SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again

Wadim Kehl [1,2,*] Fabian Manhardt [2,*] Federico Tombari [2] Slobodan Ilic [2,3] Nassir Navab [2]
[1] Toyota Research Institute, Los Altos [2] Technical University of Munich [3] Siemens R&D, Munich
wadim.kehl@tri.global fabian.manhardt@tum.de tombari@in.tum.de

Abstract

We present a novel method for detecting 3D model instances and estimating their 6D poses from RGB data in a single shot. To this end, we extend the popular SSD paradigm to cover the full 6D pose space and train on synthetic model data only. Our approach competes or surpasses current state-of-the-art methods that leverage RGB-D data on multiple challenging datasets. Furthermore, our method produces these results at around 10Hz, which is many times faster than the related methods. For the sake of reproducibility, we make our trained networks and detection code publicly available.[1]

1. Introduction

While category-level classification and detection from images has recently experienced a tremendous leap forward thanks to deep learning, the same has not yet happened for what concerns 3D model localization and 6D object pose estimation. In contrast to large-scale classification challenges such as PASCAL VOC [9] or ILSVRC [26], the domain of 6D pose estimation requires instance detection of known 3D CAD models with high precision and accurate poses, as demanded by applications in the context of augmented reality and robotic manipulation.

Most of the best performing 3D detectors follow a view-based paradigm, in which a discrete set of object views is generated and used for subsequent feature computation [31, 14]. During testing, the scene is sampled at discrete positions, features computed and then matched against the object database to establish correspondences among training views and scene locations. Features can either be an encoding of image properties (color gradients, depth values, normal orientations) [12, 16, 18] or, more recently, the result of learning [4, 29, 5, 6, 17]. In either case, the accuracy of both detection and pose estimation hinges on three aspects: (1) the coverage of the 6D pose space in terms of viewpoint and scale, (2) the discriminative power of the features to tell objects and views apart and (3) the robustness of matching towards clutter, illumination and occlusion.

CNN-based category detectors such as YOLO [25] or SSD [22] have shown terrific results on large-scale 2D datasets. Their idea is to inverse the sampling strategy such that scene sampling is not anymore a set of discrete input points leading to continuous output. Instead, the input space is dense on the whole image and the output space is discretized into many overlapping bounding boxes of varying shapes and sizes. This inversion allows for smooth scale search over many differently-sized feature maps and simultaneous classification of all boxes in a single pass. In order to compensate for the discretization of the output domain, each bounding box regresses a refinement of its corners.

The goal of this work is to develop a deep network for object detection that can accurately deal with 3D models and 6D pose estimation by assuming an RGB image as unique input at test time. To this end, we bring the concept of SSD over to this domain with the following contributions: (1) a training stage that makes use of synthetic 3D model information only, (2) a decomposition of the model pose space that allows for easy training and handling of symmetries and (3) an extension of SSD that produces 2D detections and infers proper 6D poses.

We argue that in most cases, color information alone can already provide close to perfect detection rates with good poses. Although our method does not need depth data, it is readily available with RGB-D sensors and almost all recent state-of-the-art 3D detectors make use of it for both feature computation and final pose refinement. We will thus treat depth as an optional modality for hypothesis verification and pose refinement and will assess the performance of our method with both 2D and 3D error metrics on multiple challenging datasets for the case of RGB and RGB-D data.

Throughout experimental results on multiple benchmark datasets, we demonstrate that our color-based approach is competitive with respect to state-of-the-art detectors that leverage RGB-D data or can even outperform them, while being many times faster. Indeed, we show that the prevalent trend of overly relying on depth for 3D instance detection is not justified when using color correctly.

[1] https://wadimkehl.github.io/

* The first two authors contributed equally to this work.

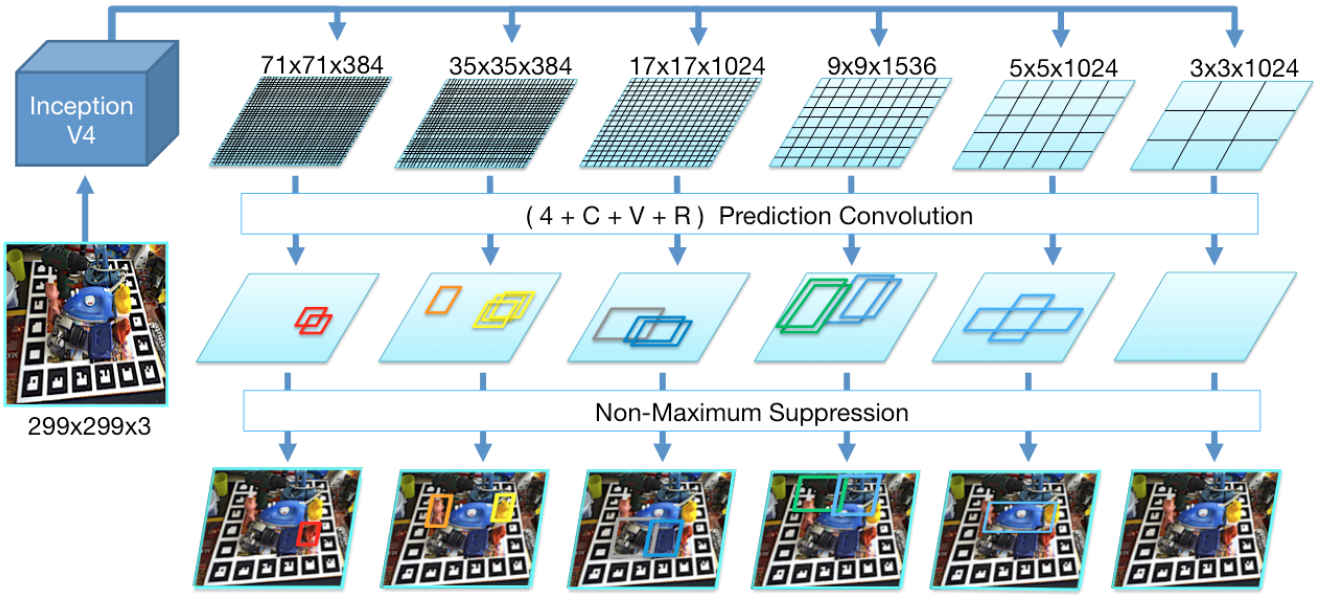


Figure 1: Schematic overview of the SSD-style network prediction. We feed our network with a 299 × 299 RGB image and produce six feature maps at different scales from the input image using branches from InceptionV4. Each map is then convolved with trained prediction kernels of shape (4 + C + V + R) to determine object class, 2D bounding box as well as scores for possible viewpoints and in-plane rotations that are parsed to build 6D pose hypotheses. Thereby, C denotes the number of object classes, V the number of viewpoints and R the number of in-plane rotation classes. The other 4 values are utilized to refine the corners of the discrete bounding boxes to tightly fit the detected object.

2. Related work

We will first focus on recent work in the domain of 3D detection and 6D pose estimation before taking a closer look at SSD-style methods for category-level problems.

To cover the upper hemisphere of one object with a small degree of in-plane rotation at multiple distances, the authors in [14] need 3115 template views over contour gradients and interior normals. Hashing of such views has been used to achieve sub-linear matching complexity [18, 16], but this usually trades speed for accuracy. Related scale-invariant approaches [16, 4, 29, 6, 17] employ depth information as an integral part for either feature learning or extraction, thus avoiding scale-space search and cutting down the number of views by around an order of magnitude. Since they require depth to work, they can fail when depth is missing or erroneous. While scale can be inferred with RGB-D data, there has not been yet any convincing work to eradicate the requirement of in-plane rotated views. Rotation-invariant methods are based on local keypoints in either 2D [32] or 3D [7, 3, 30] by explicitly computing or voting for an orientation or a local reference frame, but they fail for objects of poor geometry or texture.

