Deep Template-based Object Instance Detection

Jean-Philippe Mercier, Mathieu Garon, Philippe Giguère and Jean-François Lalonde
Laval University
Quebec City, Canada

Abstract

Much of the focus in the object detection literature has been on the problem of identifying the bounding box of a particular class of object in an image. Yet, in contexts such as robotics and augmented reality, it is often necessary to find a specific *object instance—a unique toy or a custom industrial part for example—rather than a generic object class. Here, applications can require a rapid shift from one object instance to another, thus requiring fast turnaround which affords little-to-no training time. What is more, gathering a dataset and training a model for every new object instance to be detected can be an expensive and time-consuming process. In this context, we propose a generic 2D object* instance *detection approach that uses example viewpoints of the target object at test time to retrieve its 2D location in RGB images, without requiring any additional training (i.e. fine-tuning) step. To this end, we present an end-to-end architecture that extracts global and local information of the object from its viewpoints. The global information is used to tune early filters in the backbone while local viewpoints are correlated with the input image. Our method offers an improvement of almost 30 mAP over the previous template matching methods on the challenging Occluded Linemod [3] dataset (overall mAP of 50.7). Our experiments also show that our single generic model (not trained on any of the test objects) yields detection results that are on par with approaches that are trained specifically on the target objects.*

1. Introduction

Object detection is one of the key problems in computer vision. While there has been significant effort and progress in detecting generic object classes (e.g. detect all the phones in an image), comparatively little attention has been devoted to detect specific object instances (e.g. detect *this particular* phone model). Recent approaches on this topic [30, 41, 44, 22] have achieved very good performance in detecting object instances, even under challenging occlusions. By relying on textured 3D models as a way to specify the object instances to be detected, these methods propose to train detectors tailored for these objects. Because they know the objects to be detected at training time, these approaches essentially *overfit* to the objects themselves: they become specialized at detecting them (and only them).

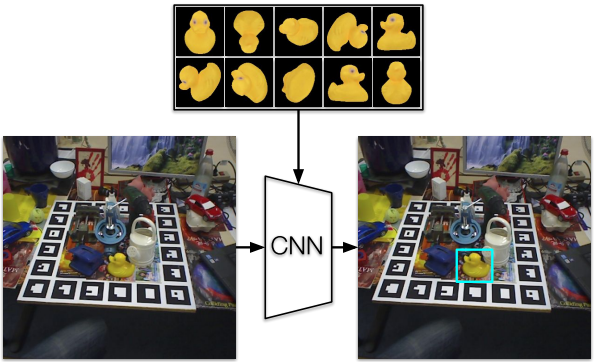

Figure 1: Overview of the proposed method. At test time, our network predicts the 2D location (in an RGB image) of a target object (unseen during training) represented by templates acquired from various viewpoints.

While this is a promising and active research direction, requiring knowledge of the objects to be detected at training time might not always be practical. For instance, if a new object needs to be detected, the entire training process must be started over. This implies first generating a full training dataset and then optimizing the network. Also, using a single network per object can be a severe constraint in embedded applications where memory is a limited resource.

In this work, we explore the case of training a *generic* 2D instance detector, where the specific object instance to be detected is only known at test time. The object to be found is represented by a set of images of that object captured from different viewpoints (fig. 1). In order to simplify the data capture setup and to facilitate comparisons to previous work on standard datasets, in this work we employ 3D models of the test objects and render different viewpoints. If a 3D model is not accessible, it would be possible to instead capture a few viewpoints of the object on a plain background.

This paper is akin to a line of work which has received somewhat less attention recently, that of template match-

ing. These techniques scan the image over a dense set of sub-windows and compare each of them with a template representing the object. A canonical example is Linemod [12], which detects a 3D object by treating several views of the object as templates, and by efficiently searching for matches over the image. While very efficient, traditional template matching techniques can be quite brittle, especially under occlusion, and yield large amounts of false positives.

In this paper, we revive this line of work and propose a novel instance detection method. Using a philosophy sharing resemblance to meta-learning [39], our method uses a large-scale 3D object dataset and a rendering pipeline to learn a versatile template representation. At test time, our approach takes as input multiple viewpoints of any object and detects it from a single RGB image immediately, without any additional training (fig. 1).

