CosyPose: Consistent multi-view multi-object 6D pose estimation

Yann Labbé[1,2], Justin Carpentier[1,2], Mathieu Aubry[3], and Josef Sivic[1,2,4]

[1] École normale supérieure, CNRS, PSL Research University, Paris, France
[2] INRIA, Paris
[3] LIGM (UMR 8049), École des Ponts, UPE, France
[4] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague
{yann.labbe,justin.carpentier,josef.sivic}@inria.fr,
mathieu.aubry@imagine.enpc.fr

Supplementary material

The supplementary material is organized as follows. In Sec. 1 we present additional qualitative results of our multi-view multi-object 6D pose estimation approach. We discuss in detail some examples to illustrate key benefits of our method as well as point out the main limitations. We then provide more examples randomly selected from the results on the T-LESS and YCB-Video datasets. In Sec. 2, we give more details of our single-view single-object 6D object pose estimator. In Sec. 3 we illustrate the object candidate matching strategy on a simple 2D example. In Sec. 4, we give additional details about our parametrization and initialization of the object-level bundle adjustment problem, introduced in Sec. 3.4 of the main paper. Finally, Sec. 5 presents the datasets used in the main paper and recalls the metrics that are used for each dataset.

1 Additional multi-view multi-object results

Each scene reconstruction is presented with a dedicated figure and we provide close-ups on various parts of the visualization to illustrate the different aspects in detail. The explanation is provided in the caption of each figure.

Layout of the figures. In each figure presented below, four (on T-LESS) or five (on YCB-Video) RGB images were used to reconstruct each scene. In each figure, each row corresponds to results associated with one image and different columns present the results of different stages of our method. The last column shows the ground truth scene. The different columns are described next.

- "Input image" is the (RGB) image used as input to the method.
- "2D detections" shows the detections obtained by the object detector (RetinaNet on T-LESS, PoseCNN on YCB-Video), after removing detections that

have scores below 0.3. The color of each 2D bounding box illustrates the object label predicted for this detection, each color is associated with a unique type of 3D object in the object database. Note that the colors for each type of 3D object are shared for all visualizations corresponding to one scene (one figure) but not shared across the figures because of the high number of objects in the database.
- "Object candidates" illustrates the 6D object poses predicted for each 2D detection. The candidates considered as outliers (those who have not been matched with a candidate from another view and are discarded) are marked with red color and are transparent. The candidates considered inliers are shown in green. Inliers are used in the final scene reconstruction. Note that the red and green colors in this (3rd) column are only used to indicate inliers and outliers and there is no correspondence with red and green colors in the 4th column that denote the different object types.
- "Scene reconstruction" illustrates the scene reconstructed by our method using all the views presented in the figure. Once the scene is reconstructed, we use the recovered 6D poses of physical objects and cameras to render the scene imaged from each of the predicted viewpoints. The renderings are overlaid over the input image.
- "Ground truth" corresponds to the ground truth scene viewed from the ground truth viewpoints. These images are shown to enable visual comparison with the results of our method. The ground truth information (number of objects, types of objects, poses of cameras, poses of objects) is not used by our method.

In the following, we illustrate the main capabilities of our system.

1.1 Highlights of the capabilities of our system

Large number of objects, robustness to occlusions, symmetric objects. Our method is able to recover the state of complex scenes that contain multiple objects, even if parts or the scene are partially or completely occluded in some of the views. The poses of cameras and objects can be correctly recovered even if all objects in the scene are symmetric. An example is presented in Fig. 1. Note how some objects are missing in each individual view but our method is able to recover correctly all objects.

Multiple object instances. Our method is able to successfully identify the correct number of objects and their labels even if there are multiple objects of the same type in the image, objects are partially occluded in some views and multiple types of objects have very similar visual appearance. An example is presented in Fig. 2

Cluttered scenes with distractors. Our method is also robust to distractor objects that are not in the database of objects. We present in Fig. 3 a complex

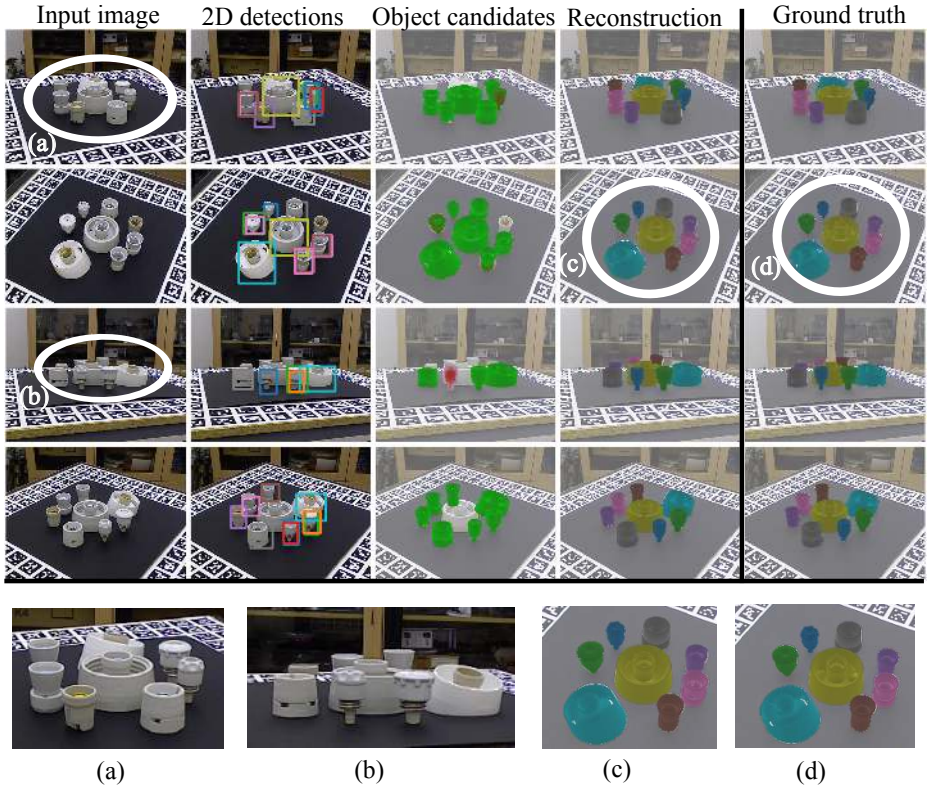

Fig. 1. Highlight I: Scene with many symmetric objects and occlusions. Our method is able to correctly identify and predict the poses of the 8 symmetric objects present in the scene. Please note how object poses and labels/colors are similar in the output of our method, shown in close-up (c), and the ground truth, shown in close-up (d). This is particularly challenging because of the high object density, varying level of occlusions and the fact that all objects of the scene are symmetric, as shown in close-ups (a) and (b).

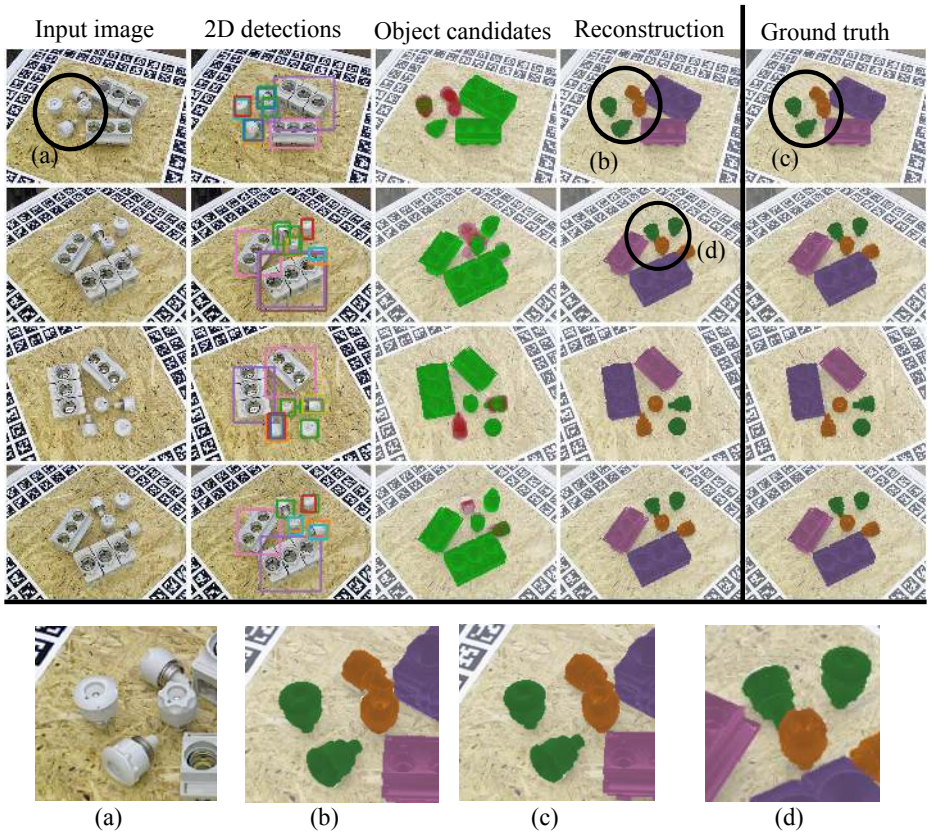

Fig. 2. Higlight II: Scene with multiple object instances of the same object type. Note how our method is able to correctly identify all objects in this challenging scene. Object poses and labels/colors predicted by our method, shown in close-up (b) are very similar to the ground truth, shown in close-up (c). This is particularly challenging because the green and orange objects have similar visual appearance, are close to each other in the scene, and objects are partially occluded in some of the views, as shown in close-ups (a) and (d).

example with many distractors where our method is able to successfully recover all objects in the scene, which are in the object database while filtering out the other ones. This is especially important for robotic applications in unstructured environments where the objects of interests are known and should not be confused with other background objects.

High accuracy. One of the key components of our approach is scene refinement (section 3.4 in the main paper), which significantly improves the accuracy of pose predictions using information from multiple views. In Fig 4, we show an example of a reconstruction that highlights the accuracy that can be reached by our method using only 4 input images.

1.2 Detailed examples

We now explain in detail few simpler examples that demonstrate how our system works and how it achieves the kind of results presented in the previous section.

Robustness to missing detections. In some situations, objects are partially or completely occluded in some of the views. As a result, 2D detections for one physical object are missing in some views. If this physical object is visible in other views, our reconstruction method is able to estimate it's pose with respect to the other objects. If all cameras can be positioned with respect to the rest of the scene using other non-occluded objects, our approach can also position the partially occluded object with respect to all cameras, even if there were initially no candidates corresponding to the object in these views. An example is shown in Fig. 5.

Robustness to incorrect detections. In T-LESS, many objects have similar visual appearance. As a result, the 2D detector often makes mistakes, predicting incorrect labels for some of the detections in some views. Our method is able to handle multiple 2D detections that have different labels at the same location in the image. In this case, a pose hypothesis is generated for each of the label hypothesis. If the object candidate cannot be matched with another view - either because the incorrect label is predicted in only one view or because the poses are not consistent - our method is able to discard this object candidate. An example is shown in Fig. 6. Please see the discussion "Duplicate objects" and Fig. 7 for examples where an object is consistently mis-identified across multiple views.

Duplicate objects. When multiple objects share the same visual appearance as it is the case in the T-LESS dataset, there are often multiple label hypotheses that are consistent across views for the same physical object. Because these objects look similar to each other and match the observed image, the pose estimation network (which tries to match a rendering with the observed image, regardless of the object type) predicts reasonable poses for each label that are

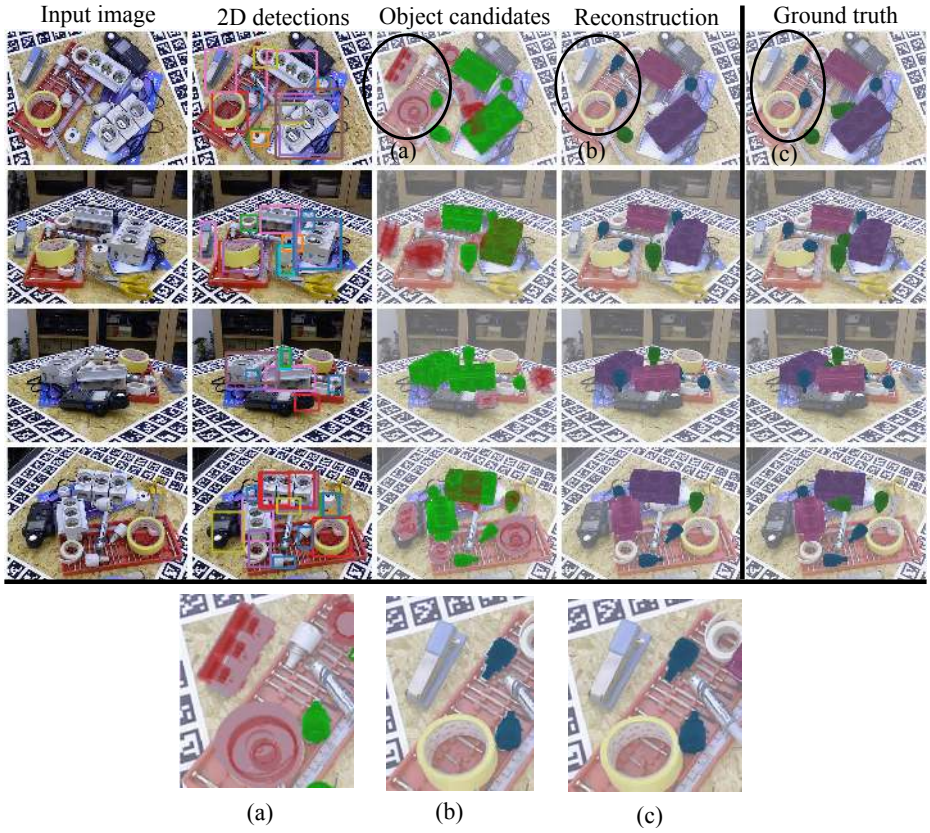

Fig. 3. Highlight III: Scene with multiple distractors. Our method is also robust to distractor objects that are not in the database of objects. Our method correctly localizes and estimates the pose of all databse objects in the scene (cf. our reconstruction (4th column) and the ground truth (5th column)) despite the presence of several distractor objects (objects not colored in the ground truth). A single-view approach (Object candidates, 3rd column) incorrectly detects three of the distractor objects and places them in the scene because they look similar to some objects of the database, as shown in the close-up (a). Our robust multi-view approach is able to filter these outliers: the objects estimated at the positions of the distractors are marked in red in (a). Distractor objects have been filtered in the final reconstruction as shown in the close-up (b) (cf. ground truth close-up (c)).

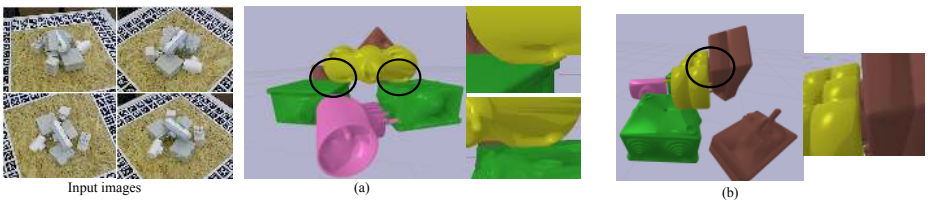

Fig. 4. Highlight IV: Accuracy of our approach. Left: input images. Then (a) and (b) shows the output scene imaged from two viewpoints different from the views used for the reconstruction. Please note in (a) how the yellow object is accurately estimated to only touch the green objects, and in (b) how the brown object is correctly plugged inside the yellow object.

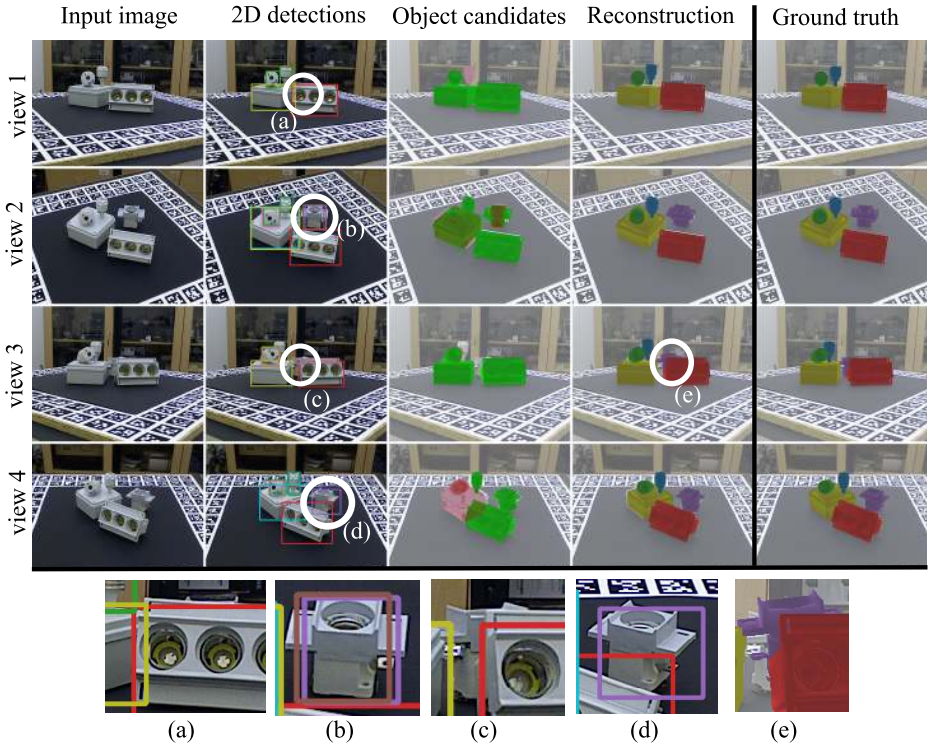

Fig. 5. Example I: robustness to missing detections. One of the objects (marked by purple circle) in the scene is detected in two views (b) (d), but not in the other two views due to partial (c) or complete (a) occlusion. Our method is able to (i) position the views 1 and 3 with respect to the scene using the other visible candidate objects and (ii) position the purple object with respect to these other objects using views 2 and 4, where the purple object is visible. Once the scene is reconstructed, it is also possible to directly recover the pose of the purple object with respect to views, where it was not originally detected, like in (e).

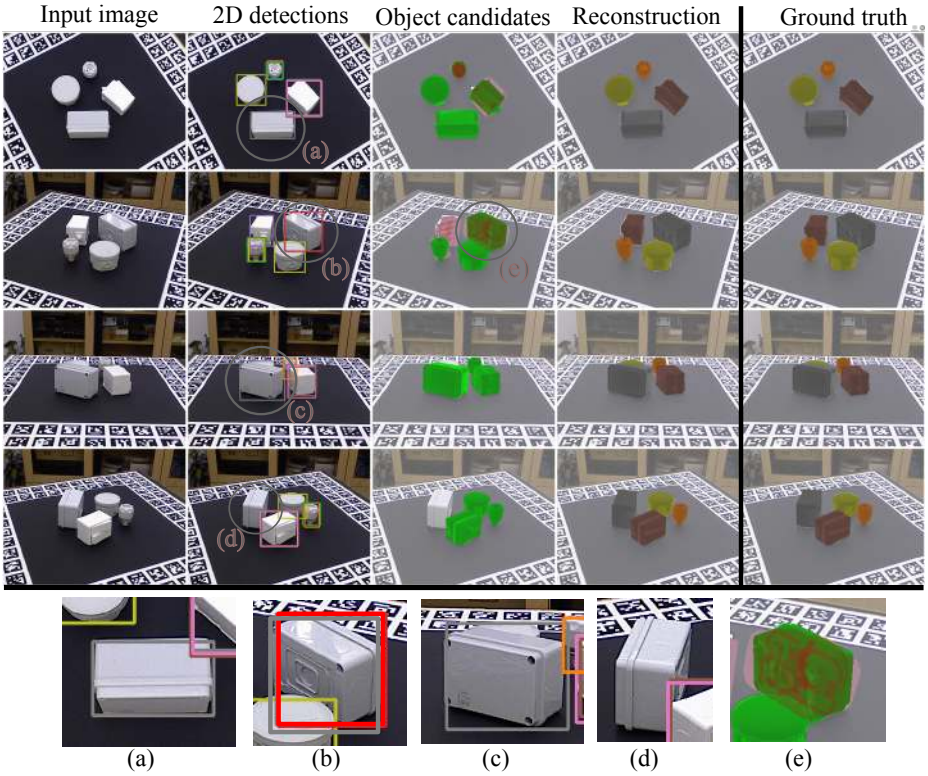

Fig. 6. Example II: Robustness to incorrect detection labels. One of the objects that is correctly identified in two views (a) (c), has two label hypotheses in view (b) and is not detected in view (d). Our method keeps the two hypotheses in (b) and predicts two 6D object candidates (e) but it is able to discard one of them because it's label is not consistent with the other views: one of the two object candidates is marked as an outlier (red) in (e). In our final scene reconstruction, the gray object is correctly recognized (it has the same color (gray) in out output "Reconstruction" and in the "Ground truth").

consistent across different views. These candidates are matched across views and multiple objects with different labels are predicted in the final scene at the same spatial position. In our visualization, we remove these duplicate objects by using a simple 3D non-maximum suppression (NMS) strategy on the estimated physical objects of the final scene. If multiple objects are too close to each other in the 3D scene, we keep the object with the highest score – the sum of the 2D detection scores of all inlier object candidates that are associated with one physical 3D object. Duplicate objects and 3D non-maximum suppression are illustrated in Fig. 7, including one correct and one incorrect example. The column "Reconstruction" in all figures corresponds to the output of our method *after* the 3D NMS.

Robustness to distractors and false positives. The complex scenes in the T-LESS dataset also have background distractor objects that are not in the object database. Some of these distractors look similar to objects in the database and can be incorrectly detected, sometimes in multiple images. In these cases, the pose estimator most often produces 6D pose estimates that are not consistent across views because the input real images are outside of the training distribution (they display objects that are not used to generate the training data). Because these estimates are not consistent across views, our method is able to filter them and mark them as outliers (red), thus gaining robustness with respect to these distractors. An example is shown in Fig. 8.

1.3 Limitations

We now describe the most challenging scenarios that our method is currently not able to recover from. For each of these, we briefly discuss possible improvements.

Limitation I: consistent mistakes If two incorrect 6D object candidates are consistent across at least two views, an (incorrect) object will be present in the reconstructed scene. Such failure case typically happens when two viewpoints are similar to each other. An example is shown in Fig. 9. If two views are very similar, the incorrect candidates will be matched together. Note that this failure mode could be resolved by using a higher number of views, and by only considering physical objects that have a sufficiently high number of associated object candidates.

Limitation II: Objects missing in the final reconstruction. Our current approach requires that a candidate in one view is matched with at least one candidate from another view. If a candidate detection and pose estimate is correct in one view but not in any other view, it will be missing from the final reconstruction. An example is presented in Fig. 10. Note that in this case, all camera poses are still estimated correctly. An interesting direction to overcome this problem would be to grow the number of object candidates in each view by reprojecting the detection from other views, as done in guided matching.

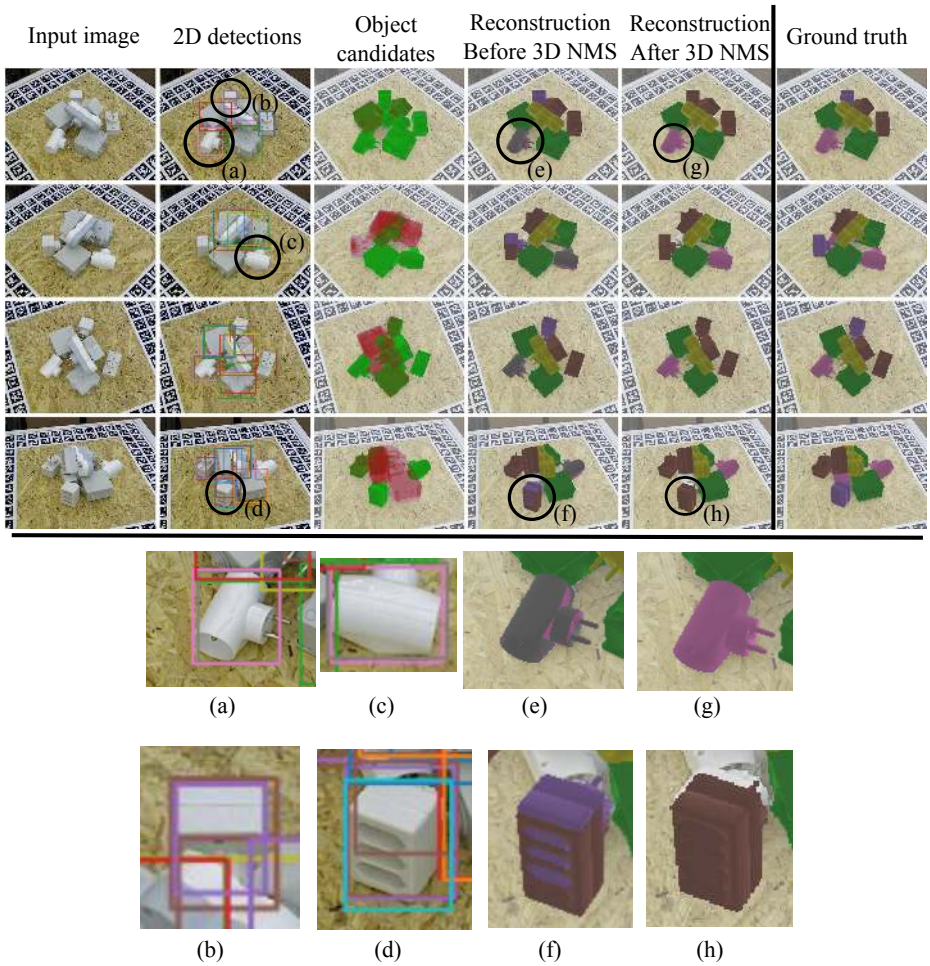

Fig. 7. Example III: Duplicate objects. 2D detections with two different labels (grey and pink) are predicted for the same object consistently across two views, (a) and (c). Because the 3D models of the pink and grey objects are similar, the poses predicted in both views are consistent and thus both pairs of object candidates are associated to separate objects. In the final scene reconstruction, two objects (grey and pink) overlap at the same 3D location (e). We use a 3D non-maximum suppression strategy to retain only a single hypothesis. In the final output (after NMS), the correct object is retained (pink), c.f. the ground truth column. In some cases, incorrectly identified objects are kept as shown in (b), (d), (f), (h).

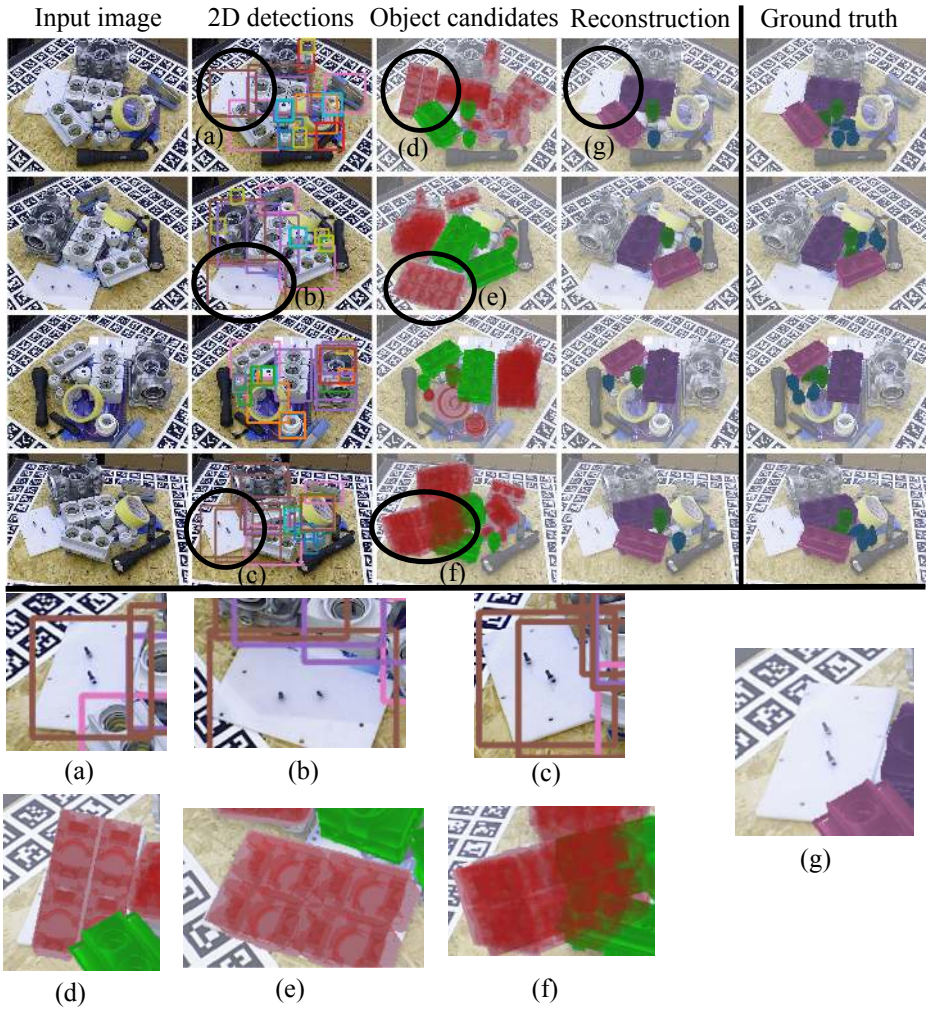

Fig. 8. Example IV: Robustness to false positive detections. One of the distractor objects is incorrectly detected in three views, see close-up (a), (b) and (c), with a consistent label (brown). For each of these detections, a 6D object candidate is generated, see close-ups (d), (e) and (f), but the poses are inconsistent across views because the pose estimation network has not been trained for this object. These candidates are filtered by our robust candidate matching strategy and considered outliers (red), see (d), (e) and (f). Note how this distractor is not present in the final scene reconstruction, as shown in close-up (g).

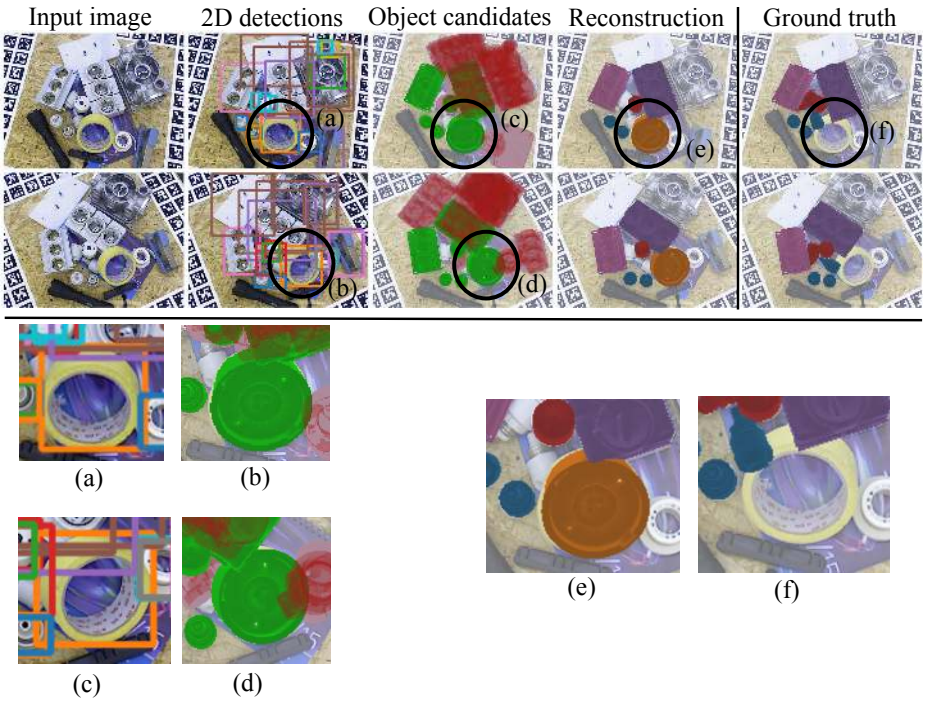

Fig. 9. Limitation I: Consistent mistakes. One of the distractors is incorrectly detected as an orange object (from the object database), as shown in close-ups (a) and (c). The two viewpoints are quite similar and as a result the two estimated object poses are consistent, as shown in (c) and (d). The object is present in the final reconstruction (e) but it does not correspond to the ground truth object (f).