Although rarely mentioned, all of the view-based methods cover only a very small, predefined 6D pose space. Placing the object differently, e.g. on its head, would lead to failure if this view had not been specifically included during training. Unfortunately, additional views increase computation and add to overall ambiguity in the matching stage. Even worse, for all discussed methods, scene sampling is crucial. If too coarse, objects of smaller scale can be missed whereas a fine-grained sampling increases computation and often leads to more false positive detections. Therefore, we explore a path similar to works on large-scale classification where dense feature maps on multiple scales have produced state-of-the-art results. Instead of relying on classifying proposed bounding boxes [10, 11, 21], whose performance hinges on the proposals' quality, recent single-shot detectors [25, 22] classify a (large) discrete set of fixed bounding boxes. This streamlines the network architecture and gives freedom to the a-priori placement of boxes.

As for works regressing the pose from RGB images, the related works of [24, 23] recently extended SSD to include pose estimates for categories. [23] infers 3D bounding boxes of objects in urban traffic and regresses 3D box corners and an azimuth angle whereas [24] introduces an additional binning of poses to express not only the category but also a notion of local orientation such as 'bike from the side' or 'plane from below'. The difference to us is that they train on real images to predict poses in a very constrained subspace. Instead, our domain demands training on synthetic model-based data and the need to encompass the full

6D pose space to accomplish tasks such as grasping or AR.

3. Methodology

The input to our method is an RGB image that is processed by the network to output localized 2D detections with bounding boxes. Additionally, each 2D box is provided with a pool of the most likely 6D poses for that instance. To represent a 6D pose, we parse the scores for viewpoint and in-plane rotation that have been inferred from the network and use projective properties to instantiate 6D hypotheses. In a final step, we refine each pose in every pool and select the best after verification. This last step can either be conducted in 2D or optionally in 3D if depth data is available. We present each part now in more detail.

3.1. Network architecture

Our base network is derived from a pre-trained InceptionV4 instance [27] and is fed with a color image (resized to 299×299) to compute feature maps at multiple scales. In order to get our first feature map of dimensionality $71 \times 71 \times 384$, we branch off before the last pooling layer within the stem and append one 'Inception-A' block. Thereafter, we successively branch off after the 'Inception-A' blocks for a $35 \times 35 \times 384$ feature map, after the 'Inception-B' blocks for a $17 \times 17 \times 1024$ feature map and after the 'Inception-C' blocks for a $9 \times 9 \times 1536$ map.[2] To cover objects at larger scale, we extend the network with two more parts. First, a 'Reduction-B' followed by two 'Inception-C' blocks to output a $5 \times 5 \times 1024$ map. Second, one 'Reduction-B' and one 'Inception-C' to produce a $3 \times 3 \times 1024$ map.

From here we follow the paradigm of SSD. Specifically, each of these six feature maps is convolved with prediction kernels that are supposed to regress localized detections from feature map positions. Let (w_s, h_s, c_s) be the width, height and channel depth at scale s. For each scale, we train a $3 \times 3 \times c_s$ kernel that provides for each feature map location the scores for object ID, discrete viewpoint and in-plane rotation. Since we introduce a discretization error by this grid, we create B_s bounding boxes at each location with different aspect ratios. Additionally, we regress a refinement of their four corners. If C, V, R are the numbers of object classes, sampled viewpoints and in-plane rotations respectively, we produce a $(w_s, h_s, B_s \times (C+V+R+4))$ detection map for the scale s. The network has a total number of 21222 possible bounding boxes in different shapes and sizes. While this might seem high, the actual runtime of our method is remarkably low thanks to the fully-convolutional design and the good true negative behavior, which tend to yield a very confident and small set of detections. We refer to Figure 1 for a schematic overview.

[2] We changed the padding of Inception-B s.t. the next block contains a map with odd dimensionality to always contain a central position.

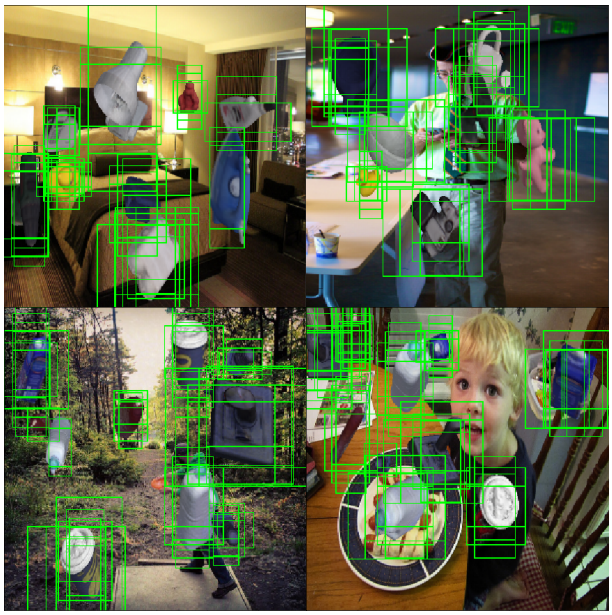


Figure 2: Exemplary training images for the datasets used. Using MS COCO images as background, we render object instances with random poses into the scene. The green boxes visualize the network's bounding boxes that have been assigned as positive samples for training.

Viewpoint scoring versus pose regression The choice of viewpoint classification over pose regression is deliberate. Although works that do direct rotation regression exist [19, 28], early experimentation showed clearly that the classification approach is more reliable for the task of detecting poses. In particular, it seems that the layers do a better job at scoring discrete viewpoints than at outputting numerically accurate translations and rotations. The decomposition of a 6D pose in viewpoint and in-plane rotation is elegant and allows us to tackle the problem more naturally. While a new viewpoint exhibits a new visual structure, an in-plane rotated view is a non-linear transformation of the same view. Furthermore, simultaneous scoring of all views allows us to parse multiple detections at a given image location, *e.g.* by accepting all viewpoints above a certain threshold. Equally important, this approach allows us to deal with symmetries or views of similar appearance in a straight-forward fashion.

3.2. Training stage

We take random images from MS COCO [20] as background and render our objects with random transformations into the scene using OpenGL commands. For each rendered instance, we compute the IoU (intersection over union) of each box with the rendered mask and every box b with IoU > 0.5 is taken as a positive sample for this object class. Additionally, we determine for the used transformation its

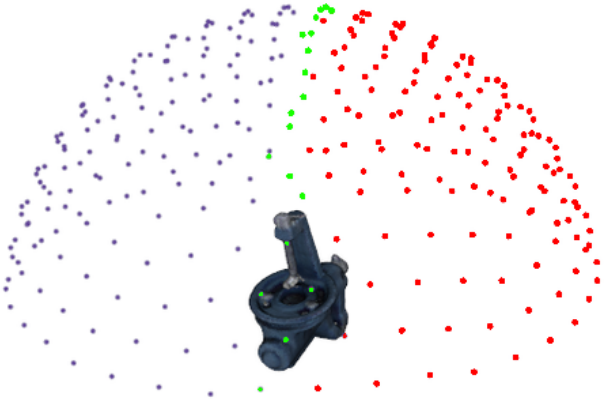


Figure 3: Discrete 6D pose space with each point representing a classifiable viewpoint. If symmetric, we use only the green points for view ID assignment during training whereas semi-symmetric objects use the red points as well.

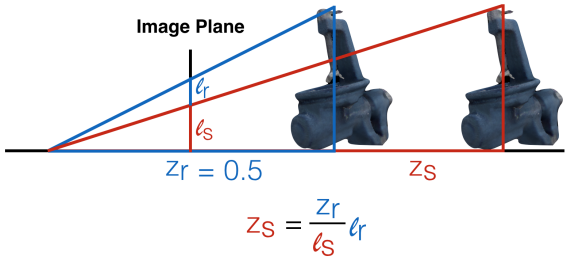


Figure 4: For each object we precomputed the perfect bounding box and the 2D object centroid with respect to each possible discrete rotation in a prior offline stage. To this end, we rendered the object at a canonical centroid distance $z_r = 0.5m$. Subsequently, the object distance z_s can be inferred from the projective ratio according to $z_s = \frac{l_r}{l_s} z_r$, where l_r denotes diagonal length of the precomputed bounding box and l_s denotes the diagonal length of the predicted bounding box on the image plane.

closest sampled discrete viewpoint and in-plane rotation as well as set its four corner values to the tightest fit around the mask as a regression target. We show some training images in Figure 2.