Our main contribution is the design of a novel deep learning architecture which can localize instances of a target object from a set of input templates. Instead of matching pixel intensities directly such as other template matching methods, our network is trained to localize an instance from a joint embedding space. Our approach is trained exclusively on synthetic data and operates by using a single RGB image as input. In addition, we introduce a series of extensions to the architecture which improve the detection performance such as tunable filters to adapt the feature extraction process to the object instance in the early layers of a pretrained backbone. We quantify the contribution of each extension through a detailed ablation study. Finally, we present extensive experiments that demonstrate that our method can successfully detect object instances that were not seen during training. In particular, we report performances that significantly outperform the state-of-the-art on the well-known Occluded Linemod [3] dataset. Notably, we attain a mAP of 50.71%, which is almost 30% better than LINE-2D [12] and on par with methods that overfit on the object instance during training.

2. Related work

Our work is most related to two areas: object instance detection in RGB images, and 2D tracking in RGB images.

Object instance detection. Our work focuses on retrieving the 2D bounding box of a *particular object instance*. This is in contrast with well-known methods such as Faster-RCNN [32] and SSD [26] or with methods that bear more resemblance to our approach such as CoAE [19], which all provide 2D locations of *object classes*. Detecting a specific object is challenging due to the large variety of objects that can be found in the wild. Descriptor-based and template-based methods are useful in such context. Generic features such as gradient histograms [12] and color histograms [35] can be computed and then retrieved from an object codebook.

Recent progress in deep learning enabled the community to develop approaches that automatically learn features from the 3D model of an object using neural network [30, 41, 44, 22] or random forest [4] classifiers. While these methods perform exceptionally well on known benchmarks [17], they share the important limitation that training these deep neural networks requires a huge amount of labeled data tailored to the object instances to be detected. Consequently, gathering the training dataset for specific objects is both costly and time-consuming. Despite this, efforts have been made to capture such real datasets [3, 13, 16, 33, 7, 34] and to combine them together [17]. A side effect is that it confines most deep learning methods to the very limited set of objects present in these datasets, as the weights of a network are specifically tuned to detect only a single [22] or a few instances [22, 30]. The difficulty of gathering a real dataset can be partially alleviated using simple rendering techniques [14, 22, 30] combined with data augmentation such as random backgrounds and domain randomization [36, 43, 37], but still suffers from a domain gap with real images. Recently, Hodan et al. [18] demonstrated that the domain gap can be minimized with physics-based rendering. Despite this progress, all of the above methods share the same limitation, in that they all require significant time (and compute power) to train a network on a new object. This implies a slow turn-around time, where a practitioner must wait hours before a new object can be detected.

To circumvent these limitations, we propose a novel generic network architecture that is trained to detect a target object that is unavailable at training time. Our method is trained on a large set of objects and can generalize to new, different objects at test time. Our architecture bears resemblance to TDID [1] that uses a template to detect a particular instance of an object. We show in our experiments that our method performs significantly better than [1] on objects not seen during training.

Tracking in 2D images. Our work shares architectural similarities with 2D image-based tracking, for which approaches use a template of the object as input, typically identified as a bounding box in the first frame of the video. In contrast, we focus on single frame detection. Thus, we employ known viewpoints of the object acquired offline. Many of these tracking approaches propose to use an in-network cross-correlation operation (sometimes denoted as $\star_d$) between a template and an image in feature space [40, 5, 23]. Additionally, recent 6-DOF trackers achieve generic instance tracking using simple viewpoint renders from a 3D model [9, 24, 27]. These methods are limited by the requirement of a previous temporal state in order to infer the current position. Our method takes inspiration from these two lines of work by first using the in-network cross-correlation and second, our experiments show that using renders is sufficient to *locate* a specific object instance from a single RGB image.