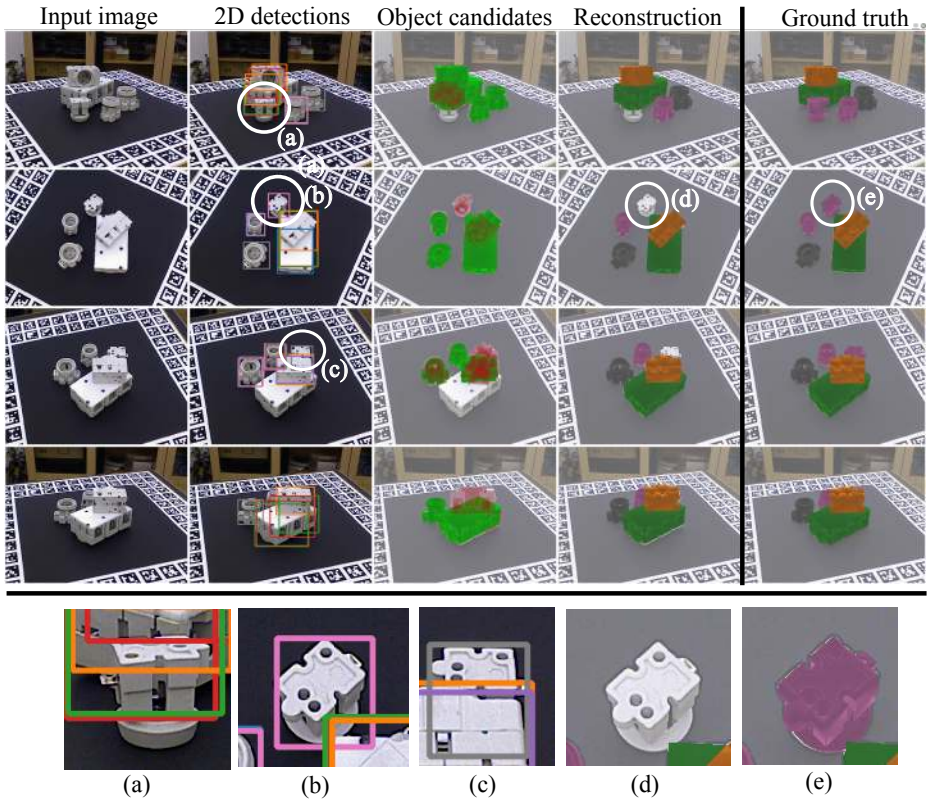

Fig. 10. Limitation II: missing objects. An object is detected correctly in one view as shown in the close-up (b), but the detection is missing in other views, shown in close-up (a), or the detection is incorrect and inconsistent, as shown in close-up (c). The object candidate (b) cannot be matched with another candidate and thus is missing from the final reconstruction, as shown in close-up (d) of the output (cf. ground truth close-up (e)).

Limitation III: Incorrect estimates of camera pose. To position the camera with respect to the scene, our method requires that there are at least three object candidate inliers in the view: two for positioning the camera with respect to the scene, and another one to validate the camera pose hypothesis. Sometimes, however, there is insufficient number of inliers. This typically happens if only two objects are visible, or if there is a small number of objects visible and some of the detections are incorrect. An example is shown in Fig. 11.

1.4 Random examples

At the end of the PDF, we provide four examples (sets of 4 or 5 images and associated output reconstructions) randomly selected for each scene of the T-LESS and YCB-Video datasets.

On YCB-Video, we adopt a slighlty different visualization because the objects are textured. For clarity and because objects are mostly visually distinct, we remove object colors to distinguish between different object types. To distinguish inlier and outlier object candidates, we mark the color of the initial 2D detection with green (inlier) or red (outlier). Again, note that the red and green colors from this column are only used to indicate inliers and outliers and there are no correspondences with other red and green colors depicting the labels of objects.

2 Single-view single-object baseline

We now detail our single-view single-object pose estimation network introduced in Sec. 4.1 of the main paper. Our method builds on DeepIM [1] but includes several extensions and improvements.

Given a single image I_a and a 2D detection $D_{a,\alpha}$ associated with an object label $l_{a,\alpha}$, our method outputs an hypothesis for the pose of the object with respect to the camera. This pose is noted $T_{C_a,O_{a\alpha}}$. In this section, we focus on one view and one object and thus omit the a and α subscripts.

Similar to DeepIM [1], we use a deep neural network that takes as input two images and iteratively refines the pose. The first image is the (real) input image I cropped on a region of the image showing the object, denoted I^c. At iteration k, the second image is a (synthetic) rendering of the object with label l rendered in a pose $T^{k-1}_{C,O}$ that corresponds to the object pose estimated at the previous iteration. The network outputs an updated refined pose $T^k_{C,O}$. The initial pose $T^0_{C,O}$ can be provided by any coarse 6D pose estimation method (such as PoseCNN [2]) but we also show that we can simply use a canonical pose of the object for $T^0_{C,O}$ as explained in the "Coarse estimation" pagraph below. We now detail our method and present the main differences with [1].

Network architecture. The network takes as input the concatenation of the synthetic and real cropped images. Both images are resized to the input resolution: 320×240. The backbone is EfficientNet-B3 [3] followed by spatial average

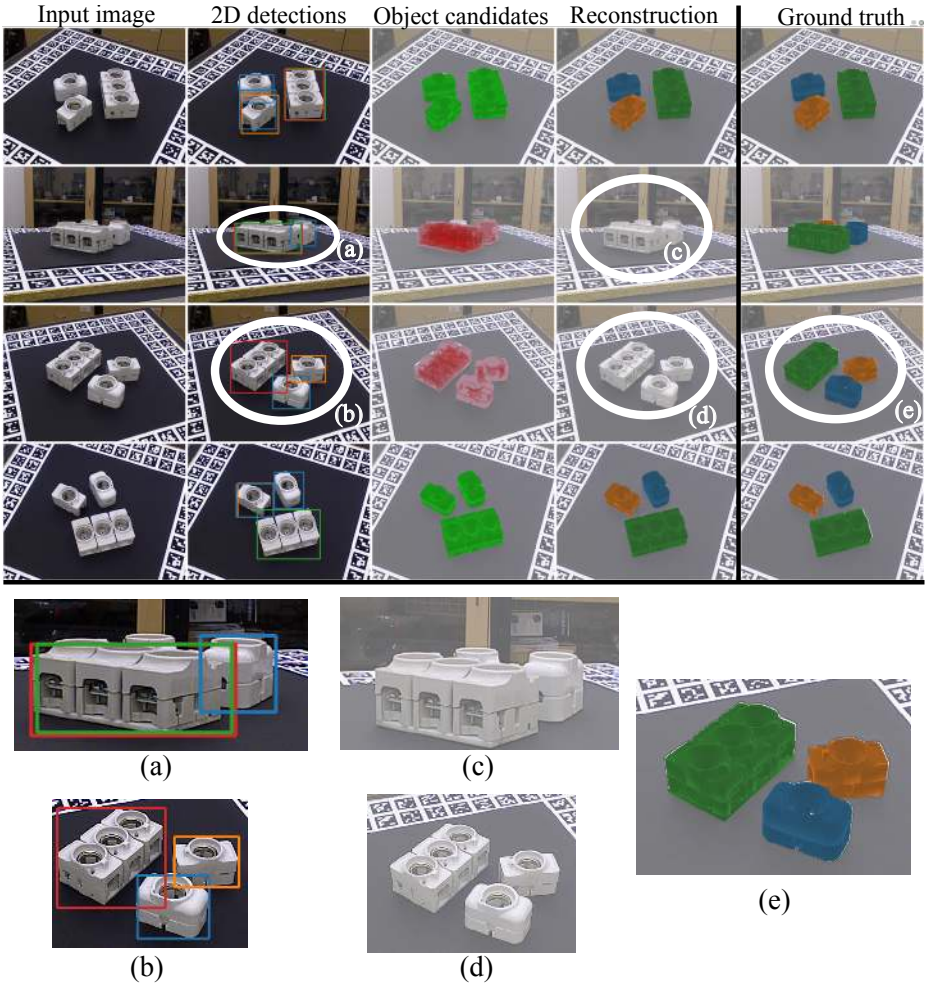

Fig. 11. Limitation III: incorrect estimates of camera pose. If one view has only two visible objects, as shown in close-up (a), the corresponding camera view with respect to the rest of the scene cannot be estimated as it requires at least three correctly estimated objects. As a result the objects are not reprojected in the image (c). This also happens if three candidates are detected in one view, as shown in close-up (b), but one of the object candidates is not consistent with the other views (here red object instead of green object).

pooling. The prediction layer is a simple fully connected layer which outputs 9 values corresponding to one vector $[v_x, v_y, v_z]$ for the translation and two vectors e_1, e_2 to predict the rotation component of T_{CO}. A rotation matrix R is recovered from e_1, e_2 using [3] by simply orthogonalizing the basis defined by the two vectors e_1, e_2. Please see "Rotation parametrization" for the equations to recover the rotation matrix R from e_1, e_2. Compared to DeepIM [1], the main difference is that we use a more recent network architecture (DeepIM is based on FlowNet [4]) and we do not include auxiliary predictions of flow and mask. This makes the method simpler and easier to train. Our input resolution of 320×240 is also smaller than 640×480 used by DeepIM, reducing memory consumption and allowing to use larger batches while training.

Transformation parametrization. Similar to DeepIM, we use the object-independent rotation and translation parametrization which consists in predicting a rotation of the camera around the object, a xy translation $[v_x, v_y]$ in image space (in pixels) for the center of the rendered object and a relative displacement v_z along the depth axis of the camera. Given the input pose T^k_{CO} and the outputs of the network ($[v_x, v_y, v_z]$ and $R = f(e_1, e_2)$), the pose update is obtained from the following equations:

$$x^{k+1} = \left(\frac{v_x}{f^C_x} + \frac{x^k}{z^k} \right) z^{k+1} \tag{1}$$

$$y^{k+1} = \left(\frac{v_y}{f^C_y} + \frac{y^k}{z^k} \right) z^{k+1} \tag{2}$$

$$z^{k+1} = v_z z^k \tag{3}$$

$$R^{k+1} = R R^k, \tag{4}$$

where $[x^k, y^k, z^k]$ is the 3D translation vector of T^k_{CO}, R^k the rotation matrix of T^k_{CO}, f^C_x and f^C_y are the focal lengths that correspond to the (fictive) camera associated with the **cropped** input image I^C. Finally, $[x^{k+1}, y^{k+1}, z^{k+1}]$ and R^{k+1} are the parameters of the output pose estimate T^{k+1}_{CO}. The differences with DeepIM are twofold. First, we use a linear parametrization of the relative depth (eq. (3)), instead of $z^{k+1} = z^k e^{-v_z}$, which we found more stable to train. Second, we use the intrinsics f^C_x, f^C_y of the cropped camera associated with the input (cropped) image. DeepIM uses the intrinsics parameters of the non-cropped camera f_x, f_y and fix them to 1 during training because the intrinsic parameters of the input camera are fixed on their datasets. We use the cropped focal lengths instead because (a) cropping and resizing the crop of the input image changes the apparent focal length and (b) the focal lengths of the input images are not fixed on T-LESS. Using the cropped focal lengths forces the network to only predict xy translations in pixels and the network can therefore become invariant to the intrinsic parameters of the input (cropped) camera.

Rotation parametrization. Given two vectors e_1 and e_2 (6 values) predicted by the neural network, we recover a rotation parametrization R by following [3]:

$$e'_1 = \frac{e_1}{||e_1||_2} \tag{5}$$

$$e'_3 = \frac{e'_1 \wedge e_2}{||e_2||_2} \tag{6}$$

$$e'_2 = e'_3 \wedge e'_1, \tag{7}$$

where $\wedge$ is the cross product between two 3D vectors. This representation has been shown to be better than quaternions (used by DeepIM) to regress with a neural network [3].

Cropping strategy. DeepIM uses (a) the input 2D detections and (b) the bounding box defined by T^k_{CO} and the vertices of the object l to define the size and location of the crop in the real input image during training. Indeed, the ground truth bounding box is known during training. At test time, only (b) is used by DeepIM because ground truth bounding boxes are not available. In our case, we only use (b) while training and testing. The intrinsic parameters of the cropped camera are also used to directly render the cropped synthetic image at a resolution of 320×240 instead of rendering at a larger resolution followed by cropping.

Symmetric disentangled loss. A standard loss for 6D pose estimation is ADD-S[2] which allows to predict pose of symmetric objects. Our loss is inspired by ADD-S loss with two main differences. First, we enumerate all the possible symmetries to find the best matching between the vertices of the predicted model and the ground truth model instead of finding the nearest neighbors. This is similar in spirit to the approach of [5] to handle object symmetries. Second, we disentangle depth v_z and translation predictions v_x, v_y, following the recommendations from [6].

More formally, we define the update function F which takes as input the initial estimate of the pose T^k_{CO}, the outputs of the neural network $[v_x, v_y, v_z]$ and R, and outputs the updated pose, i.e. the function such that

$$T^{k+1}_{CO} = F(T^k_{CO}, [v_x, v_y, v_z], R), \tag{8}$$

where the closed form of F is expressed in equations (1)(2)(3)(4) of the supplementary material. We also write $[\hat{v}_x, \hat{v}_y, \hat{v}_z]$ and $\hat{R}$ the target predictions, i.e. the predictions such that
$\hat{T}_{CO} = F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, \hat{v}_z], \hat{R})$, where $\hat{T}_{CO}$ is the ground truth pose of the object. Our loss function is then:

$$\mathcal{L}(T^k_{CO}, [v_x, v_y, v_z], R) = D_l(F(T^k_{CO}, [v_x, v_y, \hat{v}_z], \hat{R}), \hat{T}_{CO}) \tag{9}$$

$$+ D_l(F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, v_z], \hat{R}), \hat{T}_{CO}) \tag{10}$$

$$+ D_l(F(T^k_{CO}, [\hat{v}_x, \hat{v}_y, \hat{v}_z], R), \hat{T}_{CO}), \tag{11}$$

where D_l is the symmetric distance defined in the Sec. 3.2 of the main paper, with the L_2 norm replaced by the L_1 norm. The different terms of this loss separate the influence of: xy translation (9), relative depth (10) and rotation (11). We refer to [6] for additional explanations of the loss disentanglement.

Coarse estimation. To perform coarse estimation on T-LESS, we use the same network architecture, parametrization and losses defined above. As input T^0_{CO} we provide a canonical input pose that corresponds to the object being rendered at a distance of 1 meter of the camera in the center of the input 2D bounding box. The coarse and refinement networks use the same architecture, but the weights are distinct. Each network is trained independently.

Training data. Due to the complexity of annotating real data with 6D pose at large scale, most recent methods [7, 1, 8] generate additionnal synthetic training data. In our experiments, we use the real training images provided by YCB-Video and the images of the real objects displayed individually on black backgrounds provided by T-LESS. In addition, we generate one million synthetic training images on each dataset using a simple procedure described next.

We randomly sample 3 to 9 objects from the set of 3D models considered, place them randomly in a 3D box of size 50 cm and sample randomly the orientation of each object. Half of the images are generated with objects flying in the air, the other half is generated by taking the images after running physics simulation for a few seconds, generating physically feasible object configurations. This is similar to the approach described in [9, 10], though none of our rendered images are photorealistic. The camera is pointed at the center of the 3D box, its position is sampled uniformly above the box center at the same range of distance as the one of the real training data, and its roll angle is sampled between (-10, 10) degrees. On T-LESS, the distance to the object is fixed in the real reconstructed training images and we use instead the range of distances of the testing set provided (which is explicitly allowed by the guidelines of the BOP challenge [11][5]). We do not use any information from the testing set beside this distance interval.

On the T-LESS dataset, we generate data using the CAD models only. We add random textures on the CAD models following work on domain randomization [12–14]. We also paste images from the Pascal VOC dataset in the background with a probability 0.3, following [1]. On the T-LESS dataset, we add data augmentation to the input RGB images while training, following [8]. Data augmentation includes gaussian blur, contrast, brightness, color and sharpness filters from the Pillow library [15].

Examples of training images are shown in Fig. 12. Finally, when training the refinement network, we use the same distribution as DeepIM for the input poses.

[5] See https://bop.felk.cvut.cz/challenges/ Sec 2.2.

Fig. 12. Training images for our single-view single-object pose estimation networks. Examples of training images used for training the networks on T-LESS and YCB-Video.

Training procedure. All of the networks (refinement network on YCB-Video, coarse network on T-LESS, refinement network on T-LESS) are trained using the same procedure. We use the Adam optimizer [16] with a learning rate of 3.10^{-4} and default momentum parameters. Networks are trained using Pytorch and synchronous distributed training on 32 gpus, with 32 images per GPU for a total batch size of 1024. The networks are randomly initialized and we use the following training procedure. First, the network is trained for 80k iterations on synthetic data only. Then, the network is trained for another 80k iterations on both real and synthetic training images. In this second phase, the real training images account for around 25% of each batch. Following [17], we also use a warm-up phase where we progressively increase the learning rate from 0 to 3.10^{-4} during the first 5k iterations.

Experimental findings. On YCB-Video, we found that pre-training the model on synthetic data yields an improvement of approximately 2 points on the AUC of ADD(-S) metric. Without this pre-training phase, our model performed comparably to the results reported by DeepIM. Note that this is hard to directly compare because the synthetic training images are different from the ones used by DeepIM.

On T-LESS, we found that the data augmentation is crucial as also pointed out by [8]. Without data augmentation, the performance of the coarse and refinement networks is poor, with a $e_{vsd} < 0.3$ score of around 35% compared to 64% when training with data augmentation.

3 Object candidate matching: additional illustration

In Fig. 13, we illustrate our method for "Sampling of relative camera poses sampling" described in Sec 3.3 of the main paper with a simple 2D example.

4 Scene refinement

Initialization. There are multiple ways to initialize the optimization problem defined in equation (6) of the main paper. We use the following procedure. We start by picking a random camera and setting it's coordinate frame as the world coordinate frame. Then, we iterate over all cameras, trying to initialize each one. In order to initiliaze a camera a, we randomly sample another camera b which is already initialized (placed in the world coordinate frame) and use the relative pose between these two cameras $T_{C_a C_b}$ estimated while running RANSAC (relative camera pose sampling in Sec. 3.2) to place camera a in the world coordinate frame. Once all the cameras have been initialized, we initalize objects by randomly picking an object p an initializing it using a candidate associated with this physical object from a random view.

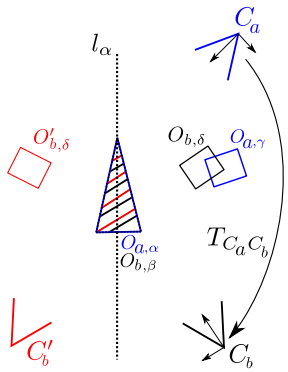

Fig. 13. Relative camera pose estimation. Given two pairs of object candidates $(O_{a,\alpha}, O_{b,\beta})$ and $(O_{a,\gamma}, O_{b,\delta})$, we estimate the relative camera pose $T_{C_a C_b}$ that best aligns candidates $O_{a,\gamma}$, $O_{b,\delta}$. In this example, the red camera pose C'_b is also valid due to the symmetries of the triangular object l_α. It is discarded because the error between $O'_{b,\delta}$ and $O_{a,\gamma}$ is bigger than between $O_{b,\delta}$ and $O_{a,\gamma}$.

Rotation parametrization. We use the same rotation parametrization as the one used for our single-view single-object network for which the equations are provided in Sec. 2 of this supplementary material.

5 Datasets and metrics

5.1 Datasets

In this section, we give details of the datasets used in our experiments.

YCB-Video. The YCB-Video [2] dataset is made of 92 scenes with around 1000 images per scene. The dataset is split into 80 scenes for training and 12 scenes for testing. It is mostly challenging due to the variations in lightning conditions, significant image noise and occlusions. The objects are picked from a subset of 21 objects from the YCB object set [18] for which reconstructed 3D models are available. The models are presented in Fig. 15. These models are used to generate additional synthetic training images.

There is at most one object of each instance per scene and most of the objects are visually distinct with the exception of the large and extra-large clamps. When testing, we follow previous works [1, 2, 19] and evaluate on a subset of 2949 keyframes. The variety of the viewpoints for each scene is limited as the camera is usually moved in front of the scene, but not completely around it.

T-LESS. The T-LESS [20] dataset is made of 20 scenes featuring multiple industry-relevant objects. There are 30 object instances, all of them are texture-less and most of them are symmetric. The reconstructed 3D models of these

Fig. 14. Objects of the YCB-Video dataset. The 21 reconstructed object models of the YCB-Video dataset. Taken from [2].

objects are presented in Fig. 15. Many objects have similar visual appearance, making the class prediction task challenging for the object detector. The images in the dataset are taken all around the scene. Scene complexity varies from 3 objects of different types to up 18 objects with 7 belonging to the same type. In single-view experiments we consider all images of the testing scenes to provide meaningful comparison with [8, 21]. For multiview experiments we consider the subset of the BOP19 challenge [11]. We use the CAD models for generating synthetic images and for evaluation.

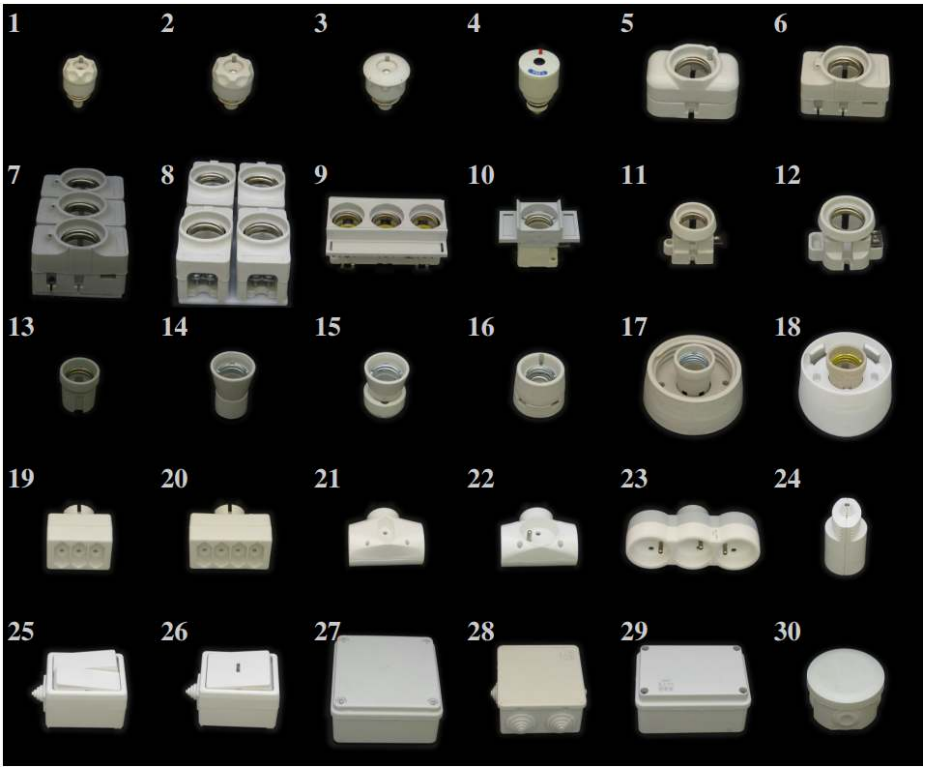

Fig. 15. Objects of the T-LESS dataset. The 30 reconstructed object models of the T-LESS dataset. Notice how multiple objects share visual appearances such as (1) (2); (5) (6); (14) (15) (16); (25) (26). Taken from [20].

5.2 Metrics

In this section, we give some details about the metrics reported in the main paper. We refer to [11, 22] for more information about these metrics.

The ADD (average distance) metric is introduced in [22] and is typically used to measure the accuracy of pose estimation for non-symmetric objects. Given a

label l of an object and following the notation introduced in Sec. 3.1 of the main paper, this metric is computed as :

$$\text{ADD}(l, T, \hat{T}) = \frac{1}{H_l} \sum_h ||\hat{T}X_l^h - TX_l^h||_2, \tag{12}$$

where T is the predicted object pose, $\hat{T}$ is the ground truth pose, X_l^h are the vertices of the 3D models and H_l is the number of vertices of the model of the object l.

For symmetric objects, the average distance is computed using the closest point distance and noted ADD-S:

$$\text{ADD-S}(l, T, \hat{T}) = \frac{1}{H_l} \sum_h \min_g ||\hat{T}X_l^h - TX_l^g||_2. \tag{13}$$

The notation ADD(-S) corresponds to computing ADD for non symmetric objects and ADD-S for symmetric objects. It is also common to report the percentage of objects for which the pose is estimated within a given threshold such as 10% of it's diameter. We use the notations ADD-S $<$ 0.1d and ADD(-S) $<$ 0.1d for this metric.

The authors of PoseCNN [2] also proposed to report the area under the accurracy-threshold curve for a threshold (on ADD-S, or ADD(-S)) varying between 0 to 10cm. We note this metric as AUC of ADD(-S) or AUC of ADD-S and we use the implementation provided with the evaluation code[6] of YCB-Video.

When evaluating on the T-LESS dataset, we also report the Visual Surface Discrepancy metric (vsd). This metric is invariant to object symmetries and takes into account the visibility of the object. As in [8, 21], the pose is considered correct when the error is less than 0.3 with $\tau = 20mm$ and $\delta = 15mm$. We note this metric $e_{\text{vsd}} < 0.3$ and use the official implementation code of the BOP challenge [11][7]. There are multiple instances of objects in multiple scenes of the T-LESS dataset. When comparing with prior work [8, 21] on all images of the primesense camera, we only evaluate the prediction which has the highest detection score for each class, and only objects visible more than 10% are considered as ground truth targets. This corresponds to the SiSo task.

When evaluating our multi-view method, we follow the more recent 6D localization protocol of the ViVo BOP challenge which considers the top-k predictions with highest score for each class in each image, where k is the number of ground truth objects of the class in the scene. Note that the metrics of the BOP challenge do not penalize making many incorrect predictions for classes that are not in the scene, which happens in most methods and is problematic for practical application. We thus propose to analyze precision-recall tradeoff similar to the standard practice in object detection, using ADD-S$<$0.1d to count true positives.

When computing the mean of ADD-S errors in our scene refinement ablation, we only consider as true positives predictions the ones which have an ADD-S

[6] https://github.com/yuxng/YCB_Video_toolbox
[7] https://github.com/thodan/bop_toolkit

error lower than half of the diameter of the object, to ensure that the prediction is matched to the correct ground truth object. Without limiting the error to this threshold and using only class labels and scores, some predictions may be matched to ground truth objects which are at a very different location in the scene. This tends to increase the errors while not being representative only of the 6D pose accuracy of the predictions.

References

1. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 683–698
2. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In: Robotics: Science and Systems XIV. (2018)
3. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 5745–5753
4. Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE international conference on computer vision. (2015) 2758–2766
5. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 2642–2651
6. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 1991–1999
7. Zakharov, S., Shugurov, I., Ilic, S.: Dpod: 6d pose object detector and refiner. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 1941–1950
8. Sundermeyer, M., Marton, Z.C., Durner, M., Brucker, M., Triebel, R.: Implicit 3d orientation learning for 6d object detection from rgb images. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 699–715
9. Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., Birchfield, S.: Deep object pose estimation for semantic robotic grasping of household objects. In: Conference on Robot Learning (CoRL). (2018)
10. Tremblay, J., To, T., Birchfield, S.: Falling things: A synthetic dataset for 3d object detection and pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. (2018) 2038–2041
11. Hodan, T., Michel, F., Brachmann, E., Kehl, W., GlentBuch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., et al.: Bop: Benchmark for 6d object pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 19–34
12. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2017) 23–30

13. Loing, V., Marlet, R., Aubry, M.: Virtual training for a real application: Accurate Object-Robot relative localization without calibration. Int. J. Comput. Vis. **126**(9) (September 2018) 1045–1060
14. Labbé, Y., Zagoruyko, S., Kalevatykh, I., Laptev, I., Carpentier, J., Aubry, M., Sivic, J.: Monte-carlo tree search for efficient visually guided rearrangement planning. IEEE Robotics and Automation Letters **5**(2) (2020) 3715–3722
15. Clark, A., et al.: The pillow imaging library. https://github.com/python-pillow/pillow. IEEE Robot. Autom. Mag. **22**(3) (September 2015) 36–52
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. (December 2014)
17. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: Training ImageNet in 1 hour. (June 2017)
18. Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., Dollar, A.M.: Benchmarking in manipulation research: Using the Yale-CMU-Berkeley object and model set. IEEE Robot. Autom. Mag. **22**(3) (September 2015) 36–52
19. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: Pixel-wise voting network for 6dof pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 4561–4570
20. Hodan, T., Haluza, P., Obdržálek, Š., Matas, J., Lourakis, M., Zabulis, X.: T-LESS: An RGB-D dataset for 6D pose estimation of Texture-Less objects. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV). (March 2017) 880–888
21. Park, K., Patten, T., Vincze, M.: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 7668–7677
22. Hinterstoisser, S., Lepetit, V., Ilic, S., Holzer, S., Bradski, G., Konolige, K., Navab, N.: Model based training, detection and pose estimation of Texture-Less 3D objects in heavily cluttered scenes. In: Computer Vision – ACCV 2012, Springer Berlin Heidelberg (2013) 548–562

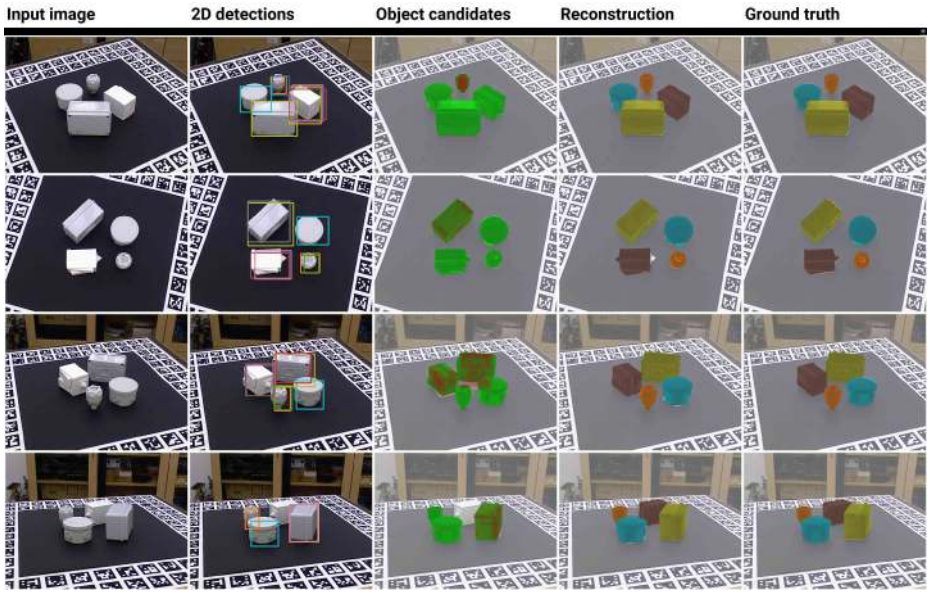
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 16.

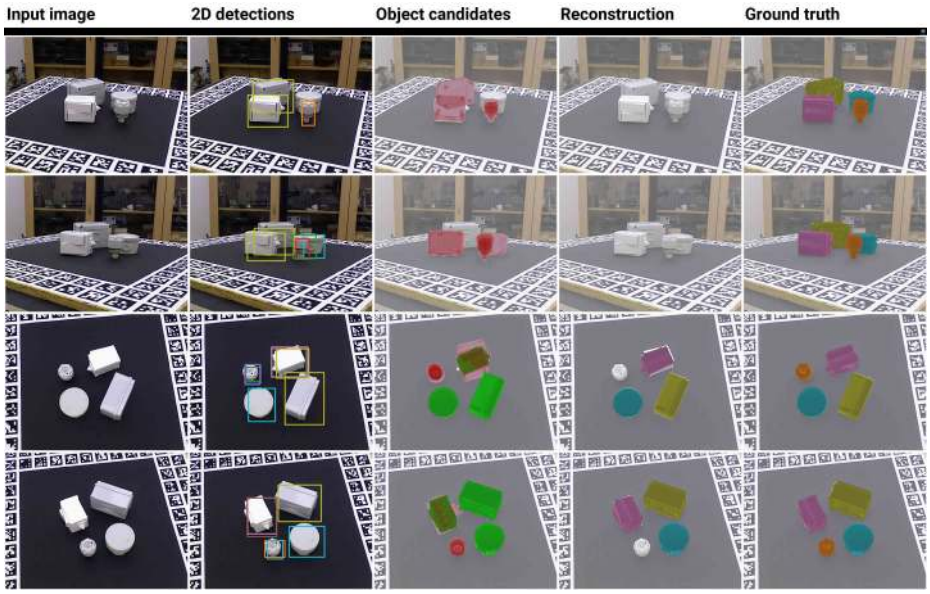
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 17.

