Latent-Class Hough Forests for 6 DoF Object Pose Estimation

Alykhan Tejani, Rigas Kouskouridas, Andreas Doumanoglou, Danhang Tang, and Tae-Kyun Kim, *Member, IEEE*

Abstract—In this paper we present *Latent-Class Hough Forests*, a method for object detection and 6 DoF pose estimation in heavily cluttered and occluded scenarios. We adapt a state of the art template matching feature into a scale-invariant patch descriptor and integrate it into a regression forest using a novel template-based split function. We train with positive samples only and we treat class distributions at the leaf nodes as latent variables. During testing we infer by iteratively updating these distributions, providing accurate estimation of background clutter and foreground occlusions and, thus, better detection rate. Furthermore, as a by-product, our *Latent-Class Hough Forests* can provide accurate occlusion aware segmentation masks, even in the multi-instance scenario. In addition to an existing public dataset, which contains only single-instance sequences with large amounts of clutter, we have collected two, more challenging, datasets for multiple-instance detection containing heavy 2D and 3D clutter as well as foreground occlusions. We provide extensive experiments on the various parameters of the framework such as patch size, number of trees and number of iterations to infer class distributions at test time. We also evaluate the *Latent-Class Hough Forests* on all datasets where we outperform state of the art methods.

Index Terms—3D Object detection, pose estimation, hough forests, one-class training, 6 DoF pose estimation

✦

1 INTRODUCTION

AMONG the most challenging tasks in computer vision is the one of estimating the 3D pose of an object due to its practical implication and its fundamental importance in applications like robotic manipulation [19] and tracking [7]. In order to efficiently fulfill the 3D object detection and pose estimation task, a computer vision method should tackle several cascading issues that hinder its effective application. Although recent emergence of consumer depth sensors provides additional cue in favour of textureless objects, background clutter, partial occlusions and large scale changes still put barriers to this problem. Template matching techniques [16] can tackle clutter and occlusion to some degree, but have inherent weakness due to their holistic nature. Point-to-Point approaches fail in cases of planar, self-similar or similar to background clutter objects [10] due to the fact that similar point features vote for different pose parameters. Moreover, these methods were only evaluated with the assumption of only one instance existing in the scene, while the case of multiple object instances, i.e., precision-recall, is left unexplored.

- A. Tejani can be contacted at E-mail: alykhan.tejani@gmail.com.
- R. Kouskouridas is with WIREWAX Ltd, London W1T 2RB, United Kingdom. E-mail: rkouskou@gmail.com.
- A. Doumanoglou and T.-K. Kim are with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, United Kingdom. E-mail: {a.doumanoglou12, tk.kim}@imperial.ac.uk.
- D. Tang is with perceptiveIO, Inc, San Francisco, CA 94103. E-mail: danhang.tang@gmail.com.

Another important perspect is that, prior arts in 3D object pose estimation [10], [16] utilize 3D mesh models to generate training samples. This implies that only positive samples are used and, thus, such techniques fall into the category of one-class learning. On the other hand, to explicitly tackle the aforementioned challenges, a more traditional way in 2D detection is to augment negative samples (clutter and occluder) during training. Due to the complexity of negative scenarios, this often results in huge amount of training samples, thus, increasing the computation burden of the system. And yet no such work can guarantee covering all cases. At present, there is a big disparity in the number of RGB-D versus RGB only datasets, adding a further challenge in mining for negative depth samples.

Highly motivated by these challenges, we present a novel method, called *Latent-Class Hough Forests*, for 3D object detection and pose estimation. Unlike traditional Hough Forest [12], which explicitly exploits class information, our method utilizes only the regression term during the training stage. Also differing from a regression forest, *Latent-Class Hough Forests* take into account class distributions at leaf nodes by considering them, during testing, as latent variables being updated iteratively, in order to provide more and more accurate voting results. As a byproduct, our system can also produce accurate occlusion-aware segmentation masks. Fig. 1 shows the effect of our inference algorithm. At the first iteration, the forest has no background information, thus all pixels are considered as foreground, which results in a noisy vote map. As the iterative process goes on, it is evident that background pixels are greatly suppressed and both the occlusion mask and vote map become more accurate and clean.

The paper in hand presents a novel method for object detection and 6 DoF pose estimation in real scenarios by
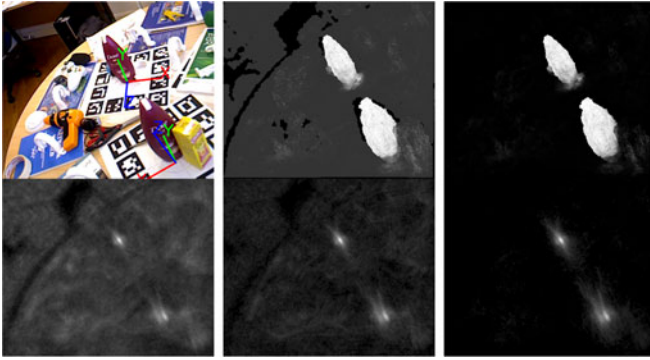
Fig. 1. An illustration of intermediate results of the iterative process. Column 1: Initial pose estimation and the corresponding Hough map. Columns 2-3: Foreground probability masks and the respective Hough maps after 5 and 10 iterations, respectively.

adopting a part-based strategy into the random forest framework. Our previous conference version [40] has been extended by a more detailed discussion of the key ingredients of the method and an extensive analysis of the parameters of the technique. Moreover, we propose an additional dataset inspired by industrial settings as well as reporting more experiments on three different datasets. Our main contributions can be summarized as follows:

- We propose *Latent-Class Hough Forests*, a novel patch-based approach to 3D object detection and pose estimation; It performs one-class learning at the training stage, and iteratively infers latent class distributions at test time.
- We adapt the state of the art 3D holistic template feature, LINEMOD [14], to be a scale invariant patch descriptor and integrate it into the random forest framework via a novel template-based splitting function.
- During the inference stage, we jointly estimate objects' 3D location and pose as well as a pixel-wise visibility map, which can be used as an occlusion aware figure-ground segmentation.
- We provide two new, more challenging public datasets for *multi-instance* 3D object detection and pose estimation, comprising *near and far range 2D and 3D clutter* as well as *foreground occlusions* in domestic and industrial scenarios. To the best of our knowledge, we are the first to provide a fully-annotated bin-picking dataset.

In the remainder of this paper we first discuss related work in Section 2 before introducing our method in Section 3. Following this, in Section 4, we provide a quantitative and qualitative analysis of our results as well as a comparison to current state of the art methods. Finally, in Section 5, we conclude with some final remarks and a discussion of future work.

2   RELATED WORK

Throughout the years several techniques for the detection and registration of objects in 3D environments have been proposed. According to the literature, three main categories can be distinguished: Template matching, learning-based methods and Point-to-Point techniques. The simplicity along with facile training sessions render template matching methods as one of the most widely used solutions for object detection tasks. From the very known baseline techniques of LINEMOD [14] and its extension [31], to the classic implementation of Distance Transform approaches [24], template matching methods have found application in contemporary vision tasks and robotics modules, respectively. On the other hand, learning-based approaches depend on laborious training sessions with numerous training samples with view to extract highly representative object models [8], [20], [25], [33]. Point-to-Point techniques build point pair features to construct object models based on point clouds. A representative method of this category is the one presented by Drost et al. [10]. In turn, simple pixel-based features have been also employed to tackle the object pose estimation problem. More recently, Brachmann et al. [4] introduced a new representation in form of a joint 3D object coordinate and class labelling (extended for tracking in [21]), which, however, suffers in cases of occlusions.

Moreover, in [23] a method for fine pose estimation by representing geometric and appearance information as a collection of 3D shared parts and objectness, has been presented. Song et al. [36] proposed a computationally expensive approach to the 6 DoF pose estimation problem that slides exemplar SVMs in the 3D space, while in [3] shape priors are learned by a soft labelling random forest for 3D object classification and pose estimation. Wu et al. [43] designed a model that learns the joint distribution of voxel data and category labels using a Convolutional Deep Belief Network, while the posterior distribution for classification is approximated by Gibbs sampling. From the relevant literature we could also identify the works of Aldoma et al. [1] and Buch et al. [6] that propose a final step of fine pose refinement for false positive/outlier removal. Last, Wohlhart et al. [42] showed how a Convolutional Neural Network can be trained to learn a 3D pose estimation-wise descriptor.

It is well understood that modeling objects as a collection of parts increases robustness to intra-class variation, pose change and even occlusion. The implicit shape model, introduced by Leibe et al. [22], learns, via unsupervised clustering, class-specific visual codebooks and spatial distributions for each entry. Codebook entries are then detected in the test image and used to cast probabilistic votes in the Hough space based on the learned spatial distributions. Gall and Lempitsky showed, with the class-specific Hough Forest [12], how part-based modeling can be effectively combined with generalized Hough voting for object detection under the random forest framework [5]. Tang et al. [37] combined Hough Forest and DOT [15] with a template matching split function, which, although highly efficient, requires extensive and diverse background images for training.