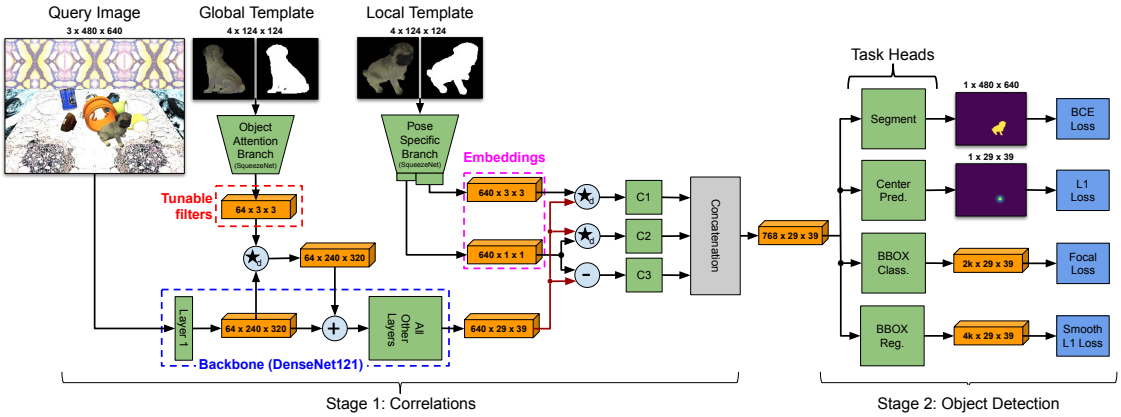

Figure 2: Our proposed architecture, shown during training. In stage 1, the network learns to localize an object solely from a set of templates. Object-specific features are learned by the "object attention" and "pose-specific" branches, and are subsequently correlated/subtracted with the generic features of the backbone network. In stage 2, the network leverages the learned representation to perform different tasks: binary segmentation, center and bounding box prediction.

3. Network architecture

We first introduce an overview of our proposed network architecture, depicted in fig. 2. Then, we discuss the two main stages of our architecture: 1) correlation and 2) object detection. The correlation stage borrows from classical template matching methods, where the template of an object is compared to the query image in a sliding-window fashion. The second stage is inspired from the recent literature in class-based object detection.

3.1. Architecture overview

We design an architecture that receives knowledge of the object as input, computes the template correlation as a first stage, and regresses bounding boxes around the object from the correlation results in a second stage. As shown in fig. 2, the network takes as input the RGB query image and two types of templates: 1) a *global* template used as an object attention mechanism to specialize early features in the backbone network; and 2) a *local* template that helps extract viewpoint-related features. Each template is an RGB image representing the rendered 3D object from a given viewpoint on a black background, concatenated with its binary mask to form four-channel images. The templates are obtained with a fast OpenGL render of the object with diffuse reflectance, ambient occlusion and lit by a combination of one overhead directional light and constant ambient lighting.

3.2. Correlation stage

The query image is first processed by a conventional backbone to extract a latent feature representation. The global template is fed to an "Object Attention Branch" (OAB), whose task is to inject a set of tunable filters early into this backbone network such that the features get specialized to the particular object instance. On the other hand, the local template is consumed by the "Pose-Specific Branch" (PSB) to compute an embedding of the object. The resulting features are then correlated with the backbone features using simple cross-correlation operations. Note that at test time, the backbone (85% of total computing) is processed only once per instance, while the second stage is computed for each template.

Backbone network. The role of the backbone network is to extract meaningful features from the query image. For this, we use a DenseNet121 [20] model pretrained on ImageNet [6]. Importantly, this network is augmented by adding a set of tunable filters between the first layer of the backbone (7×7 convolution layer with stride 2) and the rest of the model. These tunable filters are adjusted by the Object Attention Branch, described below.

Object attention branch (OAB). It has been widely studied that using a pretrained backbone provides better features initialization [29]. For a task related to template matching, this however limits the feature extraction process to be generic and not specialized early on to a particular instance (e.g. it is not necessary to have a high activation on blue objects if we are looking for a red object.). Thus, a specialized branch named "Object Attention Branch" (OAB) guides the low-level feature extraction of the backbone network by injecting high-level information pertaining to the object of interest. The output of the OAB can be seen as tunable filters, which are correlated with the feature map of the first layer of the backbone network. The correlation is done within a residual block, similarly to what is done in Residual Networks [11]. Our ablation study in sec. 5.3 demonstrate that these tunable filters are instrumental in conferring to a fixed backbone the ability to generalize to objects not seen during training.

The OAB network is a SqueezeNet [21] pretrained on

ImageNet, selected for its relatively small memory footprint and good performance. In order to receive a four-channel input (RGB and binary mask), an extra channel is added to the first convolution layer. The pretrained weights for the first three channels are kept and the weights of the fourth channel are initialized by the Kaiming method [10]. During training, a different pose of the target object is sampled at each iteration. For testing, a random pose is sampled once and used on all test images.

Pose-specific branch (PSB). Since an object instance can greatly vary depending on its viewpoint, a "pose-specific branch" (PSB) is trained to produce a high-level representation (*embeddings*) of the input object under various poses. This search, based on learned features, is accomplished by depth-wise correlations *and* subtraction with 1×1 local templates applied on the backbone output feature map. This correlation/subtraction approach is inspired by [1], where they have demonstrated an increased detection performance when combining these two operations with 1×1 embeddings. Siamese-based object trackers [2, 40] also use correlations, but with embeddings of higher spatial resolution. We found beneficial to merge these two concepts in our architecture, by using depth-wise correlations (denoted as $\star_d$) in both 1×1 and 3×3 spatial dimensions. The first one is devoid of spatial information, whereas the second one preserves some of the spatial relationships within a template. We conjecture that this increases sensitivity to orientation, thus providing some cues about the object pose.