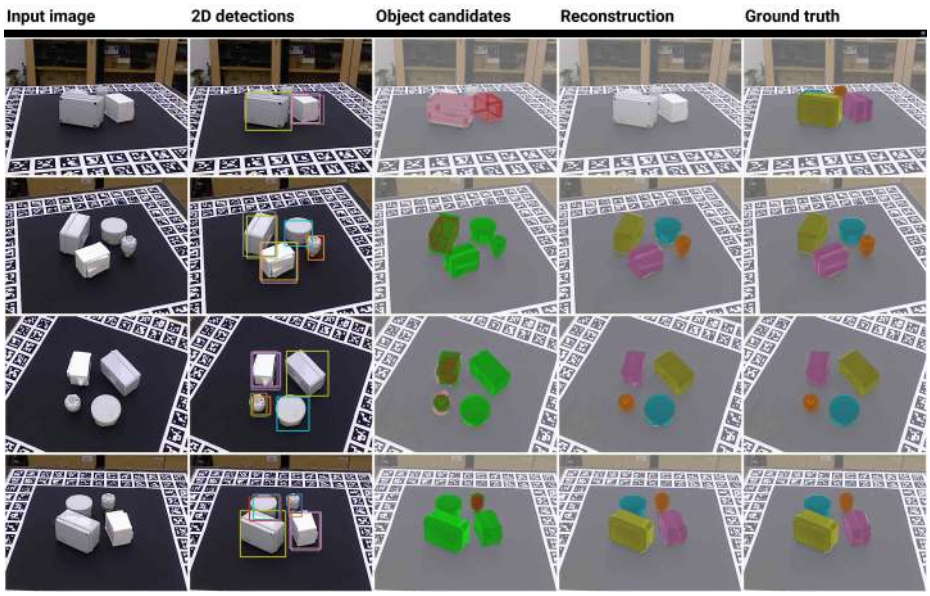
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 18.

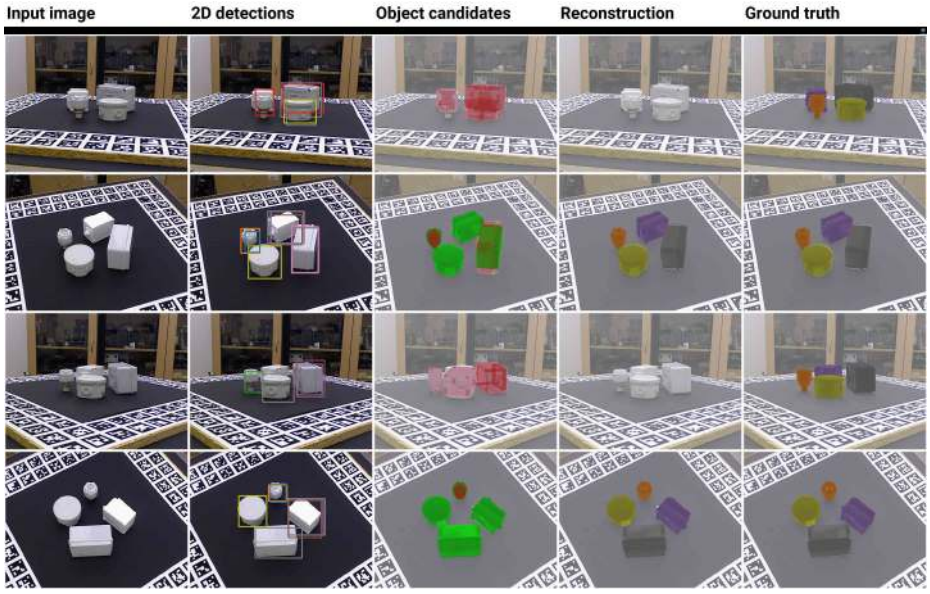
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 19.

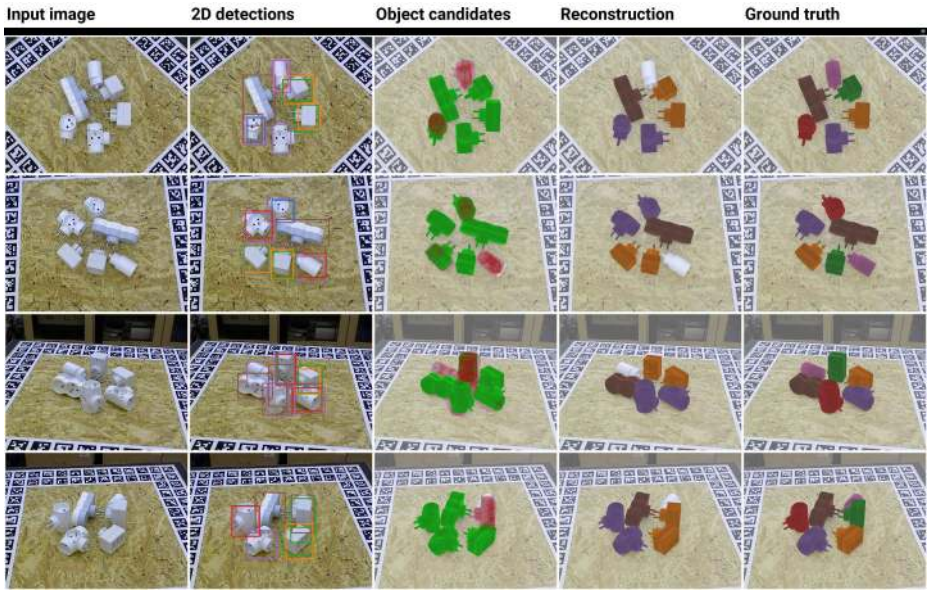
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 20.

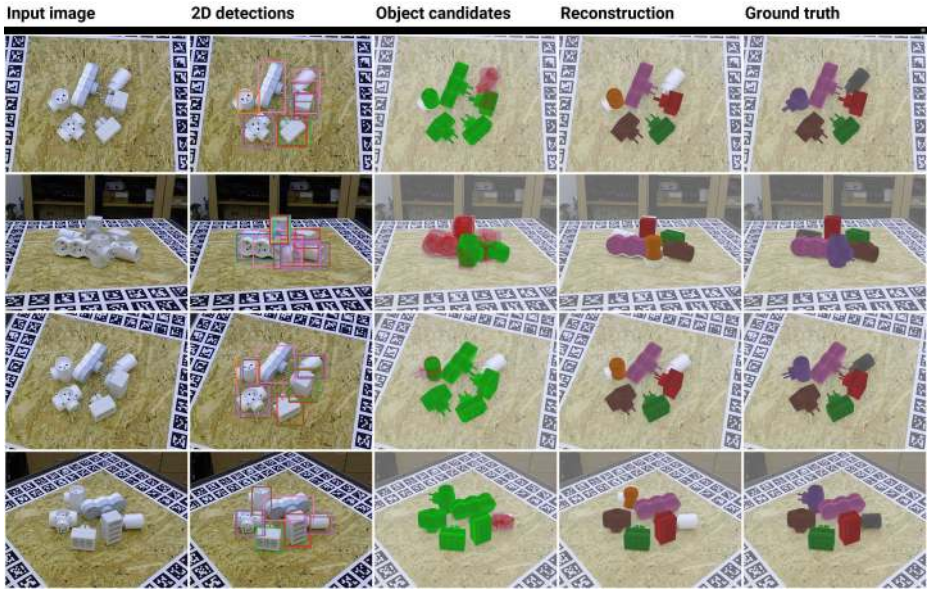
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 21.

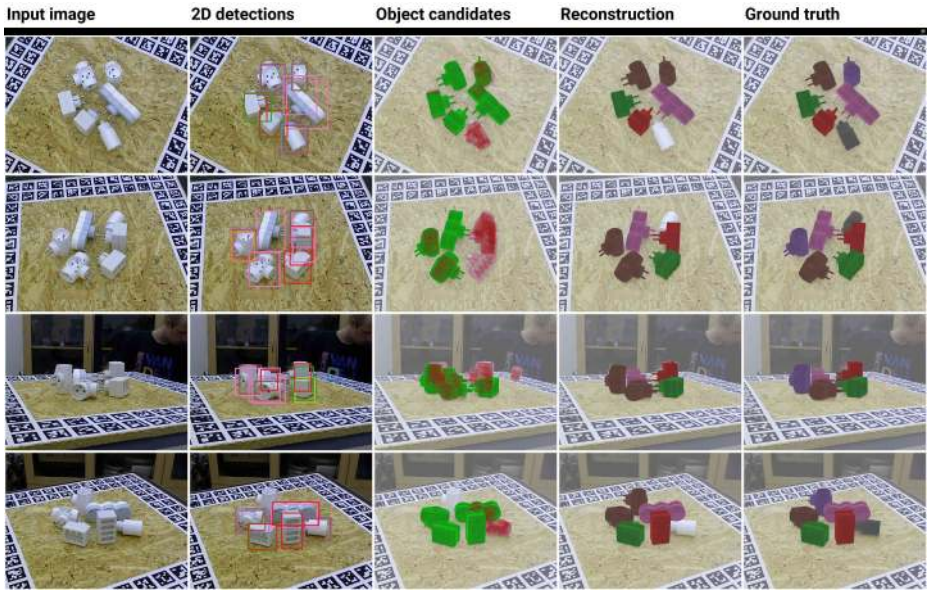
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 22.

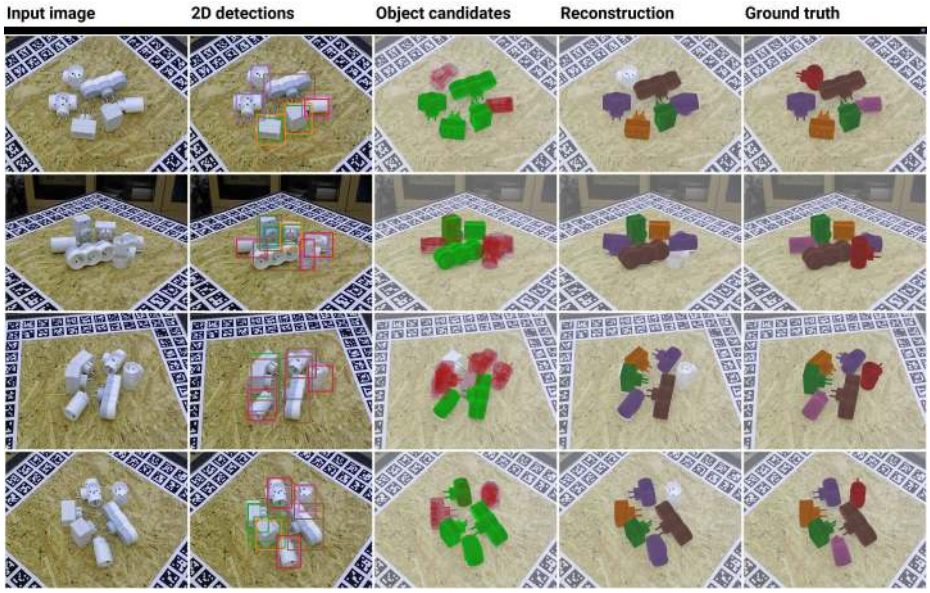
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 23.

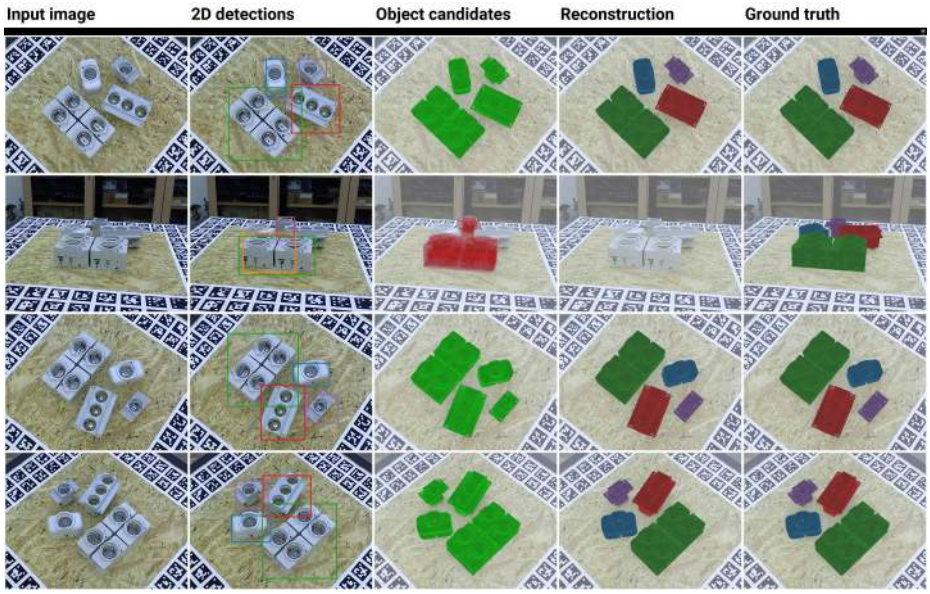
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 24.

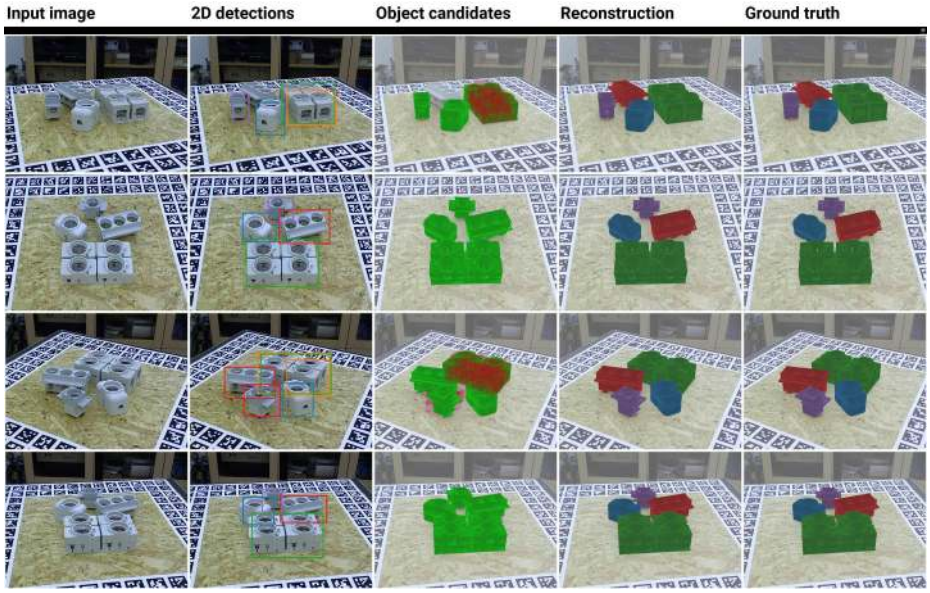
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 25.

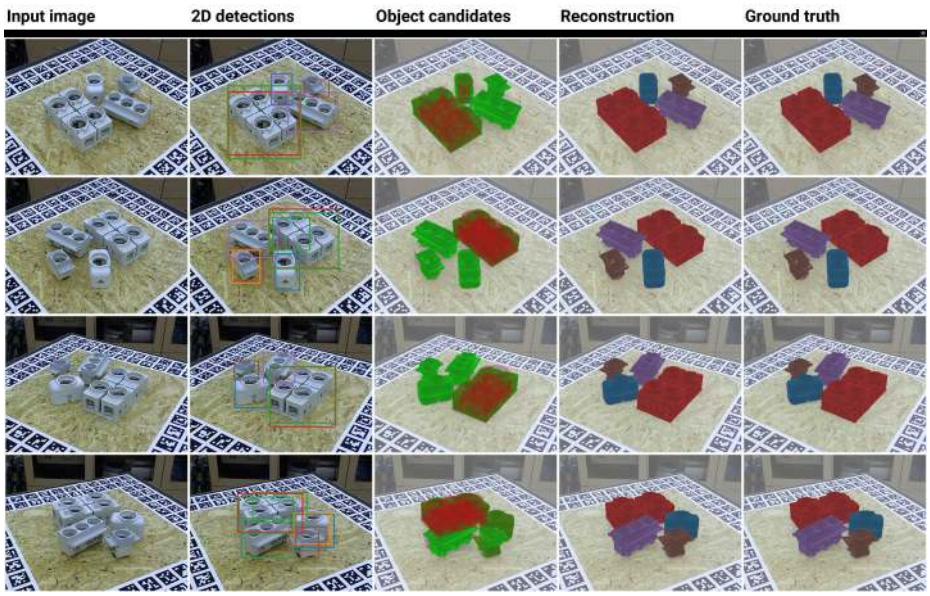
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 26.

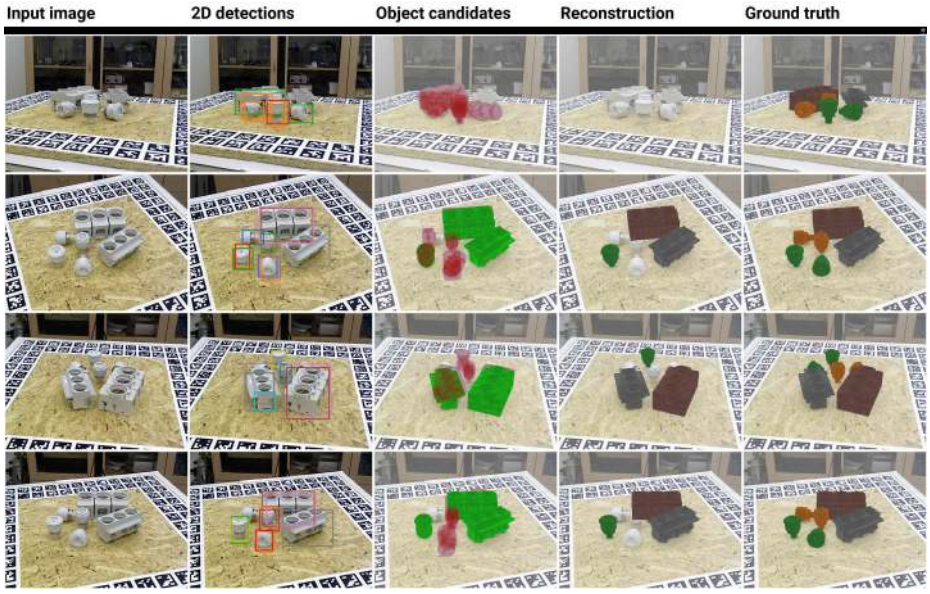
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 27.

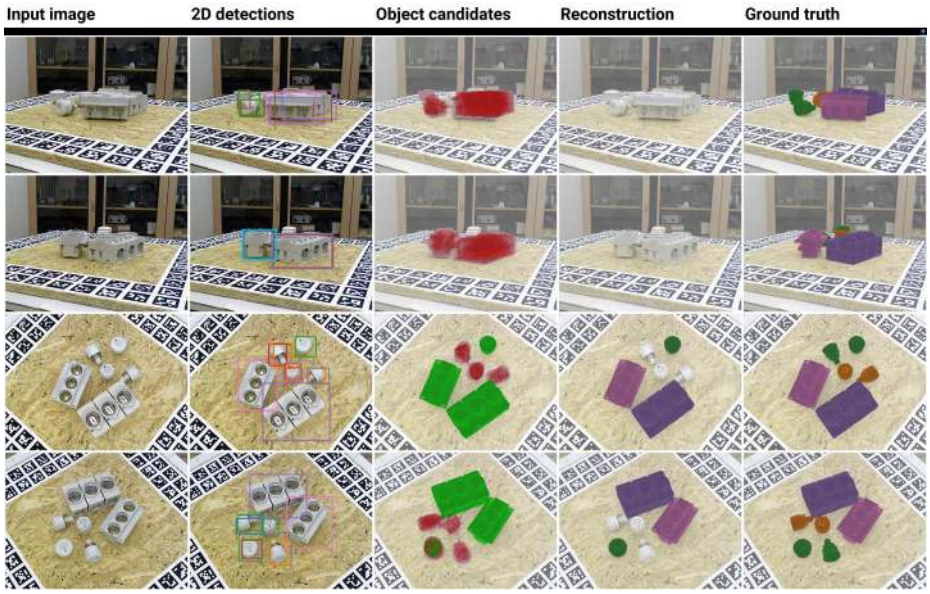
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 28.

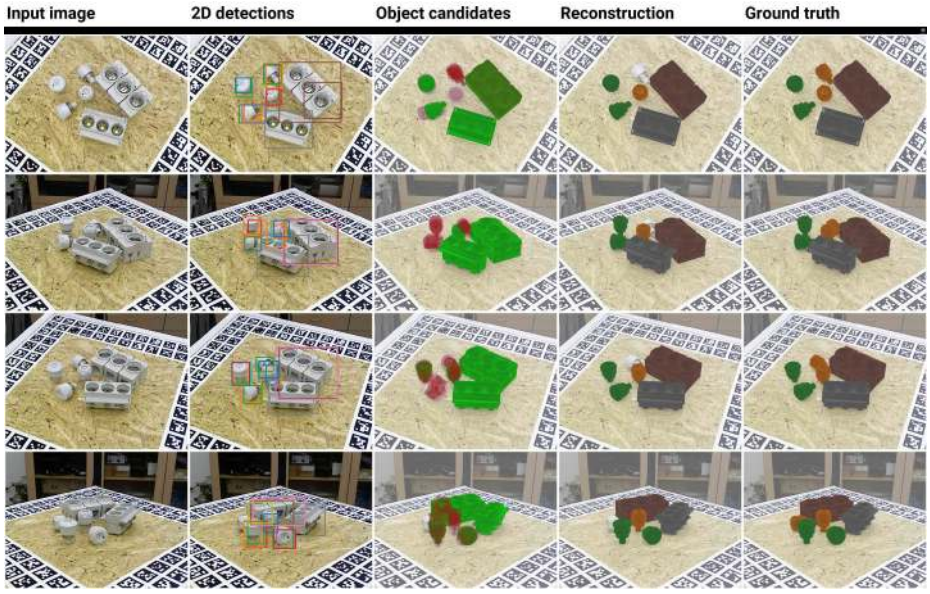
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 29.

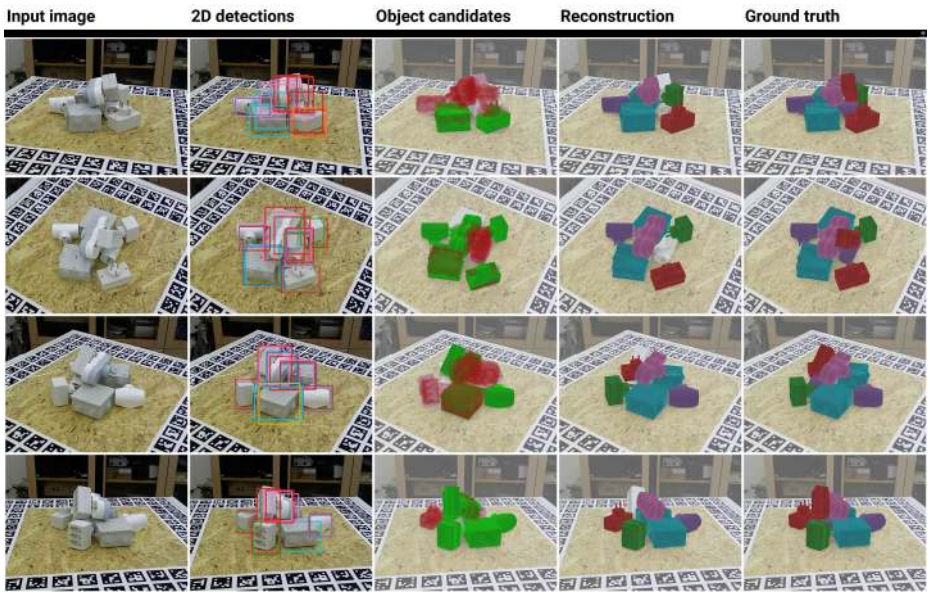
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 30.

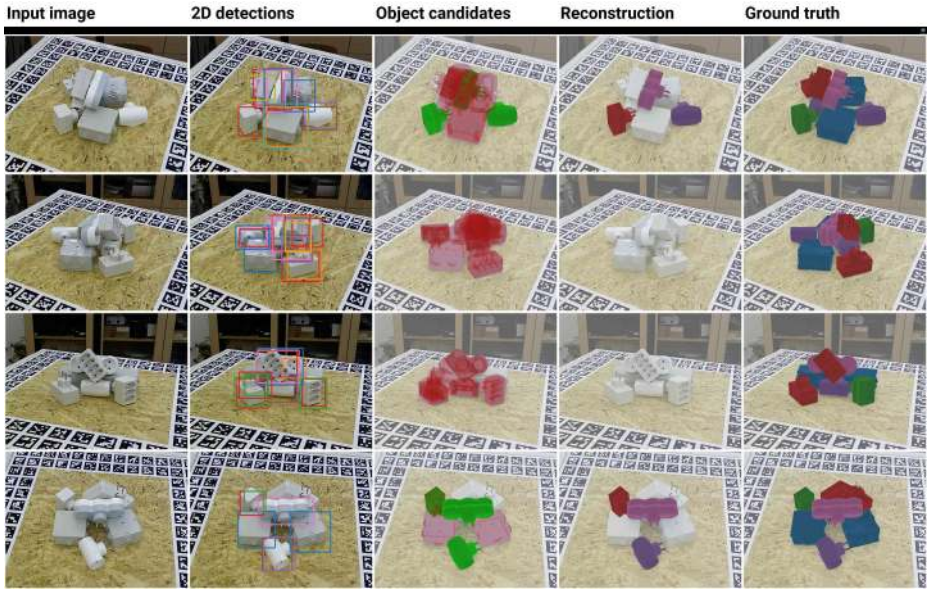
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 31.

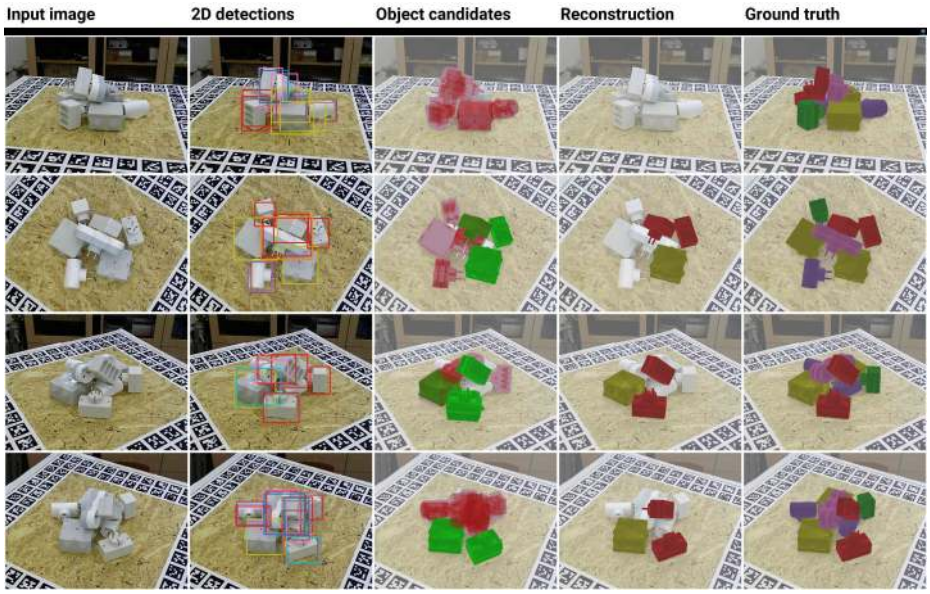
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 32.

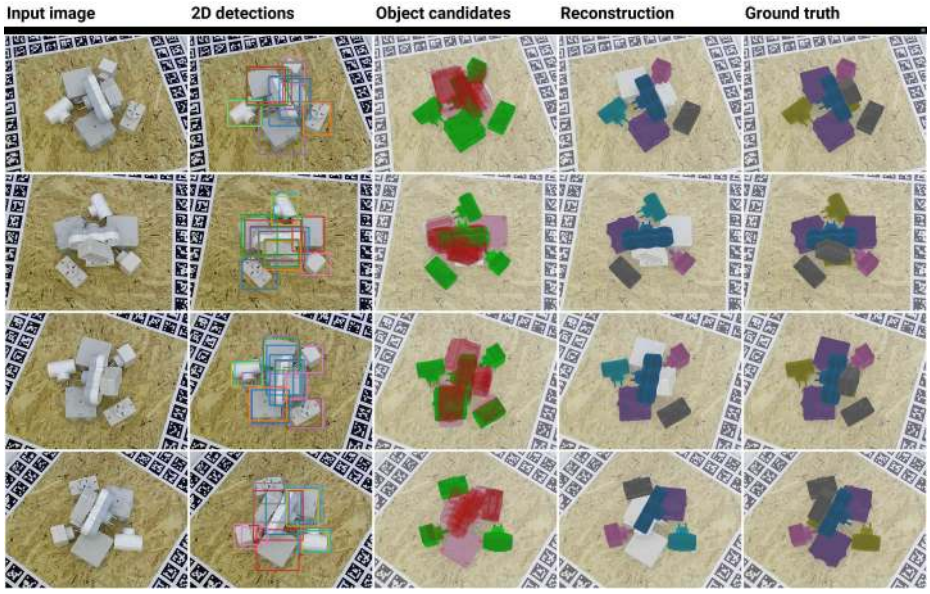
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 33.

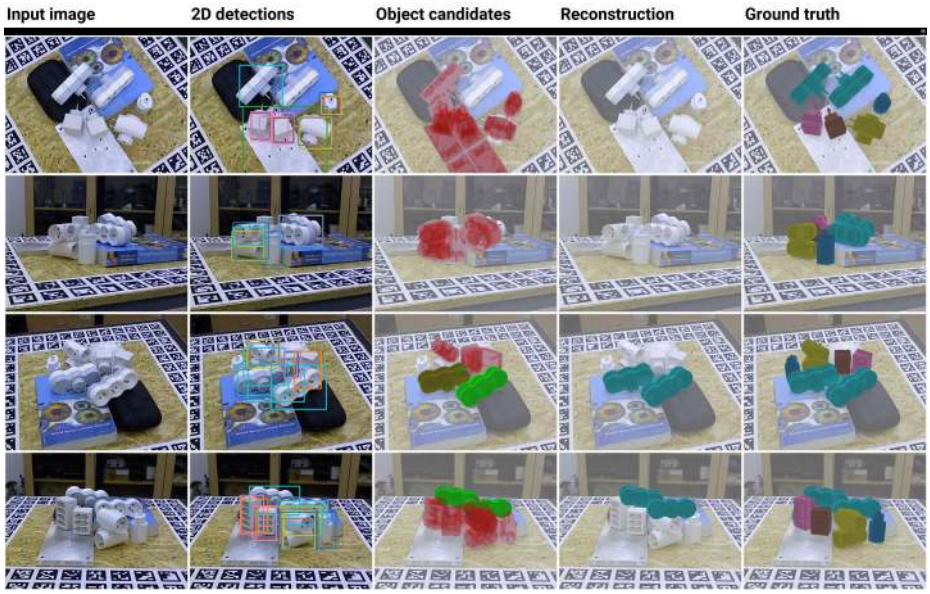
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 34.

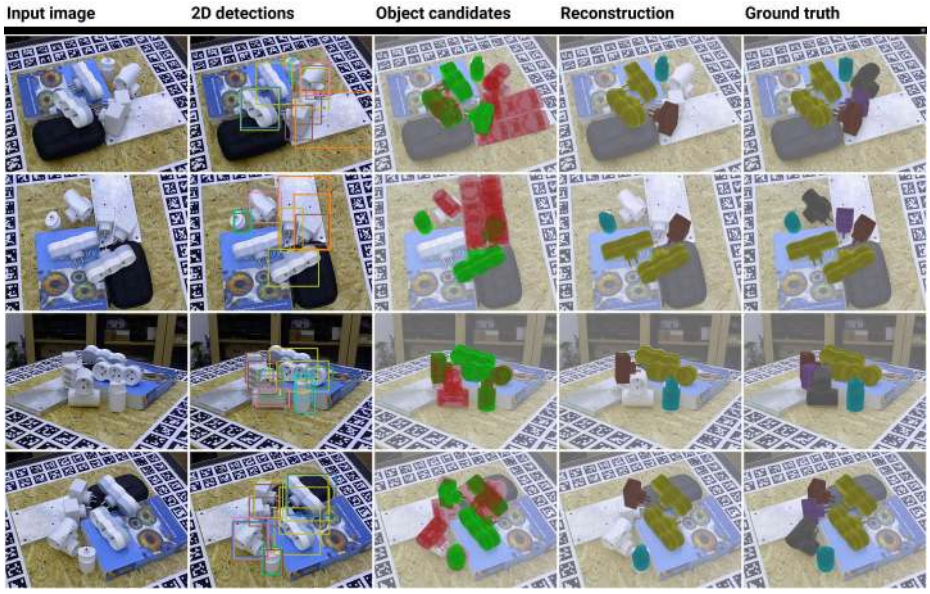
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 35.