On the other hand, one-class training stands for the learning process that depends on training without negative samples. Introduced by Moya et al. [27], [28] and further developed by Tax [39] and Scholkopf [34], these approaches lay their foundations within the support vector framework and aim to derive an enclosing decision boundary of the training data as a whole from a few supporting samples.
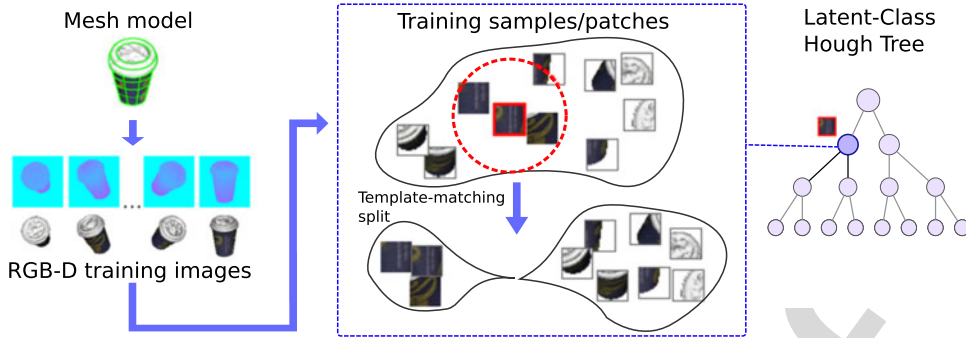
Fig. 2. During training, a random patch $\mathbb{T}$ is selected (red frame) as a template. The similarity between it and all other patches is measured and splitted based on a threshold τ (dotted circle). This process is repeated until the optimal $\mathbb{T}$ and τ are found and stored in the current node.

Other techniques such as the works of Bishop [2] and Para et al. [29] approached the problem in a probabilistic manner and tried to find the underlying density model for the training data.

Occlusion-handling is well-related to the object segmentation problem. In traditional 2D scenarios, existing methods can be categorised into tackling occlusion during training [26], [30] or the inference stage [17], [18]. Utilizing depth cues is relatively new and, only recently, Wang et al. [41] approached occlusion reasoning by explicitly learning clutter and oclusion scenarios during the training session. Our method falls into the second category where occlusion is tackled at test time.

3 PROPOSED METHOD

In the field of object detection and 3D pose estimation, LINEMOD [14], a binary RGB-D feature, has demonstrated both state of the art accuracy and efficiency. However, so far it has been combined with a holistic template matching scheme, which has inherent problem with occlusion due to the nature of holism. Moreover, as a near-neighbour search, this scheme slows down linearly as the number of templates grows. Moreover, the fact that LINEMOD is not scale invariant often leads to thousands of templates per object, in order to cover multiple scales and numerous viewpoints. To be more robust to occlusion and clutter, we start off by combining a state of the art 3D feature and a part-based detector. In this work, we choose the state of the art part-based detector Hough Forest [12]. However in our case, naively combining them does not work because: a) as a binary feature, LINEMOD only considers orientation information whilst discarding magnitude. This provides efficiency but degrades the accuracy in the case of a patch-based detector. b) No negative (clutter/occlusion) information is available during training, which means the *classification* term in Hough Forest cannot be adopted. c) Moreover, not knowing the boundary between positive and negative samples results in large amount of false positives during inference.

To address these issues, we propose *Latent-Class Hough Forests* to datamine useful clutter / occlusion information at test time and transfer the knowledge to the detector. In Section 3.1 we describe how to modify the similarity measurement of LINEMOD and integrate it into the split function of *Latent-Class Hough Forests*. Section 3.2 presents the inference process that jointly and iteratively updates the latent class distribution and voting results.

3.1 Learning

Latent-Class Hough Forests are an ensemble of randomized binary decision trees trained using the general random forest framework [5]. During training, each tree is built using a random subset of the complete training data. Each intermediate node in the tree is assigned a split function and threshold to optimize a measure of information gain; this test is then used to route incoming samples either left or right. This process is repeated until some stopping criteria is met, where a leaf node containing application-specific contextual information is formed. Each stage in this learning process is highly application dependent and we will discuss each in turn below. Fig. 2 illustrates the high level idea underlying our training module. A typical split function of a random forest can be formulated as below

$$h_i(x) = \begin{cases} 0, & S(x, \rho_i) \le \tau_i \\ 1, & S(x, \rho_i) > \tau_i \end{cases}, \qquad (1)$$

where ρ_i represents the learned parameter and τ_i is the threshold stored at node i. S is a test function that evaluates the input sample given ρ_i. The process of training a tree is to decide the optimal ρ_i and τ_i by measuring the information gain.

3.1.1 Training Data

In order to capture reasonable viewpoint coverage of the target object, we render synthetic RGB and depth images by placing a virtual camera at each vertex of a subdivided icosahedron of a fixed radius. A tree is trained from a set of patches sampled from the training images. We extract patches with size relative to the bounding box of the rendered object, while the template features are evenly spread across each patch; image gradient features are extracted only from the object contours while surface normals are taken from the body of the object. Moreover, the procedure of collection and representation of the template features is the same as the one described in [14].

3.1.2 Split Function

It has been shown in the 2D detection problem [37] that the original test function (so called two-pixel test) of Hough Forest does not work well with a binary feature. Thus, doing a naive holistic patch comparison, or the two-dimenson/two-pixel tests (as used in [11], [35], [38]) can lead to the test patches taking the incorrect route at split functions. To compensate the loss of magnitude
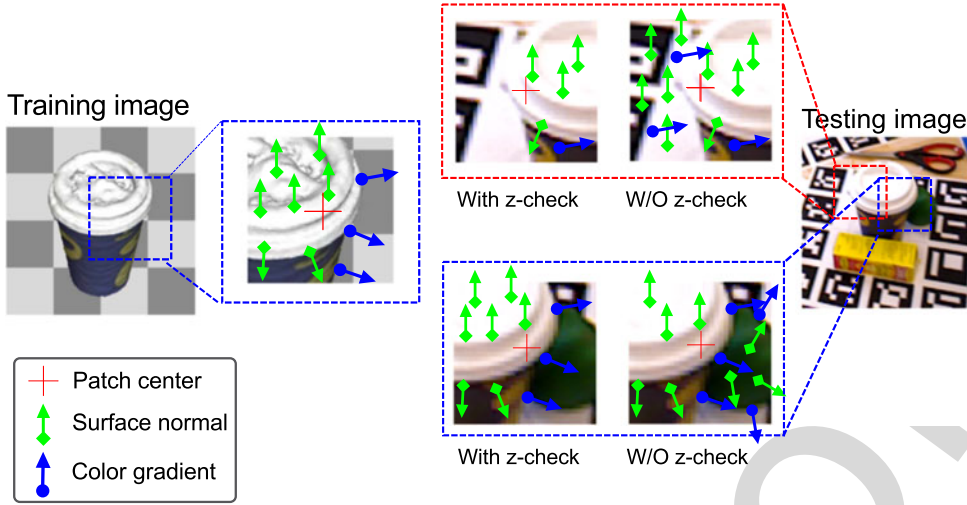
Fig. 3. The z-value check enhances the robustness against clutter and occluders. Blue patches indicate a true positive match and red ones a false positives one. Without the z-check, features of the planar background of the false patch could match the training patch and become a false positive.

information whilst keeping high efficiency levels, one possible way is to utilize the orientation information in a more effective way. To this end, a non-linear template matching test function is adopted as described below

$$S(\mathcal{X}, \rho) = S(\mathcal{X}, \mathbb{T}) = \sum_{r \in \mathcal{P}} g(\mathbf{ori}(\mathcal{X}, r), \mathbf{ori}(\mathcal{O}, r))$$
$$= \sum_{r \in \mathcal{P}} \left(\max f_m\big(\mathcal{X}(r), \mathcal{O}(r)\big) \right), \quad (2)$$

where $\mathcal{X}$ is a testing patch, $\mathbb{T} = (\mathcal{O}, \mathcal{P})$ is the template of a patch $\mathcal{O}$ with a list $\mathcal{P}$ of features. f_m is the dot product between the gradient orientation at location r of the RGB-D patch $\mathcal{X}$ and $\mathcal{O}$, respectively. The similarity measure f_m is similar to [14], while here we show how it can be adapted to work over patches. Combining Eqs. (1) and (2) gives us a non-linear split function, which performs much better than axis-aligned and linear split functions, yet, has the complexity similar to an axis-aligned one, since it involves only bit-wise operations that can be SSE-accelerated.

The test function of Eq. (2) performs well within the object boundary, but poorly around its edges. Mainly because during inference, clutter and occluder around the object make the similarity measurement between patches to fail. See Fig. 3 for an illustration of this issue. To tackle this, we modify the test function by incorporating an efficient z-value check

$$\begin{cases} S(\mathcal{X}, \mathbb{T}) & = \sum_{r \in \mathcal{P}} f(\mathcal{X}, \mathcal{O}, c, r) g(\mathbf{ori}(\mathcal{X}, r), \mathbf{ori}(\mathcal{O}, r)), \\ f(\mathcal{X}, \mathcal{O}, c, r) & = \delta(|(D(\mathcal{X}, c) - D(\mathcal{X}, r)) - (D(\mathcal{O}, c) - D(\mathcal{O}, r))| < \tau), \end{cases}$$
$$(3)$$

where, for a patch centered at position c, $D(a, b)$ retreives z-value from position b of patch a and f is an indicator function that checks if the depth difference between two patches is larger than a threshold τ.