Similar to SSD [22], we employ many different kinds of augmentation, such as changing the brightness and contrast of the image. Differently to them, though, we do not flip the images since it would lead to confusion between views and to wrong pose detections later on. We also make sure that each training image contains a 1:2 positives-negatives ratio by selecting hard negatives (unassigned boxes with high object probability) during back-propagation.

Our loss is similar to the MultiBox loss of SSD or YOLO, but we extend the formulation to take discrete views and in-plane rotations into account. Given a set of positive boxes Pos and hard-mined negative boxes Neg for a training image, we minimize the following energy:

$$L(Pos, Neg) := \sum_{b \in Neg} L_{class} + \sum_{b \in Pos} (L_{class} + \alpha L_{fit} + \beta L_{view} + \gamma L_{inplane}) \quad (1)$$

As it can be seen from (1), we sum over positive and negative boxes for class probabilities (L_{class}). Additionally, each positive box contributes weighted terms for viewpoint (L_{view}) and in-plane classification ($L_{inplane}$), as well as a fitting error of the boxes' corners (L_{fit}). For the classification terms, i.e., L_{class}, L_{view}, $L_{inplane}$, we employ a standard softmax cross-entropy loss, whereas a more robust smooth L1-norm is used for corner regression (L_{fit}).

Dealing with symmetry and view ambiguity Our approach demands the elimination of viewpoint confusion for proper convergence. We thus have to treat symmetrical or semi-symmetrical (constructible with plane reflection) objects with special care. Given an equidistantly-sampled sphere from which we take our viewpoints, we discard positions that lead to ambiguity. For symmetric objects, we solely sample views along an arc, whereas for semi-symmetric objects we omit one hemisphere entirely. This approach easily generalizes to cope with views which are mutually indistinguishable although this might require manual annotation for specific objects in practice. In essence, we simply ignore certain views from the output of the convolutional classifiers during testing and take special care of viewpoint assignment in training. We refer to Figure 3 for a visualization of the pose space.

3.3. Detection stage

We run a forward-pass on the input image to collect all detections above a certain threshold, followed by non-maximum suppression. This yields refined and tight 2D bounding boxes with an associated object ID and scores for all views and in-plane rotations. For each detected 2D box we thus parse the most confident views as well as in-plane rotations to build a pool of 6D hypotheses from which we select the best after refinement. See Figure 5 for the pooled hypotheses and Figure 6 for the final output.

3.3.1 From 2D bounding box to 6D hypothesis

So far, all computation has been conducted on the image plane and we need to find a way to hypothesize 6D poses from our network output. We can easily construct a 3D rotation, given view ID and in-plane rotation ID, and can use the bounding box to infer 3D translation. To this end, we

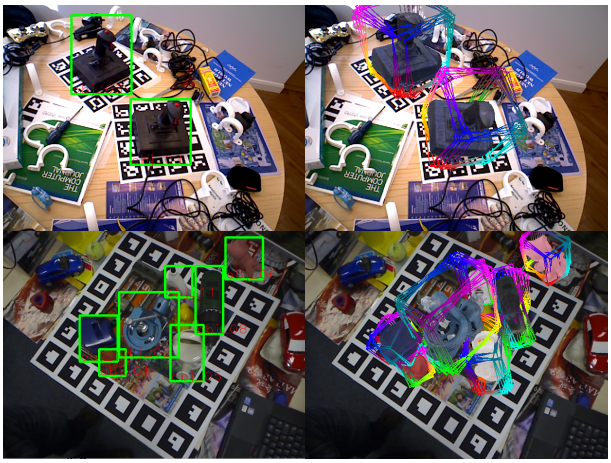


Figure 5: Prediction output and 6D pose pooling of our network on the Tejani dataset and the multi-object dataset. Each 2D prediction builds a pool of 6D poses by parsing the most confident views and in-plane rotations. Since our networks are trained with various augmentations, they can adapt to different global illumination settings.

render all possible combinations of discrete views and in-plane rotations at a canonical centroid distance $z_r = 0.5m$ in an offline stage and compute their bounding boxes. Given the diagonal length l_r of the bounding box during this offline stage and the one predicted by the network l_r, we can infer the object distance $z_s = \frac{l_r}{l_s} z_r$ from their projective ratio, as illustrated in Figure 4. In a similar fashion, we can derive the projected centroid position and back-project to a 3D point with known camera intrinsics.

3.3.2 Pose refinement and verification

The obtained poses are already quite accurate, yet can in general benefit from a further refinement. Since we will regard the problem for both RGB and RGB-D data, the pose refinement will either be done with an edge-based or cloud-based ICP approach. If using RGB only, we render each hypothesis into the scene and extract a sparse set of 3D contour points. Each 3D point X_i, projected to $\pi(X_i) = x_i$, then shoots a ray perpendicular to its orientation to find the closest scene edge y_i. We seek the best alignment of the 3D model such that the average projected error is minimal:

$$\arg\min_{R,t} \sum_i \left(||\pi(R \cdot X_i + t) - y_i||^2 \right). \qquad (2)$$

We minimize this energy with an IRLS approach (similar to [8]) and robustify it using Geman-McLure weighting. In the case of RGB-D, we render the current pose and solve with standard projective ICP with a point-to-plane formulation in closed form [2]. In both cases, we run multiple rounds of correspondence search to improve refinement and we use multi-threading to accelerate the process.

The above procedure provides multiple refined poses for each 2D box and we need to choose the best one. To this end, we employ a verification procedure. Using only RGB, we do a final rendering and compute the average deviation of orientation between contour gradients and overlapping scene gradients via absolute dot products. In case RGB-D data is available, we render the hypotheses and estimate camera-space normals to measure the similarity again with absolute dot products.

4. Evaluation

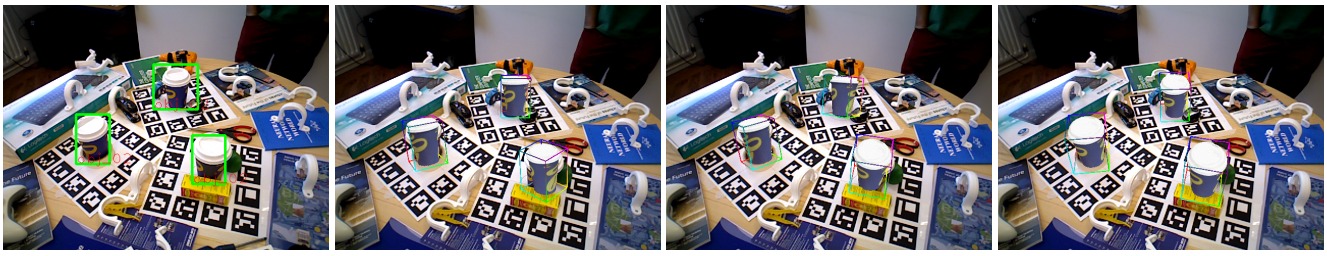
We implemented our method in C++ using TensorFlow 1.0 [1] and cuDNN 5 and ran it on a i7-5820K@3.3GHz with an NVIDIA GTX 1080. Our evaluation has been conducted on three datasets. The first, presented in Tejani et al. [29], consists of six sequences where each sequence requires the detection and pose estimation of multiple instances of the same object in clutter and with different levels of mild occlusion. The second dataset, presented in [14], consists of 15 sequences where each frame presents one instance to detect and the main challenge is the high amount of clutter in the scene. As others, we will skip two sequences since they lack a meshed model. The third dataset, presented in [4] is an extension of the second where one sequence has been annotated with instances of multiple objects undergoing heavy occlusions at times.

Network configuration and training To get the best results it is necessary to find an appropriate sampling of the model view space. If the sampling is too coarse we either miss an object in certain poses or build suboptimal 6D hypotheses whereas a very fine sampling can lead to a more difficult training. We found an equidistant sampling of the unit sphere into 642 views to work well in practice. Since the datasets only exhibit the upper hemisphere of the objects, we ended up with 337 possible view IDs. Additionally, we sampled the in-plane rotations from -45 to 45 degrees in steps of 5 to have a total of 19 bins.