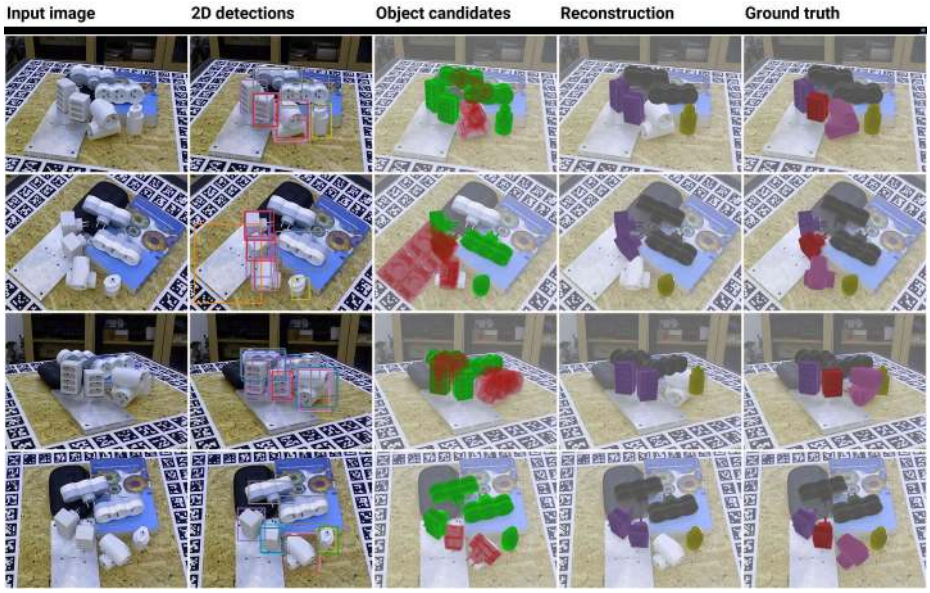
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 36.

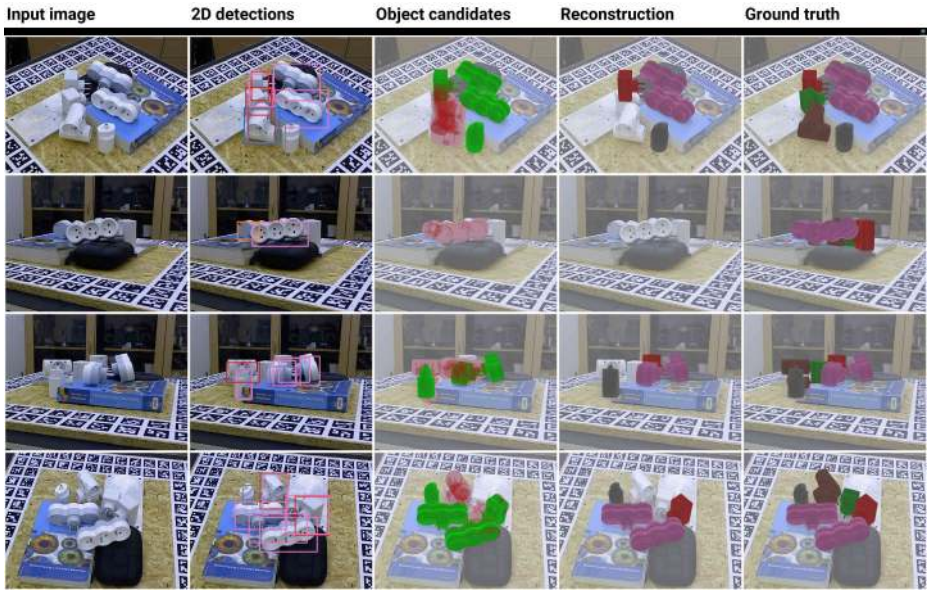
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 37.

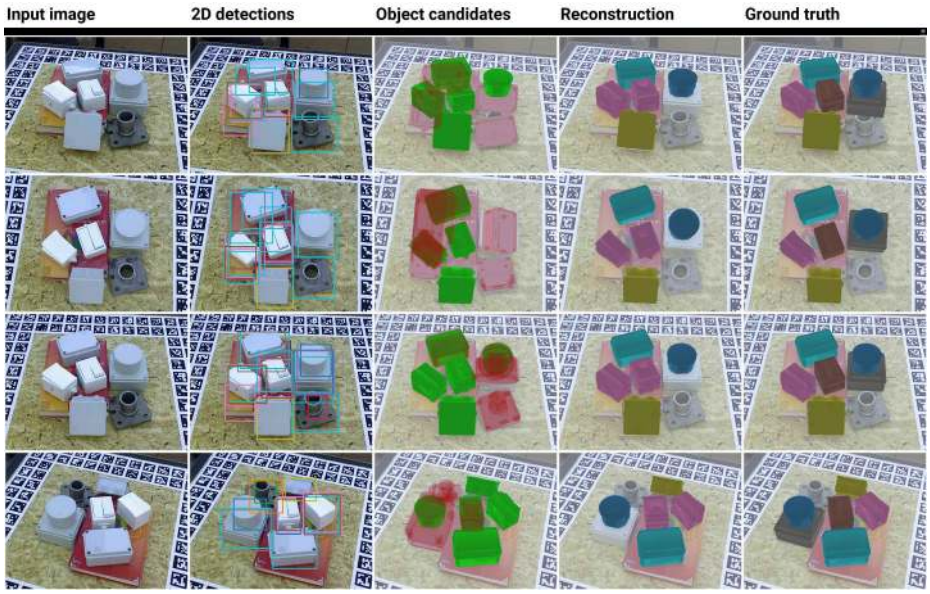
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 38.

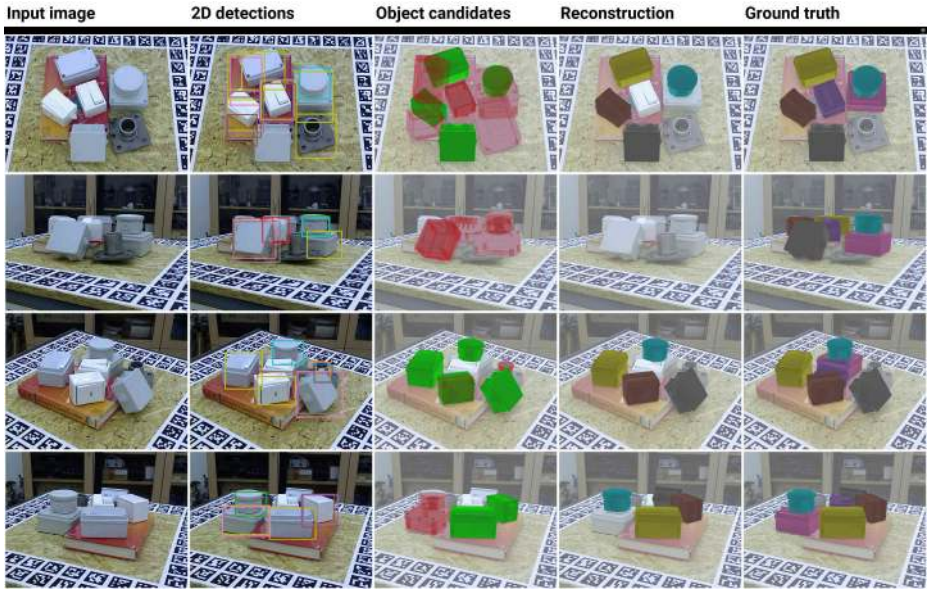
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 39.

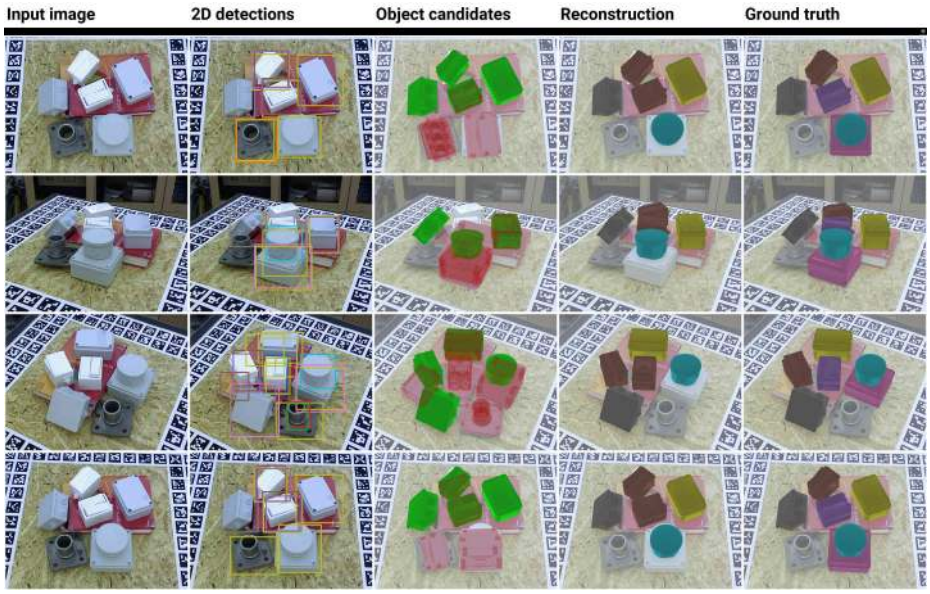
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 40.

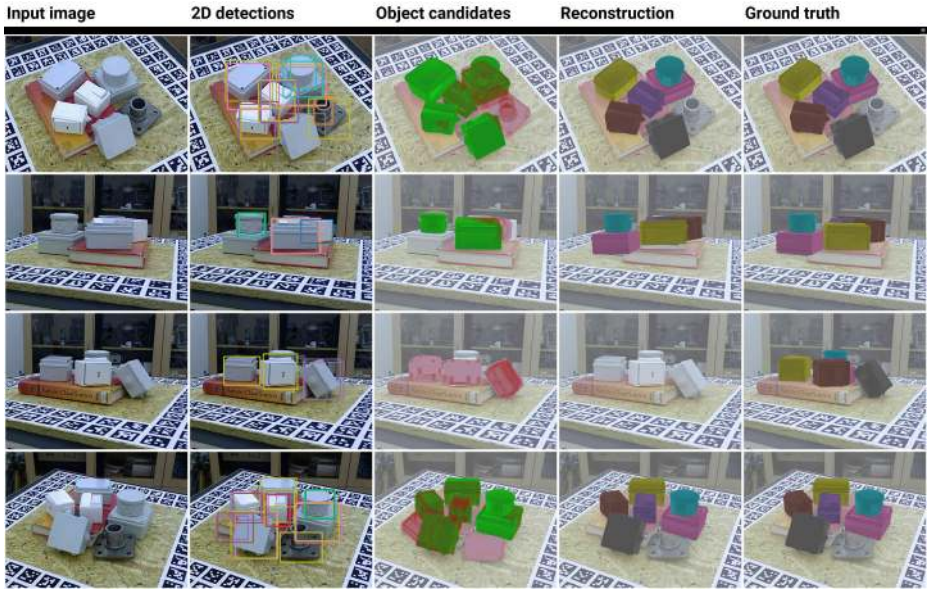
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 41.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 42.

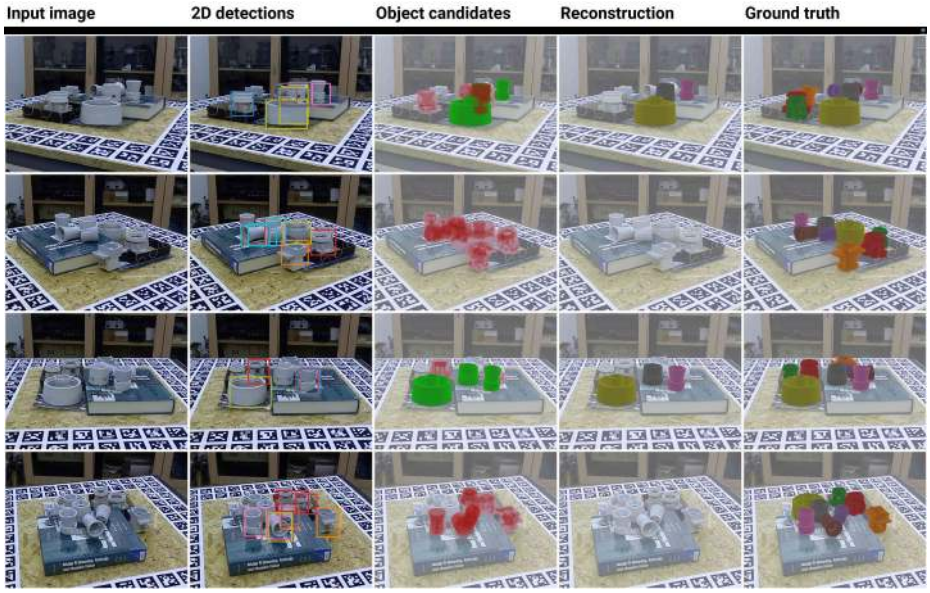
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 43.

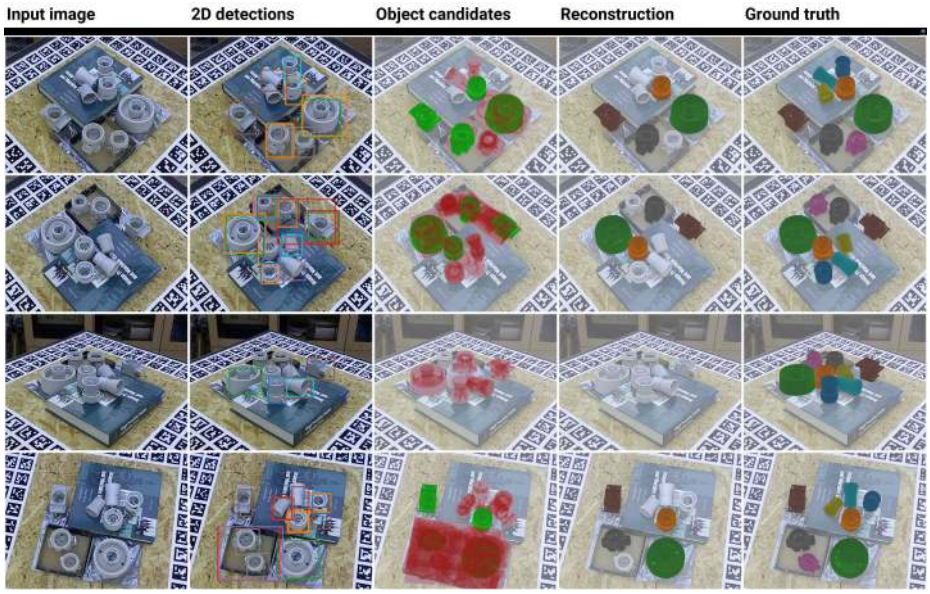
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 44.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 45.

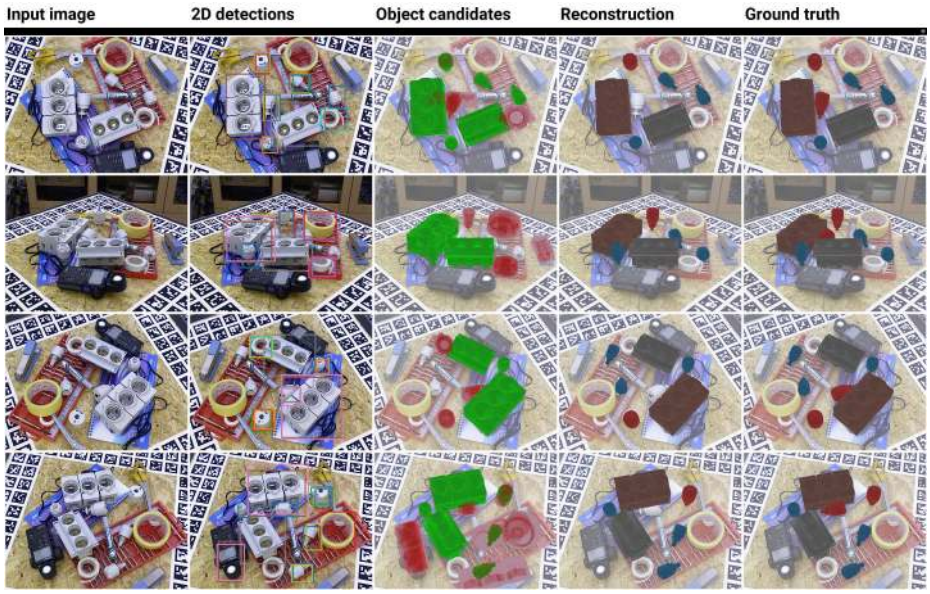
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 46.

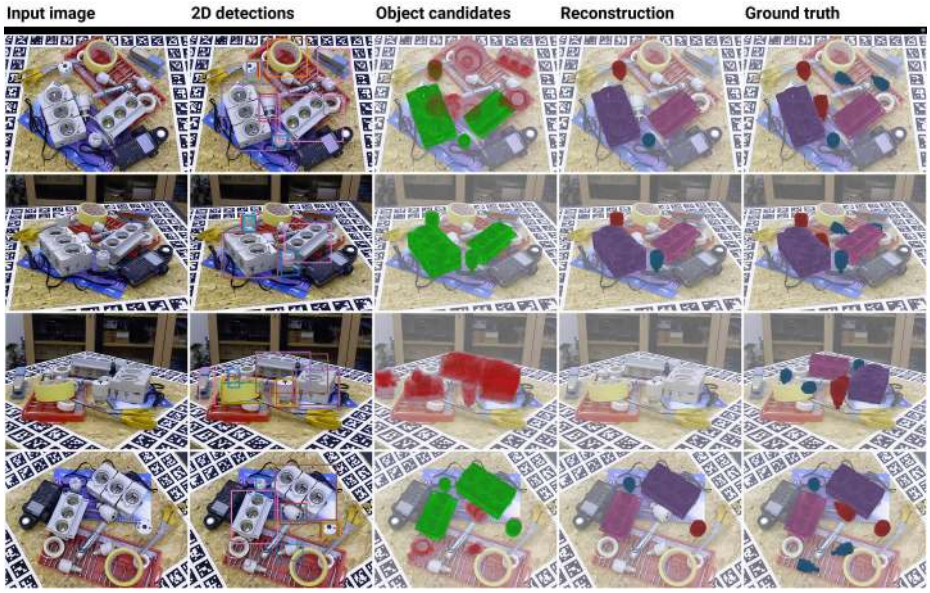
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 47.

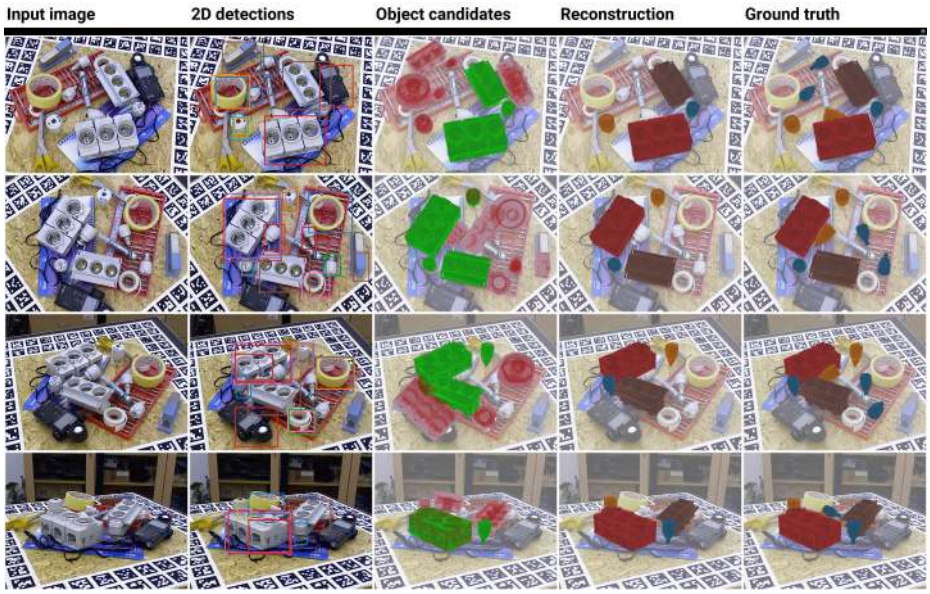
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 48.

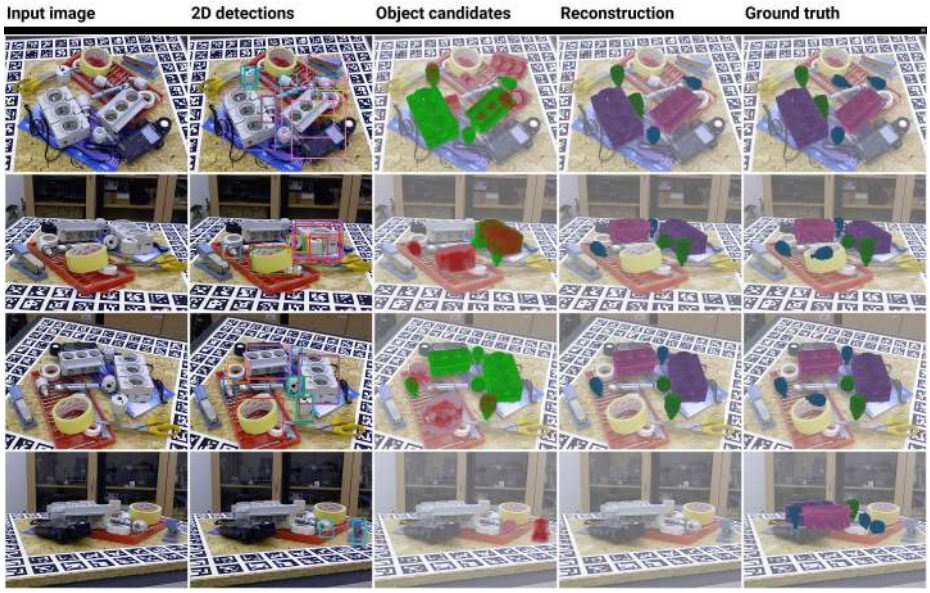
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 49.

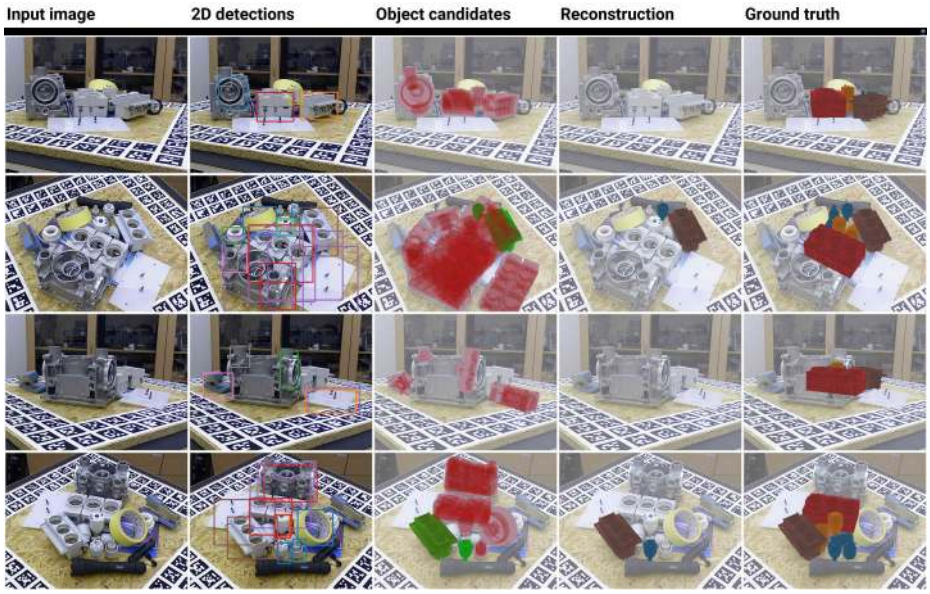
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 50.

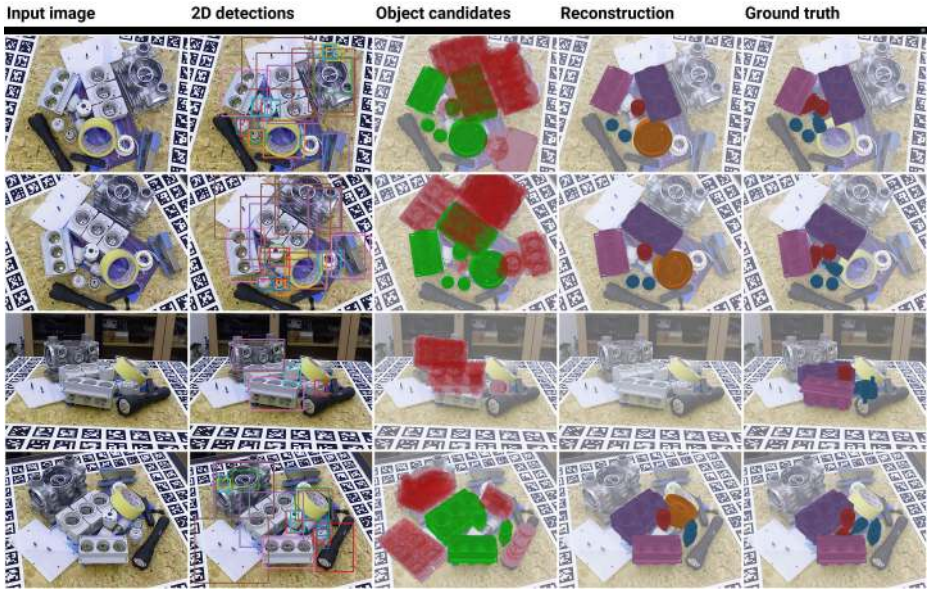
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 51.

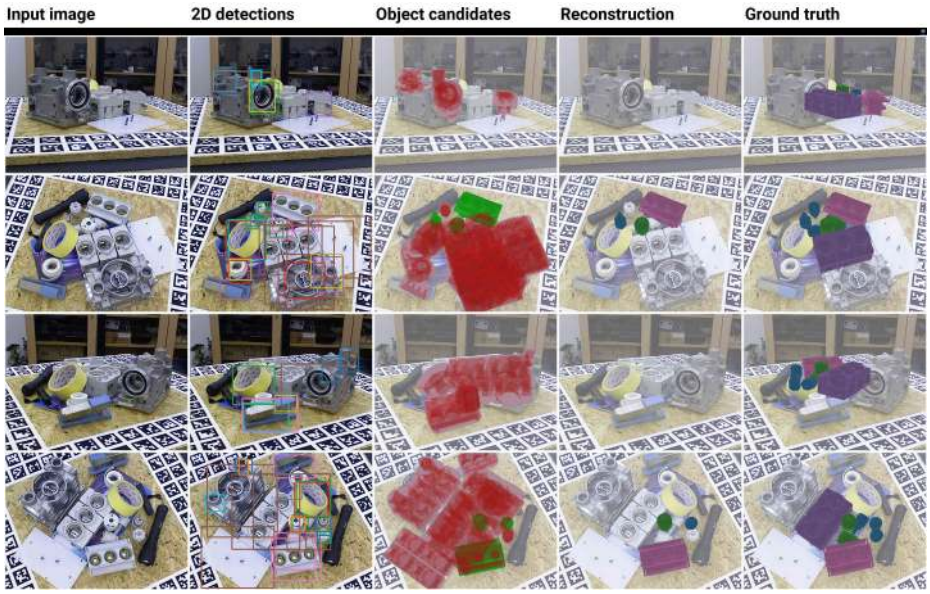
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 52.

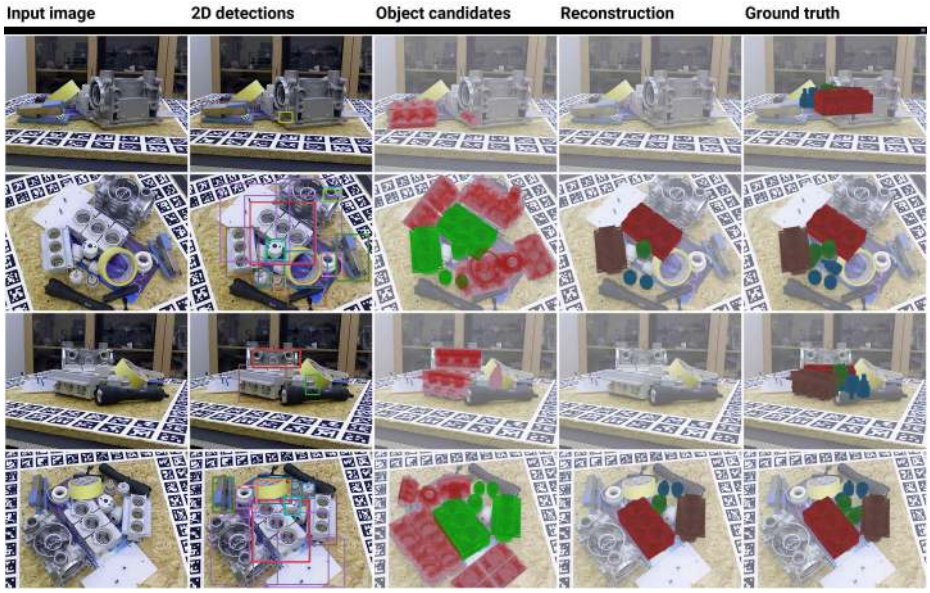
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 53.

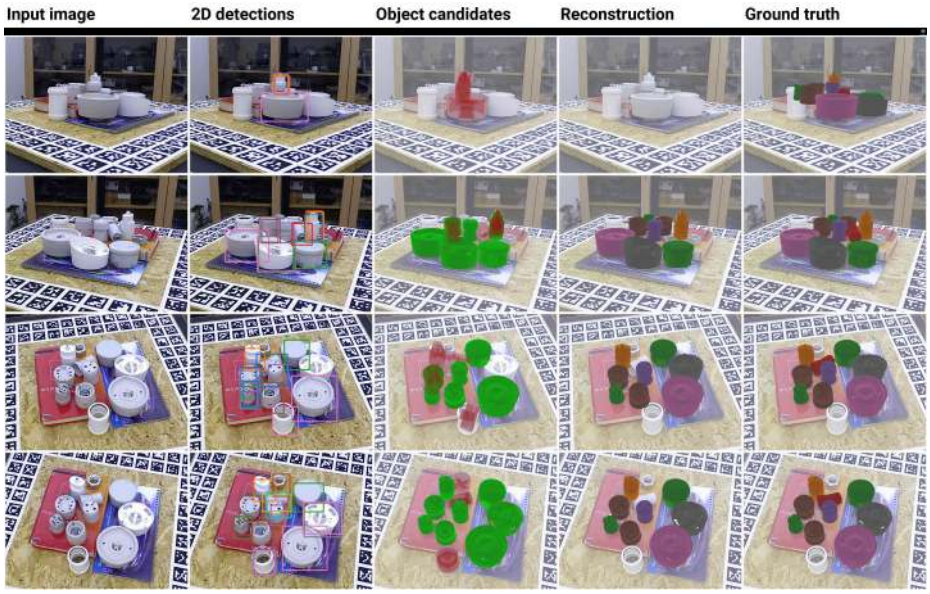
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 54.

