECCV 2014, Part II, LNCS 8690, pp. 536–551, 2014.

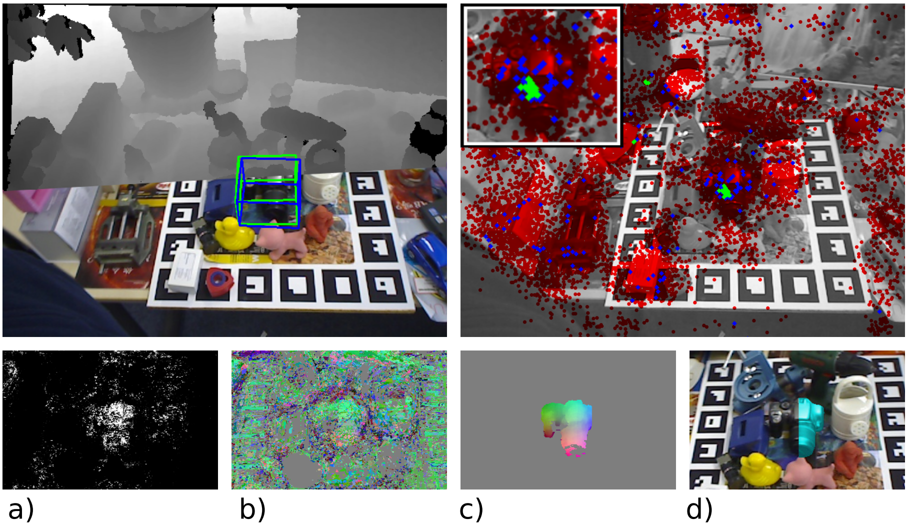

Fig. 1. Overview of our system. Top left: RGB-D Test image (upper-half depth image and lower-half RGB image). The estimated 6D pose of the query object (camera) is illustrated with a blue bounding box, and the respective ground truth with a green bounding box. Top right: Visualization of the algorithms search for the optimal pose, where the inlet is a zoom of the centre area. The algorithm optimizes our energy in a RANSAC-like fashion over a large, continuous 6D pose space. The 6D poses, projected to the image plane, which are visited by the algorithm are color coded: *red poses* are disregarded in a very fast geometry check; *blue poses* are evaluated using our energy function during intermediate, fast sampling; *green poses* are subject to the most expensive energy refinement step. Bottom, from left to right: (a) Probability map for the query object, (b) predicted 3D object coordinates from a single tree mapped to the RGB cube, (c) corresponding ground truth 3D object coordinates, (d) overlay of the *3D model in blue* onto the test image (rendered according to the estimated pose)

focus of these works has been to show that template-based techniques can be made very fast, by using specific encodings [8] or additionally a cascaded framework [21]. The typical problems of template-based techniques, such as not being robust to clutter and occlusions as well as changing lighting conditions, have been partly overcome by carefully hand-crafted templates and additional discriminative learning. Nevertheless, template-based techniques have in our view two fundamental shortcomings. Firstly, they match the complete template to a target image, i.e. encode the object in a particular pose with one “global” feature. In contrast to this, sparse feature-based representations for textured objects are “local” and hence such systems are more robust with respect to occlusions. Secondly, it is an open challenge to make template-based techniques work for articulated or deformable object instances, as well as object classes, due to the growing number of required templates.

Our approach is motivated by recent work in the field of articulated human pose estimation from a pre-segmented RGB-D image [28]. The basic idea in [28] is not to predict directly the 60-DOF human pose from an image, but to first regress an intermediate so-called object coordinate representation. This means that each pixel in the image votes for a continuous coordinate on a canonical body in a canonical pose, termed the Vitruvian Manifold. The voting is done by a random forest and uses a trained assemble of simple, local feature tests. In the next step a "geometric validation" is performed, by which an energy function is defined that compares these correspondences with a parametric body model. Finally, the pose parameters are found by energy minimization. Hence, in spirit, this is akin to the two-stage pipeline of traditional, sparse feature-based techniques but now with densely learned features. Subsequent work in [24] applied a similar idea to 6D camera pose estimation, showing that a regression forest can accurately predict image-to-world correspondences that are then used to drive a camera pose estimaten. They showed results that were considerably more accurate than a sparse feature-based baseline.

Our system is based on these ideas presented in [28,24] and applies them to the task of estimating the 6D pose of specific objects. An overview of our system is presented in Fig. 1. However, we cannot apply [28,24] directly since we additionally need an object segmentation mask. Note that the method in [28] can rely on a pre-segmented human shape, and [24] does not require segmentation. To achieve this we jointly predict a dense 3D object coordinate labelling and a dense class labelling. Another major difference to [24] is a clever sampling scheme to avoid unnecessary generation of false hypotheses in the RANSAC-based optimization.