Given the above configuration, we trained the last layers of the network and the predictor kernels using ADAM and a constant learning rate of 0.0003 until we saw convergence on a synthetic validation set. The balancing of the loss term weights proved to be vital to provide both good detections and poses. After multiple trials we determined $\alpha = 1.5$, $\beta = 2.5$ and $\gamma = 1.5$ to work well for us. We refer the reader to the supplementary material to see the error development for different configurations.

4.1. Single object scenario

Since 3D detection is a multi-stage pipeline for us, we first evaluate purely the 2D detection performance between



(a) 2D Detections (b) Unrefined (c) RGB refinement (d) RGB-D refinement

Figure 6: After predicting 2D detections (a), we build 6D hypotheses and run pose refinement and a final verification. While the unrefined poses (b) are rather approximate, contour-based refinement (c) produces already visually acceptable results. Occlusion-aware projective ICP with cloud data (d) leads to a very accurate alignment.

Sequence	LineMOD [12]	LC-HF [29]	Kehl [17]	Us
Camera	0.589	0.394	0.383	**0.741**
Coffee	0.942	0.891	0.972	**0.983**
Joystick	0.846	0.549	0.892	**0.997**
Juice	0.595	0.883	0.866	**0.919**
Milk	0.558	0.397	0.463	**0.780**
Shampoo	**0.922**	0.792	0.910	0.892
Total	0.740	0.651	0.747	**0.885**

Table 1: F1-scores on the re-annotated version of [29]. Although our method is the only one to solely use RGB data, our results are considerably higher than all related works.

our predicted boxes and the tight bounding boxes of the rendered groundtruth instances on the first two datasets. Note that we always conduct proper detection and not localization, *i.e.* we do not constrain the maximum number of allowed detections but instead accept all predictions above a chosen threshold. We count a detection to be correct when the IoU score of a predicted bounding box with the groundtruth box is higher than 0.5. We present our F1-scores in Tables 1 and 2 for different detection thresholds.

It is important to mention that the compared methods, which all use RGB-D data, allow a detection to survive after rigorous color- and depth-based checks whereas we use simple thresholding for each prediction. Therefore, it is easier for them to suppress false positives to increase their precision whereas our confidence comes from color cues only.

On the Tejani dataset we outperform all related RGB-D methods by a huge margin of 13.8% while using color only. We analyzed the detection quality on the two most difficult sequences. The 'camera' has instances of smaller scale which are partially occluded and therefore simply missed whereas the 'milk' sequence exhibits stronger occlusions in virtually every frame. Although we were able to detect the 'milk' instances, our predictors could not overcome the occlusions and regressed wrongly-sized boxes which were not tight enough to satisfy the IoU threshold. These were counted as false positives and thus lowered our recall[3].

On the second dataset we have mixed results where we can outperform state-of-the-art RGB-D methods on some sequences while being worse on others. For larger feature-rich objects like 'benchvise', 'iron' or 'driller' our method performs better than the related work since our network can draw from color and textural information. For some objects, such as 'lamp' or 'cam', the performance is worse than the related work. Our method relies on color information only and thus requires a certain color similarity between synthetic renderings of the CAD model and their appearance in the scene. Some objects exhibit specular effects (*i.e.* changing colors for different camera positions) or the frames can undergo sensor-side changes of exposure or white balancing, causing a color shift. Brachmann et al. [5] avoid this problem by training on a well-distributed subset of real sequence images. Our problem is much harder since we train on synthetic data only and must generalize to real, unseen imagery.

Our performance for objects of smaller scale such as 'ape', 'duck' and 'cat' is worse and we observed a drop both in recall and precision. We attribute the lower recall to our bounding box placement, which can have 'blind spots' at some locations and consequently, leading to situations where a small-scale instance cannot be covered sufficiently by any box to fire. The lower precision, on the other hand, stems from the fact that these objects are textureless and of uniform color which increases confusion with the heavy scene clutter.

4.1.1 Pose estimation

We chose for each object the threshold that yielded the highest F1-score and run all following pose estimation experiments with this setting. We are interested in the pose accuracy for all correctly detected instances.

[3] We refer to the supplement for more detailed graphs.

	ape	bvise	cam	can	cat	driller	duck	box	glue	holep	iron	lamp	phone
Our method	76.3	**97.1**	92.2	**93.1**	89.3	**97.8**	80.0	93.6	**76.3**	71.6	**98.2**	93.0	**92.4**
Kehl [17]	**98.1**	94.8	**93.4**	82.6	**98.1**	96.5	**97.9**	**100**	74.1	**97.9**	91.0	**98.2**	84.9
LineMOD [14]	53.3	84.6	64.0	51.2	65.6	69.1	58.0	86.0	43.8	51.6	68.3	67.5	56.3
LC-HF [29]	85.5	96.1	71.8	70.9	88.8	90.5	90.7	74.0	67.8	87.5	73.5	92.1	72.8

Table 2: F1-scores for each sequence of [14]. Note that the LineMOD scores are supplied from [29] with their evaluation since [14] does not provide them. Using color only we can easily compete with the other RGB-D based methods.

Sequence	IoU-2D	IoU-3D	VSS-2D	VSS-3D
Camera	0.973	0.904	0.693	0.778
Coffee	0.998	0.996	0.765	0.931
Joystick	1	0.953	0.655	0.866
Juice	0.994	0.962	0.742	0.865
Milk	0.970	0.990	0.722	0.810
Shampoo	0.993	0.974	0.767	0.874
Total	0.988	0.963	0.724	0.854

Table 3: Average pose errors for the Tejani dataset.

	RGB		
	Ours	Brachmann 2016 [5]	LineMOD [13]
IoU	99.4 %	97.5%	86.5%
ADD [12]	76.3%	50.2%	24.2%

	RGB-D		
	Ours	Brachmann 2016 [5]	Brachmann 2014 [4]
IoU	96.5 %	99.6%	99.1%
ADD [12]	90.9%	99.0%	97.4%

Table 4: Average pose errors for the LineMOD dataset.

Error metrics To measure 2D pose errors we will compute both an IoU score and a Visual Surface Similarity (VSS) [15]. The former is different than the detection IoU check since it measures the overlap of the rendered masks' bounding boxes between groundtruth and final pose estimate and accepts a pose if the overlap is larger than 0.5. VSS is a tighter measure since it counts the average pixel-wise overlap of the mask. This measure assesses well the suitability for AR applications and has the advantage of being agnostic towards the symmetry of objects. To measure the 3D pose error we use the ADD score from [14]. This assesses the accuracy for manipulation tasks by measuring the average deviation between transformed model point clouds of groundtruth and hypothesis. If it is smaller than $\frac{1}{10}th$ of the model diameter, it is counted as a correct pose.

Refinement with different parsing values As mentioned, we parse the most confident views and in-plane rotations to build a pool of 6D hypotheses for each 2D detection. Here, we want to assess the final pose accuracy

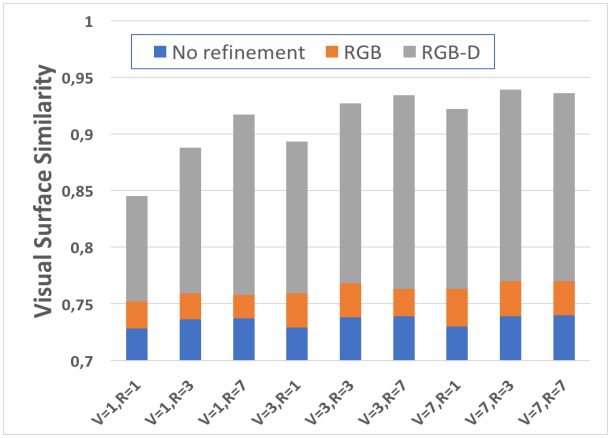


Figure 7: Average VSS scores for the 'coffee' object for different numbers of parsed views and in-plane rotations as well as different pose refinement options.

when changing the number of parsed views V and rotations R for different refinement strategies We present in Figure 7 the results on Tejani's 'coffee' sequence for the cases of no refinement, edge-based and cloud-based refinement (see Figure 6 for an example). To decide for the best pose we employ verification over contours for the first two cases and normals for the latter. As can be seen, the final poses without any refinement are imperfect but usually provide very good initializations for further processing. Additional 2D refinement yields better poses but cannot cope well with occluders whereas depth-based refinement leads to perfect poses in practice. The figure gives also insight for varying V and R for hypothesis pool creation. Naturally, with higher numbers the chances of finding a more accurate pose improve since we evaluate a larger portion of the 6D space. It is evident, however, that every additional parsed view V gives a larger benefit than taking more in-plane rotations R into the pool. We explain this by the fact that our viewpoint sampling is coarser than our in-plane sampling and thus reveals more uncovered pose space when parsed, which in turn helps especially depth-based refinement. Since we create a pool of $V \cdot R$ poses for each 2D detection, we fixed $V = 3, R = 3$ for all experiments as a compromise between accuracy and refinement runtime.