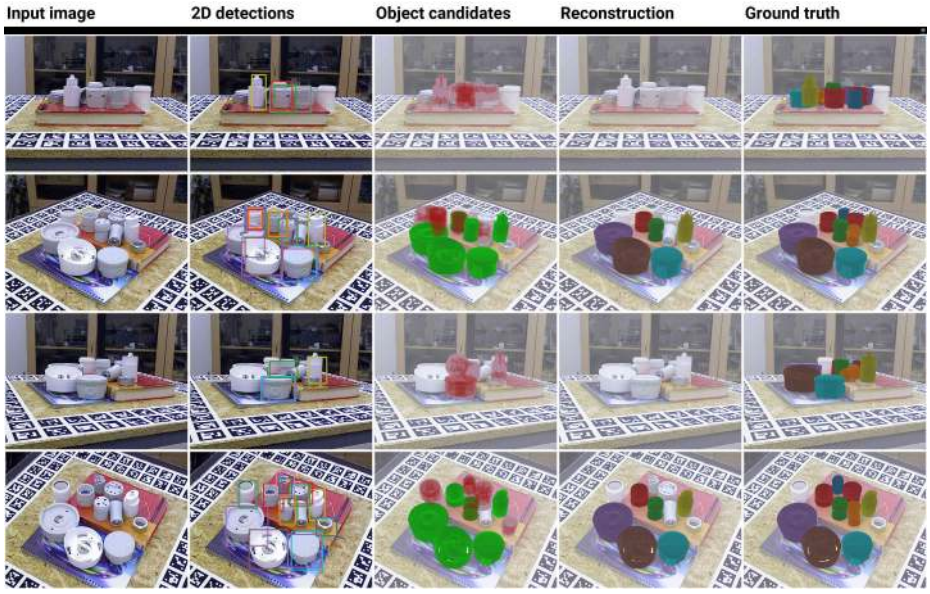
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 55.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 56.

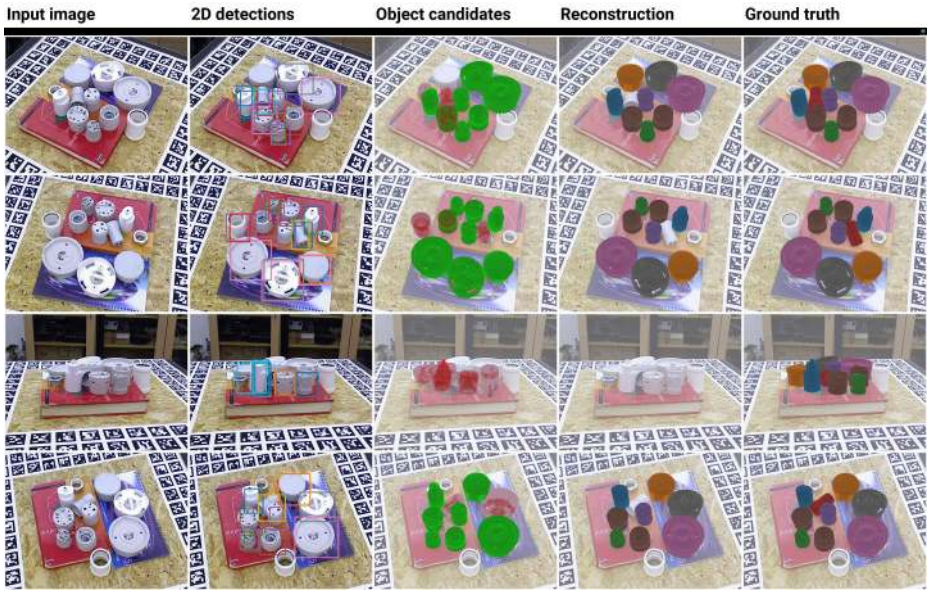
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 57.

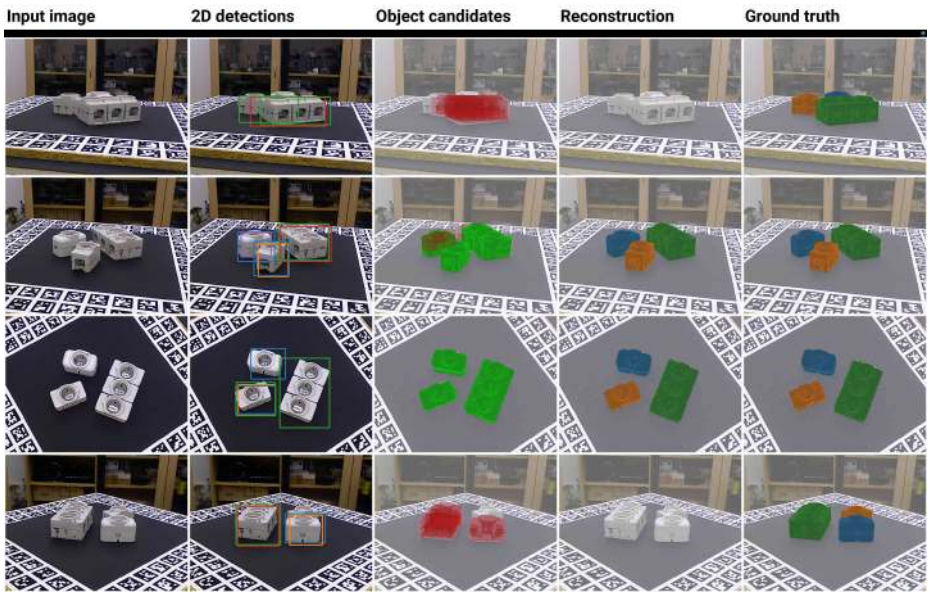
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 58.

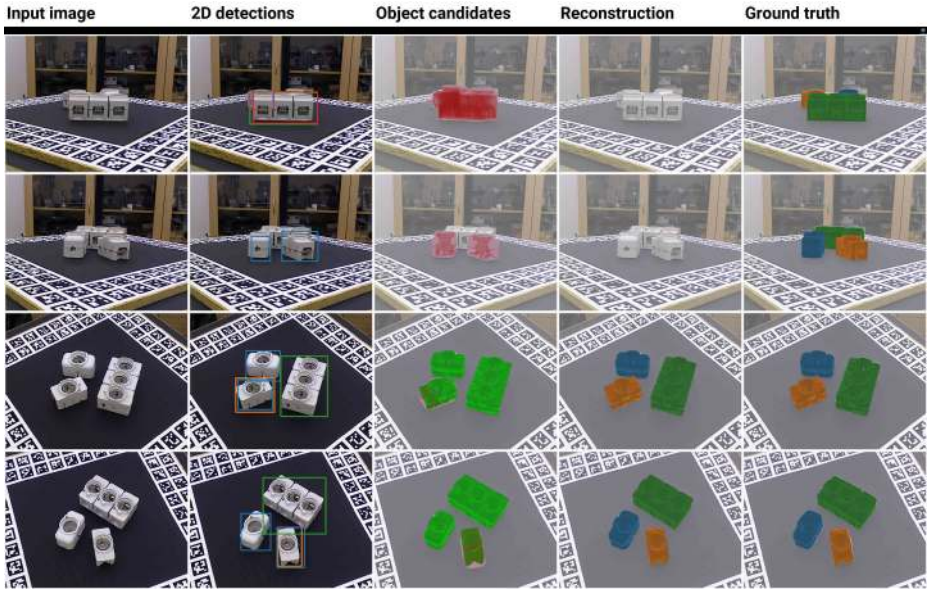
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 59.

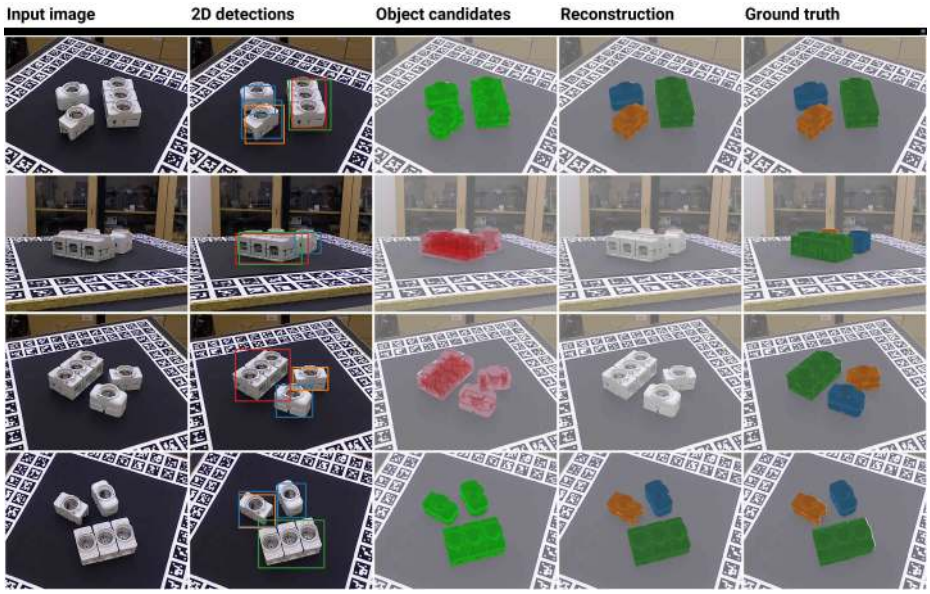
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 60.

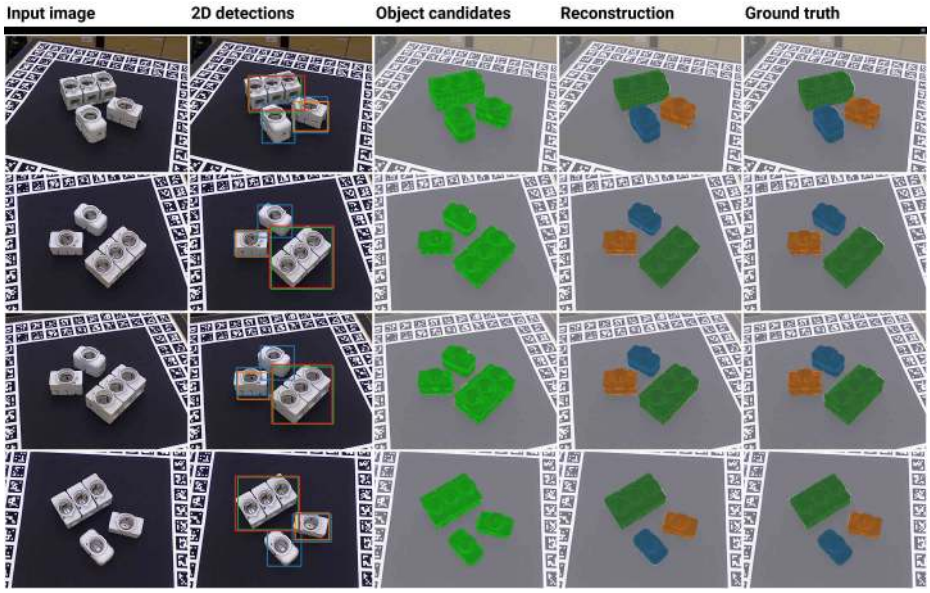
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 61.

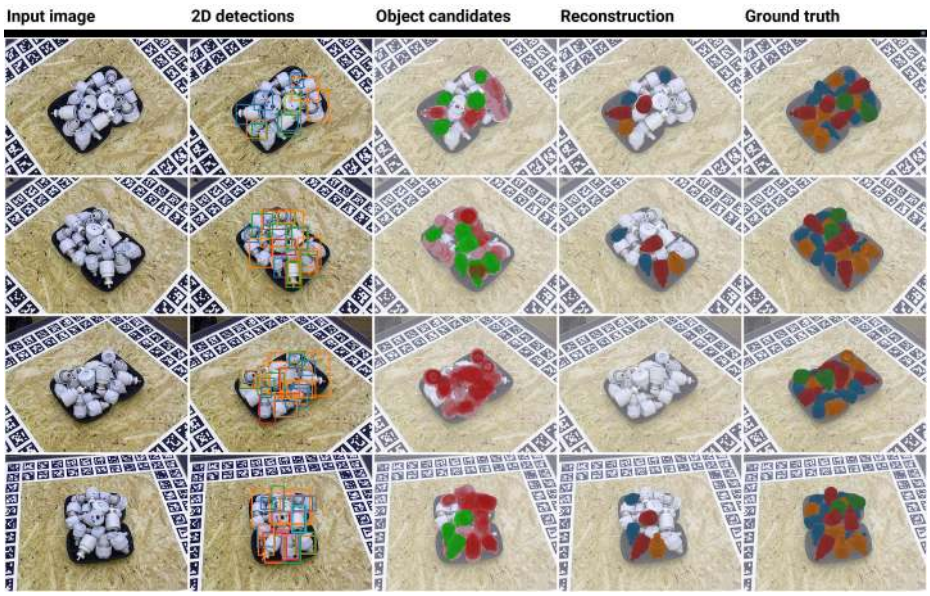
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 62.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 63.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 64.

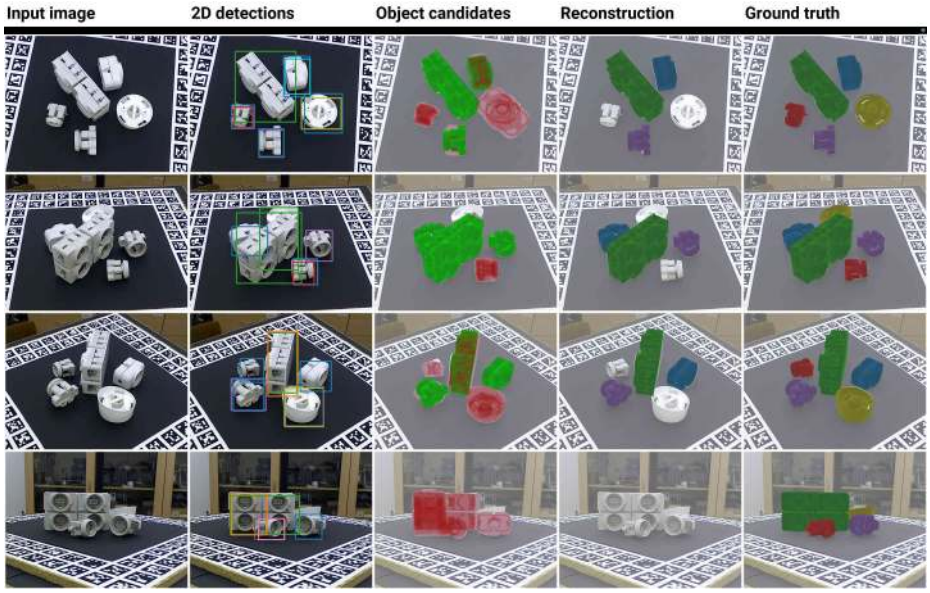
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 65.

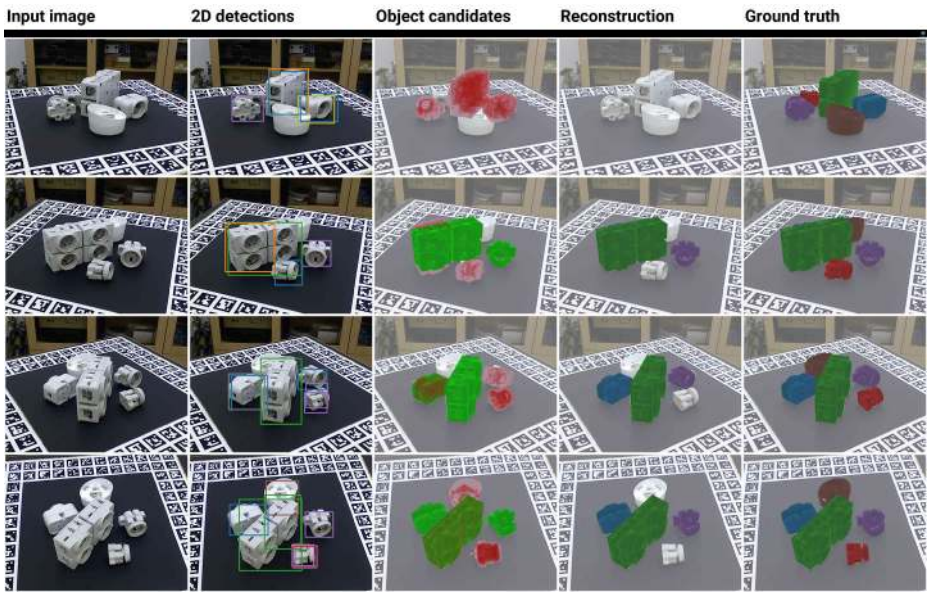
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 66.

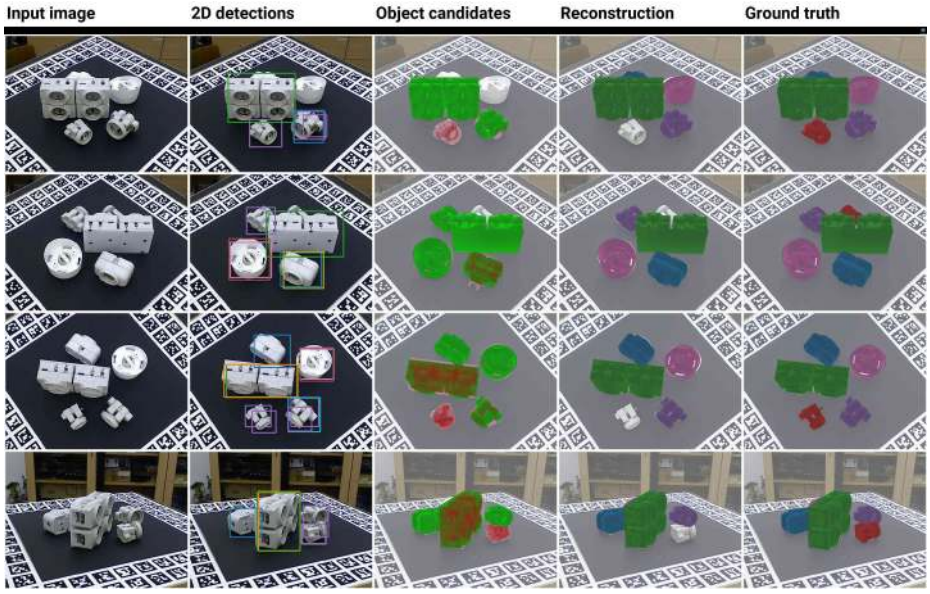
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 67.

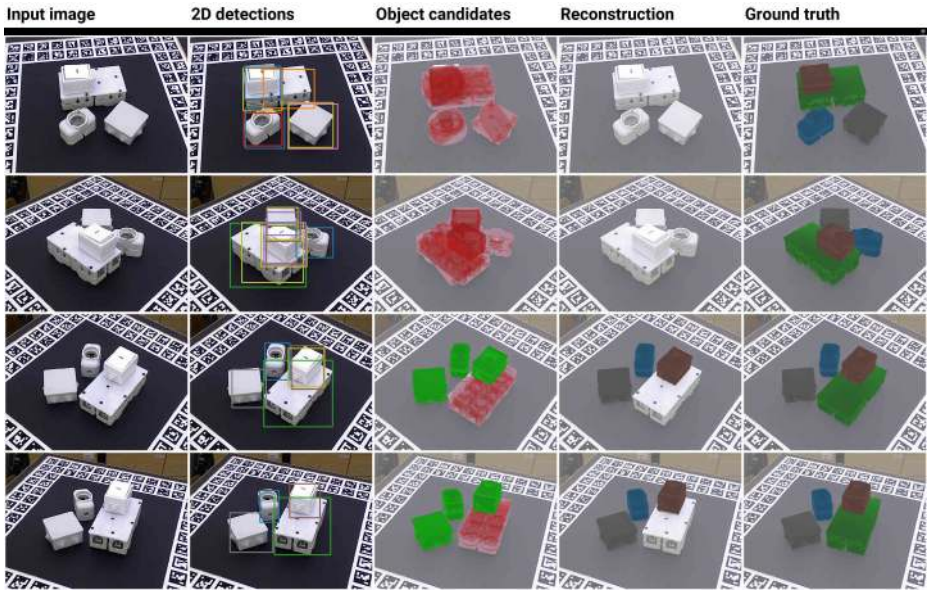
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 68.

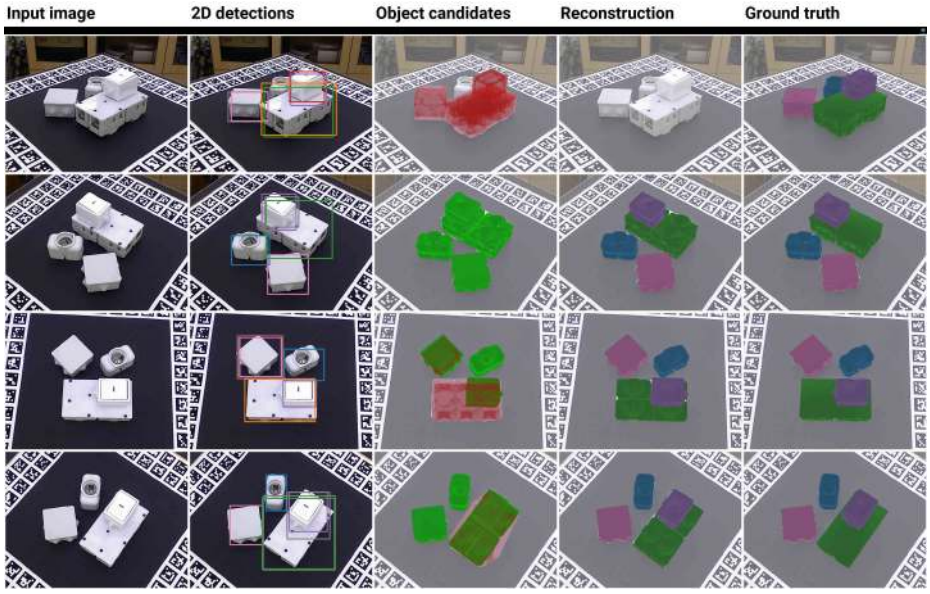
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 69.

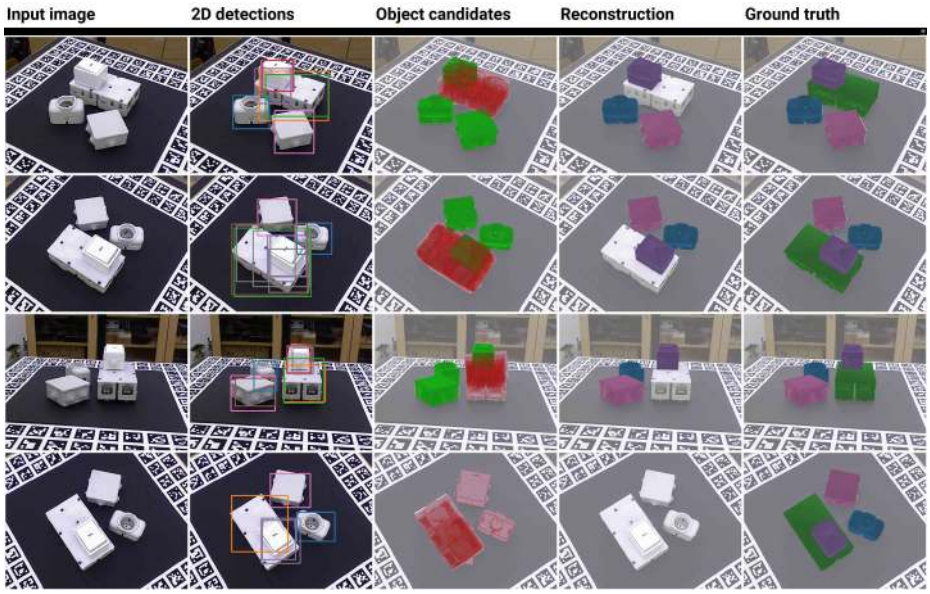
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 70.

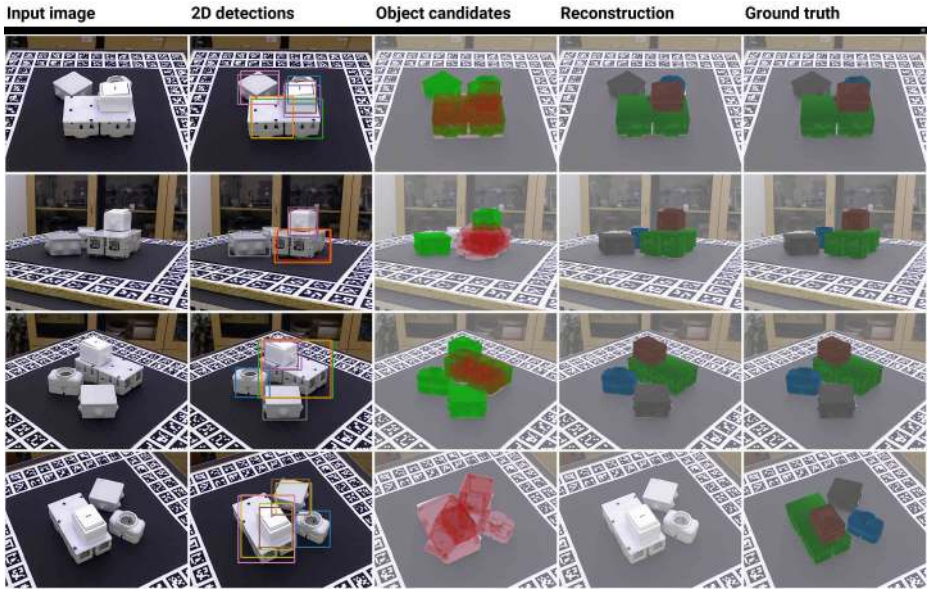
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 71.

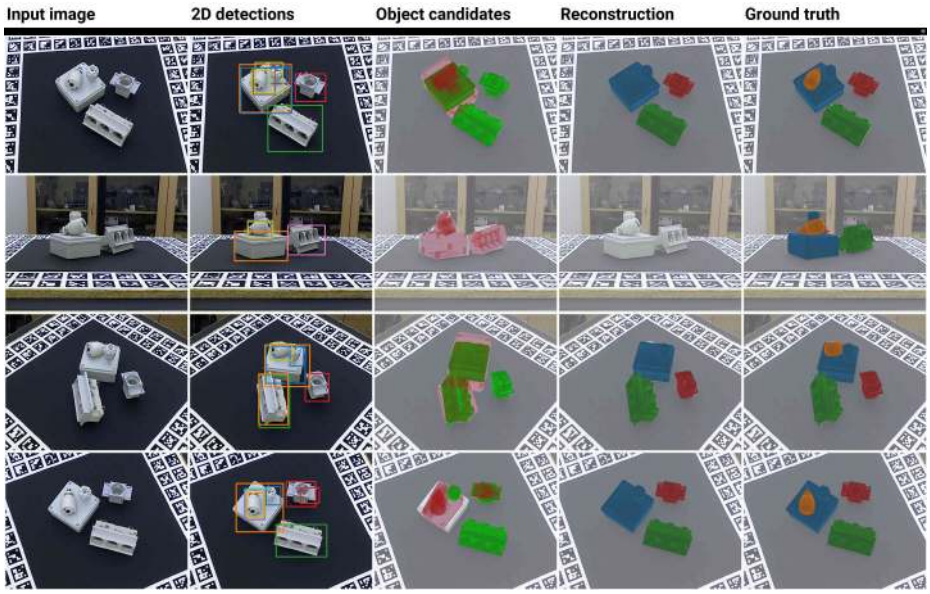
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 72.

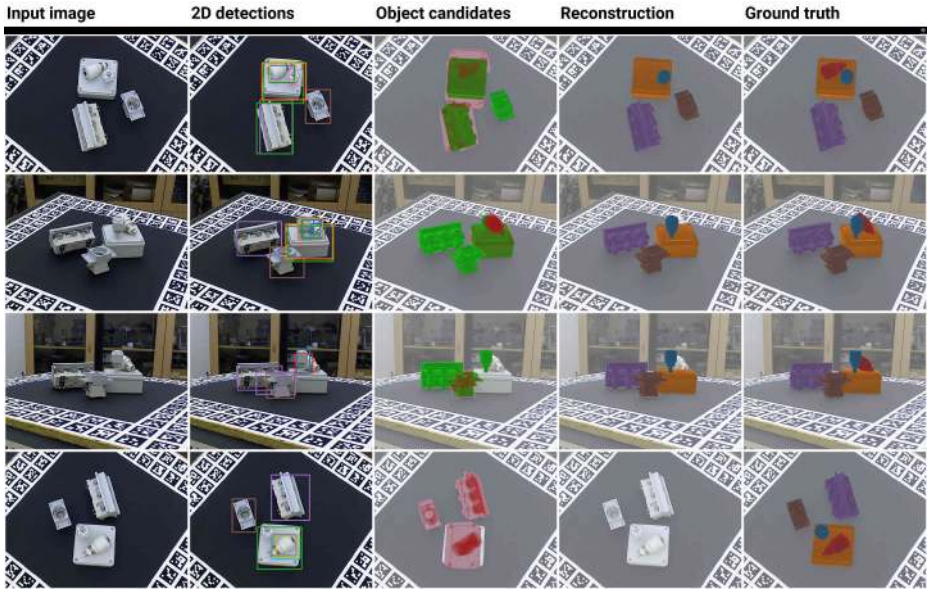
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 73.

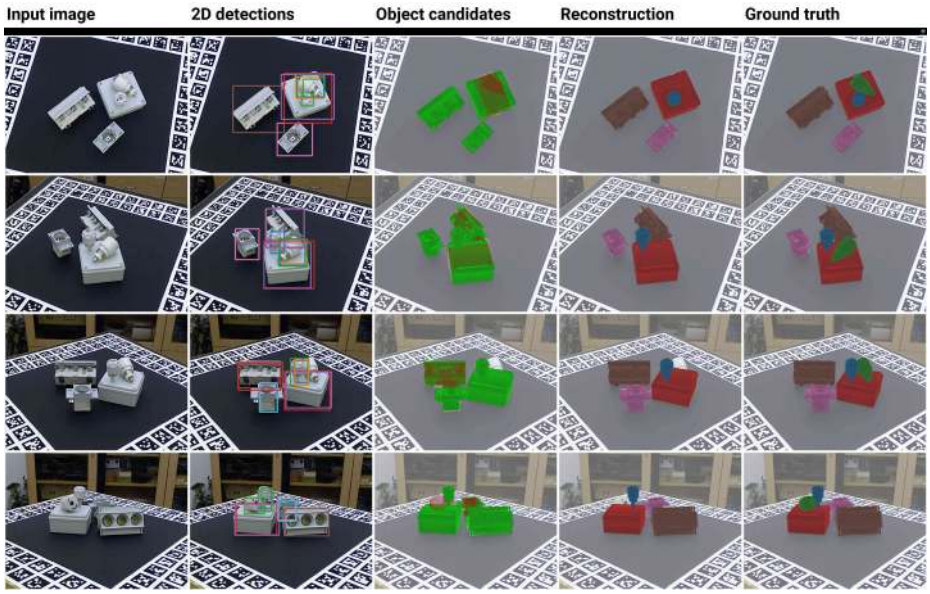
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 74.

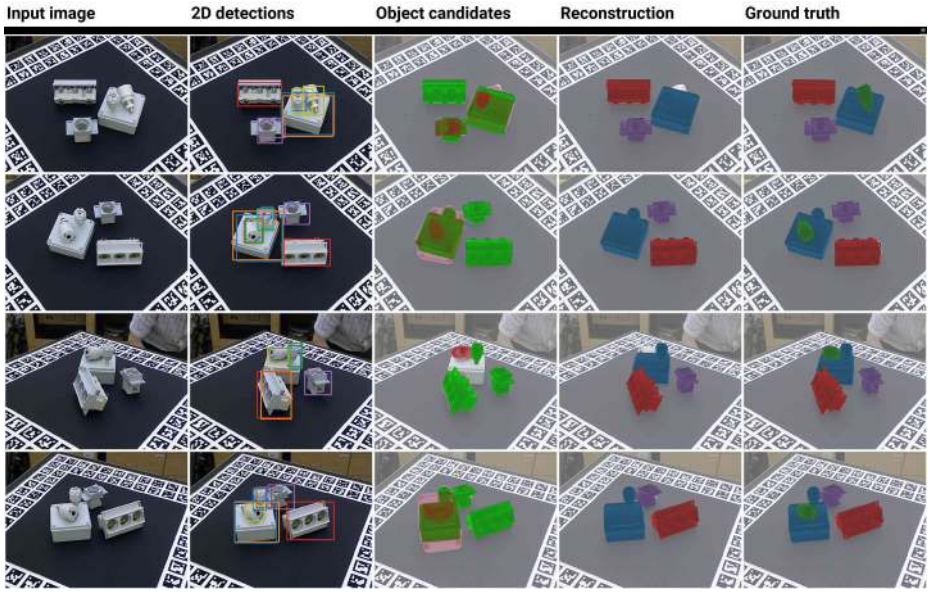
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 75.