The original LINEMOD is not scale invariant, which results in a large amount of templates that needed to be sampled from the scale space. However, inspired by [35], we achieve scale invariance by normalizing feature offsets by its depth value

$$\begin{cases} S(\mathcal{X}, \mathbb{T}) & = \sum_{r \in \mathcal{P}} f(\mathcal{X}, \mathcal{O}, c, r) g(\mathbf{ori}(\mathcal{X}, \frac{r}{D(\mathcal{X}, c)}), \mathbf{ori}(\mathcal{O}, \frac{r}{D(\mathcal{O}, c)})), \\ f(\mathcal{X}, \mathcal{O}, c, r) & = \delta\big(|\big(D(\mathcal{X}, c) - D\big(\mathcal{X}, \frac{r}{D(\mathcal{X}, c)}\big)\big) - (D(\mathcal{O}, c) \\ & \quad - D(\mathcal{O}, \frac{r}{D(\mathcal{O}, c)}))| < \tau). \end{cases}$$
$$(4)$$

During training, at each split node, we randomly choose a patch as a template $\mathbb{T}$ and measure its similarity with all other patches. Those patches with similarity larger than a threshold τ will go to one child node, whilst the rest will go to the other. This split is measured with respect to the information gain and repeated multiple times until an optimal one is found.

3.1.3 Constructing Leaf Nodes

The training data is recursively split by this process until the tree has reached a maximum depth or the number of samples arriving at a node fall below a threshold. When either one of these criteria is met a leaf node is formed from the patches reaching it. As far information gain is concerned, since no negative information is available during training, we cannot use a classification term but only the regression one. However, differing from a regression forest, *Latent-Class Hough Forests* still store the class distribution in leaf nodes. Following the approach of Girshick et al. [13] we only store the indexes of the training patches that reached at each leaf node and the modes of the distribution, which we efficiently calculate via the MeanShift algorithm. We create a class distribution at each leaf, however, as no background information reaches the leaves during training this distribution is initialized to $p_{fg}^l = 1$ and $p_{bg}^l = 0$ for the foreground and background probabilities, respectively.

3.2 Inference

After learning, we have a *Latent-Class Hough Forest* trained with positive samples only. At inference stage, we propose an iterative algorithm to datamine negative information.

3.2.1 Hough Voting

Let $E(\theta)$ represent the probability of the random event that the target object exists in the scene under the 6D pose
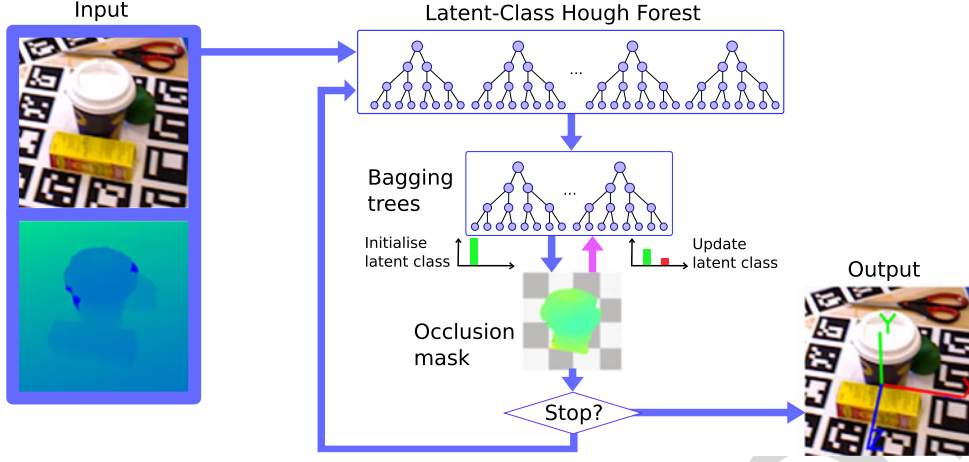
Fig. 4. Inference process: Input LINEMOD patches are extracted from RGB-D images. For each iteration, a subset of trees are drawn from the forest with bagging.

$\theta = (\theta_x, \theta_y, \theta_z, \theta_{roll}, \theta_{pitch}, \theta_{yaw})$. The goal of the voting process is to aggregate the conditional probabilities $p(E(\theta)|\mathcal{P})$ for each patch $\mathcal{P}$, given that each vote originates from a foreground patch ($p_{fg} = 1$). For a patch P reaching leaf node l of tree T, the conditional probability is formalized as follows:

$$p(E(\theta)|\mathcal{P}; \mathcal{T}) = p\left(E(\theta), p_{fg}^l = 1|\mathcal{P}\right)$$
$$= p\left(E(\theta)|p_{fg}^l = 1, \mathcal{P}\right) \cdot p\left(p_{fg}^l = 1|\mathcal{P}\right), \tag{5}$$

where p_{fg}^l is the foreground probability at the leaf node, l. Finally, for a forest, $\mathcal{F}$, we simply average the probabilities over all trees

$$p(E(\theta)|\mathcal{P}; \mathcal{F}) = \frac{1}{|\mathcal{F}|} \sum_t^{|\mathcal{F}|} p(E(\theta)|\mathcal{P}; \mathcal{T}_t). \tag{6}$$

The first factor, $p\left(E(\theta)|p_{fg} = 1, \mathcal{P}\right)$, can be estimated by passing each patch down the forest and accumulating the votes stored at the leaf, in which votes from multiple trees can be combined in an additive manner, this gives us the same probabilities as in Eq. (6) up to a constant factor. The estimation is then deferred to the ability of locating local maxima in this aggregated space. Traditionally there are two different methods of locating targets in vote space. One is to aggregate all votes in the same position of vote space, and return the peaks with non-maximum suppression [12]. The other is to treat each vote as a data point, and then use MeanShift to locate the mode. In this case an assumption of only one instance in the scene is made to avoid local minimum [11]. The former is efficient for locating but less accurate than the latter, especially when the votes are sparsely distributed.

To accommodate both efficiency and accuracy, we propose a three-stage localization technique in which we first aggregate all votes in 2D image space, use this as a score function and locate the top N vote peaks as valid hypotheses; and then use them as initialization for MeanShift to locate modes in 3D translation, $(\theta_x, \theta_y, \theta_z)$; finally find the mode in rotation, $(\theta_{roll}, \theta_{pitch}, \theta_{yaw})$ given translation.

3.2.2 Update Class Distribution

As mentioned in Section 3.1, class distributions are initialized as all foreground. For each pixel $\mathbf{x} = (\mathcal{P}, l)$, located at the center position l of patch $\mathcal{P}$, the initial probability of being foreground is $p_{fg}^{\mathbf{x}} = 1$. Inspired by [22], for each valid hypothesis θ, we backproject to obtain a consensus patch set, i.e., peak of the probability distribution in the Hough space where multiple patches cast their vote, is considered as valid only in cases where its score is above a threshold. This threshold value is different for each object while our three stage localization technique includes also thresholds per stage. All the thresholds are object-specific and are estimated via trial and error. All consensus voting patches are considered to be foreground and the rest background

$$p_{fg}^{\mathbf{x}} = \delta\left(\left(\sum_\theta \left(p_{fg}^{\mathbf{x}}|\theta\right)p(\theta)\right) > 0\right), \tag{7}$$

in which $p(\theta)$ indicates whether this hypothesis is valid or not and $(p_{fg}^{\mathbf{x}}|\theta)$ suggests whether $\mathbf{x}$ has voted for θ. As long as $\mathbf{x}$ has voted for one valid hypothesis, it is, then, considered as a foreground pixel ($\delta(\cdot) = 1$). In other words, after a valid hypothesis is found via patch voting, we move down to pixel level, where patches that voted for this hypothesis are back-projected to the scene. Since we have an estimation of the object's center in the scene and we know also its diameter, pixels (of the voting patches) that are not spatial consistent (distance to the object center larger than the diameter) are considered to belong to the background and the rest to the foreground. With this we can update the latent class distribution of each leaf node l by

$$p_{fg}^{\mathbf{l}} = \frac{\sum_{\mathbf{x}} p_{fg}^{\mathbf{x}}}{|\{\mathbf{x}|\mathbf{x} \in l\}|}, \tag{8}$$

which can be translated to the calculation of the normalized portion of foreground patches ending at leaf node l.

3.2.3 Iterative Process

The estimation obtained with previous step is inaccurate and uncertain. However, with an iterative process, we update the probability distribution $p_{fg}^{\mathbf{l}}$ for every iteration. To avoid error accumulation of each round, we propose to draw a subset of trees with bagging, such that not all the trees are updated by previous iteration. The whole process is illustrated in Fig. 4 and described in Algorithm 1.

3.2.4   Final Segmentation

With the estimated hypotheses $\{\theta\}$ and a foreground probability mask $\mathcal{Z}$ generated with $p_{fg}^{\mathbf{x}}$, we can easily obtain a final occlusion-aware segmentation by

$$\mathcal{M} = \mathbb{B}(\theta) \cap \mathcal{Z}, \qquad (9)$$

where $\mathbb{B}$ is a bounding box centered at θ. This is helpful for further refinement step such as ICP, which does not work well in case of occlusions.