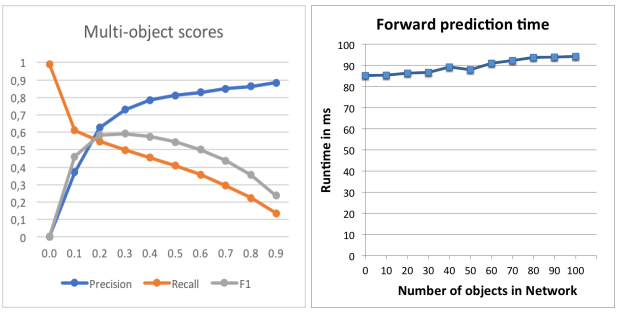


Figure 8: Left: Detection scores on the multi-object dataset for a different global threshold. Right: Runtime increase for the network prediction with an increased number of objects.

Performance on the two datasets We present our pose errors in Tables 3 and 4 after 2D and 3D refinement. Note that we do not compute the ADD scores for Tejani since each object is of (semi-)symmetric nature, leading always to near-perfect ADD scores of 1. The poses are visually accurate after 2D refinement and furthermore are boosted by an additional depth-based refinement stage. On the second dataset we are actually able to come very close to Brachmann et al. which is surprising since they have a huge advantage of real data training. For the case of pure RGB-based poses, we can even overtake their results. We provide more detailed error tables in the supplement.

4.2. Multiple object detection

The last dataset has annotations for 9 out of the 15 objects and is quite difficult since many instances undergo heavy occlusion. Different to the single object scenario, we have now a network with one global detection threshold for all objects and we present our scores in Figure 8 when varying this threshold. Brachmann et al. [5] can report an impressive Average Precision (AP) of 0.51 whereas we can report an AP of 0.38. It can be observed that our method degrades gracefully as the recall does not drop suddenly from one threshold step to the next. Note again that Brachmann et al. have the advantage of training on real images of the sequence whereas we must detect heavily-occluded objects from synthetic training only.

4.3. Runtime and scalability

For a single object in the database, Kehl et al. [17] report a runtime of around 650ms per frame whereas Brachmann et al. [4, 5] report around 450ms. Above methods are scalable and thus have a sublinear runtime growth with an increasing database size. Our method is a lot faster than the related work while being scalable as well. In particular, we can report a runtime of approximately 85ms for a single object. We show our prediction times in Figure 8 which reveals that we scale very well with an increasing number of objects in the network. While the prediction is fast, our pose refinement takes more time since we need to refine every pose of each pool. On average, given that we have about 3 to 5 positive detections per frame, we need a total of an additional 24ms for refinement, leading to a total runtime of around 10Hz.

Figure 9: One failure case where incorrect bounding box regression, induced by occlusion, led to wrong 6D hypothesis creation. In such cases a subsequent refinement cannot always recover the correct pose anymore.

4.4. Failure cases

The most prominent issue is the difference in colors between synthetic model and scene appearance, also including local illumination changes such as specular reflections. In these cases, the object confidence might fall under the detection threshold since the difference between the synthetic and the real domain is too large. A more advanced augmentation would be needed to successfully tackle this problem. Another possible problem can stem from the bounding box regression. If the regressed corners are not providing a tight fit, it can lead to translations that are too offset during 6D pose construction. An example of this problem can be seen in Figure 9 where the occluded milk produces wrong offsets. We also observed that small objects are sometimes difficult to detect which is even more true after resizing the input to 299×299. Again, designing a more robust training as well as a larger network input could be of benefit here.

Conclusion

To our knowledge, we are the first to present an SSD-style detector for 3D instance detection and full 6D pose estimation that is trained on synthetic model information. We have shown that color-based detectors are indeed able to match and surpass current state-of-the-art methods that leverage RGB-D data while being around one order of magnitude faster. Future work should include a higher robustness towards color deviation between CAD model and scene appearance. Avoiding the problem of proper loss term balancing is also an interesting direction for future research.

Acknowledgments We would like to thank Toyota Motor Corporation for funding and supporting this work.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. In *OSDI*, 2016. 5

[2] P. Besl and N. McKay. A Method for Registration of 3-D Shapes. *TPAMI*, 1992. 5

[3] T. Birdal and S. Ilic. Point Pair Features Based Object Detection and Pose Estimation Revisited. In *3DV*, 2015. 2

[4] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation using 3D Object Coordinates. In *ECCV*, 2014. 1, 2, 5, 7, 8

[5] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother. Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image. In *CVPR*, 2016. 1, 6, 7, 8

[6] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim. 6D Object Detection and Next-Best-View Prediction in the Crowd. In *CVPR*, 2016. 1, 2

[7] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: efcient and robust 3D object recognition. In *CVPR*, 2010. 2

[8] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *TPAMI*, 2002. 5

[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge: A Retrospective. *IJCV*, 2014. 1

[10] R. Girshick. Fast R-CNN. *arXiv:1504.08083*, 2015. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *TPAMI*, 2015. 2

[12] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Textureless Objects. *TPAMI*, 2012. 1, 6, 7

[13] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, 2011. 7

[14] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradsky, K. Konolige, and N. Navab. Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes. In *ACCV*, 2012. 1, 2, 5, 7

[15] T. Hodan, J. Matas, and S. Obdrzalek. On Evaluation of 6D Object Pose Estimation. In *ECCV Workshop*, 2016. 7

[16] T. Hodan, X. Zabulis, M. Lourakis, S. Obdrzalek, and J. Matas. Detection and Fine 3D Pose Estimation of Textureless Objects in RGB-D Images. In *IROS*, 2015. 1, 2

[17] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation. In *ECCV*, 2016. 1, 2, 6, 7, 8

[18] W. Kehl, F. Tombari, N. Navab, S. Ilic, and V. Lepetit. Hashmod: A Hashing Method for Scalable 3D Object Detection. In *BMVC*, 2015. 1, 2

[19] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, 2015. 3

[20] G. Lin, C. Shen, Q. Shi, A. V. D. Hengel, and D. Suter. Fast Supervised Hashing with Decision Trees for High-Dimensional Data. In *CVPR*, 2014. 3

[21] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *arXiv:1612.03144*, 2016. 2

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-y. Fu, and A. C. Berg. SSD : Single Shot MultiBox Detector. In *ECCV*, 2016. 1, 2, 4

[23] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. *arXiv:1612.00496*, 2016. 2

[24] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast Single Shot Detection and Pose Estimation. In *3DV*, 2016. 2

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1, 2

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1

[27] C. Szegedy, S. Ioffe, and V. Vanhoucke. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *arxiv:1602.07261*, 2016. 3

[28] D. J. Tan, F. Tombari, S. Ilic, and N. Navab. A Versatile Learning-based 3D Temporal Tracker : Scalable , Robust , Online. In *ICCV*, 2015. 3

[29] A. Tejani, D. Tang, R. Kouskouridas, and T.-k. Kim. Latent-class hough forests for 3D object detection and pose estimation. In *ECCV*, 2014. 1, 2, 5, 6, 7

[30] F. Tombari, S. Salti, and L. Di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010. 2

[31] M. Ulrich, C. Wiedemann, and C. Steger. Combining scale-space and similarity-based aspect graphs for fast 3D object recognition. *TPAMI*, 2012. 1

[32] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *ECCV*, 2016. 2

SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again Supplementary material

1. Object-wise detection scores

We present the detection score graphs for each object of the first two datasets in Figures 1 and 2 from which we determined the best object-wise threshold. For reproducibility, we list them in Tables 1 and 2.