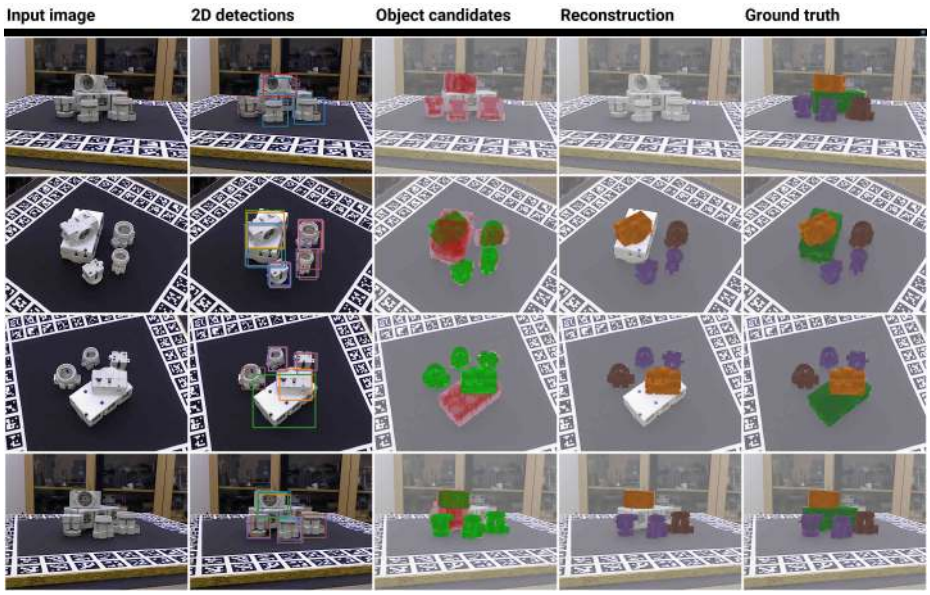
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 76.

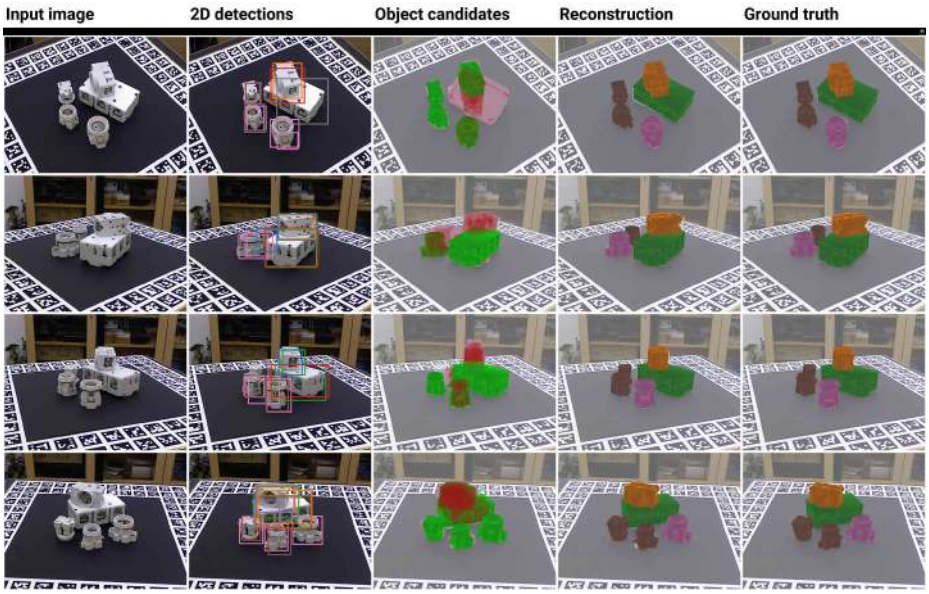
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 77.

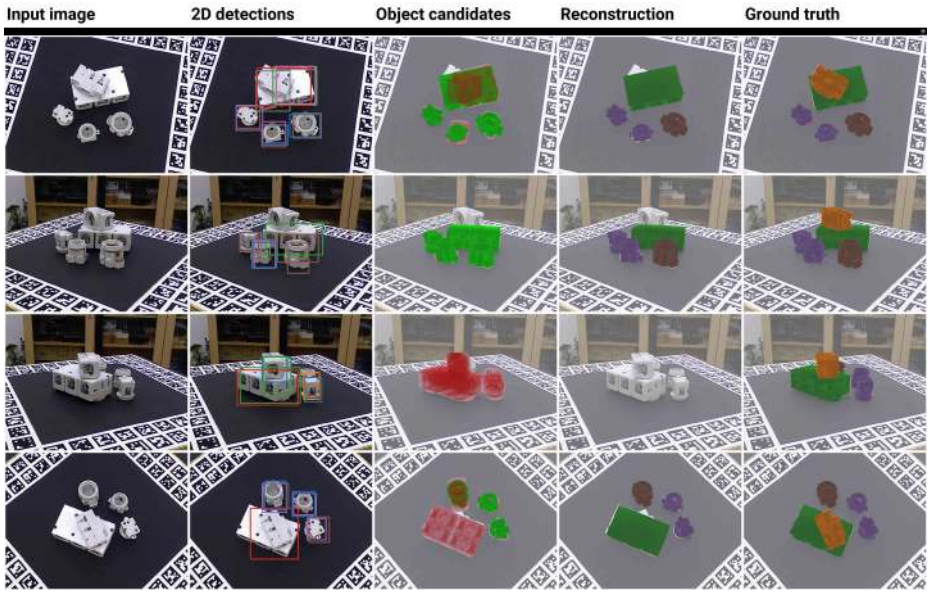
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 78.

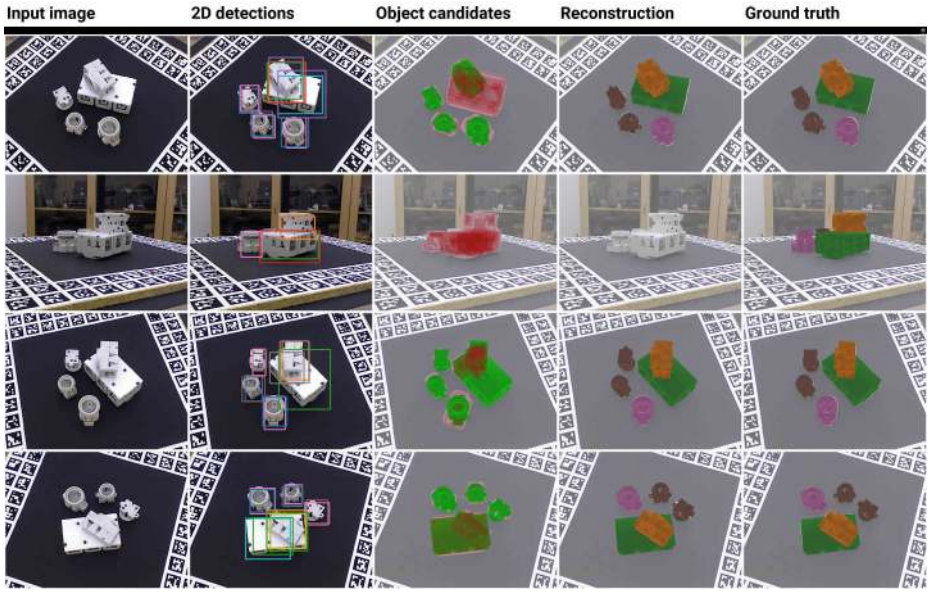
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 79.

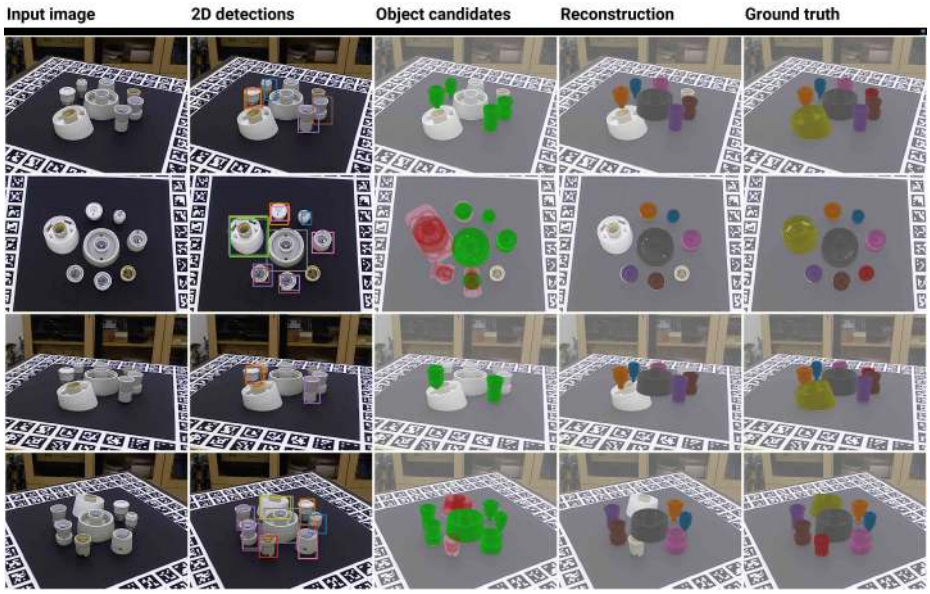
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 80.

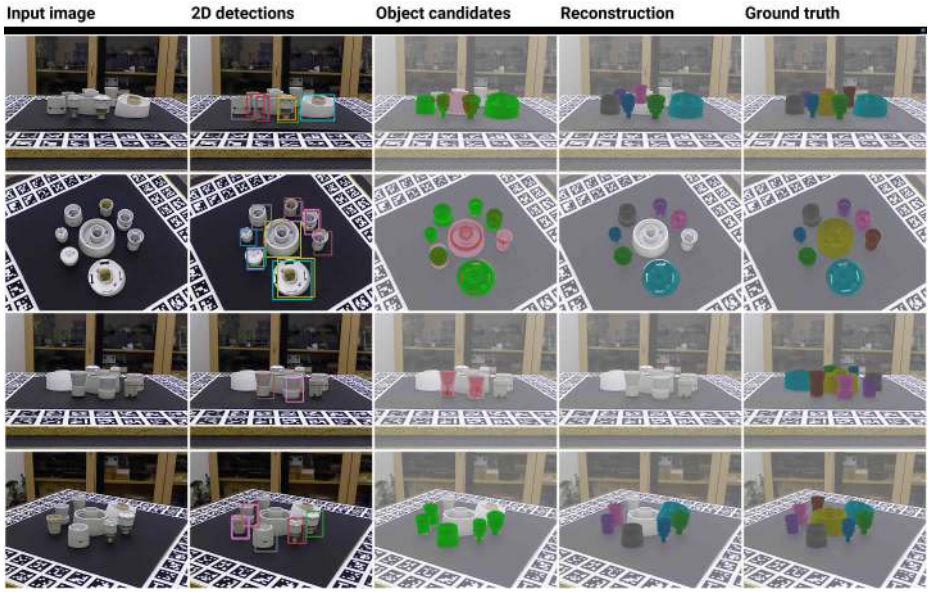
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 81.

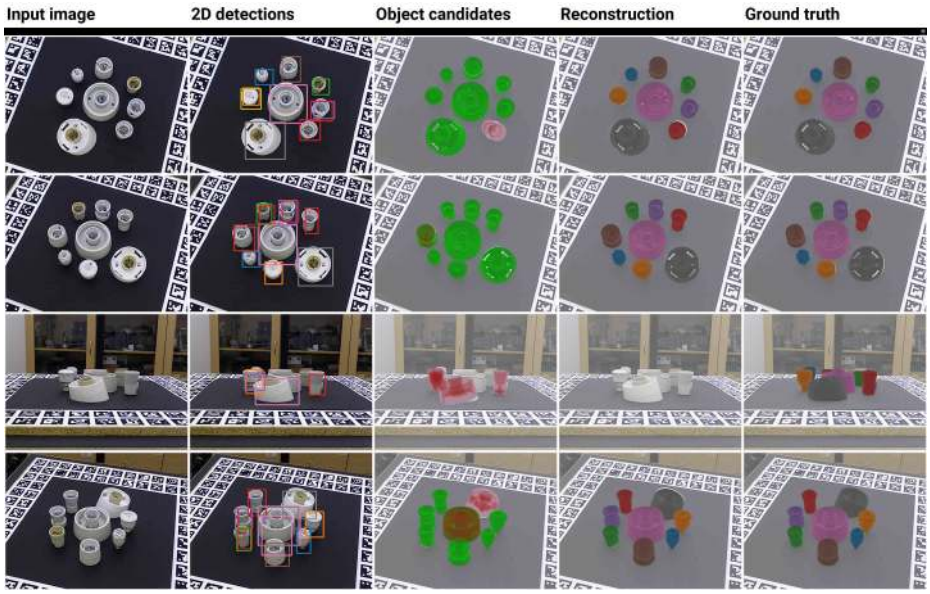
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 82.

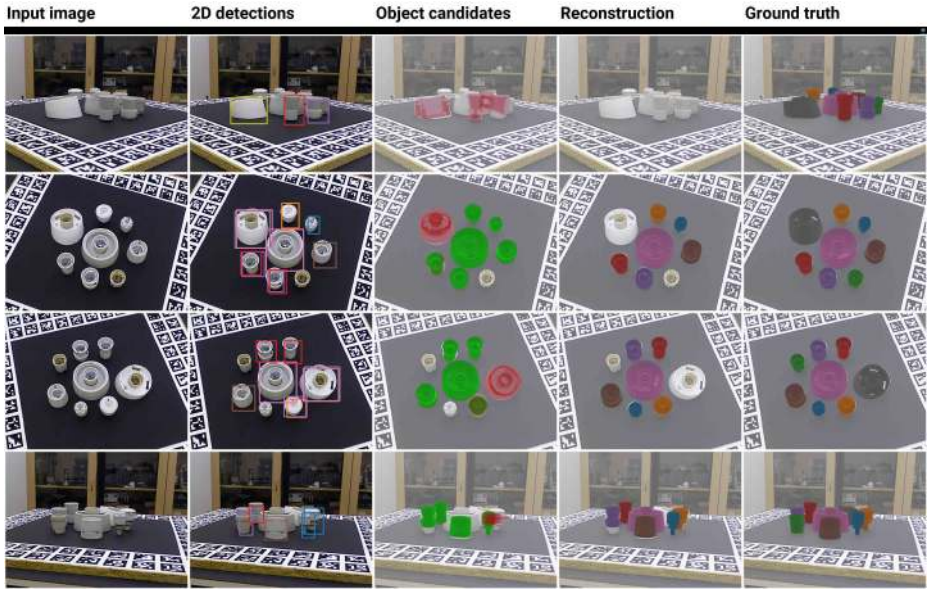
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 83.

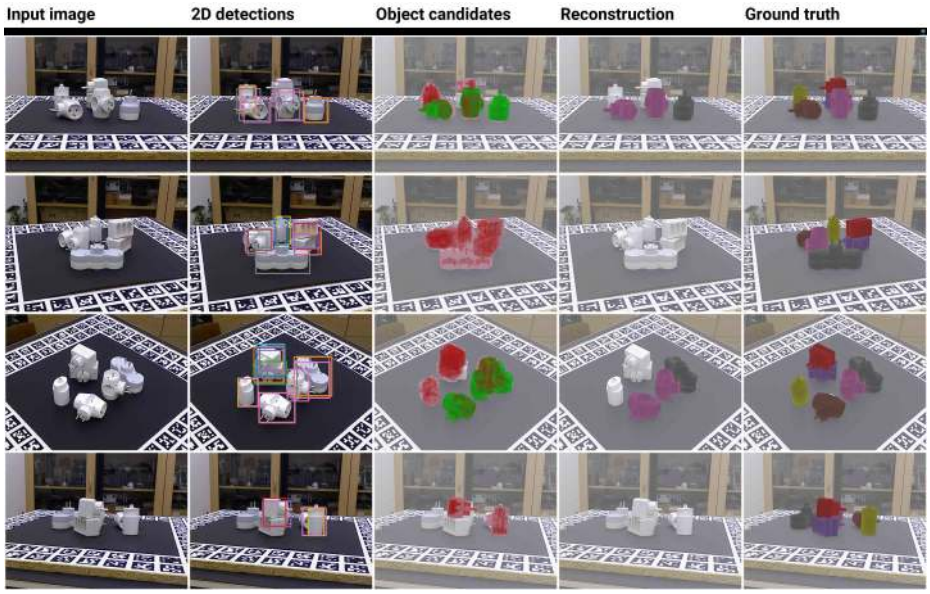
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 84.

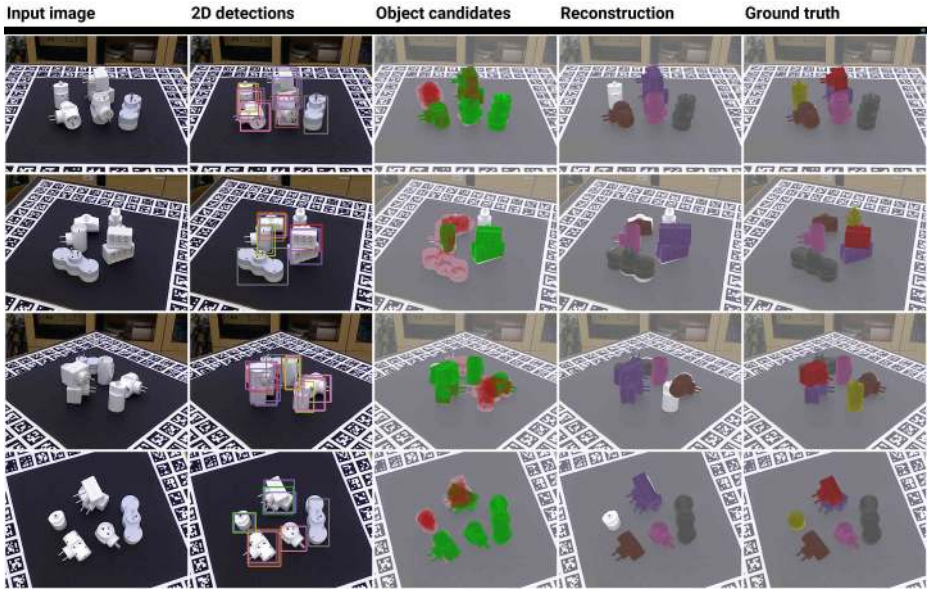
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 85.

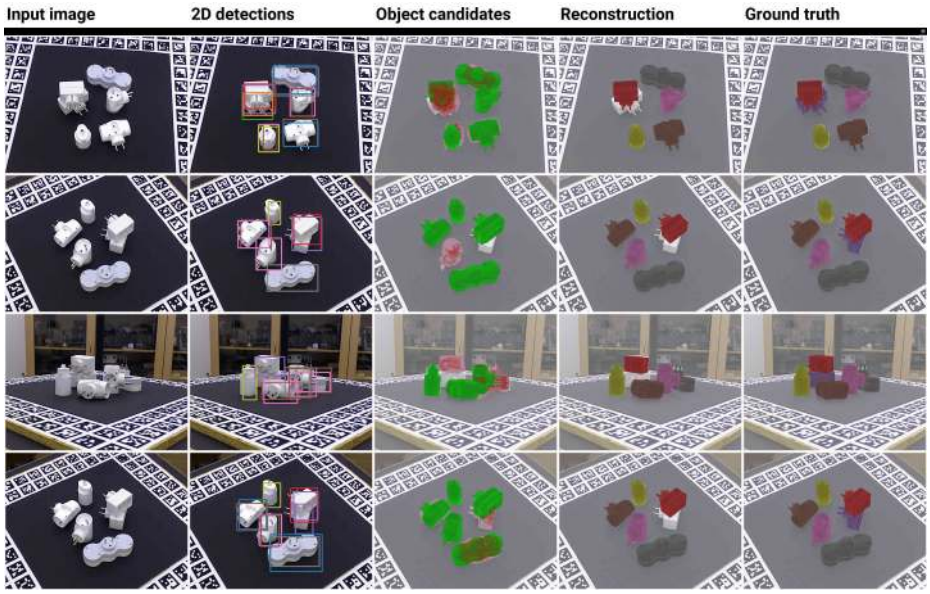
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 86.

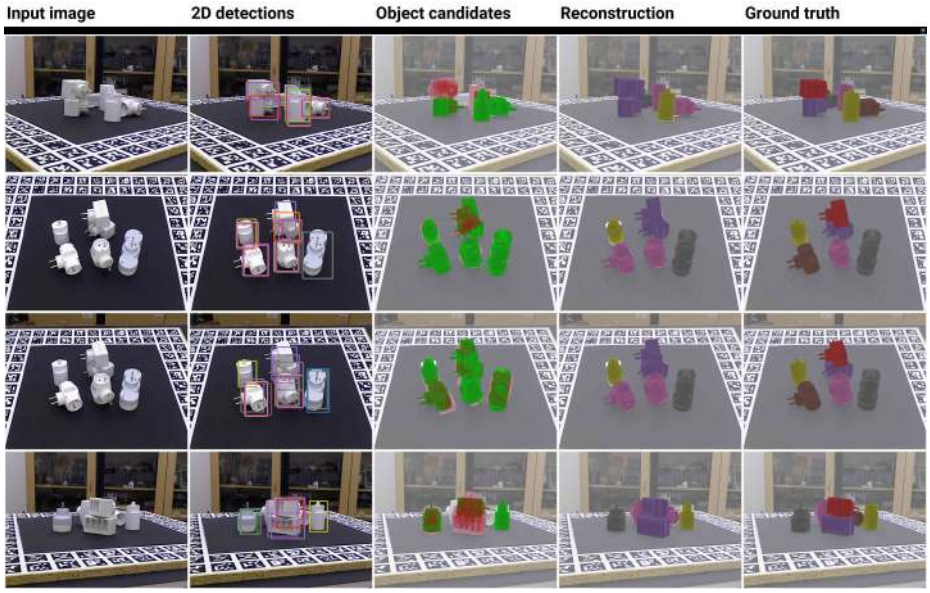
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 87.

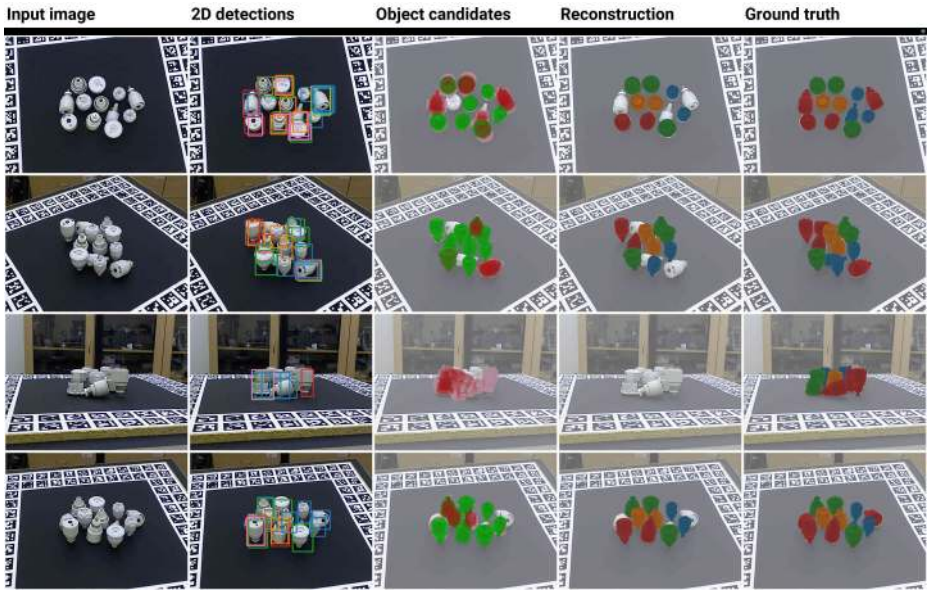
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 88.

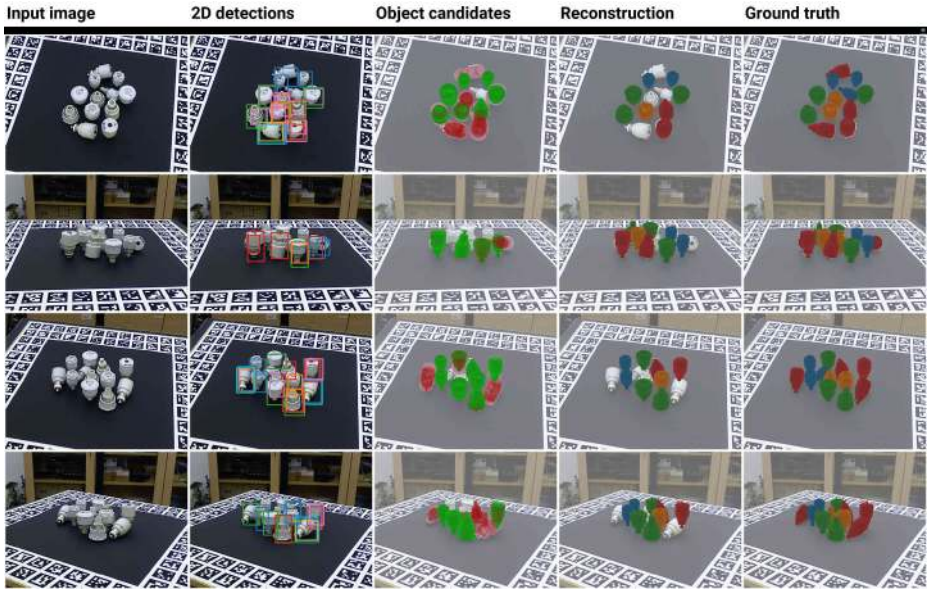
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 89.

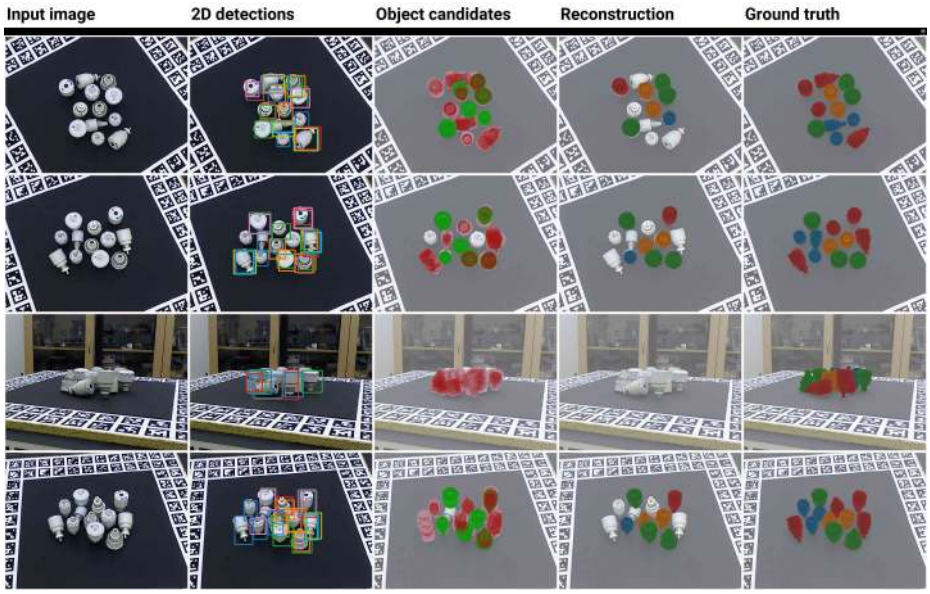
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 90.

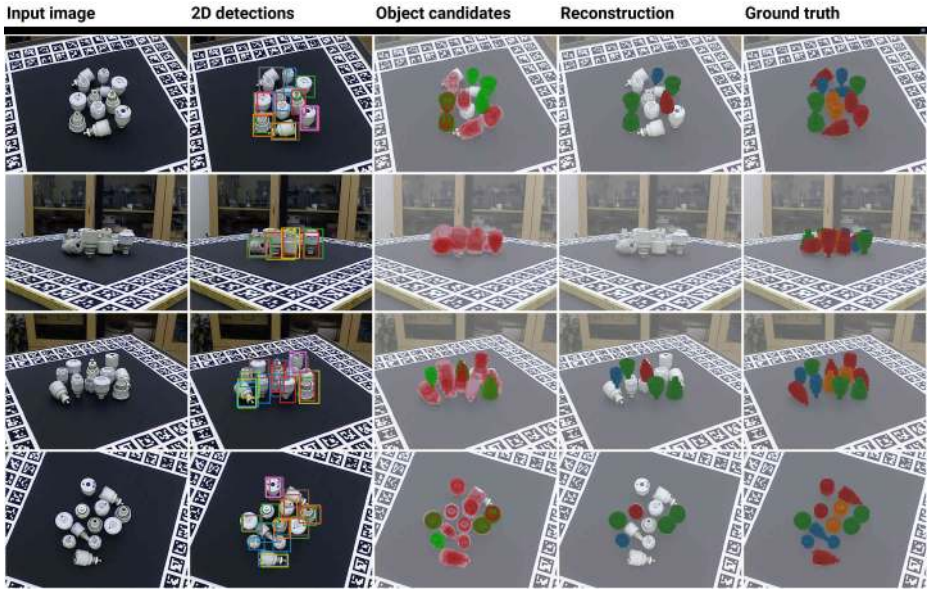
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 91.

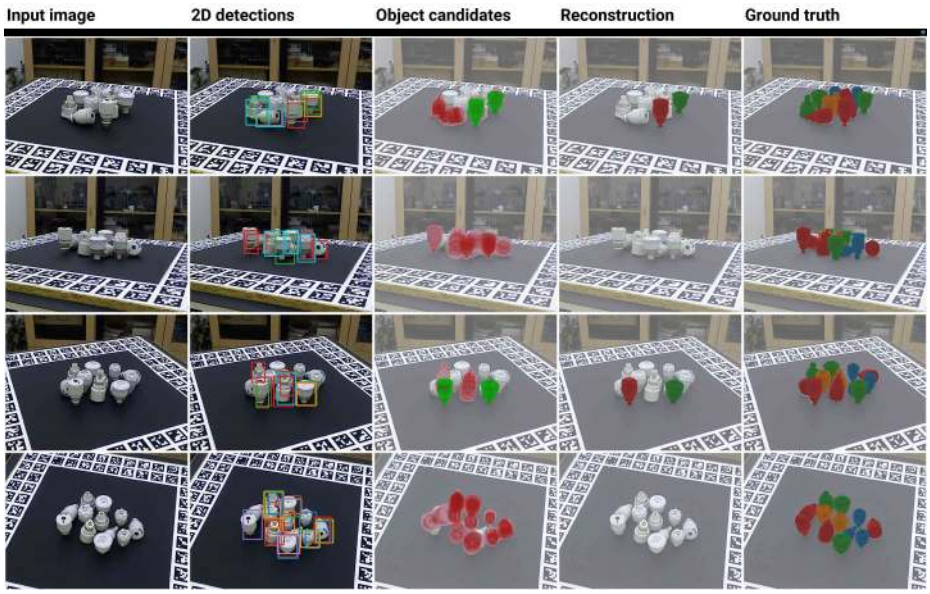
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 92.

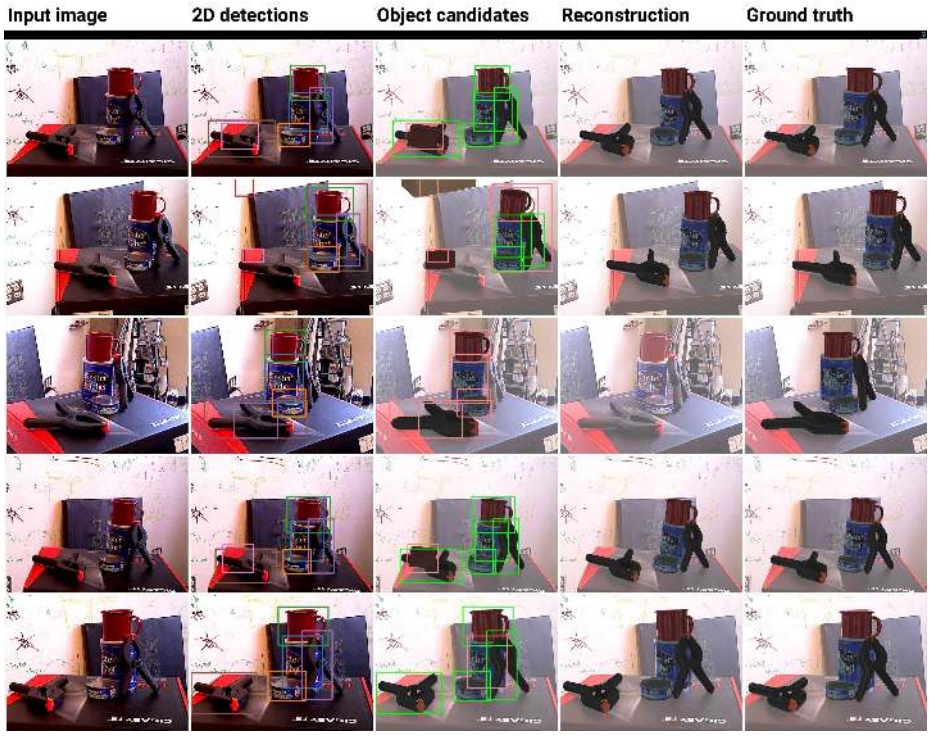
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 93.