Algorithm 1. Inference Process

Require: An input image $\mathcal{I}$; A Hough Forest $\mathcal{F}$

1: **repeat**
2:     Draw a subset of trees $\mathcal{F}^*$ with bagging
3:     Randomly sample a set of patches P from I
4:     Propagate P down $\mathcal{F}^*$ and vote for θ (Eq. (6))
5:     Backproject to obtain a foreground mask (Eq. (7))
6:     Partition P into positive and negative subsets by
     foreground mask

$$P^{fg} = \{\mathbf{x} | \mathbf{x} \in Z\}$$

$$P^{bg} = P \backslash P^{fg}$$

7:     Update the probabilities at leaf nodes (Eq. (8)) given P^{fg}
     and P^{bg}.
8: **until** Maximum iteration

---

4   EXPERIMENTS

We perform experiments on three 3D pose estimation datasets. The first is the publicly available dataset of Hinterstoisser et al. [16], which contains 13 distinct objects each associated with an individual test sequence comprising of over 1,100 images with close and far range 2D and 3D clutter. Each test image is annotated with ground truth position and 3D pose. We also introduce our two new datasets,[1] called *Domestic Environments Dataset* and *Bin-picking Dataset* to further evaluate the efficiency of our method in real scenarios. In all tests we use the metric defined in [16] to determine if an estimation is correct. More formally, for a 3D model $\mathcal{M}$, with ground truth rotation $\boldsymbol{R} = (\theta_{pitch}, \theta_{yaw}, \theta_{roll})$ and translation $\boldsymbol{T} = (\theta_x, \theta_y, \theta_z)$, given an estimated rotation, $\hat{\boldsymbol{R}} = (\hat{\theta}_{pitch}, \hat{\theta}_{yaw}, \hat{\theta}_{roll})$ and translation, $\hat{\boldsymbol{T}} = (\hat{\theta}_x, \hat{\theta}_y, \hat{\theta}_z)$, the matching score is defined as

$$m = \underset{\mathbf{x} \in \mathcal{M}}{avg} ||(\boldsymbol{R}\mathbf{x} + \boldsymbol{T}) - (\hat{\boldsymbol{R}}\mathbf{x} + \hat{\boldsymbol{T}})||, \qquad (10)$$

for non-symmetric objects and

$$m = \underset{\mathbf{x_1} \in \mathcal{M}}{avg} \underset{\mathbf{x_2} \in \mathcal{M}}{\min} ||(\boldsymbol{R}\mathbf{x_1} + \boldsymbol{T}) - (\hat{\boldsymbol{R}}\mathbf{x_2} + \hat{\boldsymbol{T}})||, \qquad (11)$$

for symmetric objects. An estimation is deemed correct if $m \leq k_m * d$, where k_m is a chosen coefficient and d is the diameter of $\mathcal{M}$. We would like to note that the symmetric version (Eq. (11)) of the matching score assumes complete symmetry around an axis, like bowls. However, other objects with, for example, square shape have an advantage if assumed symmetric.

Unlike [16], in which only the top N detections from each image are selected, we also compute precision-recall curves and present the F1-Score which is the harmonic mean of precision and recall. We argue that this is a more accurate form of comparison, as directly comparing detections is inaccurate as some images may be more challenging than others and the number of target objects may be unknown (as is the case in our new datasets).

In the proposed method a forest is currently trained for one object while there are several works (e.g., [4], [9]) that utilize a single forest for multiple objects. Our method could be extended to multiple objects per forest by adding three different objective functions that would be used for measuring information gain. During training, similar to classic Hough Forests [12], we could randomly select to perform either entropy minimization of the class distribution or entropy minimization of the regression parameters. Practically this means that we could perform multi-class classification and 6D object pose estimation, addressing, thus, one shortcoming of our method that assumes at least one object instance to always be present in the scene.

In Section 4.1 we present extensive experiments on various parameters of the framework such as patch size, number of trees and number of iterations to infer class distributions at test time. In Section 4.2.1 we perform self comparison tests highlighting the benefits of adding scale-invariance to the template similarity measure (Eq. (3)) and using co-training to update the latent class distributions (Algorithm 1). Following this, in Section 4.2.2 we compare the performance of our method on the famous dataset of Hinterstoisser et al. [16] against the state of the art works of LINEMOD [14], Drost et al. [10], Rios Cabrera et al. [31] and Brachmann et al. [4]. Moreover, in Sections 4.2.3 and 4.2.4 we present our new datasets and additional comparative analysis.

4.1   Framework Parameters

Parameter optimization was performed on a validation dataset that was created by randomly selecting a subset of our own datasets. For each object class, we train a *Latent-Class Hough Forest* with varying number of trees and patch sizes. Moreover, during inference and for the co-training stage we experiment with different number of iterations, while the number of hypotheses to be backprojected per iteration is set $N = 10$. We choose 10 as it is greater than the number of instances present in all datasets, however this number is not fixed and can be adapted based on the application. Furthermore, in all experiments for parameter optimization the coefficient k_m is set to the value of 0.15, the results with this coefficient are also found to be visually correct.

4.1.1   Patch Size and Number of Trees

Selecting the size of the patch is of paramount importance since large patches tend to match the disadvantages of a holistic template (i.e., sensitive clutter, occlusions etc.) while small ones are prone to noise. The size of the patch depends on the object's identity and it is relative to its size. We extract the bounding box of the target object by rendering its 3D mesh model and modify the patch size to be proportional to the size of the bounding box. Experimenting

---

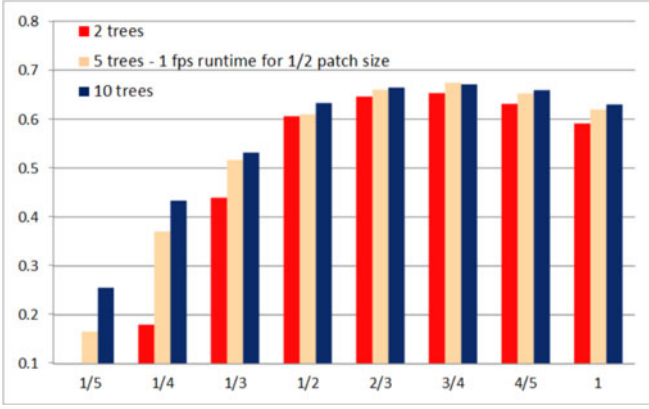1. http://www.iis.ee.ic.ac.uk/rkouskou/research/LCHF.html and http://www.iis.ee.ic.ac.uk/rkouskou/research/6D_NBV.html (scenario 2)

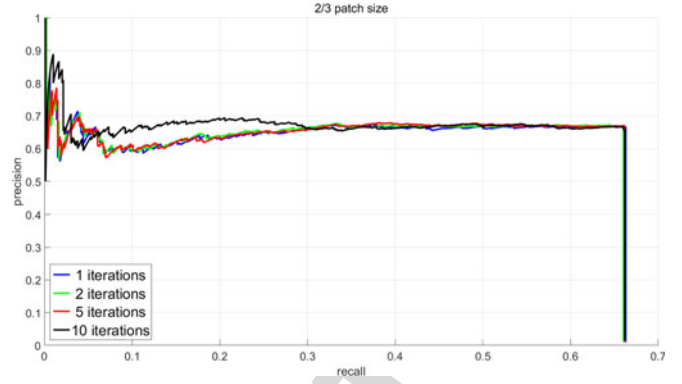Fig. 5. F1-Scores for the eight different patch sizes and the three different forests in our validation dataset.

Fig. 7. The impact of the number of iterations on the performance of our system for our validation dataset on a fixed patch size of 2/3 and a forest with 10 trees.

518 with eight different patch sizes, as shown in Fig. 5, revealed
519 that, increasing the size of the patch has a direct effect on
520 the F1-Score, while saturation occurs for patches with size
521 equal to 2/3 of the bounding box and higher. Patches
522 resembling holistic ones (i.e., 4/5 and 1) are proven to be
523 very prone to occlusions.

524 We have additionally experimented with the number of
525 the trees of our *Latent-Class Hough Forests*. It is apparent
526 that, using patches with relatively large size and forests
527 with more than five trees, puts additional computational
528 burden to our system. For instance, we can achieve real-
529 time execution (1 fps) if we use 1/2 as patch size, two trees
530 and zero iterations in the co-training stage. However, select-
531 ing larger patches and forests results in a respective increase
532 of the execution time.

533 Regarding the balance of the trees, the learned thresholds τ
534 span through the whole range of the values of the split func-
535 tion in a certain node. Usually, the nodes at the top of the tree
536 tend to be more balanced, since there is a variety of different
537 patches, whereas nodes at lower depths are less balanced.
538 However, as the tree grows, each node contains less patches
539 and balancing depends on how well the available patches
540 can be split in two sets, according to our objective function. In
541 fact, producing balanced nodes is not always the best way to
542 split the samples, since it depends on the feature space the
543 patches lay on. The forest optimizes splitting according to the
544 semantic appearance of the patches, and balanced splitting
545 does not always satisfy this objective. However, we have

noticed that for objects Coffee Cup, Juice Carton and Joystick 546
the percentage of training patches that move to the child with 547
the lowest number of samples is around 43 percent, implying 548
well balanced trees, while for the rest of the objects this 549
percentage is around 35 percent. Another way to measure the 550
balance of the trees was proposed in [32] where instead of 551
taking one patch, one could also take two or more patches 552
and assign the patch to closest one. 553

4.1.2 Bagging and Number of Iterations 554

During inference we randomly draw a subset of trees with 555
bagging and iteratively update class distributions in order 556
to datamine the negative information. The impact of bag- 557
ging is shown in Fig. 6. Given four different sizes of forests 558
(i.e., 10, 9, 8 and 5 trees) we randomly select a subset of trees 559
to obtain an initial object hypotheses set that is then used to 560
construct a consensus pixel test. To investigate the effect of 561
the number of selected trees, we compared the performance 562
levels of our method when drawing a) 1 to 9 trees from a 10 563
tree forest, b) 1 to 8 trees from a nine tree forest, c) 1 to 7 564
trees from an eight tree forest and d) 1 to 4 from a five tree 565
forest. According to the conducted experiments (see Fig. 6), 566
our method preforms better in cases we split the forest in 567
the middle, thus, creating two classifiers of equal size. 568