Camera	Coffee	Joystick	Juice	Milk	Shampoo
0.55	0.35	0.5	0.25	0.3	0.45

Table 1: Object-wise thresholds for the Tejani dataset.

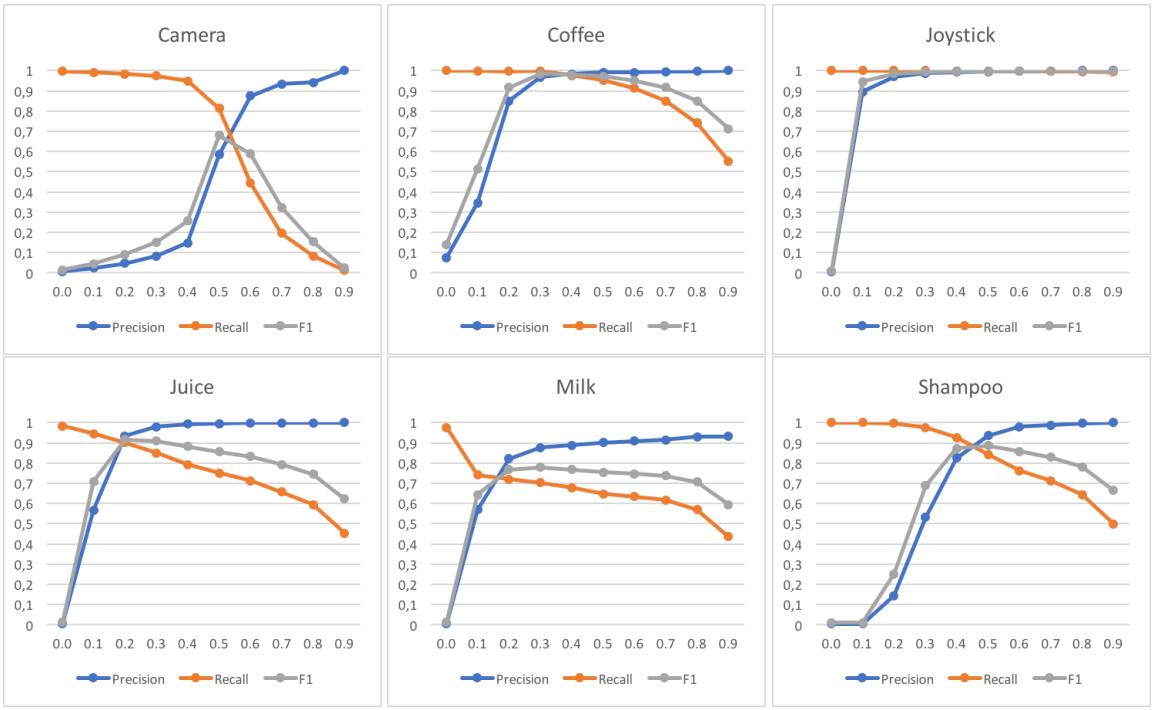


Figure 1: Plotting the detection scores for each object on the Tejani dataset for a varying threshold.

ape	bvise	cam	can	cat	driller	duck	box	glue	holep	iron	lamp	phone
0.5	0.15	0.2	0.75	0.35	0.25	0.25	0.25	0.4	0.4	0.3	0.55	0.35

Table 2: Object-wise thresholds for the LineMOD dataset.

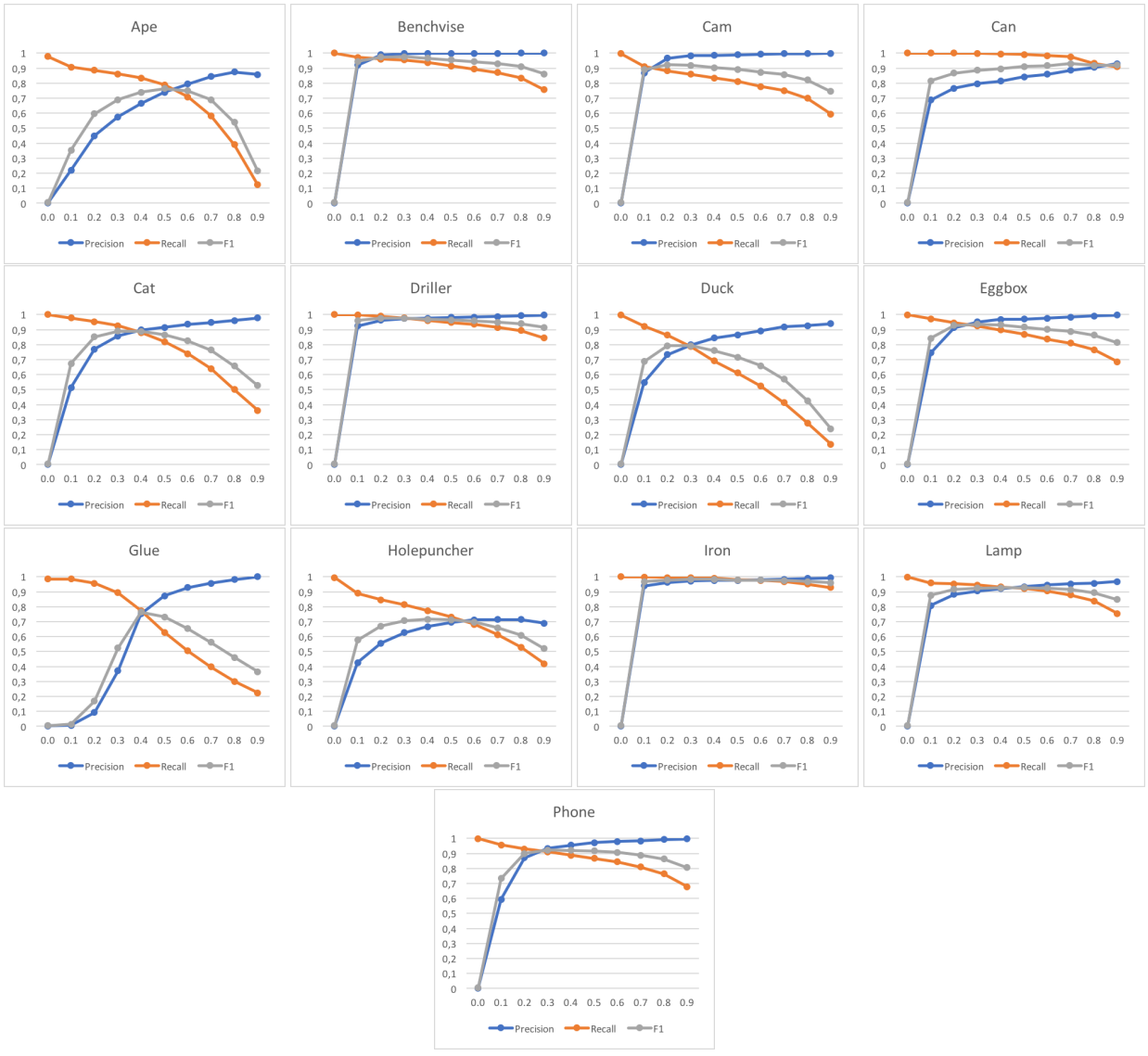


Figure 2: Plotting the detection scores for each object on the LineMOD dataset for a varying threshold.