This PSB branch has the same structure and weight initialization as the OAB, but is trained with its own specialized weights. The output of that branch are two local template embeddings: at 1×1 and 3×3 spatial resolution respectively. Depth-wise correlations (1×1 and 3×3) and subtractions (1×1) are applied between the embeddings generated by this branch and the feature maps extracted from the backbone. All of them are processed by subsequent 3×3 convolutions (C1–C3) and are then concatenated.

At test time, the object viewpoint is not known. Therefore, a stack of templates from multiple viewpoints are provided to the pose specific branch. Processing time can be saved at runtime by computing the templates embeddings in an offline phase. Note that the correlation between the local templates and the extracted features is a fast operation and can be easily applied in batch. The backbone network is only processed once per object instance.

3.3. Object detection stage

The second stage of the network deals with estimating object information from the learned correlation map. The architecture comprises a main task (bounding box prediction) and two auxiliary tasks (segmentation and center prediction).

Bounding box prediction. The bounding box classification and regression tasks are used to predict the presence and location of the object respectively (as in [25]). The classification head predicts the presence/absence of the object for k anchors at every location of the feature map while the regression head predicts a relative shift on the location (x, y) and size $(\text{width}, \text{length})$ with respect to every anchor. In our method, we have $k = 24$: 8 scales (30, 60, 90, 120, 150, 180, 210 and 240 pixels) and 3 different ratios (0.5, 1 and 2). Both heads are implemented as 5-layer convolution branches [25]. Inspired from RetinaNet [25], anchors with an Intersection-over-Union (IoU) of at least 0.5 are considered as positive examples, while those with IoU lower than 0.4 are considered as negatives. The other anchors between 0.4 and 0.5 are not used. At test time, bounding box predictions for all templates are accumulated and predictions with an (IoU) $>$ 0.5 are filtered by Non-Maximum Suppression (NMS). Also, for each bounding box prediction, a depth estimation can be made by multiplying the depth at which the local template was rendered with the size ratio between the local template size (124 pixels) and the prediction size. Predictions that have a predicted depth outside the chosen range of [0.4, 2.0] meters, which is a range that fits to most tabletop settings, are filtered out.

Segmentation and center prediction. The segmentation head predicts a pixel-wise binary mask of the object in the scene image at full resolution. The branch is composed of 5 convolution layers followed by $2\times$ bilinear upsampling layers. Additionally, the center prediction head predicts the location of the object center at the same resolution than the correlation map (29×39) to encourage a strong correlation. The correlation channels are compressed to a single channel heatmap with a 1×1 convolution layer.

Loss Functions. The network is trained end-to-end with a main (bounding box detection) and two auxiliary (segmentation and center prediction) tasks. As such, the training loss $\ell_{\text{train}} = \lambda_1 \ell_{\text{seg}} + \lambda_2 \ell_{\text{center}} + \ell_{\text{FL}} + \ell_{\text{reg}}$, where ℓ_{seg} is a binary cross-entropy loss for segmentation, ℓ_{center} is an L_1 loss for the prediction of the object center in a heatmap, ℓ_{FL} is a focal loss [25] associated with the object presence classification and ℓ_{reg} is a smooth-L_1 loss for bounding box regression. The weights λ_1, λ_2 were empirically set to 20.

4. Training data

In this section, we detail all information related to the input images (query and templates) during training. In particular, we define how the synthetic images are generated and how the dataset is augmented.

4.1. Domain randomization training images

We rely on 125 different textured 3D models gathered in majority from the various datasets of the 6D pose estimation benchmark [17] (excluding Linemod [12] since it is used for evaluation). Our fully-annotated training dataset is generated with a physic-based simulator similar to [28], for which

(a) (b) (c)

Figure 3: Examples from our domain randomization training set. In (a), objects are randomly placed in front of the camera and rendered using OpenGL with a background sampled from Sun3D dataset [42]. In (b) and (c), a physical simulation is used to drop several objects on a table with randomized parameters (camera position, textures, lighting, materials and anti-aliasing). For each render, 2 variations are saved: one with simple diffuse materials and without shadows (b), and one with more sophisticated specular materials and shadows (c).

objects are randomly dropped on a table in a physical simulation. Every simulation is done in a simple cubic room (four walls, a floor and a ceiling) containing a table placed on the floor in the middle of the room. Inspired from the success of domain randomization [36, 37], we added randomness to the simulation parameters in order to reduce the domain gap between synthetic and real images. The following parameters are randomized: the texture of the environment (walls, floor and table), lighting (placement, type, intensity and color), object materials (diffuse and specular reflection coefficients) and anti-aliasing (type and various parameters).