The quality of the resulted segmentation mask depends 569
on the number of iterations of the co-training process. 570
Since our method makes no use of negative data during 571
training, one straightforward question is whether training 572
with both positive and negative patches would reduce the 573
number of iterations required for extracting the segmenta- 574
tion mask. Towards this end, we used the background 575
dataset of [4] that contains RGB-D images of different clut- 576
tered office backgrounds. Patches were sampled from the 577
negative data, pushed through the trees and the classes 578
probability distributions were updated. As expected the 579
overall accuracy was not improved while a slightly faster 580
convergence was noticed only in very few cases. After the 581
first iteration, the extracted consensus pixel set contains 582
both positive and negative data that come directly from 583
the observable unlabelled data in the scene. We believe 584
that this is the reason why training with negative samples 585
offers slightly faster convergence only in very few cases. 586
Fig. 7 illustrates the effect of the iterative part on the per- 587
formance of our method on our validation dataset. As 588
expected, the iterative method affects only the precision of 589
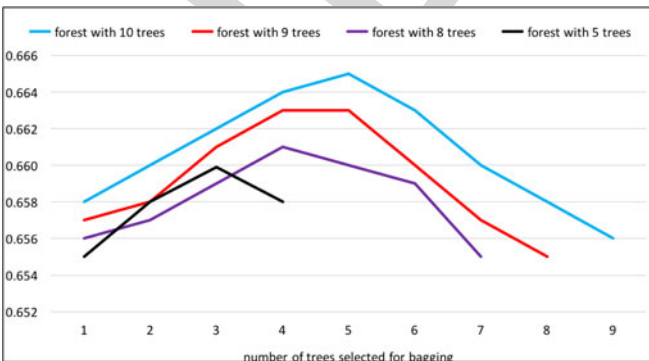

Fig. 6. The impact of bagging in our iterative process. Accuracy (F1-Scores) versus number of trees selected for bagging under four different forest sizes.
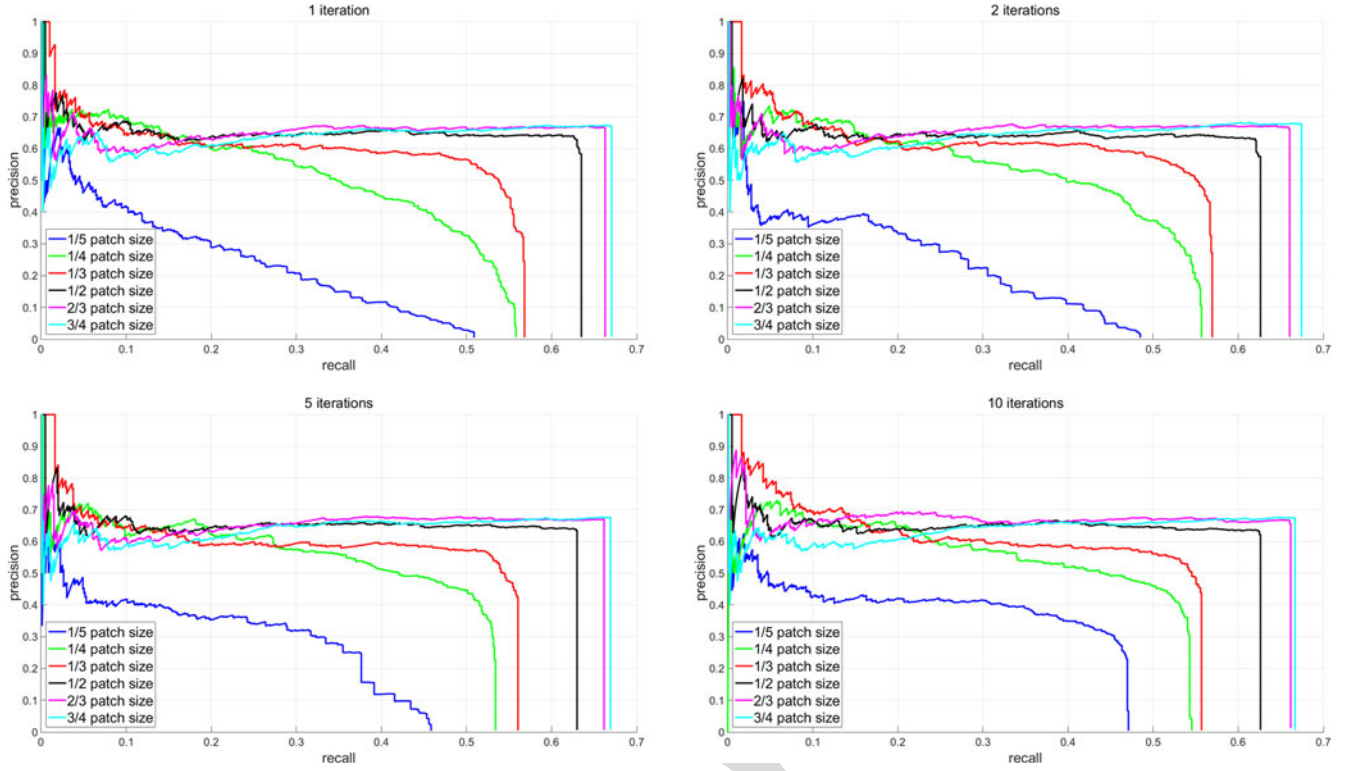
Fig. 8. Precision-Recall curves for different number of iterations and patch sizes.

TABLE 1
F1-Scores for LINEMOD [14], the Method of Drost et al. [10] and Our Approach for Each Object Class for the Dataset
of Hinterstoisser et al. [16] and Our *Domestic Environment Dataset*

Approach	LINEMOD [14]	Drost et al. [10]	SI LINEMOD	Without Iterations	With Iterations
Dataset of Hinterstoisser et al. [16]					
Sequence (# images)			**F1-Score**		
Ape(1,235)	0.533	0.628	0.631	0.799	**0.855**
Bench Vise (1,214)	0.846	0.237	0.869	0.941	**0.961**
Driller (1,187)	0.691	0.597	0.744	0.899	**0.905**
Cam (1,200)	0.640	0.513	0.711	0.636	**0.718**
Can (1,195)	0.512	0.510	0.550	0.708	**0.709**
Iron (1,151)	0.683	0.405	0.749	0.705	**0.735**
Lamp (1,226)	0.675	0.776	0.790	0.911	**0.921**
Phone (1,224)	0.563	0.471	0.655	0.660	**0.728**
Cat (1,178)	0.656	0.566	0.773	0.884	**0.888**
Hole Punch (1,236)	0.516	0.500	0.601	0.819	**0.875**
Duck (1,253)	0.580	0.313	0.659	0.888	**0.907**
Box (1,252)	0.860	0.826	**0.933**	0.736	0.740
Glue (1,219)	0.438	0.382	0.462	0.643	**0.678**
Average	0.630	0.517	0.702	0.788	**0.817**
Domestic Environment Dataset					
Sequence (# images)			**F1-Score**		
Coffee Cup (708)	0.879	0.902	0.897	0.891	**0.921**
Shampoo (1,058)	0.675	0.701	0.699	0.792	**0.836**
Joystick (1,032)	0.479	0.313	0.518	0.549	**0.572**
Camera (708)	0.526	0.503	**0.581**	0.394	0.431
Juice Carton (859)	0.561	0.684	0.559	0.883	**0.916**
Milk (860)	0.246	0.322	0.311	0.397	**0.439**
Average	0.561	0.571	0.594	0.651	**0.686**

the system and not the recall, since it does not serve as a false positive rejection framework. Higher number of iterations offer slightly better accuracy with the cost, however, of additional execution time.