To summarize, the **main contribution** of our work is a new approach that has the benefits of local feature-based object detection techniques and still achieves results that are even slightly superior, in terms of accuracy, to template-based techniques for texture-less object detection. This gives us many conceptual and practical advantages. Firstly, one does not have to train a separate system for textured and texture-less objects. Secondly, one can use the same system for rigid and non-rigid objects, such as laptops, scissors, and objects in different states, e.g. a pot with and without lid. Thirdly, by using local features we gain robustness with respect to occlusions. Fourthly, by applying a rigorous feature learning framework we are robust to changes in lighting conditions. Fig. 2 shows the benefits of our system. The main technical contribution of our work is the use of a new representation in form of a joint dense 3D object coordinate and object class labelling. An additional minor contribution is a new dataset of 10k images of 20 objects captured each under three different lighting conditions and labelled with accurate 6D pose, which will be made publicly available.

2 Related Work

There is a vast literature in the area of pose estimation and object detection, including instance and category recognition, rigid and articulated objects, and

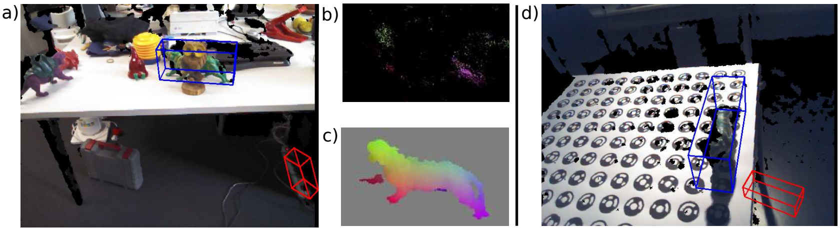

Fig. 2. Our method is able to find the correct pose, where a template-based method fails. (a) Test image showing a situation with strong occlusion. The pose estimate by our approach is shown in blue. The pose estimated by our reimplementation of the method by Hinterstoisser et al. from [8] is shown in red. (b) The coordinate predictions for a from one tree mapped to the RGB-cube and multiplied with $p_{c,i}$. (c) Ground truth object coordinates for a mapped to the RGB-cube. (d) Test image showing extreme light conditions, different from the training set. Estimated poses are displayed as in a.

coarse (quantized) and accurate (6D) poses. In the brief review below, we focus on techniques that specifically address the detection of instances of rigid objects in cluttered scenes and simultaneously infer their 6D pose. Some of the work was already mentioned above.

Template-Based Approaches. Perhaps the most traditional approach to object detection is to use templates, e.g. [11,26,7,8]. This means a rigid template is scanned across the image, and a distance measure is computed to find the best match. As the state of the art in template-based approaches, [8] uses synthetic renderings of a 3D object model to generate a large number of templates covering the full view hemisphere. They employ an edge-based distance metric which works well for textureless objects, and refine their pose estimates using ICP to achieve an accurate 6D pose. Such template-based approaches can work accurately and quickly in practice. The limitations of template-based approaches were discussed above.

Sparse Feature-Based Approaches. A popular alternative to templates are sparse feature-based approaches. These extract points of interest (often scale-invariant) from the image, describe these with local descriptors (often affine and illumination invariant), and match to a database. For example, Lowe [15] used SIFT descriptors and clustered images from similar viewpoints into a single model. Another great example of a recent, fast and scalable system is [16]. Sparse techniques have been shown to scale well to matching across vast vocabularies [18,20]. More recently a trend has been to *learn* interest points [22,10], descriptors [30], and matching [14,19,1]. Despite their popularity, a major limitation of sparse approaches for real applications is that they require sufficiently textured objects. Our approach instead can be applied densely at every image pixel regardless of texture, and can learn what are the most appropriate image features to exploit. Note that there is also large literature on contour and shape matching, which can deal with texture-less objects, e.g. [29], which is, however, conceptually different to our work.

Dense Approaches. An alternative to templates and sparse approaches are dense approaches. In these, every pixel produces some prediction about the desired output. In the generalized Hough voting scheme, all pixels cast a vote in some quantized prediction space (e.g. 2D object center and scale), and the cell with the most votes is taken as the winner. In [27,5], Hough voting was used for object detection, and was shown able to predict coarse object poses. In our work we borrow an idea from Gall et al. [5] to jointly train an objective over both Hough votes and object segmentations. However, in contrast to [5] we found a simple joint distribution over the outputs (in our case 3D object coordinates and object class labeling) to perform better than the variants suggested in [5]. Drost et al. [4] also take a voting approach, and use oriented point pair features to reduce the search space. To obtain a full 6D pose, one could imagine a variant of [5] that has every pixel vote directly for a *global* quantized 6D pose. However, the high dimensionality of the search space (and thus the necessary high degree of quantization) is likely to result in a poor estimate of the pose. In our approach, each pixel instead makes a 3D continuous prediction about only its *local* correspondence to a 3D model. This massively reduces the search space, and, for learning a discriminative prediction, allows a much reduced training set since each point on the surface of the object does not need to be seen from every possible angle. We show how these 3D object correspondences can efficiently drive a subsequent model fitting stage to achieve a highly accurate 6D object pose.

Finally, there are approaches for object class detection that use a similar idea as our 3D object coordinate representation. One of the first systems is the 3D LayoutCRF [9] which considers the task of predicting a dense part-labelling, covering the 3D rigid object, using a decision forest, though they did not attempt to fit a 3D model to those labels. After that the Vitruvian Manifold [28] was introduced for human pose estimation, and recently the scene coordinate regression forests was introduced for camera re-localization in RGB-D images [24]. Both works were discussed in detail above.