Renders. Our physics-based domain randomization dataset is composed of 10,000 images. To generate these images, we ran 250 different simulations with different sets of objects (between 4 and 13 objects in each simulation). In 50% of the simulations, objects were automatically repositioned to rest on their bottom/main surface to replicate a bias found in many tabletop datasets. For each simulation, 20 camera positions were randomly sampled on half-spheres of radius ranging from 0.8 to 1.4 meters, all pointing towards the table center with random offsets of ± 15 degrees for each rotation axis. For each sampled camera position, two image variations were rendered: one with realistic parameters (containing reflections and shadows) as shown in fig. 3-(c) and the other without, as shown in fig. 3-(b). Tremblay et al. [38] showed that using different kinds of synthetic images reduced the performance gap between synthetic and real images. Accordingly, we have generated an additional set of 10,000 simpler renders using OpenGL. For this, we rendered objects in random poses on top of real indoor backgrounds sampled from the Sun3D dataset [42] (fig. 3-(a)).

Labels. After the simulations, we kept the 6 degree of freedom pose of each object as the ground truth. We used the pose together with the 3D model to generate a visibility mask for the segmentation task, and projected the center of the 3D model in the image plane to generate the center heatmap. The ground-truth heatmap is a 2D Gaussian with an amplitude of 1 and a variance of 5 at the projected center of the object at an image resolution equivalent to the output of the network.

4.2. Templates

The following section describes the template generation procedure for training. We also remind the different procedure used at test time, as described in sec. 3.2.

For each training iteration, one of the objects from the query image is selected as the target object and all the others are considered as background. All templates are rendered with a resolution of 124×124 pixels. To render consistent templates from multiple objects of various size, we adjust the distance of the object so that its largest length on the image plane falls in the range of 100 to 115 pixels. The borders are then padded to reach the size of 124×124.

Global template (OAB): In an offline phase, 240 templates are generated for each 3D model by sampling 40 viewpoints on an icosahedron with 6 in-plane rotations per viewpoint. During training, one of the 240 templates is sampled randomly for each iteration. At test time, a single one is randomly selected for all experiments.

Local template (PSB): We apply perturbations on the orientation of the template image by sampling a random rotation axis and rotation magnitude, and adding that perturbation to the ground truth viewpoint before rendering the local template. The impact of using different rotation magnitude is quantified in table 2, with best performance obtained with random rotations perturbation in the range of 20–30° to the ground truth viewpoint. At test time, a stack of 160 templates rendered from 16 viewpoints is used.

4.3. Data augmentation

Online data augmentation is applied to synthetic images during training. We use the segmentation mask of the object

in the query image to randomly change the hue, saturation and brightness of the object and its template. We also apply augmentations on the whole query image, such as: brightness shifts, Gaussian blur and noise, horizontal and vertical flips, random translations and scale. To minimize the risk of overfitting to object color, a random hue is applied to the whole image and the template 50% of the time. Finally, we apply motion blur to the image 20% of the time by convolving a line kernel to the image, as in [8].

5. Experiments

In this section, we provide details on the training procedure and on the dataset and metrics used to evaluate our approach. We also describe the various ablation studies that validate our design choices. Finally, we present an extensive evaluation against the state-of-the-art methods.

5.1. Training details

Our complete network is trained for 50 epochs with AMSGrad [31].We used a learning rate of 10^{-4} with steps of 0.1 at epochs 20 and 40, a weight decay of 10^{-6} and mini batches of size 6. We used 1k renders as a validation set and used the remaining 19k of the generated dataset (OpenGL and physics-based) for training. Each epoch, the network was trained for 1,300 iterations and images are sampled with a ratio of 80/20 respectively from the physics-based and OpenGL renders. Once the training was complete, the network with the smallest validation loss (computed at the end of each epoch) was kept for testing.

5.2. Datasets and metrics

We evaluate on the well-known Linemod [13] and Occluded Linemod [3] datasets. Linemod consists of 15 sequences of real objects containing heavy clutter where the annotations of a single object are available per sequence. Occluded Linemod is a subset of Linemod, where annotations for 8 objects have been added by [3]. Keeping in line with previous work, we only keep the prediction with the highest score for each object and use the standard metrics listed below. We use a subset containing 25% of the Linemod dataset for the ablation studies.