In turn, Fig. 8 illustrates the conditional effect of different patch sizes for different number of iterations during inference. At this point we would like to note that, as seen in Tables 1 and 2, our system is capable of producing accurate

TABLE 2
Matching Score and Speed on the Dataset of Hinterstoisser et al. [16] for LINEMOD [14], the Methods of Drost et al. [10],
Rios Cabrera et al. [31], Brachman et al. [4] and Our Approach

Approach		LINEMOD [14]	Drost et al. [10]	Rios Cabrera et al. [31]	Brachmann et al. [4]	Our Approach
Sequence (# images)	Metric			Matching Score / Speed		
Ape(1,235)	10	95.8% / 127 ms	86.5% / 22.7 s	95.0% / **55.8 ms**	**95.8%** / 122 ms	95.7% / 1.82 s
Bench Vise (1,214)	10	98.7% / 115 ms	70.7% / 2.94 s	98.9% / **53.3 ms**	**100%** / 178 ms	99.7% / 2.11 s
Driller (1,187)	10	93.6% / 121 ms	87.3% / 2.65 s	94.3% / **54.6 ms**	**99.5%** / 173 ms	99.2% / 1.91 s
Cam (1,200)	10	97.5% / 148 ms	78.6% / 2.81 s	98.2% / **58.4 ms**	99.6% / 147 ms	**99.6%** / 1.98 s
Can (1,195)	10	95.4% / 122 ms	80.2% / 1.60 s	**96.3%** / **55.3 ms**	95.9% / 165 ms	96.1% / 2.08 s
Iron (1,151)	10	97.5% / 116 ms	84.9% / 3.18 s	98.4% / **54.3 ms**	97.6% / 166 ms	**98.5%** / 1.97 s
Lamp (1,226)	10	97.7% / 125 ms	93.9% / 2.29 s	97.9% / **54.8 ms**	**99.8%** / 188 ms	99.6% / 2.01 s
Phone (1,224)	10	93.3% / 157 ms	80.7% / 4.70 s	95.3% / **58.4 ms**	**97.6%** / 161 ms	96.7% / 2.05 s
Cat (1,178)	10	99.3% / 111 ms	85.4% / 7.52 s	99.1% / **53.5 ms**	**100%** / 132 ms	99.8% / 1.97 s
Hole Punch (1,236)	10	95.9% / 110 ms	77.4% / 8.30 s	97.5% / **54.2 ms**	**99.4%** / 204 ms	99.5% / 1.92 s
Duck (1,253)	10	95.9% / 104 ms	40.0% / 6.97 s	94.2% / **53.6 ms**	95.9% / 132 ms	**96.1%** / 1.74 s
Box (1,252)	11	99.8% / 101 ms	97.0% / 2.94 s	99.8% / **56.0 ms**	98.0% / 152 ms	98.1% / 2.10 s
Glue (1,219)	11	91.8% / 135 ms	57.2% / 4.03 s	96.3% / **58.5 ms**	**98.9%** / 142 ms	98.2% / 1.83 s
Average		96.3% / 122 ms	78.4% / 5.58 s	97.1% / **55.4 ms**	**98.3%** / 158 ms[1]	98.2% / 1.96 s

In the second column we report which matching score (Eqs. (10) or (11)) was used.
[1]*The method of Brachman et al. [4] makes use of GPU during testing. The reported execution times were obtained using our GPU TitanX.*

DoF pose estimations even without the iterative part. However, depending on the application domain we might switch to different parameters to meet the respective demands. For example, for industrial object manipulation, real-time execution usually constitutes a prerequisite, while robots operating in domestic environments, emphasize more on the accuracy levels. Towards this end, for the *Domestic Environments Dataset* we set the patch size of 2/3, use a forest of 10 trees and set the number of iterations to 10, while for our *Bin-picking Dataset*, we investigate both the aforementioned parameters and the effect of smaller patches (1/2) for forests with only two trees without any iteration at all.

Our approach is not guaranteed to converge and it is possible to produce a solution where all the foreground probabilities converge to zero. However, we have not noticed such a case throughout the exhaustive experimental analysis. The extracted consensus pixel set stems from patches that voted for a hypothesis with a score larger than a threshold, which is different for each object and estimated via trial and error. However, false positives play an important role in our method and can generate false segmentation masks, as shown in the last row of Fig. 10.

4.2 Comparative Study

In the following section we present a comparative analysis of the performance of our method against other state of the art techniques. We perform a self comparisons to investigate the role of our iterative method, while we compare against several recently published methods in three different datasets.

4.2.1 Self Comparisons

We perform two self comparisons on the dataset of Hinterstoisser et al. [16] and our *Domestic Environment Dataset*. First we compare the results of our method with and without the iterative process. As can be seen in Table 1 for the dataset of Hinterstoisser et al. [16], our approach with the iterative process improves the F1-Score by 2.8 percent

on average and up to 6.4 percent on some objects. The biggest gains are seen in objects that have large amounts of indistinct regions (image patches) for which background clutter can easily be confused. For example, the biggest improvements are seen in the Glue, Holepuncher and Phone objects that contain large planar regions. Furthermore, in Table 1 we compare the results of using holistic LINEMOD templates, to scale-invariant (SI) LINEMOD templates. As the scale-invariant version is trained using only one scale, the performance is increased six-fold (623 templates opposed to 3,738). Furthermore, the performance is also increased by 6.9 percent on average, this is due to the fact that templates are able to match at scales not seen in the template learning stage of the original LINEMOD. The impact of our iterative method is more visible in testing scenes that contain small objects with significant amount of foreground occlusions, e.g., Camera object in *Domestic Environment Dataset* or Coffee Cup object in *Bin-Picking Dataset*.

Additionally, we investigate the impact of different modalities on the performance of our system. We used a much smaller validation subset comprised of images of two of the objects contained in both our *Domestic Environment Dataset* and *Bin-Picking Dataset*. Table 3 summarizes the calculated F1-Scores, while one interesting finding is that geometrical cues seem to be more important from appearance ones when trying to detect the object Coffee Cup. In turn, although the detection levels are relatively low, the Camera object favors the usage of both RGB and D modalities.

TABLE 3
Impact of Different Modalities on the Performance of Our
Method Based on a Smaller Validation Subset

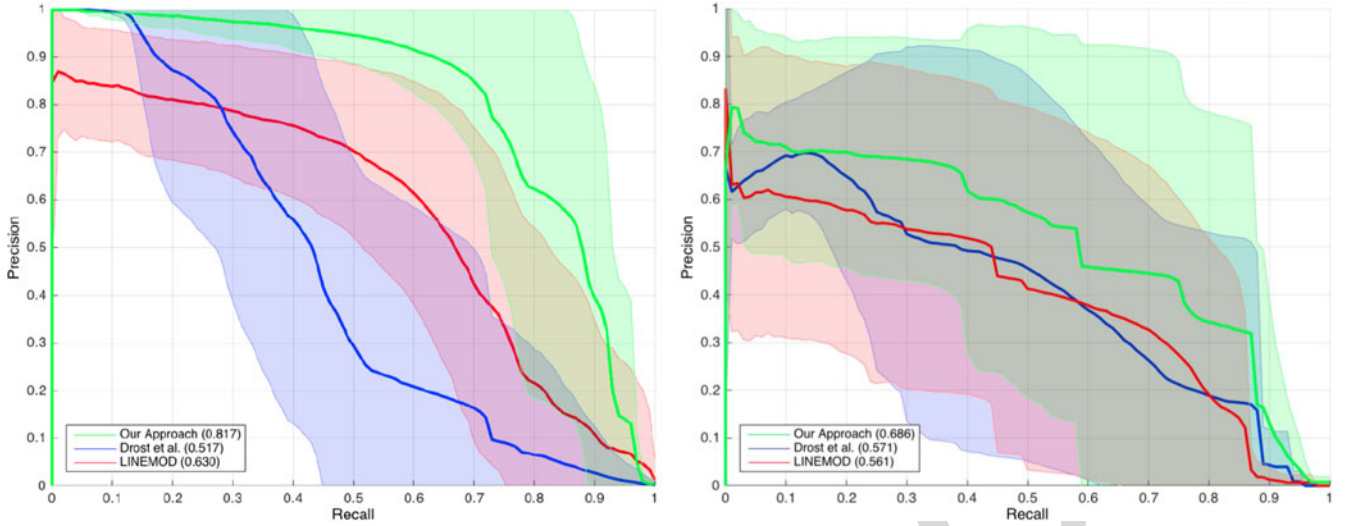| Object	RGB	Depth	RGB-D
Coffee Cup	0.748	**0.838**	0.828
Camera	0.309	0.324	**0.369**

Fig. 9. Average precision-recall curve over all objects in the dataset of LINEMOD [16] (left) and our *Domestic Environment Dataset* (right). The shaded region represents one standard deviation above and below the precision value at a given recall value.

4.2.2   1st Dataset: LINEMOD

The performance of our method was evaluated also on the famous dataset of Hinterstoisser et al. [16] against the state of the art works of LINEMOD [14], Drost et al. [10], Rios Cabrera et al. [31] and Brachmann et al. [4]. This dataset contains only one object instance per testing image in a cluttered background, however, without occlusions. Table 2 summarizes the results of the conducted comparison with respect to the performance levels and the execution times. In the second column of Table 2 we report which matching score (Eqs. (10) or (11)) was used per object, while for our method we used the same k_m threshold values with the ones presented in [16]. The performance of our method is almost identical to the one of Brachmann et al. [4] on a dataset that could be apprehended as obsolete, since the challenges it offers, i.e., background clutter with no foreground occlusions, are experimentally proven to be easily addressed. As far as execution times are concerned, the scalable method of Rios Cabrera et al. [31] provided the lowest runtimes followed by LINEMOD [14], the method of Brachmann et al. [4] and our proposed framework, respectively. Compared to the aforementioned works, our technique provides higher performance levels with the additional computational cost. We would like to note that since the evaluation metric for symmetric objects is not very representative, one could transform every hypothesis using the rotation matrices R_x that convert the object to its symmetric shapes and evaluate the detection performance using the non-symmetric measure of Eq. (10).

Implementation Details: We have implemented our own version of LINEMOD that produced the same exactly results presented in [16]. For the method of Drost et al. [10], we use a binary version kindly provided by the author and set the parameters as described in [10]. Source code of the method of Brachmann et al. [4] is publicly available. As far as the method of Rios Cabrera et al. [31] is concerned, we show in Table 2 the respective performance levels as they were presented in the original paper. Regarding runtimes, reported numbers are taken from [31].