3 Method

We first describe our decision forest that jointly predicts both 3D object coordinates and object instance probabilities. Then we will discuss our energy function which is based on forest output. Finally, we will address our RANSAC based optimization scheme.

3.1 Random Forest

We use a single decision forest to classify pixels from an RGB-D image. A decision forest is a set $\mathcal{T}$ of decision trees T^j. Pixels of an image are classified by each tree T^j and end up in one of the tree's leafs l^j. Our forest is trained in a way that allows us to gain information about which object $c \in C$ the pixel i might belong to, as well as what might be its position on this object. We will denote a pixel's position on the object by $\mathbf{y}_i$ and refer to it as the pixel's *object coordinate.* Each leaf l^j stores a distribution over possible object affiliations $p(c|l^j)$, as well as a set of object coordinates $\mathbf{y}_c(l^j)$ for each possible object affiliation c. The term $\mathbf{y}_c(l^j)$ will be referred to as *coordinate prediction.* In the following we only discuss the interesting design decisions which are specific to our problem and refer the reader to the supplementary material for a detailed description.

Design and Training of the Forest. We build the decision trees using a standard randomized training procedure [2]. We quantized the continuous distributions $p(\mathbf{y}|c)$ into $5 \times 5 \times 5 = 125$ discrete bins. We use an additional bin for a background class. The quantization allows us to use the standard information gain classification objective during training, which has the ability to cope better with the often heavily multi-model distributions $p(\mathbf{y}|c)$ than a regression objective [6]. As a node split objective that deals with both our discrete distributions, $p(c|l^j)$ and $p(\mathbf{y}|c, l^j)$, we use the information gain over the joint distribution. This has potentially $125|C| + 1$ labels, for $|C|$ object instances and background, though many bins are empty and the histograms can be stored sparsely for speed. We found the suggestion in [5] to mix two separate information gain criteria to be inferior on our data.

An important question is the choice of features evaluated in the tree splits. We looked at a large number of features, including normal, color, etc. We found that the very simple and fast to compute features from [24] performed well, and that adding extra feature types did not appear to give a boost in accuracy (but did slow things down). The intuitive explanation is that the learned combination of simple features in the tree is able to create complex features that are specialized for the task defined by the training data and splitting objective. The features in [24] consider depth or color differences from pixels in the vicinity of pixel i and capture local patterns of context. The features are depth-adapted to make them largely depth invariant [23]. Each object is segmented for training. If a feature test reaches outside the object mask, we have to model some kind of background to calculate feature responses. In our experiments we will use uniform noise or a simulated plane the object sits on. We found this to work well and to generalize well to new unseen images. Putting objects on a plane allows the forest to learn contextual information.

For training, we use randomly sampled pixels from the segmented object images and a set of RGB-D background images. After training the tree structure based on quantized object coordinates, we push training pixels from all objects through the tree and record all the continuous locations $\mathbf{y}$ for each object c at each leaf. We then run mean-shift with a Gaussian kernel and bandwidth 2.5cm. We use the top mode as prediction $\mathbf{y}_c(l^j)$ and store it at the leaf. We furthermore store at each leaf the percentage of pixels coming from each object c to approximate the distribution of object affiliations $p(c|l^j)$ at the leaf. We also store the percentage of pixels from the background set that arrived at l^j, and refer to it as $p(bg|l^j)$.

Using the Forest. Once training is complete we push all pixels in an RGB-D image through every tree of the forest, thus associating each pixel i with a distribution $p(c|l_i^j)$ and one prediction $\mathbf{y}_c(l_i^j)$ for each tree j and each object c. Here l_i^j is the leaf outcome of pixel i in tree j. The leaf outcome of all trees for a pixel i is summarized in the vector $\mathbf{l}_i = (l_i^1, \ldots, l_i^j, \ldots, l_i^{|\mathcal{T}|})$. The leaf outcome of the image is summarized in $L = (\mathbf{l}_1, \ldots, \mathbf{l}_n)$. After the pixels have been classified we calculate for the object c we are interested in and for each pixel i in the image a number $p_{c,i}$ by combining the $p(c|l_i^j)$ stored at the leafs l_i^j. The number $p_{c,i}$ can be seen as the approximate probability $p(c|\mathbf{l}_i)$, that a pixel i belongs to object c given it ended up in all its leaf nodes $\mathbf{l}_i = (l_i^1, \ldots, l_i^j, \ldots, l_i^{|\mathcal{T}|})$. We will thus refer to the number $p_{c,i}$ as object probability. We calculate the object probability as

$$p_{c,i} = \frac{\prod_{j=1}^{|\mathcal{T}|} p(c|l_i^j)}{\prod_{j=1}^{|\mathcal{T}|} p(bg|l_i^j) + \sum_{\hat{c} \in C} \prod_{j=1}^{|\mathcal{T}|} p(\hat{c}|l_i^j)}. \tag{1}$$

A detailed deduction for Eq. 1 can be found in the supplementary material.