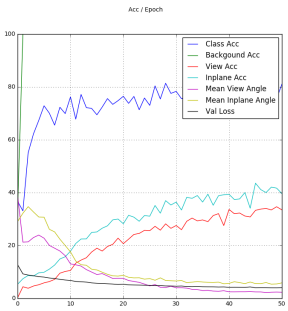
2. Detailed pose errors for the LineMOD dataset

	ape	bvise	cam	can	cat	driller	duck	box	glue	holep	iron	lamp	phone
IoU-2D	0.99	1.00	0.99	1.0	0.99	0.99	0.98	0.99	0.98	0.99	0.99	0.99	1.00
IoU-3D	0.96	0.98	0.98	0.99	0.95	0.95	0.95	0.98	0.89	0.97	0.97	0.98	0.93
VSS-2D	0.73	0.67	0.73	0.75	0.67	0.66	0.71	0.78	0.72	0.70	0.74	0.66	0.72
VSS-3D	0.84	0.88	0.90	0.86	0.81	0.84	0.83	0.88	0.75	0.77	0.85	0.84	0.81
ADD-2D	0.65	0.80	0.78	0.86	0.70	0.73	0.66	1.00	1.00	0.49	0.78	0.73	0.79
ADD-3D	0.85	0.94	0.94	0.94	0.86	0.85	0.82	1.00	1.00	0.73	0.95	0.87	0.87

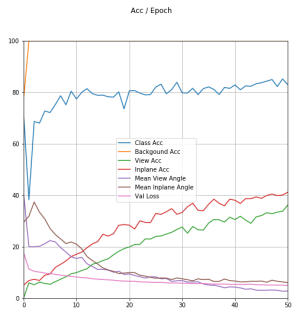
Table 3: Object-wise pose errors for the LineMOD dataset.

3. Error development for different loss term weights

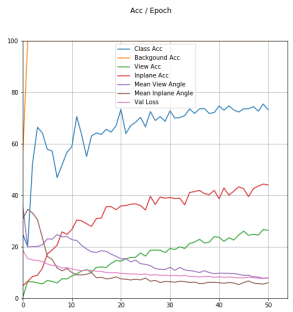
We plot the average error on a synthetic validation set. While the accuracies for class, viewpoint and in-plane rotations increase, the networks converge at different levels. We also plot the more important mean angular deviation for viewpoint and in-plane rotation since this is usually the expected error of the pooled hypotheses before refinement.



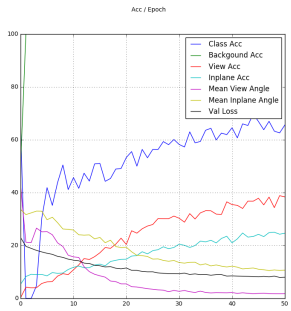
(a) $\alpha = 1, \beta = 1, \gamma = 1$



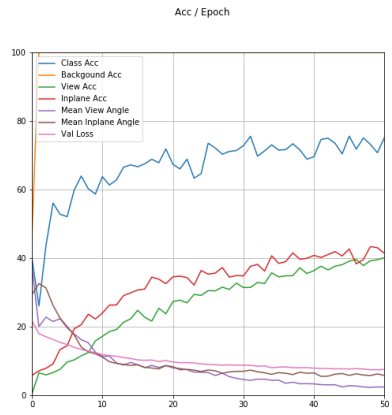
(b) $\alpha = 3, \beta = 1, \gamma = 1$



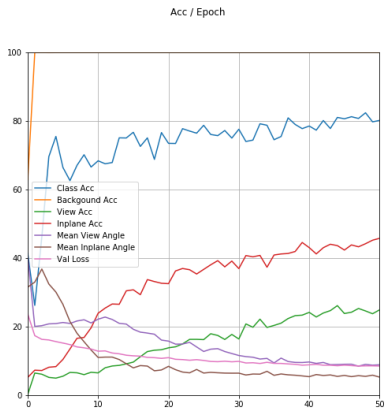
(c) $\alpha = 1, \beta = 1, \gamma = 3$



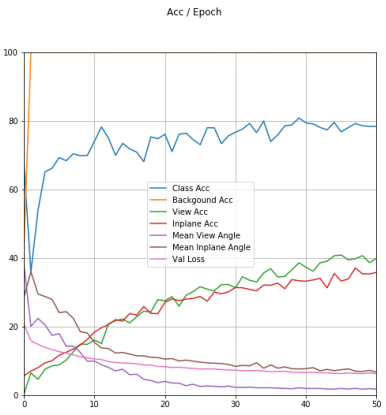
(d) $\alpha = 1, \beta = 3, \gamma = 1$



(e) $\alpha = 1, \beta = 2, \gamma = 2$



(f) $\alpha = 3, \beta = 1, \gamma = 3$



(g) $\alpha = 2, \beta = 2, \gamma = 1$

Figure 3: Development of training error on a synthetic validation set.

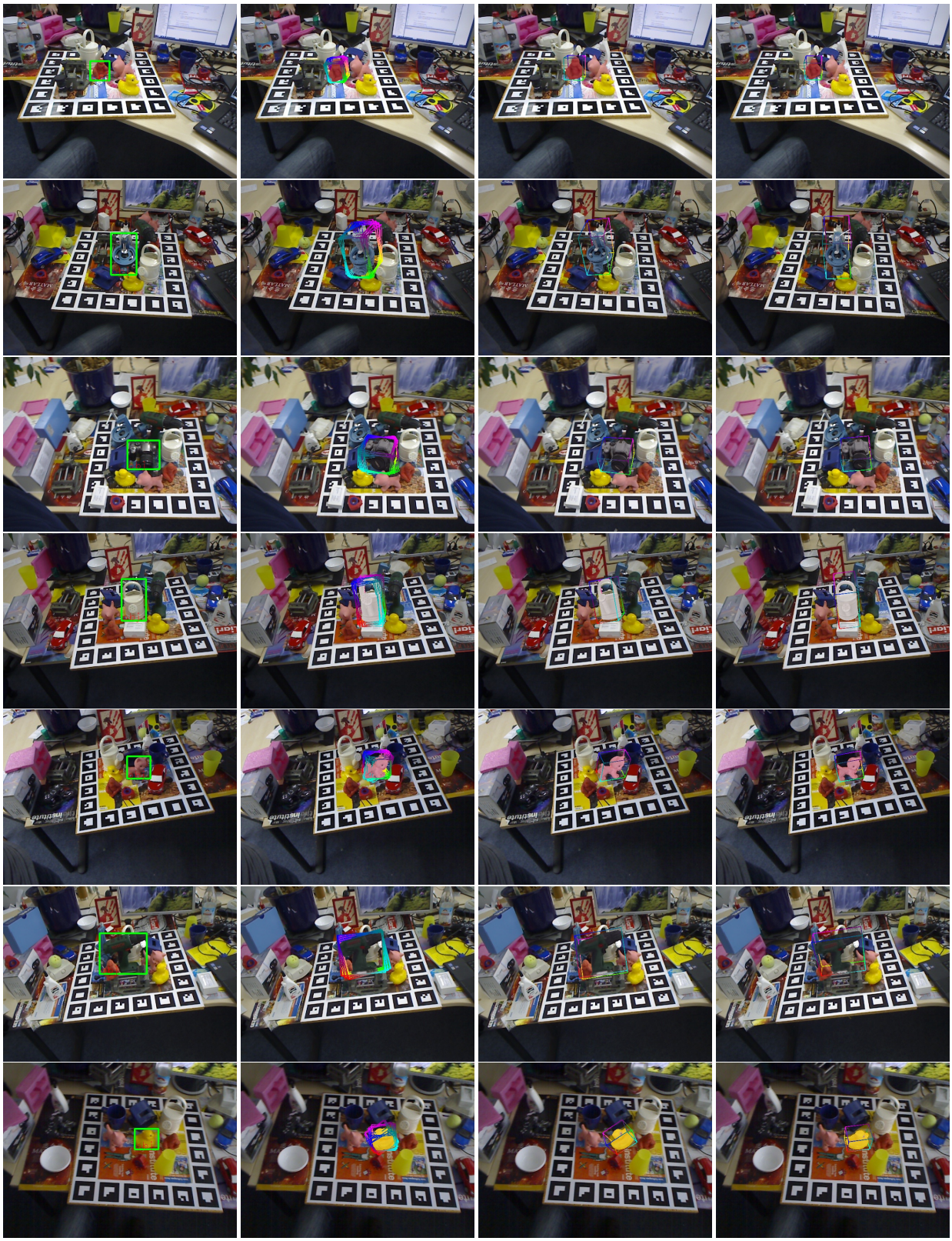


Figure 4: Qualitative results on the LineMOD dataset. From left to right: 2D prediction, hypothesis pool, result after 2D refinement, result after 3D refinement.