4.2.3   2nd Dataset: Domestic Environments

Our *Domestic Environment Dataset* consists of 6 objects placed on top of a table simulating, thus, a kitchen table in a house. We provide a dense 3D reconstruction of each object obtained via a commercially available 3D scanning tool [35]. Additionally, we provide for each object, similarly to [16], an individual testing sequence containing over 700 images annotated with ground truth position and 3D pose. Testing sequences were obtained by a freely moving handheld RGB-D camera and ground truth was calculated using marker boards and verified manually. Testing images were sampled to produce sequences uniformly distributed in the pose space by $[0° - 360°]$, $[-80° - 80°]$ and $[-70° - 70°]$ in the yaw, roll and pitch angles, respectively. Unlike the dataset of [16], our testing sequences contain *multiple object instances* and *foreground occlusions* along with near and far range 2D and 3D clutter, making it more challenging for the task of 3D object detection and pose estimation. Some example frames from this dataset can be seen in Fig 10.

In Fig. 9 we show the average precision-recall curves across all objects and in Table 1 we show the F1-Score per object for each dataset. All methods provided evidence of worse performance levels on the new dataset, which is to be suspected due to the introduction of occlusions as well as multiple object instances. It is evident that we outperform both state of the

TABLE 4
Comparison of Two Versions of Our *Latent-Class Hough Forests* and the Method of Brachmann et al. [4]

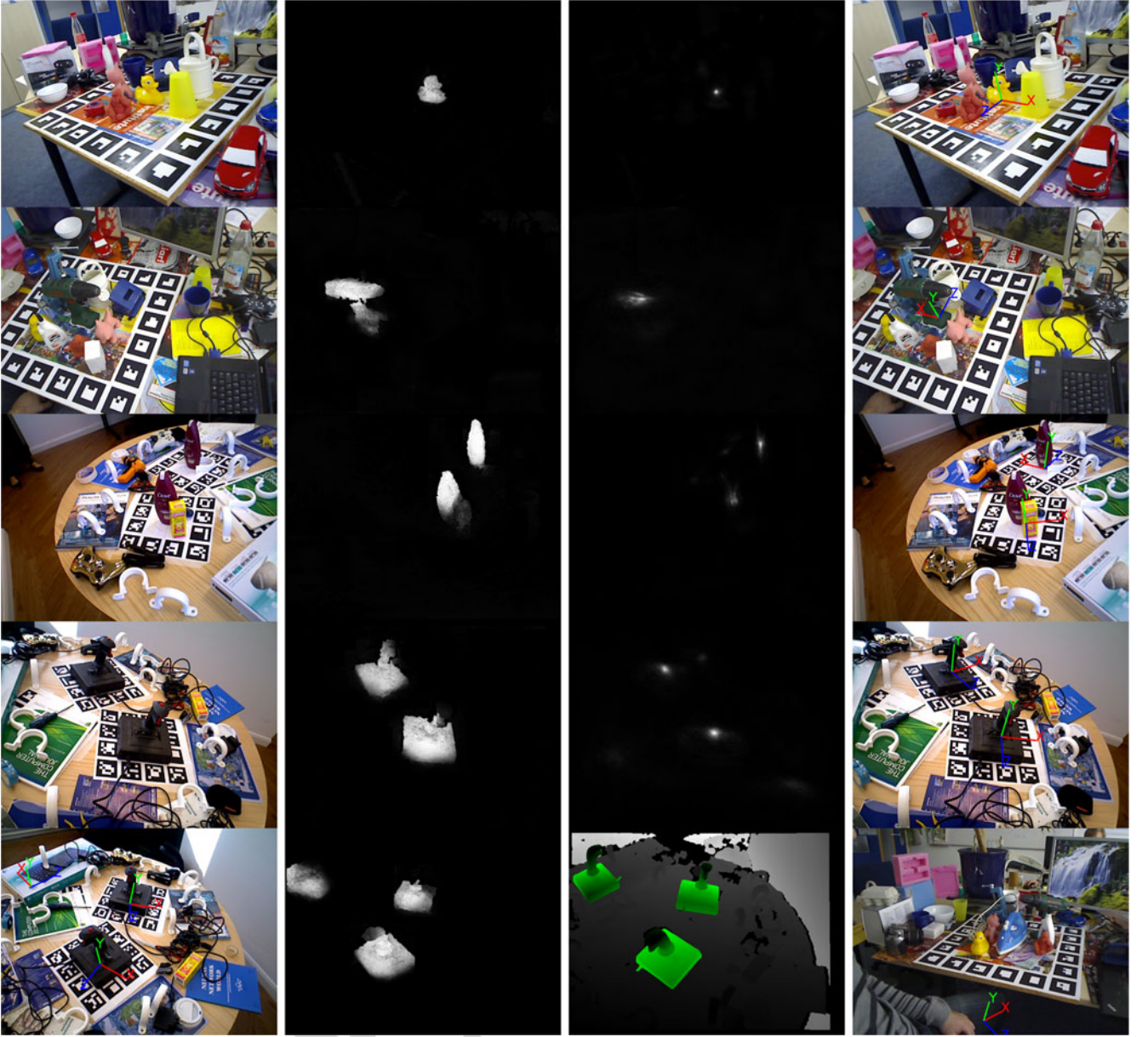| **Domestic Environment Dataset**		
	Latent-Class Hough Forests	
Object	[4]	5 trees 1/2 patch	10 trees 2/3 patch
Coffee Cup	91.9%	93.1%	**95.2%**
Shampoo	83.3%	87.6%	**88.8%**
Joystick	**76.3%**	71.6%	74.9%
Camera	72.2%	76.1%	**80.8%**
Juice Carton	90.2%	91.6%	**93.9%**
Milk	48.9%	51.2%	**52.7%**
Average	77.1%	78.5%	**81.1%**

Fig. 10. Rows 1-4 show, from left to right, the original RGB image, the final segmentation mask, the final Hough vote map and the augmented 3D axis of the estimated result. The final row shows some incorrect results, from left to right: one false positive leading to a false segmentation mask and wrong 3D rendering and finally a false negative.

arts in both datasets. The method of Drost et al. [10] has considerably lower precision values due to the fact that it does not take object boundaries into consideration, thus large planar regions of the target object can have a large surface overlap in the background clutter causing many false positives in addition to the true positives. Conversely, our method maintains high levels of precision at high recall which is due to the inference process simplifying the Hough space.

We have also compared our method with the one of Brachmann et al. [4] that is designed to estimate only one object per image. For a fair comparison, we compare the scores of the top hypothesis produced by the respective methods per image. Table 4 shows the detection results for the method of Brachmann et al. [4] and two different versions of our *Latent-Class Hough Forests*. In Fig. 10 we present some qualitative results on both our *Domestic Environment Dataset* and the one of LINEMOD [16]. A video demonstrating the efficiency of our method in this dataset is also available.[2]

TABLE 5
Percentages of Accurate Detections and F1-Scores for Two Versions of Our *Latent-Class Hough Forests* and the Method of Brachmann et al. [4]

Bin-Picking Dataset		
	Latent-Class Hough Forests	
Object	[4]	5 trees 1/2 patch	10 trees 2/3 patch
Coffee Cup	89.4%	90.1% (0.521)	**91.2%** (0.542)
Juice Carton	87.6%	89.6% (0.484)	**90.4%** (0.492)
Average	88.5%	89.8% (0.502)	**90.8%** (0.517)

2. https://www.youtube.com/watch?v=idY3Q7wg5rk

Fig. 11. Rows 1, 3 and 5 show the augmented 3D axis of the estimated result for our *Bin-picking Dataset*. Rows 2, 4 and 6 illustrate the overlaid mesh of the registered objects for the scene right above. The final row shows cases of false positives.

4.2.4 3rd Dataset: Bin-Picking

One of the most widely encountered application in industry is the one of robotic manipulation and, specifically, the one of manipulating similar objects placed in bins (e.g., grasping bolts and screws from a box). It is apparent that, this particular working scenario is very challenging for any vision algorithm since several cascading issues arise (e.g., severe occlusions, foreground clutter etc.). Annotating objects stacked in bins can not be done via the methodology we used for our *Domestic Environments Dataset*, where we placed markers beneath each testing object in order to acquire the necessary ground truth measurements. Since there is no simple way of obtaining the latter, Liu et al. [24] devised a statistical evaluation of the pose estimation consistency across multiple viewpoints of the camera. They utilized a robot arm to register the different viewpoints, while the final metric is a histogram of the deviations from the median pose estimation.

Unilke [24], we put manual labor to provide, to the best of our knowledge, the first fully annotated *Bin-picking Dataset*. The building of the dataset was divided into two phases, first, registering the viewpoints and second, annotating the objets. A freely moving handheld RGB-D camera was used to capture the testing frames, which, in turn, are registered by manually selecting key-point correspondences across all sequences. Afterwards, we annotate each object in the testing scenes by manually projecting the 3D mesh of the targets onto the respective point clouds of the testing sequences. The dataset comprises of multiple instances of two objects (Juice Carton and Coffee Cup) in more than 150 testing images.