3.2 Energy Function

Our goal is to estimate the 6 DOF pose H_c for an object c. The pose H_c is defined as the rigid body transformation (3D rotation and 3D translation) that maps a point from object space into camera space. We formulate the pose estimation as an energy optimization problem. To calculate the energy we compare synthetic images rendered using H_c with the observed depth values $D = (d_1, \ldots, d_n)$ and the results of the forest $L = (\mathbf{l}_1, \ldots, \mathbf{l}_n)$. Our energy function is based on three components:

$$\hat{E}_c(H_c) = \lambda^{\mathrm{depth}} E_c^{\mathrm{depth}}(H_c) + \lambda^{\mathrm{coord}} E_c^{\mathrm{coord}}(H_c) + \lambda^{\mathrm{obj}} E_c^{\mathrm{obj}}(H_c). \tag{2}$$

While the component $E_c^{\mathrm{depth}}(H)$ punishes deviations between the observed and ideal rendered depth images, the components $E_c^{\mathrm{coord}}(H)$ and $E_c^{\mathrm{obj}}(H)$ punish deviations from the predictions of the forest. Fig. 3 visualizes the benefits of each component. The parameters λ^{depth}, λ^{coord} and λ^{obj} reflect the reliability of the different observations. We will now describe the components in detail.

The Depth Component. This component is defined as

$$E_c^{\mathrm{depth}}(H_c) = \frac{\sum_{i \in M_c^D(H_c)} f(d_i, d_i^*(H_c))}{|M_c^D(H_c)|}, \tag{3}$$

where $M_c^D(H_c)$ is the set of pixels belonging to object c. It is derived from the pose H_c by rendering the object into the image. Pixels with no depth observation d_i are excluded. The term $d_i^*(H_c)$ is the depth at pixel i of our recorded 3D model for object c rendered with pose H_c. In order to handle inaccuracies in the 3D model we use a robust error function: $f(d_i, d_i^*(H)) = \min\left(||\mathbf{x}(d_i) - \mathbf{x}(d_i^*(H))||, \tau_d\right) / \tau_d$, where $\mathbf{x}(d_i)$ denotes the 3D coordinates in the

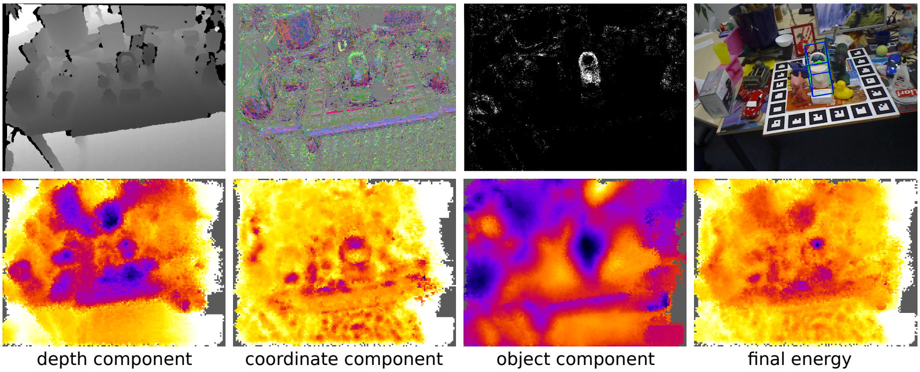

Fig. 3. Benefits of the different energy components. While different energy components display strong local minima, their combination usually shows the strongest minimum at the correct pose. The energies were calculated for different poses and projected into image space using minimum projection. White stands for high energy dark blue for low energy. Each component is displayed below data it is related to. Left to right: depth component of test image together with $E_c^{\text{depth}}(H_c)$, predicted object coordinates from one of the trees with $E_c^{\text{coord}}(H_c)$, object probabilities with $E_c^{\text{object}}(H_c)$, the RGB-components of test image is displayed with the final energy $E_c(H_c)$. The *estimated pose (blue)* and *ground truth pose (green)* are shown as bounding box.

camera system derived from the depth d_i. The denominator in the definition normalizes the depth component to make it independent of the object's distance to the camera.

The Object Component. This component punishes pixels inside the ideal segmentation M_c^D which are, according to the forest, unlikely to belong to the object. It is defined as

$$E_c^{\text{obj}}(H_c) = \frac{\sum_{i \in M_c^D(H_c)} \sum_{j=1}^{m} - \log p(c|l_i^j)}{|M_c^D(H_c)|}. \tag{4}$$

The Coordinate Component. This component punishes deviations between the object coordinates $\mathbf{y}_c(l_i^j)$ predicted by the forest and ideal object coordinates $\mathbf{y}_{i,c}(H_c)$ derived from a rendered image. The component is defined as

$$E_c^{\text{coord}}(H_c) = \frac{\sum_{i \in M_c^L(H_c)} \sum_{j=1}^{m} g(\mathbf{y}_c(l_i^j), \mathbf{y}_{i,c}(H_c))}{|M_c^L(H_c)|}. \tag{5}$$

where $M_c^L(H_c)$ is the set of pixels belonging to the object c excluding pixels with no depth observation d_i and pixels where $p_{c,i} < \tau_{pc}$. The latter is necessary because we find that pixels with small $p_{c,i}$ do not provide reliable coordinate predictions $\mathbf{y}_c(l_i^j)$. The term $\mathbf{y}_{i,c}(H_c)$ denotes the coordinates in object space at pixel i of our 3D model for object c rendered with pose H_c. We again use a robust error function $g(\mathbf{y}_c(l_i^j), \mathbf{y}_{i,c}(H_c)) = \min\left(||\mathbf{y}_c(l_i^j) - \mathbf{y}_{i,c}(H_c)||^2, \tau_y\right)/\tau_y$.