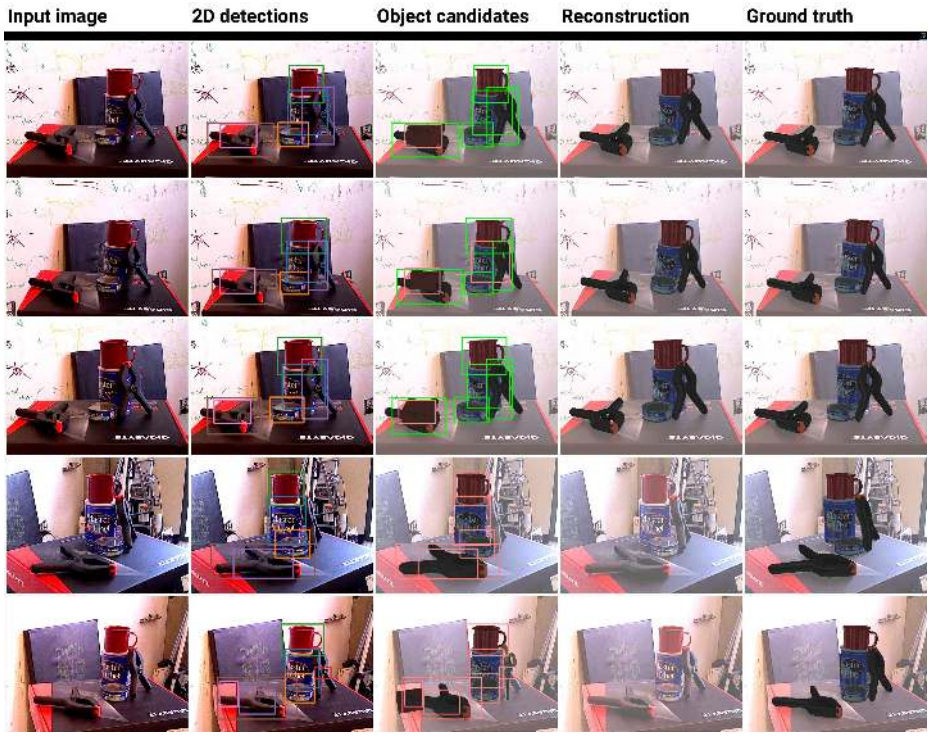
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 94.

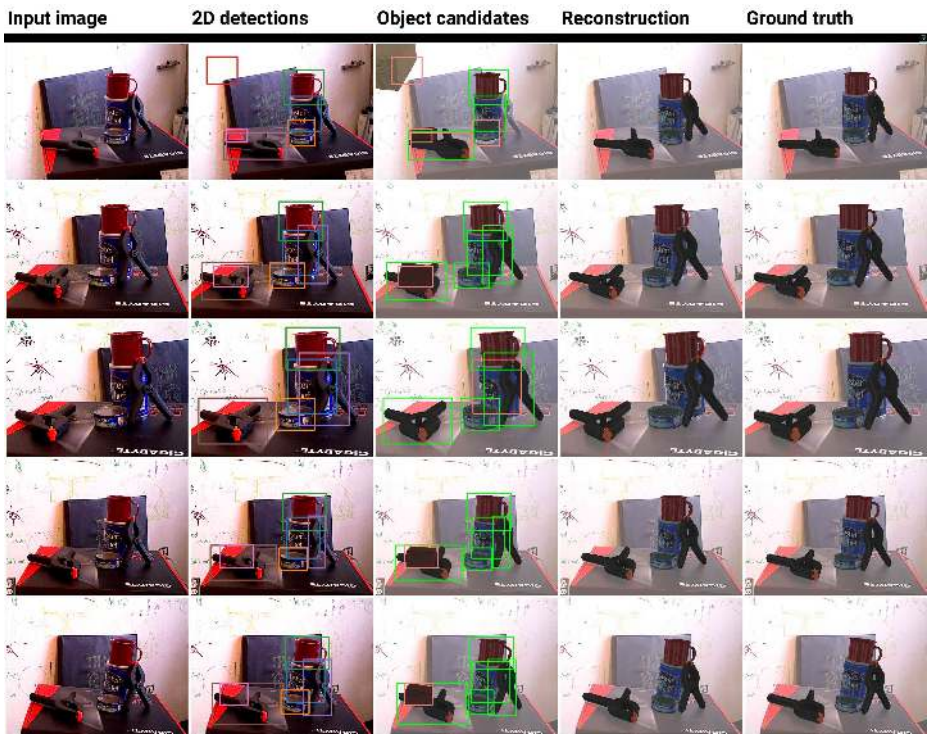
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 95.

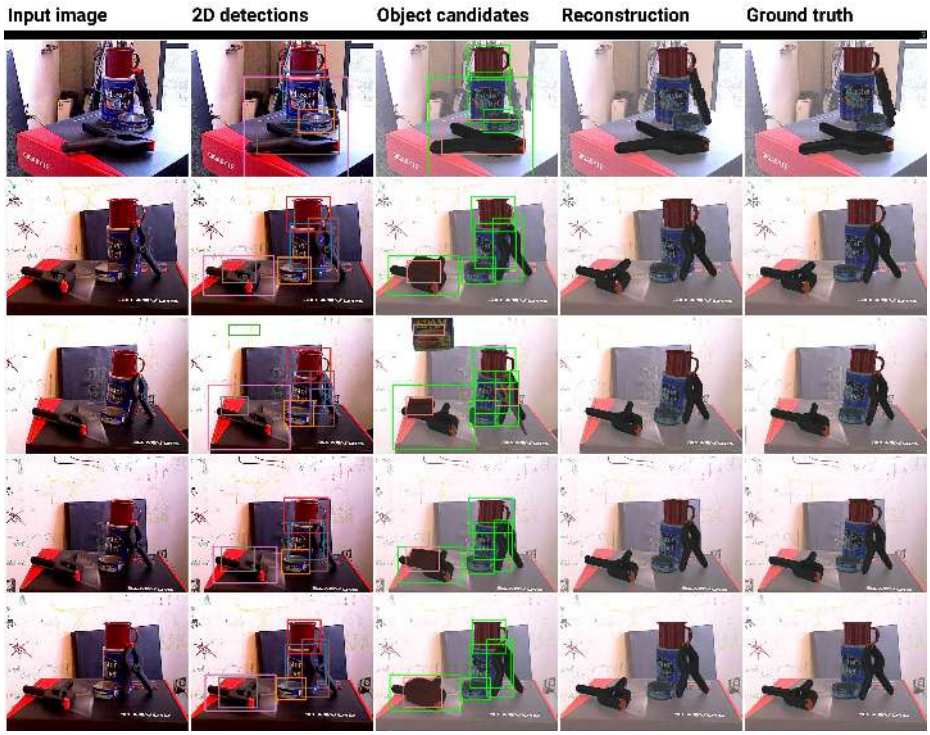
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 96.

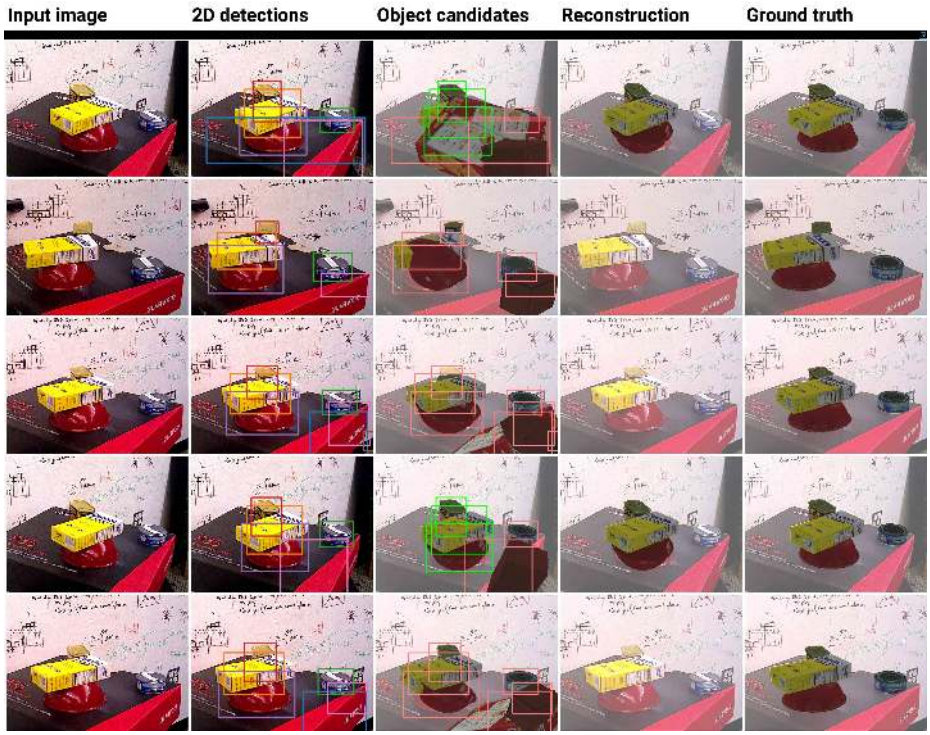
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 97.

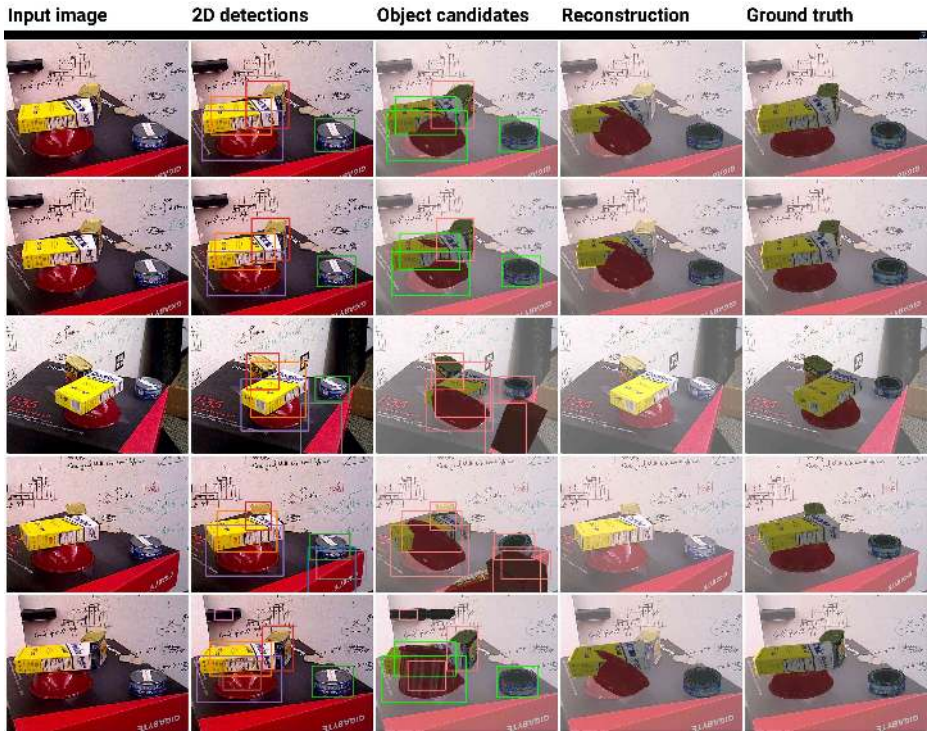
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 98.

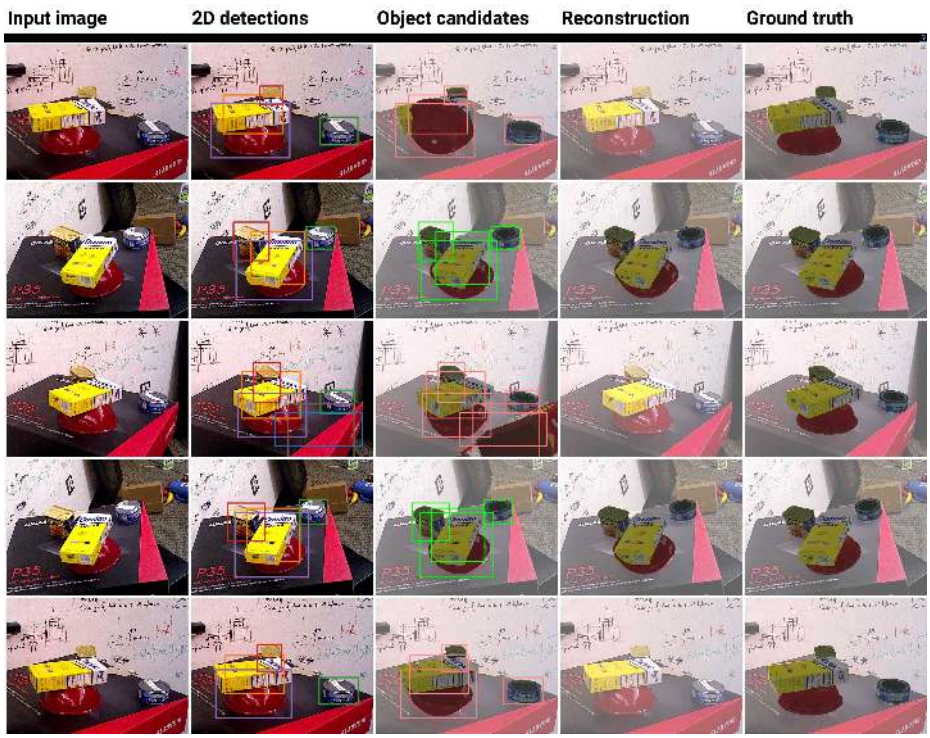
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 99.

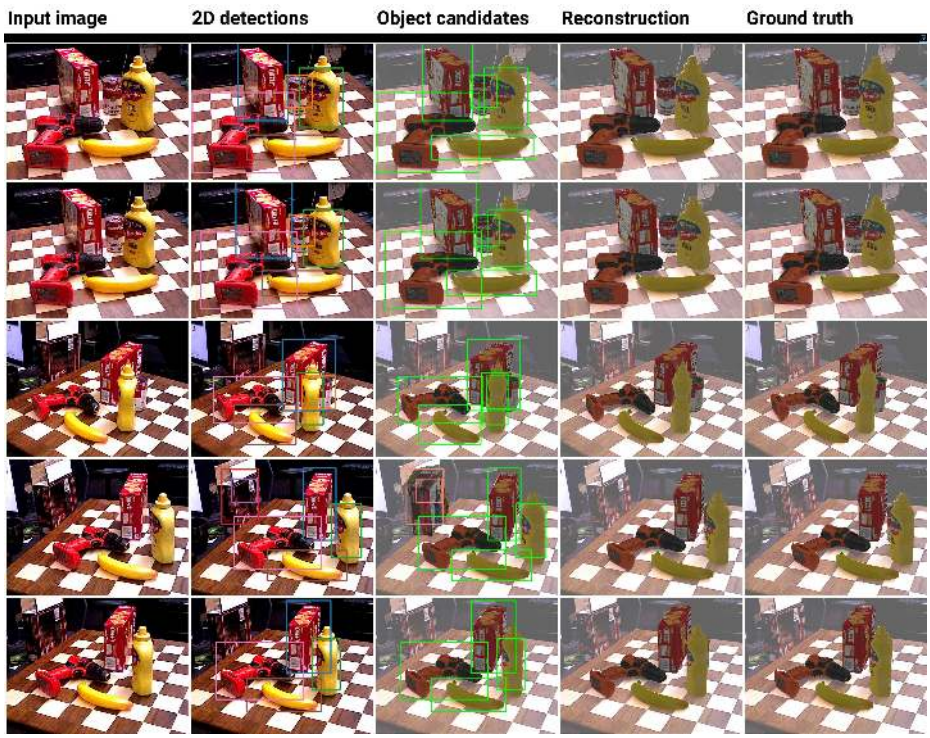
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 100.

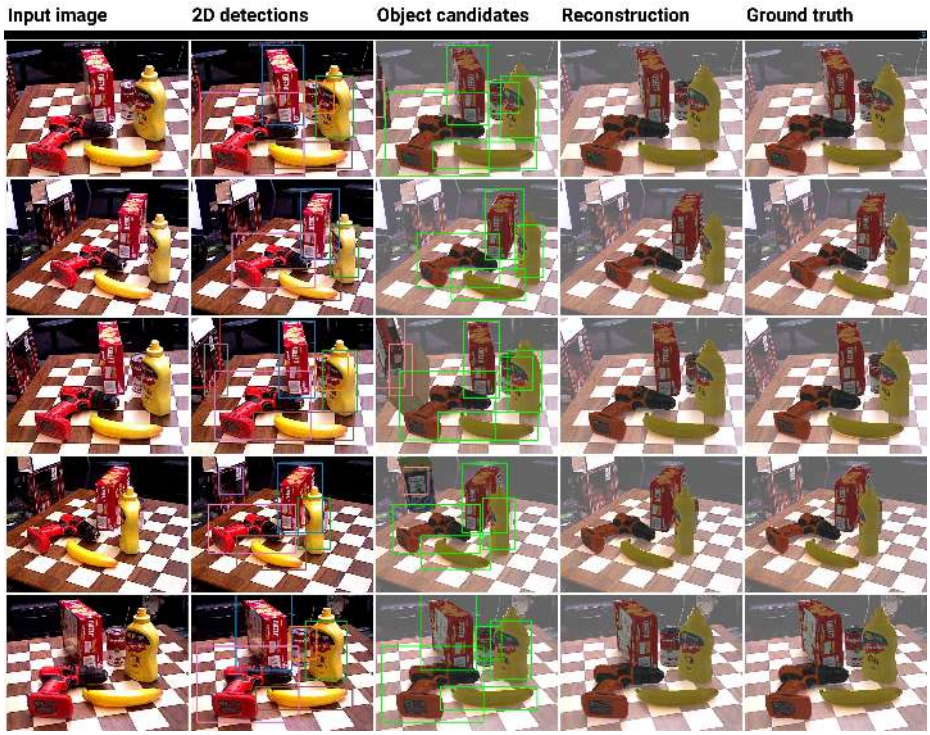
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 101.

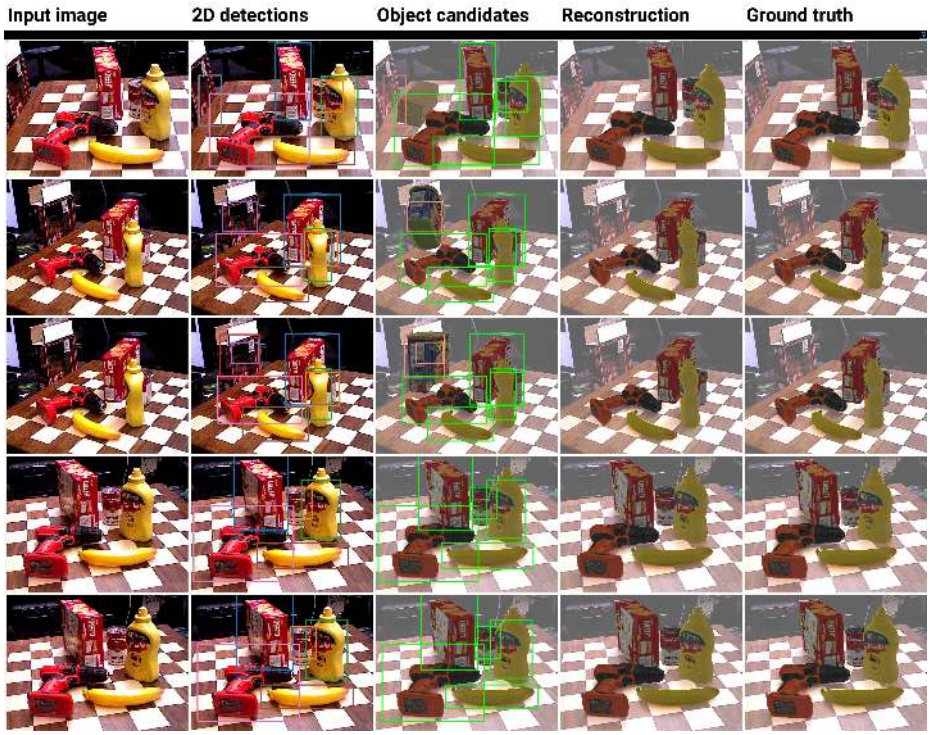
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 102.

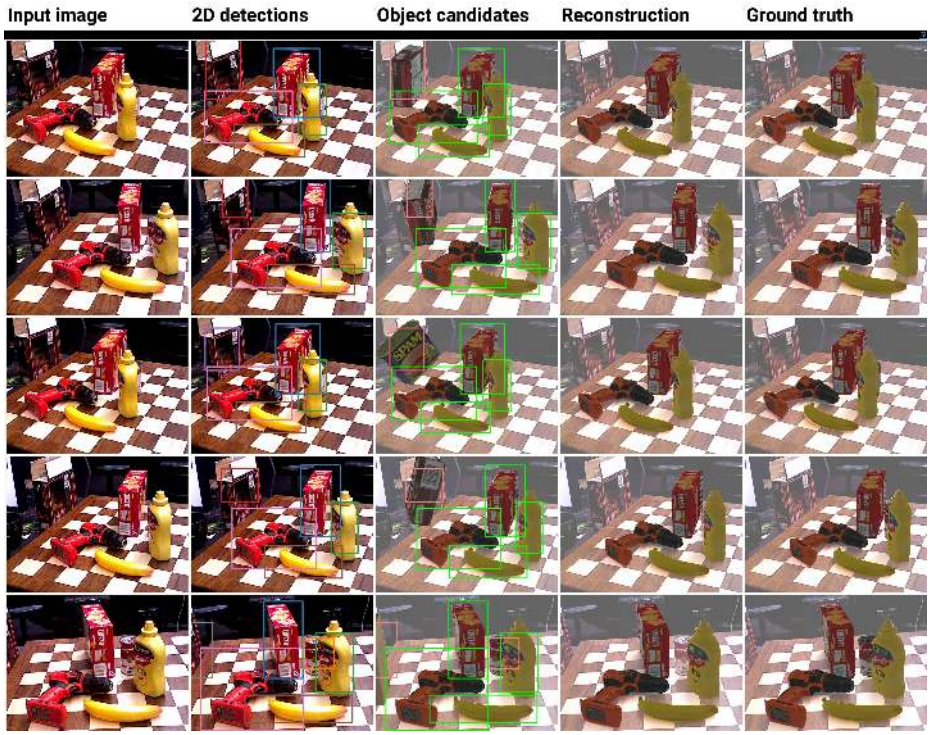
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 103.

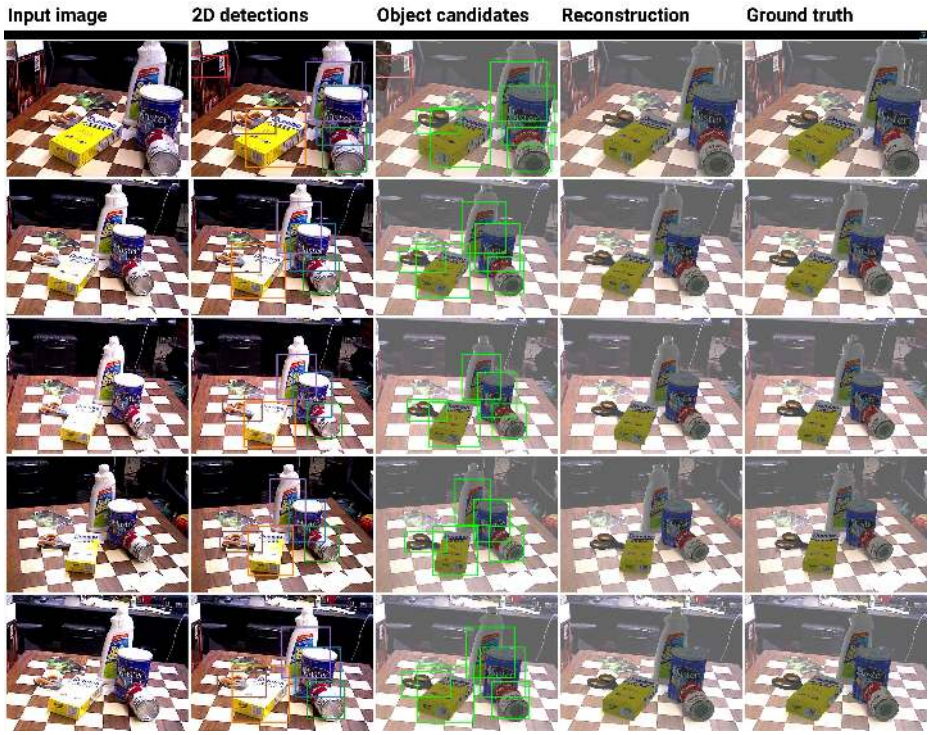
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 104.

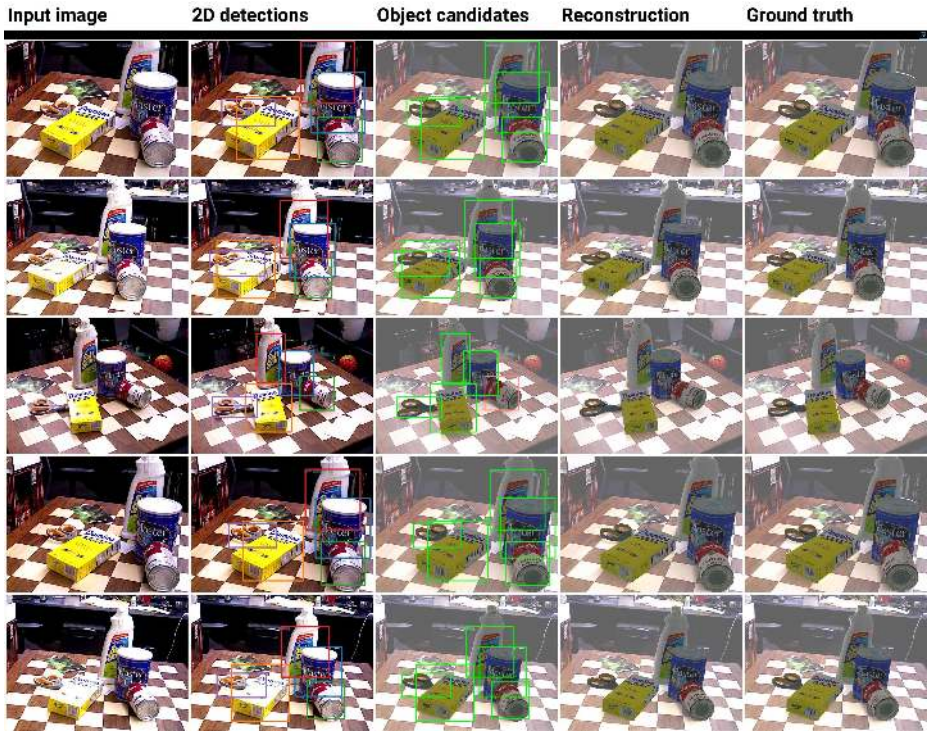
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 105.

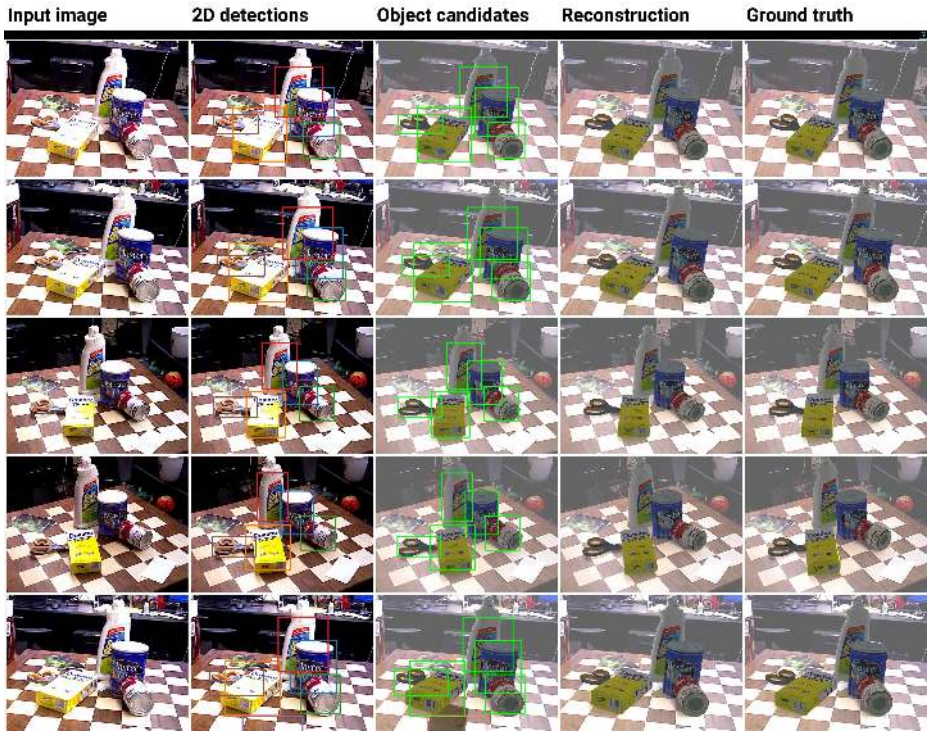
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 106.

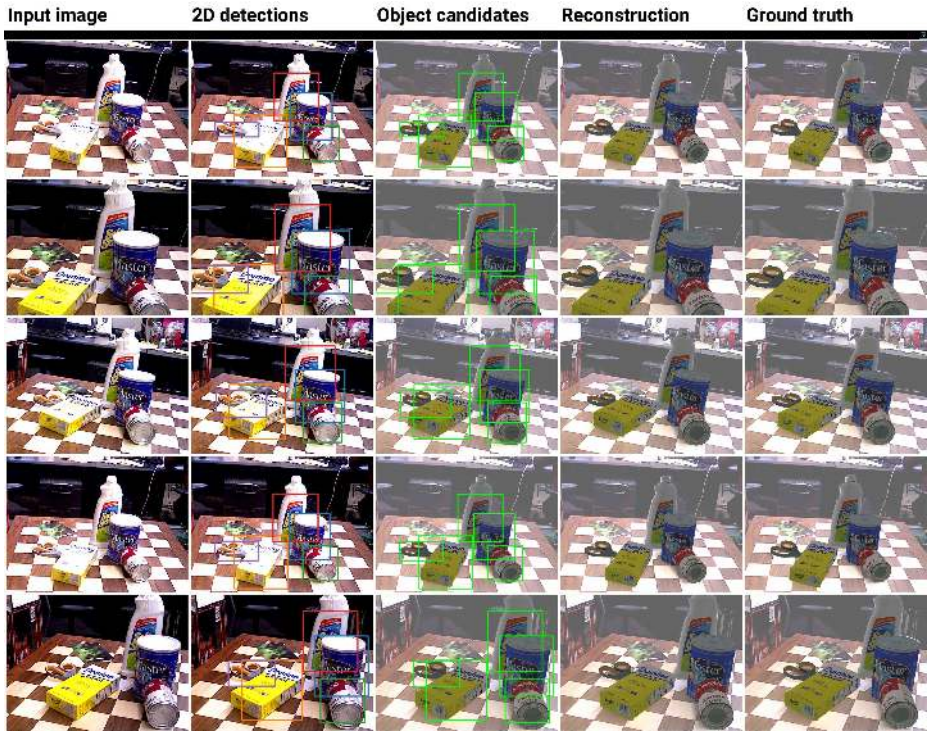
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 107.