Figure 5: Qualitative results on the LineMOD dataset. From left to right: 2D prediction, hypothesis pool, result after 2D refinement, result after 3D refinement.

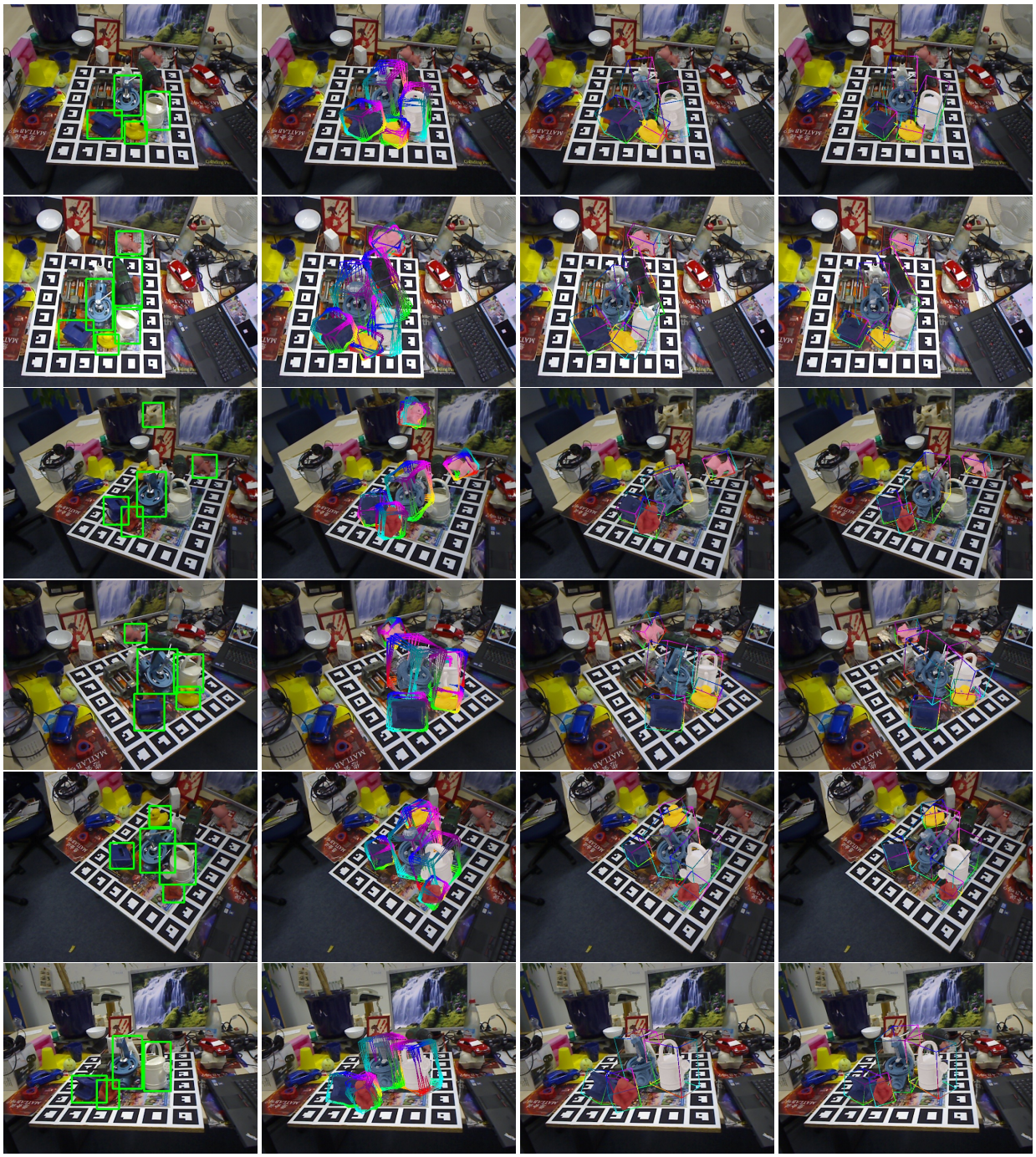


Figure 6: Qualitative results on the multi-object dataset. From left to right: 2D prediction, hypothesis pool, result after 2D refinement, result after 3D refinement.

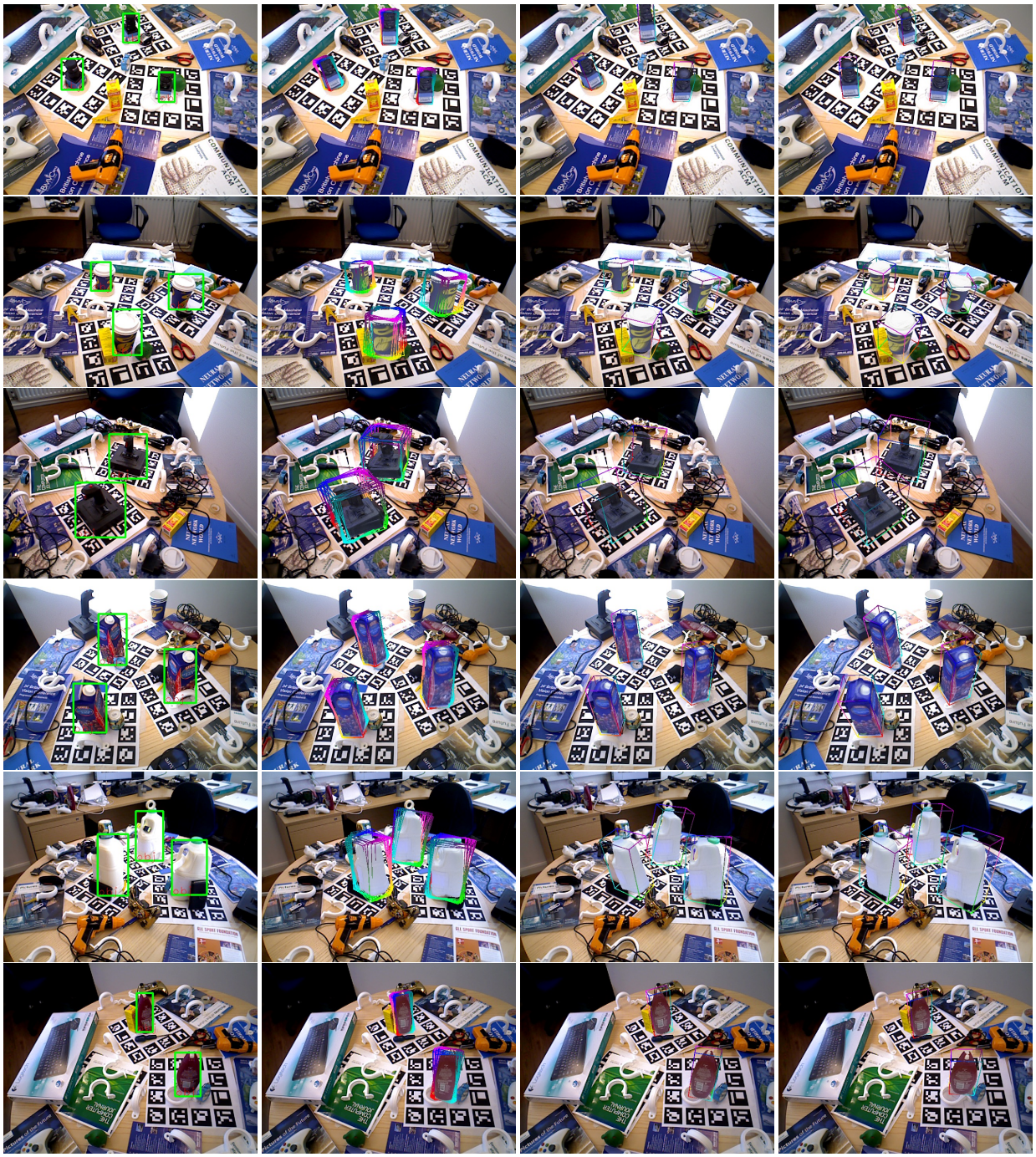


Figure 7: Qualitative results on the Tejani dataset. From left to right: 2D prediction, hypothesis pool, result after 2D refinement, result after 3D refinement.