Final Energy Function. Since our energy terms are all normalized, stability can be an issue whenever the number of pixels to be considered becomes very small. To address the problem we use the following stable formulation:

$$E_c(H_c) = \begin{cases} \hat{E}_c(H_c), & \text{if } |M_c^L(H_c)| > 100 \\ \infty, & \text{otherwise} \end{cases} \tag{6}$$

3.3 Optimization

In order to find the solution to the task in Eq. 6 we use a RANSAC-based algorithm. It samples pose hypotheses based on observed depth values and the coordinate predictions from the forest. Subsequently, these hypotheses are evaluated and refined. A visualization of the process can be found in Fig. 1. We will now describe the procedure in detail.

Sampling of a Pose Hypothesis. We first draw a single pixel i_1 from the image using a weight proportional to the previously calculated $p_{c,i}$ each pixel i. We draw two more pixels i_2 and i_3 from a square window around i_1 using the same method. The width of the window is calculated from the diameter of the object and the observed depth value d_{i_1} of the pixel $w = f\delta_c/d_i$ where $f = 575.816$ pixels is the focal length. Sampling is done efficiently using an integral image of $p_{c,i}$. We randomly choose a tree index j_1, j_2 and j_3 for each pixel. Finally, we use the Kabsch algorithm to calculate the pose hypothesis H_c from the 3D-3D-correspondences $(\mathbf{x}(i_1), \mathbf{y}_c(l_{i_1}^{j_1}))$, $(\mathbf{x}(i_2), \mathbf{y}_c(l_{i_2}^{j_2}))$ and $(\mathbf{x}(i_3), \mathbf{y}_c(l_{i_3}^{j_3}))$.

We map each of the three predicted positions $\mathbf{y}_c(l_{i_\bullet}^{j_\bullet})$ into camera space using H_c and calculate a transformation error $e_{i_\bullet,j_\bullet}(H_c) = ||\mathbf{x}(i_\bullet) - H_c\mathbf{y}_c(l_{i_\bullet}^{j_\bullet})||$, which is simply the Euclidean distance to their counterpart. We accept a pose hypothesis H_c only if none of the three distances is larger than 5% of the object's diameter δ_c. The process is repeated until a fixed number of 210 hypotheses are accepted. All accepted hypotheses are evaluated according to Eq. 6.

Refinement. We refine the top 25 accepted hypotheses. To refine a pose H_c we iterate over the set of pixels $M_c^D(H_c)$ supposedly belonging to the object c as done for energy calculation. For every pixel $i \in M_c^D(H_c)$ we calculate the error $e_{i,j}(H_c)$ for all trees j. Let $\hat{j}$ be the tree with the smallest error $e_{i,\hat{j}}(H_c) \leq e_{i,j}(H_c) | \forall j \in \{1, \ldots, |\mathcal{T}|\}$ for pixel i. Every pixel i where $e_{i,\hat{j}}(H_c) < 20\text{mm}$ is considered an inlier. We store the correspondence $(\mathbf{x}(i_1), \mathbf{y}_c(l_i^{\hat{j}}))$ for all inlier pixels and use them to reestimate the pose with the Kabsch algorithm. The process is repeated until the energy of the pose according Eq. 6 no longer decreases, the number of inlier pixels drops below 3, or a total of 100 iterations is reached.

The Final Estimate. The pose hypothesis with the lowest energy after refinement is chosen as final estimate. The estimates in Figs. 1 to 3 as well as our quantitative results in the experiments section were obtained using the exact algorithm described above. Our formulation of the task as energy optimization problem, however, allows for the use of any general optimization algorithm to further increase the precision of the estimate.

4 Experiments

Several object instance detection datasets have been published in the past [21,3], many of which deal with 2D poses only. Lai et al. [13] published a large RGB-D dataset of 300 objects that provides ground truth poses in the form of approximate rotation angles. Unfortunately, such annotations are to coarse for the accurate pose estimation task we try to solve. We evaluated our approach on the recently introduced Hinterstoisser et al. [8] dataset and our own dataset. The Hinterstoisser dataset provides synthetic training and real test data. Our dataset provides real training and real test data with realistic noise patterns and challenging lighting conditions. On both datasets we compare to the template-based method of [8]. We also tested the scalability of our method and comment on running times. In the supplementary material we provide additional experimental results for an occlusion dataset, for a detection task, and regarding the contribution of our individual energy terms. We train our decision forest with the following parameters. At each node we sample 500 color features and depth features. In each iteration we choose 1000 random pixels per training image, collect them in the current leafs and stop splitting if less than 50 pixels arrive. The tree depth is not restricted. A complete set of parameters can be found in the supplement.