In Table 5 we show the efficiency of our method in this challenging dataset in the form of accurate detection and F1-Scores. Similar to our previous comparison with the method of Brachmann et al. [4], we evaluate the scores of the top hypothesis produced by the respective methods per image. F1-Scores are calculated only for our method which is designed to work with multi-instance objects. In Fig. 11 we present several qualitative results on the *Bin-picking Dataset*. Our system is capable of achieving near real-time execution without any GPU processing. A video demonstrating the efficiency of our method in the *Bin-picking Dataset* is also available.[3]

5 CONCLUSION

In this paper we have introduced a novel framework for accurate 3D detection and pose estimation of multiple object instances in cluttered and occluded scenes. We have demonstrated that these challenges can be efficiently met via the adoption of a state of the art template matching feature into a patch-based regression forest. During training we employ a one-class learning scheme, i.e., training with positive samples only rather than involving negative examples. In turn, during inference, we engage the proposed *Latent-Class Hough Forests* that iteratively produce a more accurate estimation of the clutter/occluder distribution by considering class distribution as latent variables. As a result, apart from accurate detection results we can, further, obtain highly representative occlusion-aware masks facilitating further tasks

such as scene layout understanding, occlusion aware ICP or online domain adaption to name a few. Our method is evaluated using both the public dataset of Hinterstoisser et al. [16] and our new challenging ones containing foreground occlusion (severe in cases of the *Bin-Picking Dataset*) and multiple object instances. Experimental evaluation provides evidence of our novel *Latent-Class Hough Forest* outperforming all baselines highlighting the potential benefits of part-based strategies to address the issues of such a challenging problem.

REFERENCES

[1] A. Aldoma, F. Tombari, L. Di Stefano , and M. Vincze, "A global hypotheses verification method for 3D object recognition," in *Proc. 12th Eur. Conf. Comput. Vis.*, 2012, pp. 511–524.
[2] C. M. Bishop, "Novelty detection and neural network validation," *IEE Proc. Vis. Image Signal Process.*, vol. 141, no. 4, pp. 217–222, 1994.
[3] U. Bonde, V. Badrinarayanan, and R. Cipolla, "Robust instance recognition in presence of occlusion and clutter," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 520–535.
[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D object pose estimation using 3D object coordinates," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 536–551.
[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, 2001.
[6] A. G. Buch, Y. Yang, N. Kruger, and H. G. Petersen, "In search of inliers: 3D correspondence by local and global voting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2075–2082.
[7] A. Crivellaro, M. Rad, Y. Verdie, K. Moo Yi , P. Fua, and V. Lepetit, "A novel representation of parts for accurate 3D object detection and tracking in monocular images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 4391–4399.
[8] A. Doumanoglou, V. Balntas, R. Kouskouridas, and T.-K. Kim, "Siamese regression networks with efficient mid-level feature extraction for 3D object pose estimation," *arXiv:1607.02257*, 2016.
[9] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "6D object detection and next-best-view prediction in the crowd," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3583–3592.
[10] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 998–1005.
[11] G. Fanelli, J. Gall, and L. Van Gool, "Real time head pose estimation with random regression forests," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 617–624.
[12] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky, "Hough forests for object detection, tracking, and action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2188–2202, Nov. 2011.
[13] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 415–422.
[14] S. Hinterstoisser, et al., "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 858–865.
[15] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 2257–2264.
[16] S. Hinterstoisser, et al. "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Proc. 11th Asian Conf. Comput. Vis.*, 2012, pp. 548–562.
[17] E. Hsiao and M. Hebert, "Occlusion reasoning for object detection under arbitrary viewpoint," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3146–3153.
[18] I. Kokkinos and P. Maragos, "Synergy between object recognition and image segmentation using the expectation-maximization algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 8, pp. 1486–1501, Aug. 2009.

3. https://www.youtube.com/watch?v=dh2VtnnsGuY

[19] R. Kouskouridas, K. Charalampous, and A. Gasteratos, "Sparse pose manifolds," *Auton. Robots*, vol. 37, pp. 191–207, 2014.

[20] R. Kouskouridas, A. Gasteratos, and C. Emmanouilidis, "Efficient representation and feature extraction for neural network-based 3D object pose estimation," *Neurocomputing*, vol. 120, pp. 90–100, 2013.

[21] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-DOF model based tracking via object coordinate regression," in *Proc. 12th Asian Conf. Comput. Vis.*, 2014, pp. 384–399.

[22] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Proc. ECCV Workshop Statist. Learn. Comput. Vis.*, 2004, pp. 17–32.

[23] J. J. Lim, A. Khosla, and A. Torralba, "FPM: Fine pose parts-based model with 3D CAD models," in *Proc. 13th Eur. Conf. Comput. Sci.*, 2014, pp. 478–493.

[24] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, Y. Taguchi, and T. K. Marks, "Fast object localization and pose estimation in heavy clutter for robotic bin picking," *Int. J. Robot. Res.*, vol. 31, pp. 951–973, 2012.

[25] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-SVMs for object detection and beyond," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 89–96.

[26] M. Mathias, R. Benenson, R. Timofte, and L. van Gool, "Handling occlusions with Franken-classifiers," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1505–1512.

[27] M. M. Moya and D. R. Hush, "Network constraints and multi-objective optimization for one-class classification," *Neural Netw.*, vol. 9, pp. 463–474, 1996.

[28] M. Moya, M. Koch, and L. Hostetler, "One-class classifier networks for target recognition applications," Sandia Nat. Labs., Albuquerque, NM, USA, Tech. Rep. SAND-93-0084C, 1993.

[29] L. Parra, G. Deco, and S. Miesbach, "Statistical independence and novelty detection with information preserving nonlinear maps," *Neural Comput.*, vol. 8, pp. 260–269, 1996.

[30] B. Pepikj, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 3286–3293.

[31] R. Rios-Cabrera and T. Tuytelaars, "Discriminatively trained templates for 3D object detection: A real time scalable approach," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 2048–2055.

[32] M. Ristin, M. Guillaumin, J. Gall, and L. Van Gool, "Incremental learning of NCM forests for large-scale image classification," in *IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3654–3661.

[33] C. Sahin, R. Kouskouridas, and T.-K. Kim, "A learning-based variable size part extraction architecture for 6D object pose recovery in depth," *arXiv: 1701.02166*, 2017.

[34] B. Schölkopf, J. C. Platt, J. Shawe-Taylor , A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[35] J. Shotton, et al., "Real-time human pose recognition in parts from single depth images," *Commun. ACM*, vol. 56, pp. 116–124, 2013.

[36] S. Song and J. Xiao, "Sliding shapes for 3D object detection in depth images," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 634–651.

[37] D. Tang, Y. Liu, and T.-K. Kim, "Fast pedestrian detection by cascaded random forest with dominant orientation templates," in *Proc. British Mach. Vis. Conf.*, 2012, pp. 58.1–58.11.

[38] D. Tang, T.-H. Yu, and T.-K. Kim, "Real-time articulated hand pose estimation using semi-supervised transductive regression forests," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 3224–3231.

[39] D. M. Tax, *One-Class Classification*. TU Delft, The Netherlands, Delft Univ. Technol., 2001.

[40] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-class Hough forests for 3D object detection and pose estimation," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 462–477.

[41] T. Wang, X. He, and N. Barnes, "Learning structured Hough voting for joint object detection and occlusion reasoning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 1790–1797.

[42] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3109–3118.

[43] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1912–1920.

Alykhan Tejani received the MEng degree in computing from Imperial College London and the MPhil degree from the Computer Vision and Learning Lab, Imperial College London. He has also worked at the mobile app company Blippar, where he was part of the team developing their visual search and augmented reality platforms. His current research interests include large-scale object recognition and 3D object pose estimation.

Rigas Kouskouridas received the PhD degree from the Department of Production and Management Engineering, Democritus University of Thrace, in 2013. Afterwards, he was a postdoctoral research associate in the Imperial Computer Vision and Learning Lab, Electrical and Electronic Engineering Department, Imperial College London. Currently, he is computer vision and machine learning scientist in WIREWAX Interactive Video Technology.

Andreas Doumanoglou received the MEng degree in electrical and electronic engineering from Aristotle University of Thessaloniki, in 2009. Currently, he is working toward the PhD degree in the Imperial Computer Vision and Learning Lab, Electrical and Electronic Engineering Department, Imperial College London. He works in collaboration with CERTH-ITI and his areas of interests include robot vision and machine learning.
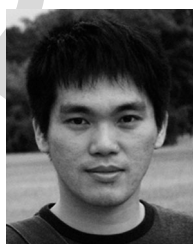
Danhang Tang received the BSc degree from Sun Yat-Sen University, the MSc (1st Hons.) degree from the University College London, and the PhD degree from Imperial College London. He is a senior scientist and founding team member with perceptiveIO, Inc. Before that, he was a visiting researcher with Microsoft Research Redmond. From 2007 to 2009, he worked as a system architect for Evryx Technologies Ltd., in support for SnapNow, one of the first image recognition apps in the world. During this time he co-drafted the visual search specification for the China Telecom.
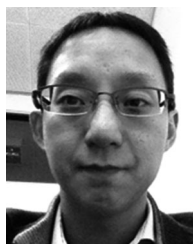
Tae-Kyun Kim received the PhD degree from the University of Cambridge. He is currently an associate professor, and the leader of Computer Vision and Learning Lab, Imperial College London, since 2010. On the topics of hand pose, face recognition by image sets, 6D object pose, action/activity, robot vision, he has published more than 40 top-tier journal and conference papers. His co-authored algorithm is an international standard of MPEG-7 ISO/IEC for face image retrieval, and he is a recipient of KUKA best robotics paper award at ICRA14. He has been co-chairing CVPR HANDS workshops and ICCV/ECCV object pose workshops, and is an associate editor of the *Image and Vision Computing Journal*, and the *IPSJ Transactions on Computer Vision and Applications*. He is the general chair of BMVC17. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.