Linemod. The standard metric for this dataset is the "2D bounding box" metric proposed in [3]. The metric calculates the ratio of images for which the predicted bounding box has an intersection-over-union (IoU) with the ground truth higher than 0.5.

Occluded Linemod. The standard mean average precision (mAP) is used to evaluate the performance of multi-object detection. To allow for direct comparison, we regroup the predictions made for different objects and apply NMS on predictions with an IoU > 0.5. We use the same methodology as in [3]: the methods are evaluated on 13 of the 15 objects of the Linemod dataset (the "bowl" and "cup" objects are left out). Of the remaining 13 objects, 4 are never found in the images, yet those are still detected and kept in the evaluation (as an attempt to evaluate the robustness to missing objects). The mAP is therefore computed by using all the predictions on the 9 other objects left.

Network	Δ performance (%)
w/o tunable filters (OAB)	-19.76
w/o auxiliary tasks	-7.73
w/o 3×3 correlation (PSB)	-5.37

Table 1: Network architecture ablation study. Removing tunable filters resulted in the most notable performance drop.

5.3. Ablation studies

Network architecture. We evaluate the importance of different architecture modules (presented in sec. 3). For each test, a specific module is removed and its performance is compared to the full architecture. Tab. 1 shows that removing the "Object Attention Branch" resulted in the largest performance drop (almost 20%). Also, removing the higher-resolution 3×3 embeddings and auxiliary tasks reduced performances by approximately 5% and 8% respectively.

Importance of local template perturbation during training. A perfect match between the template pose and the target object pose in the scene is unlikely. As such, the training procedure must take this into account by adding orientation perturbations to local templates at train time. Here, we investigated what is the desirable magnitude of such perturbations. In tab. 2, a random rotation of 0° represents local templates selected with the same orientation as the object in the scene. Perturbations are then added by randomly sampling a rotation axis (in spherical coordinates) and a magnitude. A network was retrained for each amount of perturbation. Tab. 2 illustrates that perturbations of 20–30° seems to be optimal. Networks trained with too small perturbations may not be able to detect objects under all their possible configurations, resulting in small performances drop of less than 5%, and those trained with too big perturbations are more prone to false detections (the network is trained to allow for bigger differences in appearance and shape between the template and scene object), resulting in a bigger drop of 16% for rotations of 180°.

Number of local templates. The impact of providing various numbers of local templates to the network at test time is evaluated, both in terms of accuracy and speed, in tab. 3. Timings are reported on a Nvidia GeForce GTX 1080Ti. To generate a varying number of templates, we first selected 16 pre-defined viewpoints spanning a half-sphere on top of the object. Each template subsequently underwent 5 (80 templates), 10 (160 templates) and 20 (320 templates) in-plane

Random rotations	Δ performance (%)
0°	-4.33
± 10°	-3.12
± 20°	0
± 30°	-0.42
± 40°	-5.18
± 180°	-16.07

Table 2: Impact of perturbing the pose of local templates (instead of using the the ground truth pose) during training.

# of templates	Δ performance (%)	runtime (ms)
80	-2.80	230
160	0.00	430
320	+0.03	870
1 (oracle)	+16.75	60

Table 3: Bounding box detection performance and runtime for various numbers of local templates at test time. The oracle sets an upper bound of performance by providing a single template with the ground truth object pose.

Global template selection	Δ performance (%)
Random Pose	+1.21
Empty	-32.47
Wrong Object	-38.15

Table 4: Robustness towards different selection of global templates at test time.

rotations. Tab. 3 compares performances with that obtained with an oracle who provided a template with the ground truth object pose. Overall, performance ceases to improve beyond 160 templates.

Global template selection. In tab. 4, we show that the object pose of the global template does not impact significantly the detection performance. For the first test, we report the average performance of 5 different evaluations in which a random global template was selected. The performance slightly improved compared to the random template used in all other tests, suggesting that the template selection in every other test was suboptimal. However, it also shows that the templates were not cherry-picked for optimal performance on the test datasets. Secondly, we show that using a template of the good object is primordial. Using empty templates (all 0's) or providing templates from another object results in a dramatic performance drop of more than 30%, thus hinting about the discriminative power of the OAB.

Number of objects in the training set. The network was retrained on subsets of objects of the synthetic dataset. The remaining objects were considered as background clutter.