Dataset of Hinterstoisser et al. Hinterstoisser et al. [8] provide colored 3D models of 13 texture-less objects[1] for training, and 1000+ test images of each object on a cluttered desk together with ground truth poses. The test images cover the upper view hemisphere at different scales and a range of ±45° in-plane rotation. The goal is to evaluate the accuracy in pose estimation for one object per image. It is known which object is present. We follow exactly the test protocol of [8] by measuring accuracy as the fraction of test images where the pose of the object was estimated correctly. The tight pose tolerance is defined in the supplementary material. In [8] the authors achieve a strong baseline of 96.6% correctly estimated poses, on average. We reimplemented their method and were able to reproduce these numbers. Their pipeline starts with an efficient template matching schema, followed by two outlier removal steps and iterative closest point adjustment. The two outlier removal steps are crucial to achieve the reported results. In essence they comprise of two thresholds on the color and depth difference, respectively, between the current estimate and the test image. Unfortunately the correct values differ strongly among objects and have to be set by hand for each object[2]. We also compare to [21] who optimize the Hinterstoisser templates in a discriminative fashion to boost performance and speed. They also rely on the same two outlier removal checks but learn the object dependent thresholds discriminatively.

To produce training data for our method we rendered all 13 object models with the same viewpoint sampling as in [8], but skipped scale variations because

[1] We had to omit 2 objects since proper 3D models were missing.

[2] We verified this in private communication with the authors. These values are not given in the article.

of our depth-invariant features. Since our features may reach outside the object segmentation during training we need a background model to compute sensible feature responses. For our color features we use randomly sampled colors from a set of background images. The background set consists of approx. 1500 RGB-D images of cluttered office scenes recorded by ourselfs. For our depth features we use an infinite synthetic ground-plane as background model. In the test scenes all objects stand on a table but embedded in dense clutter. Hence, we regard the synthetic plane as an acceptable prior. Additionally, we also show results for a background model of uniform depth noise, and uniform RGB noise. The decision forest is trained for all 13 objects and a background class, simultaneously. For the background class we sample RGB-D patches from our office background set. To account for variance in appearance between purely synthetic training images and real test images we add Gaussian noise to the response of the color feature[25]. After optimizing our energy, we deploy no outlier removal steps, in contrast to [8,21].

Table 1. Results on the Hinterstoisser et al. dataset with synthetic training data, real training data and different background models (plane, noise). We see that our approach is consistently superior to [8,21].

	Synth. Training				Real Training	
	Linemod[8]	DTT-3D[21]	Our(plane)	Our(noise)	Our(plane)	Our(noise)
Avg.	96.6%	97.2%	98.3%	92.6%	98.1%	97.4%
Med.	97.1%	97.5%	98.9%	92.1%	99.6%	98.8%
Max.	99.9%	99.8%	100.0%	99.7%	100.0%	100%
Min.	91.8%	94.2%	95.8%	84.4%	91.1%	89.2%

Table 1 summarizes the results. We score an average of 98.3% with the synthetic plane background model. Hence we improve on both systems of [8] and [21]. See Fig. 4 for qualitative results. Using uniform noise as background model, we still report excellent results with 92.6% correctly estimated poses on average.

To verify that our approach is not restricted to synthetic training data, we performed an experiment where we trained with real images for each object. Since the dataset includes only one scene per object, we had to split each sequence into training and test. We sampled training images with at least 15° angular distance, to have an approximately regular coverage of the upper hemisphere similar to the Hinterstoisser et al. setup. The maximum distance of training images is $\approx 25°$ making this test slightly harder than in the synthetic setup. All other images are test images. To remove the background in the training images we do an intersection test with the 3D object bounding box. We substitute the background pixels with the two background model variants already discussed above. We do not add noise to the feature responses. In this experiment, we observe excellent accuracy which is stable even with the simple noise background model (compare right two columns in Table 1).

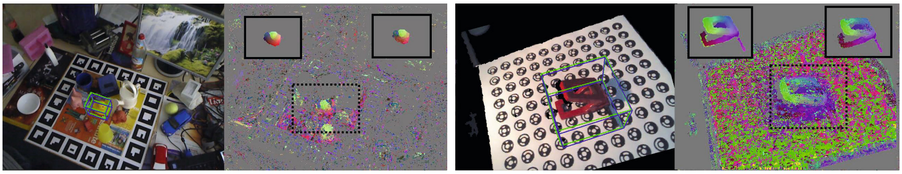

Fig. 4. Examples for pose estimation with our system (blue bounding box) versus the ground truth pose (green bounding box). The left test image shows an object from the Hinterstoisser et al. dataset[8], the right test image shows an object from our dataset. Next to each test image are the predicted object coordinates $\mathbf{y}$ from one tree of the forest. The inlay figures show the ground truth object coordinates (left) and the best object coordinates (right), where "best" is the best prediction of all trees with respect to ground truth (for illustration only).

Our Dataset. We recorded 20 textured and texture-less objects under three different lighting conditions: bright artificial light (*bright*), darker natural light (*dark*), and directional spot light (*spot*). For each light setting we recorded each object on a marker board in a motion that covers its upper view hemisphere. The distance to the object varied during the recording but the in-plane rotation was kept fixed. We added in-plane rotation artificially afterwards in the range of ±45°. We used KinectFusion [17,12] to record the external camera parameters for each frame. This serves as pose ground truth and is used to generate the object coordinates per pixel for training the decision forest. Recordings of the same object but different lighting conditions were registered using the marker board. Images that were used for training were segmented with the 3D object bounding box. An overview over the dataset and details about the recording procedure can be found in the supplement. We sampled training images with at least 15° angular distance. The maximal angular distance of training images is $\approx 25°$. We did not place our objects on a synthetic plane, because they were already recorded on a planar board. Depth features reaching outside the object mask during training will just use the depth in the original training image. For color features we sampled randomly from another set of office backgrounds that do not contain our objects.

To evaluate how well our approach generalizes with respect to varying lighting conditions, we trained our decision forest with the *bright* and *dark* training sets. Again we added Gaussian noise to the response of the color feature for robustness. In a first run we tested with images of the *bright* set that were not used for training. Here, the forest did not need to generalize to new lighting conditions but only to unseen views, which it does with excellent accuracy (avg. 95%, see Table 2). As before we measured performance as the percentage of correctly estimated poses of one object per test image which is always present. In a second run we tested with the complete *spot* set to demonstrate the capability of generalization to a difficult new lighting condition. We report an average rate of correctly estimated poses of 88.2%.

To demonstrate that the template based approach of Linemod[8] does not generalize as well with respect to lighting change we used our re-implementation to extract templates for one object based on the training set described above. Note, that the training set contains each view only with one scale. This can be problematic for Linemod if the test-data shows scale variation not covered by the training data. Hence, we render each training image from 2 larger and 3 smaller distances in 10cm steps. This gives us 6 different scales for each training image similar to the setup in [8]. As in [8] we tuned the outlier removal parameters by hand. However, we found that we had to disable these tests completely to get any detections under new lighting conditions. In a validation run we extracted templates from the *bright* training set and tested on the *bright* test set. Following the procedure of [8], we can estimate correct poses in 80.1% of the images. We account the difference to the performance on the Hinterstoisser dataset[8] to the fact that the object is textured and that our images are noisy. If we test with the same templates on the *spot* set, performance drops to 57.1%. Since our tree has seen both *bright* and *dark* in training we apply the following testing procedure to Linemod for a fair comparison. We also extract templates from the *dark* training set and apply it to the *spot* test set, observing 55.3%. For the final score, we consider an image solved by Linemod if one of the template sets, *dark* or *bright*, lead to the correct pose. Then we observe accuracy of 70.2%. So even under testing conditions in favor of Linemod performance drops by 10%. On the same object, we report accuracy of 96.9% on the *bright* test set (included in training lighting), and 91.8% on the *spot* test set (not seen in training).

Table 2. Accuracy on our dataset when testing with different lighting conditions. *Bright* lighting appeared in the training set, whereas *spot* lighting did not. We report average and median accuracy for our 20 objects. We also compare to Linemod [8] on one object. Details are given in the text.

	All		Toy(Battle Cat)			
Test Condition	Avg.	Med.	Our	[8](*dark*)	[8](*bright*)	[8](combined)
bright	95.6%	97.7%	96.9%	-	80.1%	-
spot	88.2%	93.0%	91.8%	55.3%	57.1%	70.2%

Scalability. We show the potential of our method with respect to scalability in two different ways: scalability in the object count, and scalability in the space of poses. The first concerns the number of objects the system is able to identify, while the latter concerns the range of poses it can recognize. We start with a forest that was trained for 5 objects, and a set of training images sampled form *dark* and *bright* lighting conditions, with an angular distance of min. 45°. We add ±45° in-plane rotation to each training image. During test, we consider images of the *spot* set which are at maximum 10° apart from the closest training image. This results in the same test difficulty as in the previous experiments. Performance is measured for one object (Stuffed Cat). We modify this setup in two ways. Firstly, we increased the object count to 30 by combining our dataset with real images of the Hinterstoisser dataset. We sampled the Hinterstoisser sets to have approximately the same amount of training images for our objects and the additional Hinterstoisser objects. Secondly, we increased the number of in-plane rotated training images 4-fold to the full ±180°. The results are shown in Fig. 5.

	default	6×objects	4×poses
Forest time	102ms	138ms	113ms
Opt. time	443ms	424ms	517ms
Accuracy	95%	94%	95%

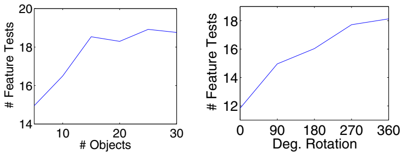

Fig. 5. Left: Running time of our system with increasing object count and pose range. Accuracy stays stable. Right: Illustration of the sub-linear growth of the decision forest.

As the number of objects and the range of poses increase, the evaluation time of the tree does increase slightly, but considerably less than 6× resp. 4×. The runtime of the energy optimization is effected slightly due to variation in the forest prediction, and the accuracy of the system stays stable. Below the table in Fig. 5 we plot the sub-linear growth in the average number of feature tests per pixel with increasing object number and range of poses. Our proposed pipeline is linear in the number of objects but we demonstrate that with the forest the first essential step of our discriminatively trained method behaves sub-linearly in the number of objects.