Figure 4: Qualitative results on the Occluded Linemod dataset [3], showing good (green), false (blue) and missed (red) detections. For reference, the 15 objects are shown in the bottom row (image from [17]). To generate these results, all objects (except objects 3 and 7) are searched in each image.

# of objects	Δ performance (%)
15	-31.58
30	-15.97
63	-10.42
90	-3.58
125	0

Table 5: Impact of the number of objects used in training.

Tab. 5 shows the performance w.r.t the quantity of objects used during training. While using few objects still performs reasonably well, more objects does improve generalization.

Similar objects in the training set. While no single object were present in both training and test sets, it is possible that the training set contained objects that shared similarities to objects in the test set. To evaluate the potential impact this might have, we removed all cups from our training set (13 were found), trained a network on the resulting set, and evaluated its performance on the test set. Doing so reduced the overall score by less than 1%, but the average performance solely on cups slightly improved (not statistically significant). This experiment demonstrates that the network does not overfit on a particular class of instances.

5.4. Comparative evaluation to the state of the art

We report an evaluation on Linemod and Occluded Linemod (OL) in tab. 6 and compare with other state-of-the-art RGB-only methods. Competing methods are divided into 2 main groups: those who do know the test objects at train time ("known objects"), and those who do not. Ap-

Methods	Known objects	Real images	Linemod (2D BBox)	OL (mAP)
Brachmann et al. [4]	Yes	Yes	97.50	51.00
SSD-6D [22]	Yes	No	99.40	38.00
DPOD [44]	Yes	No	N/A	48.00
Hodan et al. [18]	Yes	No	N/A	55.90*
Tjaden et al. [35]	No	Yes	78.50	N/A
LINE-2D [12]	No	No	86.50	21.0
TDID corrs. [1]	No	No	54.37	34.13
SiamMask corrs. [40]	No	No	68.23	41.47
Ours	No	No	77.92	50.71

Table 6: Quantitative comparison to the state of the art, with 2D bounding box metric on Linemod and mean average precision (mAP) on Occluded Linemod (OL). The 2D bounding box metric calculates the recall for the 2D bounding boxes with the highest prediction score. For both metrics, predictions are considered good if the IoU of the prediction and the ground truth is at least 0.5 (0.75 for Hodan et al. [18]). The methods are separated first according to their prior knowledge of test objects and then if real images similar to the test set are used to optimize the performance. Our approach is the most robust of all methods that were not trained for the test objects, having a good score on Linemod and the best score on Occluded Linemod.

proaches such as [4, 22, 44, 18] are all learning-based methods that were specifically trained on the objects. On the other hand, [35, 12] and [1, 40] are respectively template matching and learning-based methods that do not include a specific training step targeted towards specific object instances. It is worth noting that even though [35] is classified as not needing known objects at training time, it still requires an initialization phase using real images (to build a dictionary of histogram features). As in [4], they thus use parts of the Linemod dataset as a training set that covers most of the object viewpoints. These methods have therefore an unfair advantage compared to our approach and Line-2D, since they leverage domain-specific information (lighting, camera, background) of the evaluation dataset.

Our method is evaluated without prior knowledge of the Linemod objects. It can be directly compared with Line-2D [12] which also uses templates as input. On the standard Linemod dataset, Line-2D outperforms our method by 8.5% on the "2D bounding box" metric. The better results of Line-2D on Linemod can be explained in part by an additional and naive post-processing color-based check that rejects false positives [4] while we report the performance of our approach without any post-processing. We note that this naive approach fails if minor occlusions occurs. In contrast, our method outperforms Line-2D by almost 30% in mAP on the more difficult Occluded Linemod. Our approach also provides competitive performance that is on par or close to all other methods that test on known objects and/or have access to real images. Fig. 4 shows qualitative results on Occluded Linemod. We also compare our approach with TDID [1] and SiamMask [40]. We replaced their original backbones by the same architecture (DenseNet) we are using. As specified by those methods, a siamese backbone replaced our 2 branches approach (OAB and PSB). TDID uses a 1×1 embedding whereas a 3×3 embedding is used for SiamMask. All implementations were trained on the same task following the same procedure than our approach and their scores are reported in tab. 6. Overall, our proposed approach significantly outperforms these two baselines.