Running Times. The complete running time of our pose estimation approach is the sum of forest prediction time and energy optimization time. Forest predictions are generated once per frame and the results are reused for every object in that frame. Our CPU implementation of the random forests takes 160ms avg. per frame on the dataset of [8]. Based on these predictions, energy optimization is done per object. We implemented energy evaluation on the GPU and report 398ms avg. per object on the dataset of [8] with the parameter settings suggested above. However, we found that a set of reduced parameters results in a large speed up while maintaining accuracy. We reduced the number of hypotheses from 210 to 42, the number of refinement steps from 100 to 20 and refined only the best 3 hypotheses. This still achieves 96.4% avg. accuracy on the dataset of [8] while reducing the average energy optimization time to 61ms.

Acknowledgements. This work has partially been supported by the European Social Fund and the Federal State of Saxony within project VICCI (#100098171). We thank Holger Heidrich for his reimplementation of Linemod. We also thank Stephan Ihrke, Daniel Schemala, Patrick Sprung and Sören König for their help preparing the different datasets und their contributions to our implementation.

References

1. Bo, L., Ren, X., Fox, D.: Unsupervised feature learning for RGB-D based object recognition. In: ISER (2012)
2. Criminisi, A., Shotton, J.: Decision Forests for Computer Vision and Medical Image Analysis. Springer (2013)
3. Damen, D., Bunnun, P., Calway, A., Mayol-Cuevas, W.: Real-time learning and detection of 3D texture-less objects: A scalable approach. In: BMVC (2012)
4. Drost, B., Ulrich, M., Navab, N., Ilic, S.: Model globally, match locally: Efficient and robust 3D object recognition. In: CVPR (2010)
5. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough Forests for object detection, tracking, and action recognition. IEEE Trans. on PAMI 33(11) (2011)
6. Girshick, R., Shotton, J., Kohli, P., Criminisi, A., Fitzgibbon, A.: Efficient regression of general-activity human poses from depth images. In: ICCV (2011)
7. Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., Lepetit, V.: Gradient response maps for real-time detection of texture-less objects. IEEE Trans. on PAMI (2012)
8. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Lee, K.M., Matsushita, Y., Rehg, J.M., Hu, Z. (eds.) ACCV 2012, Part I. LNCS, vol. 7724, pp. 548–562. Springer, Heidelberg (2013)
9. Hoiem, D., Rother, C., Winn, J.: 3D LayoutCRF for multi-view object class recognition and segmentation. In: CVPR (2007)
10. Holzer, S., Shotton, J., Kohli, P.: Learning to efficiently detect repeatable interest points in depth data. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 200–213. Springer, Heidelberg (2012)
11. Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the hausdorff distance. IEEE Trans. on PAMI (1993)
12. Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., Fitzgibbon, A.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: UIST (2011)
13. Lai, K., Bo, L., Ren, X., Fox, D.: A large-scale hierarchical multi-view rgb-d object dataset. In: ICRA. IEEE (2011)
14. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. IEEE Trans. on PAMI 28(9) (2006)
15. Lowe, D.G.: Local feature view clustering for 3d object recognition. In: CVPR (2001)
16. Martinez, M., Collet, A., Srinivasa, S.S.: Moped: A scalable and low latency object recognition and pose estimation system. In: ICRA (2010)
17. Newcombe, R., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: ISMAR (2011)
18. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)
19. Ozuysal, M., Calonder, M., Lepetit, V., Fua, P.: Fast keypoint recognition using random ferns. IEEE Trans. on PAMI (2010)
20. Philbin, J., Chum, O., Isard, M., Sivic, J., Zisserman, A.: Object retrieval with large vocabularies and fast spatial matching. In: CVPR (2007)

21. Rios-Cabrera, R., Tuytelaars, T.: Discriminatively trained templates for 3D object detection: A real time scalable approach. In: ICCV (2013)
22. Rosten, E., Porter, R., Drummond, T.: FASTER and better: A machine learning approach to corner detection. IEEE Trans. on PAMI 32 (2010)
23. Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A.: Real-time human pose recognition in parts from a single depth image. In: CVPR (2011)
24. Shotton, J., Glocker, B., Zach, C., Izadi, S., Criminisi, A., Fitzgibbon, A.: Scene coordinate regression forests for camera relocalization in rgb-d images. In: CVPR (2013)
25. Shotton, J., Girshick, R.B., Fitzgibbon, A.W., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., Blake, A.: Efficient human pose estimation from single depth images 35(12) (2013)
26. Steger, C.: Similarity measures for occlusion, clutter, and illu-mination invariant object recognition. In: DAGM-S (2001)
27. Sun, M., Bradski, G., Xu, B.-X., Savarese, S.: Depth-encoded hough voting for joint object detection and shape recovery. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part V. LNCS, vol. 6315, pp. 658–671. Springer, Heidelberg (2010)
28. Taylor, J., Shotton, J., Sharp, T., Fitzgibbon, A.: The Vitruvian Manifold: Inferring dense correspondences for one-shot human pose estimation. In: CVPR (2012)
29. Ferrari, V., Schmid, F.J., From, C.: images to shape models for object detection. In: IJCV (2009)
30. Winder, S., Hua, G., Brown, M.: Picking the best DAISY. In: CVPR (2009)