6. Discussion

We have proposed a method for detecting specific object instances in an image that does not require knowledge of the object at training time. At test time, the proposed network takes multiple viewpoints of the object as input, and predicts the location of this object from a single RGB image. Our experiments show that while the network has not been trained with any of the test objects, it is significantly more robust to occlusion than previous template-based methods (30% improvement in mAP over Line-2D [12]). It is also highly competitive with networks that are specifically trained on the object instances. Numerous ablation studies show the importance of each part of our proposed network.

Limitations. False positives arise from clutter with similar color/shape as the object, as shown in fig. 4. We hypothesize that our late correlation at small spatial resolution (templates of 3×3 and 1×1) prevents the network from leveraging detailed spatial information related to the object shape. Another limitation is that the method requires 0.43s to detect a single object instance in an image (c.f. tab. 3), scaling linearly with the number of objects. The main reason for this is the object attention branch (OAB), which makes the backbone features instance-specific via tunable filters, which needs to be recomputed for each object. Also, while capturing a 3D model has become increasingly simpler ([15] show that it can be done in less than 5 minutes with commodity hardware), this may not always be practical. While our experiments rely on such 3D models to allow for quantitative evaluation on standard datasets (e.g. Linemod) for which only the 3D model is available, obtaining multiple viewpoints of an object could also be done simply by photographing it against a uniform background.

Future directions. By providing a generic and robust 2D instance detection framework, this work opens the way for new methods that can extract additional information about the object, such as its full 6-DOF pose. We envision a potential cascaded approach, which could first detect unseen objects, and subsequently regress the object pose from a high-resolution version of the detection window.

References

[1] Phil Ammirato, Cheng-Yang Fu, Mykhailo Shvets, Jana Kosecka, and Alexander C Berg. Target driven instance detection. *arXiv preprint arXiv:1803.04610*, 2018.

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European Conference on Computer Vision*, 2016.

[3] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *European Conference on Computer Vision*, 2014.

[4] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[5] Achal Dave, Pavel Tokmakov, Cordelia Schmid, and Deva Ramanan. Learning to track any object. *arXiv preprint arXiv:1910.11844*, 2019.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2009.

[7] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[8] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *The IEEE International Conference on Computer Vision*, Oct 2017.

[9] Mathieu Garon, Denis Laurendeau, and Jean-François Lalonde. A framework for evaluating 6-DOF object trackers. In *European Conference on Computer Vision*, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision*, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2016.

[12] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *IEEE International Conference on Computer Vision*, 2011.

[13] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Asian Conference on Computer Vision*, 2012.

[14] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. In *European Conference on Computer Vision*, 2018.

[15] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In *IEEE International Conference on Computer Vision Workshops*, 2019.

[16] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *IEEE Winter Conference on Applications of Computer Vision*, 2017.

[17] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *European Conference on Computer Vision*, 2018.

[18] Tomas Hodan, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. *arXiv preprint arXiv:1902.03334*, 2019.

[19] Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. One-shot object detection with co-attention and co-excitation. In *Advances in Neural Information Processing Systems*, pages 2725–2734, 2019.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2017.

[21] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[22] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *IEEE International Conference on Computer Vision*, 2017.

[23] Xin Li, Chao Ma, Baoyuan Wu, Zhenyu He, and Ming-Hsuan Yang. Target-aware deep tracking. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2019.

[24] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *European Conference on Computer Vision*, 2018.

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision*, 2017.

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[27] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *European Conference on Computer Vision*, 2018.

[28] Chaitanya Mitash, Kostas E Bekris, and Abdeslam Boularias. A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[29] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[30] Mahdi Rad and Vincent Lepetit. Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *IEEE International Conference on Computer Vision*, 2017.

[31] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *International Conference on Learning Representations*, 2019.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015.

[33] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters*, 1(2):1179–1185, 2016.

[34] Alykhan Tejani, Danhang Tang, Rigas Kouskouridas, and Tae-Kyun Kim. Latent-class hough forests for 3d object detection and pose estimation. In *European Conference on Computer Vision*, 2014.

[35] Henning Tjaden, Ulrich Schwanecke, and Elmar Schomer. Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms. In *IEEE International Conference on Computer Vision*, 2017.

[36] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[37] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *IEEE On Computer Vision And Pattern Recognition Workshops*, 2018.

[38] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[39] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 2016.

[40] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *IEEE Conference on Computer Vision And Pattern Recognition*, 2019.

[41] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[42] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *IEEE International Conference on Computer Vision*, 2013.

[43] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. Deceptionnet: Network-driven domain randomization. *IEEE International Conference on Computer Vision*, 2019.

[44] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: Dense 6d pose object detector in rgb images. In *IEEE International Conference on Computer Vision*, 2019.