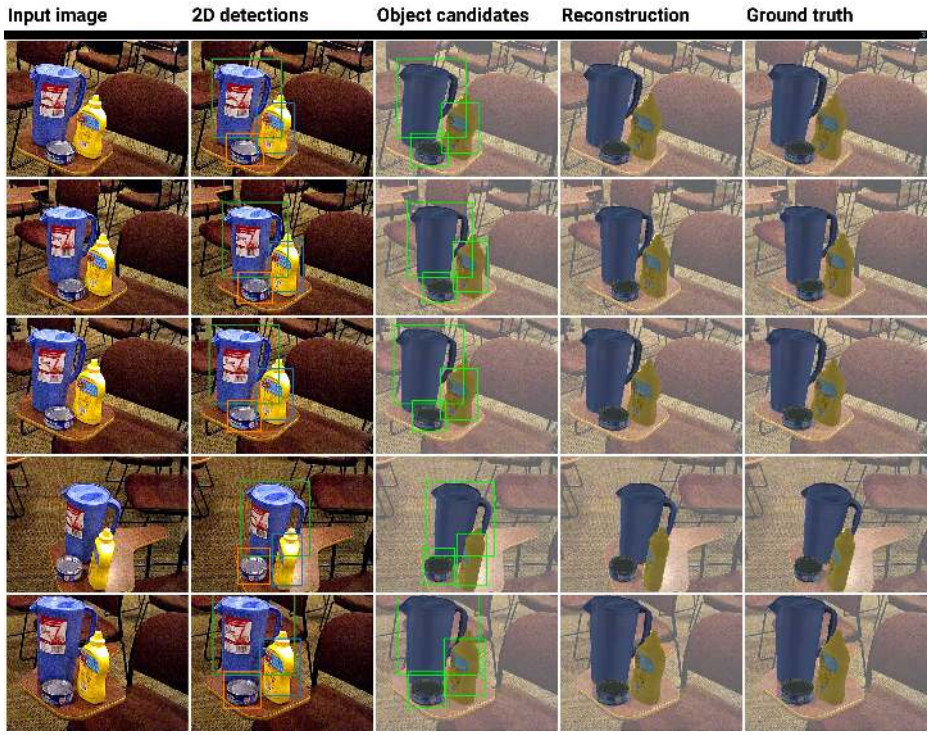
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 108.

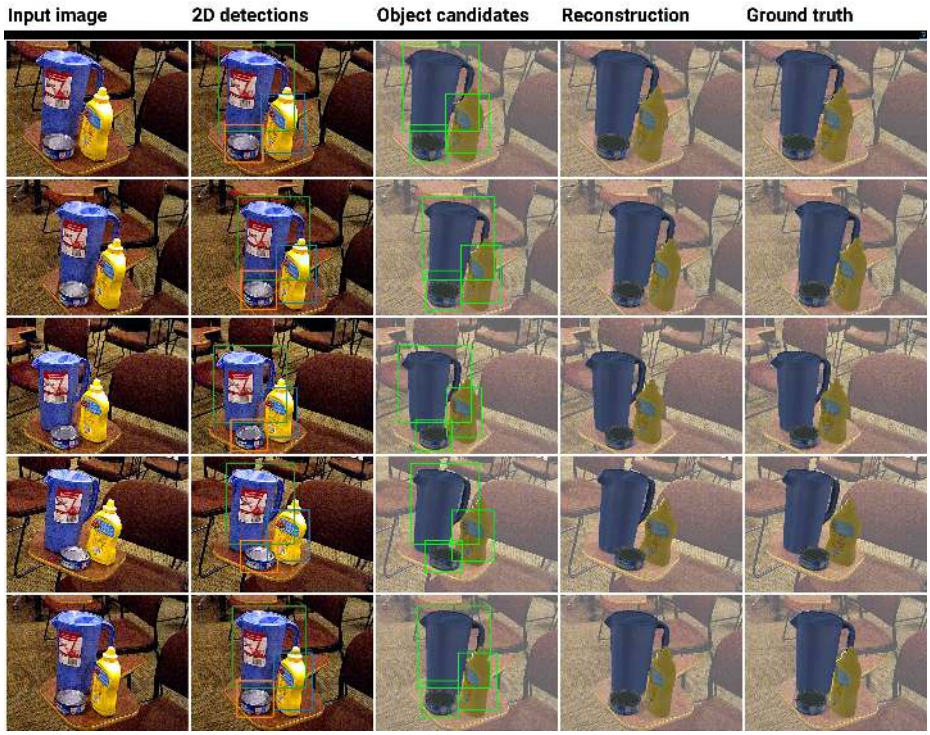
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 109.

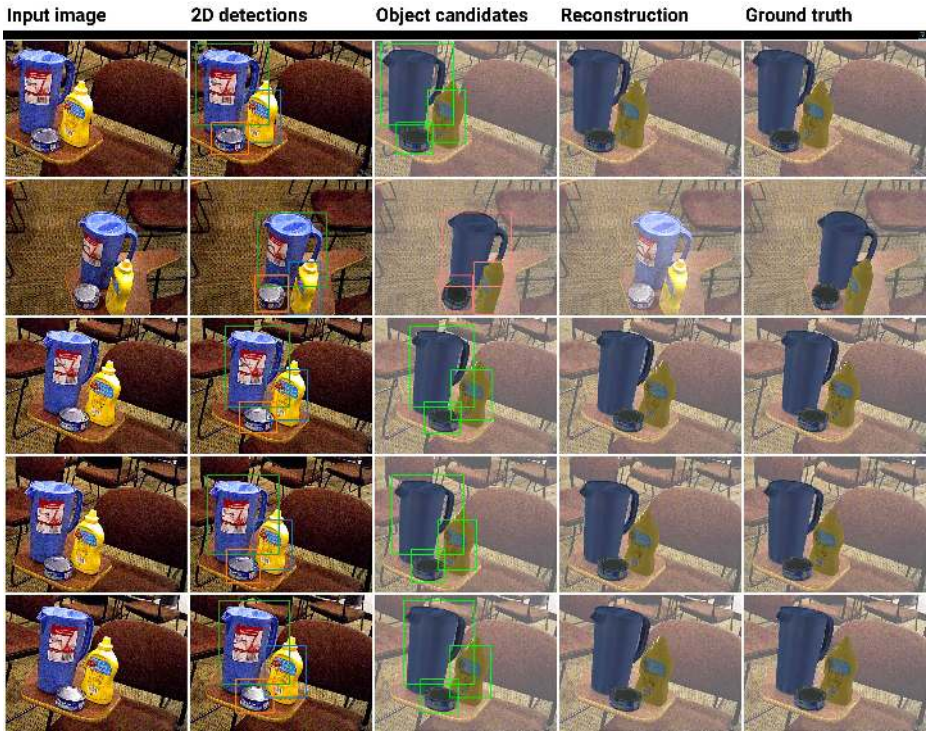
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 110.

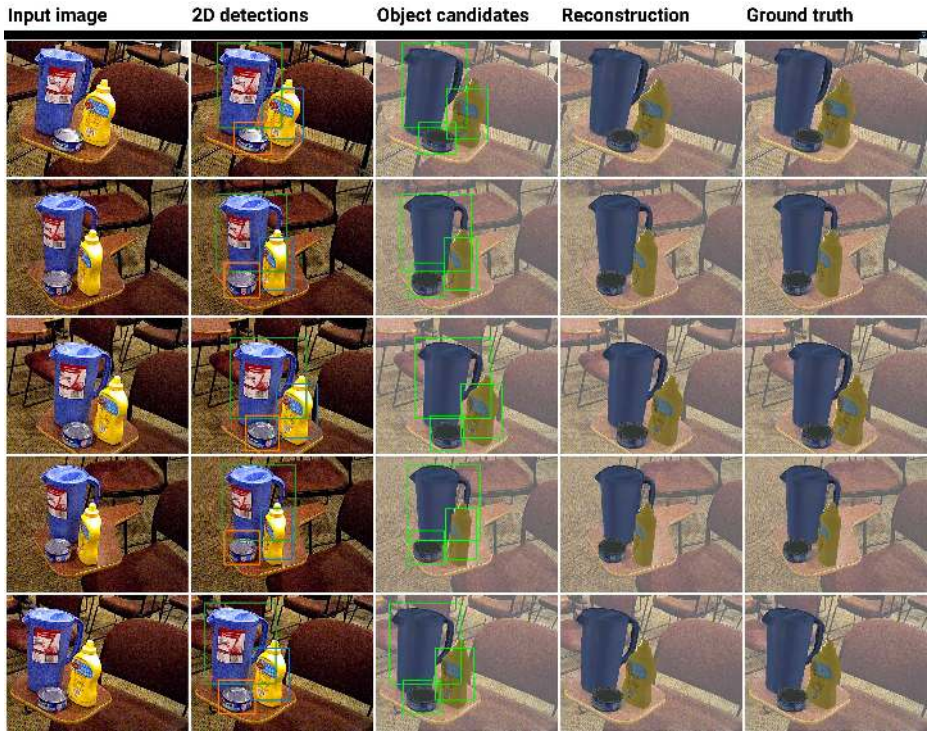
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 111.

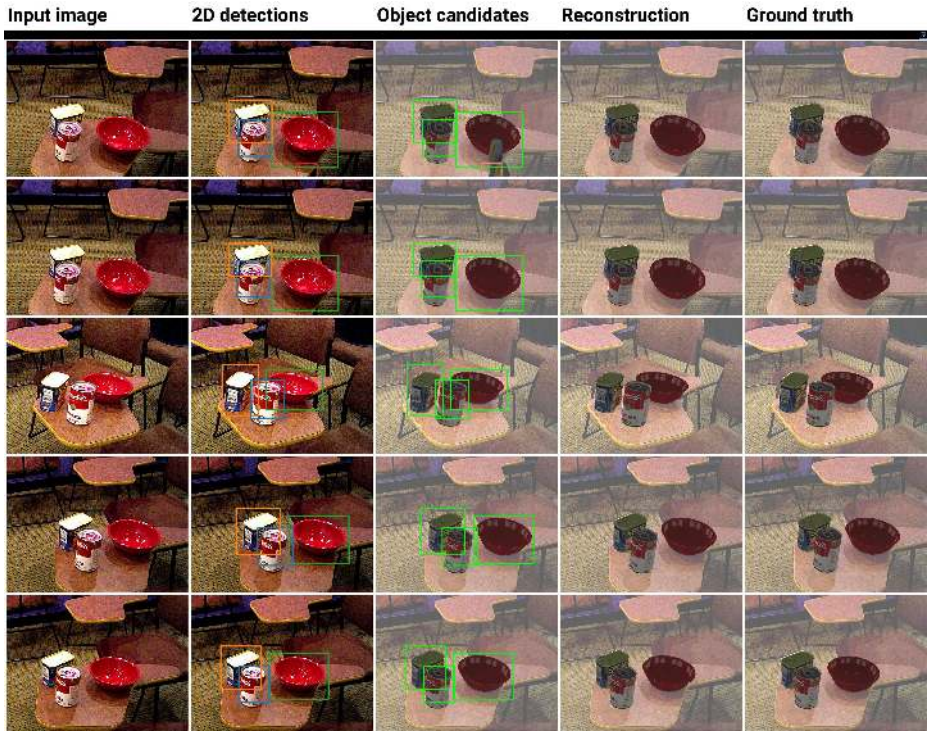
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 112.

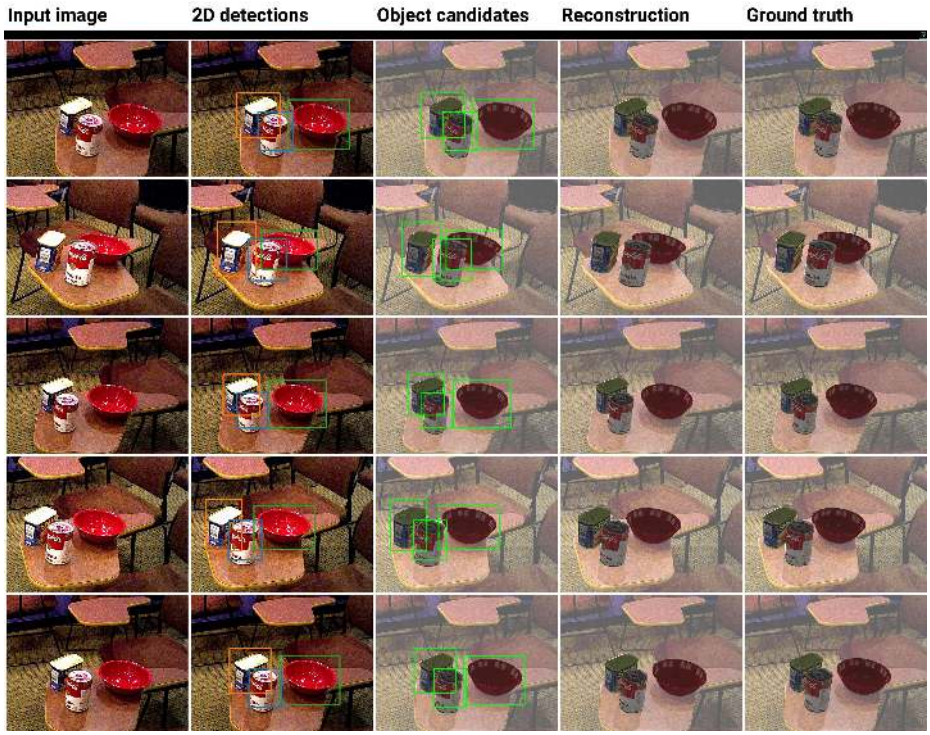
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 113.

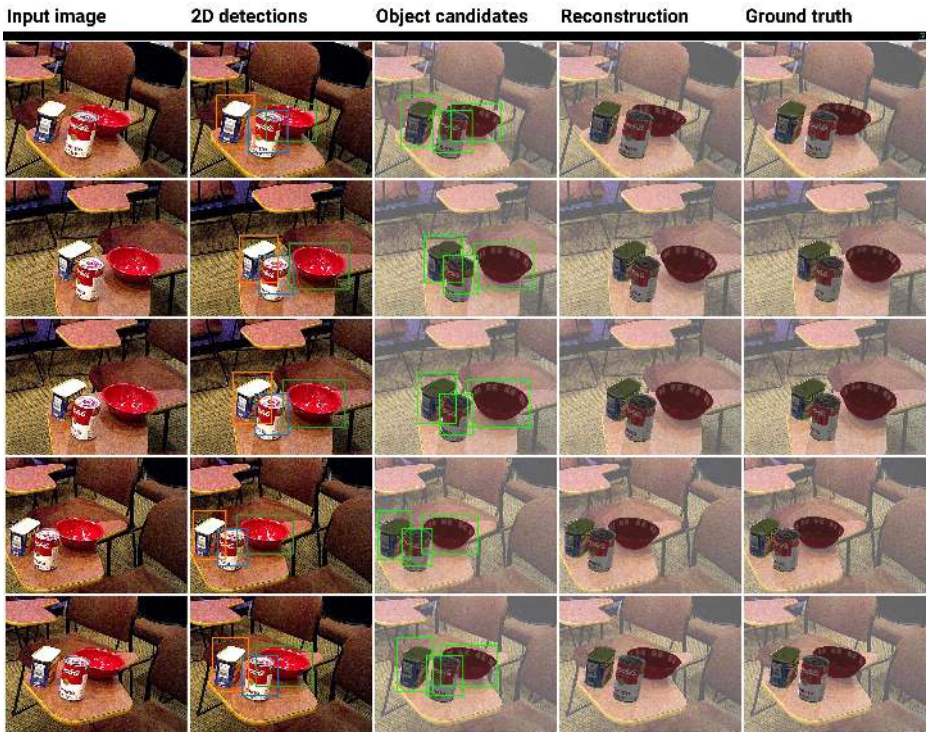
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 114.

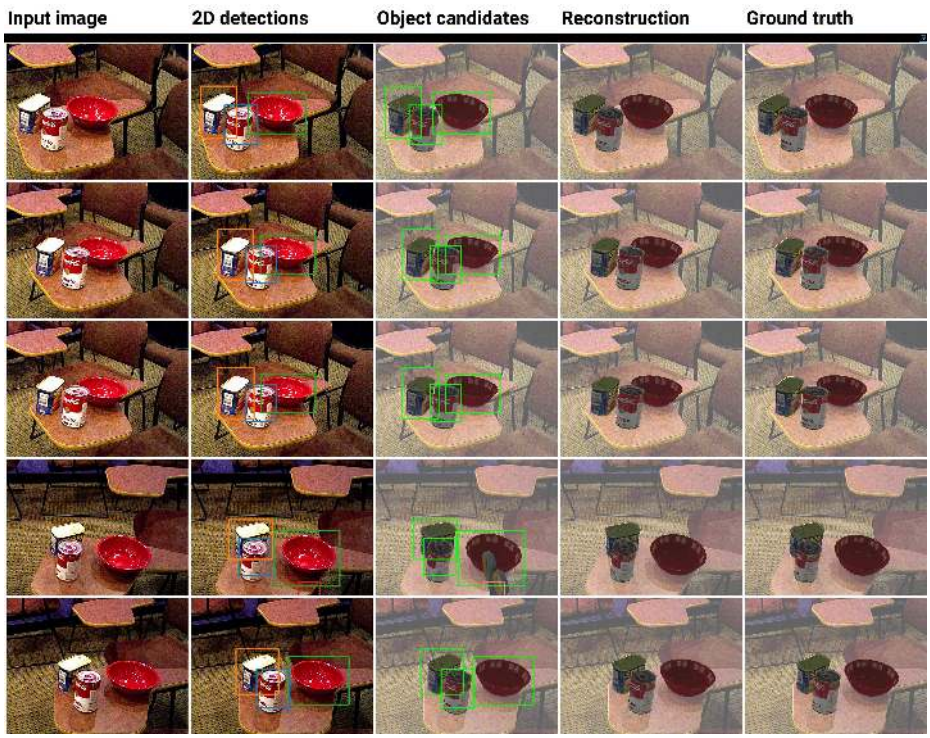
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 115.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 116.

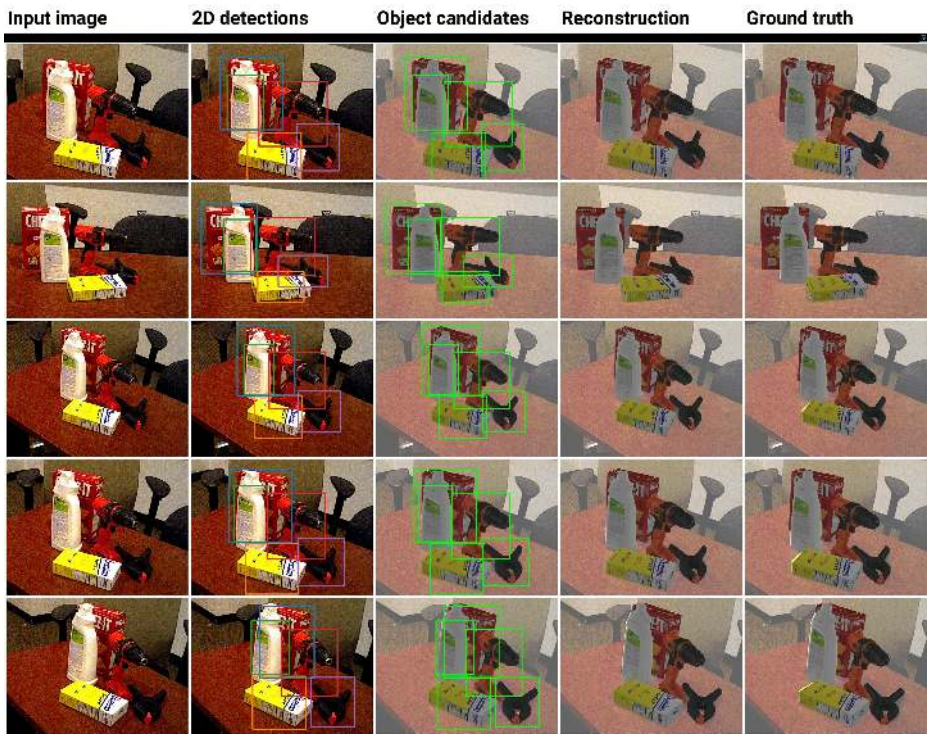
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 117.


Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 118.

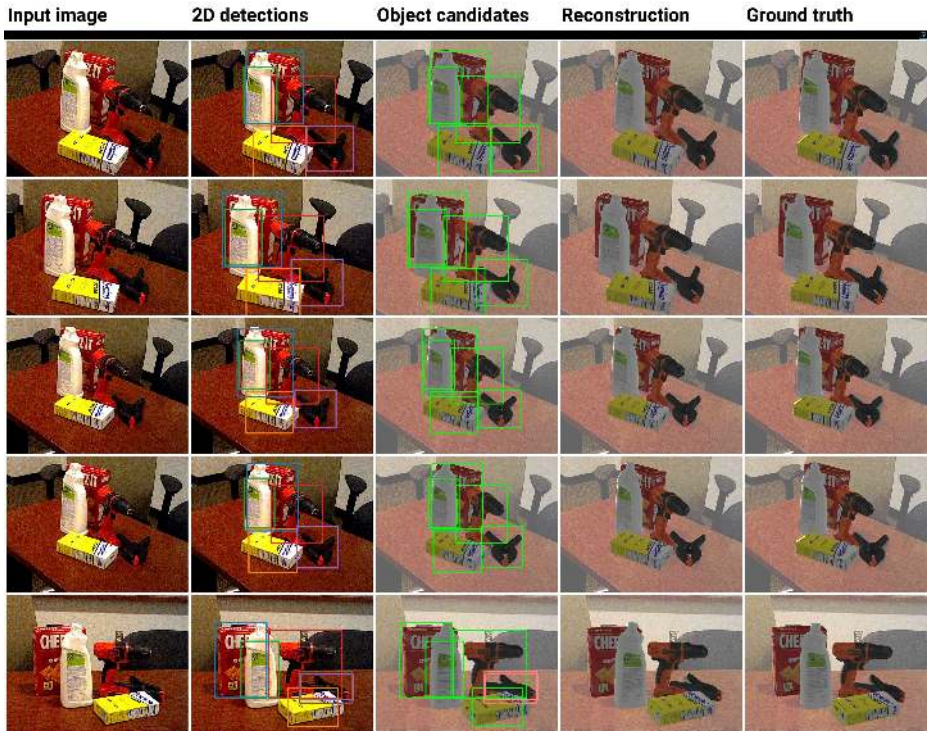
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 119.

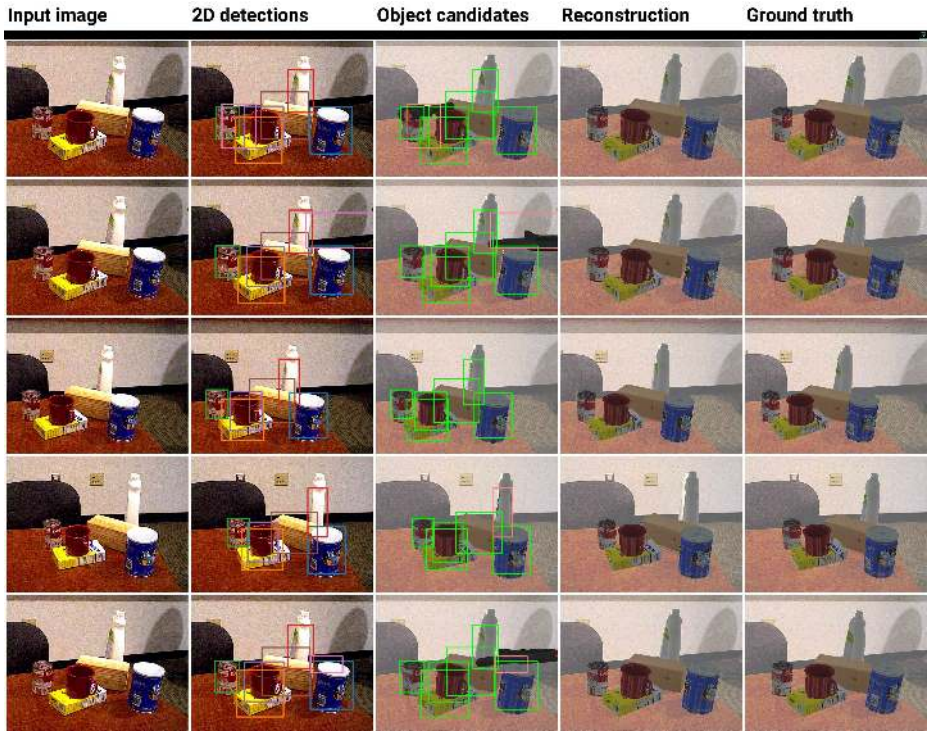
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 120.

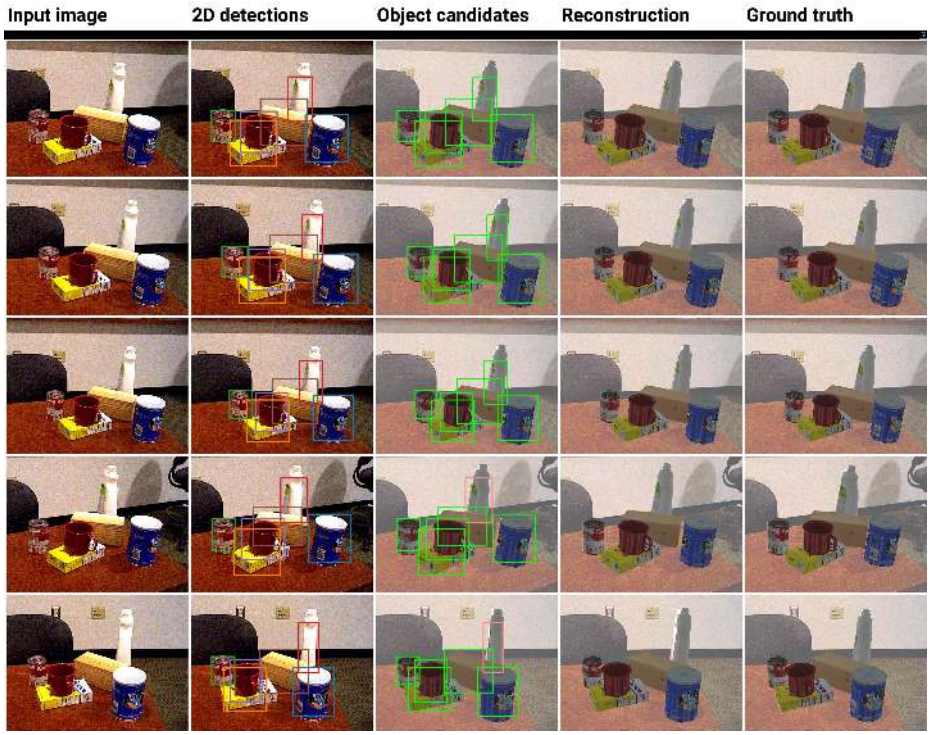
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 121.

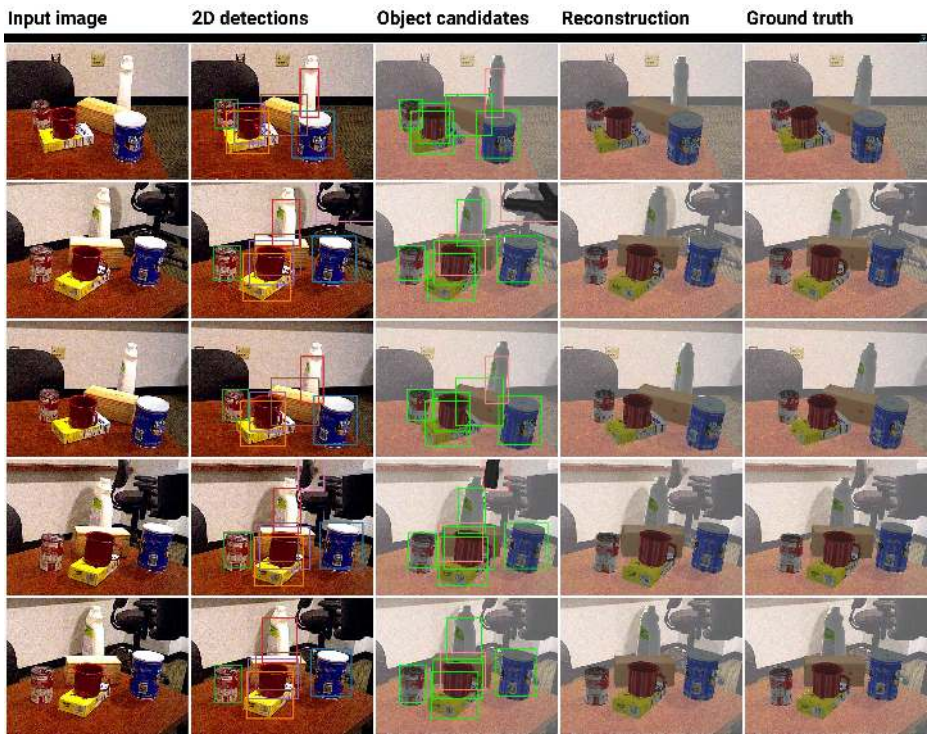
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 122.

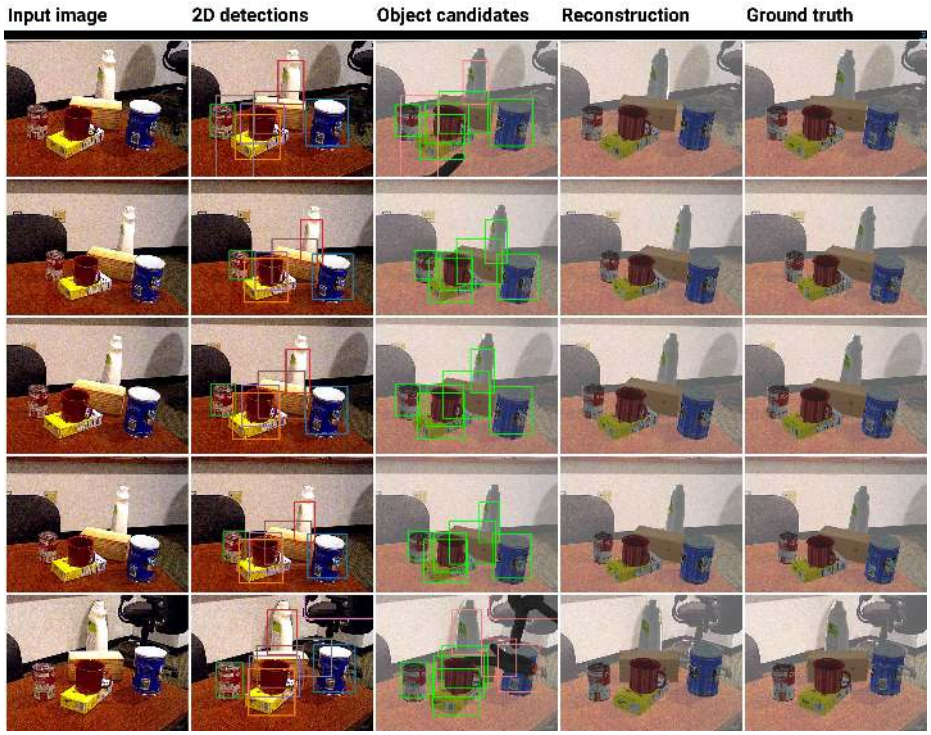
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 123.

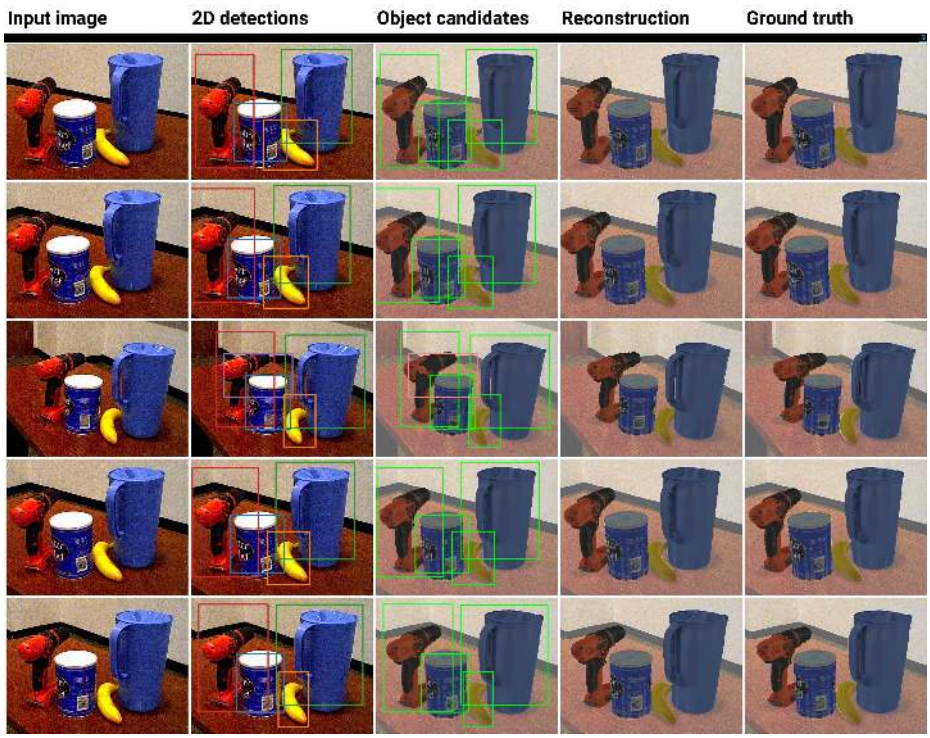
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 124.

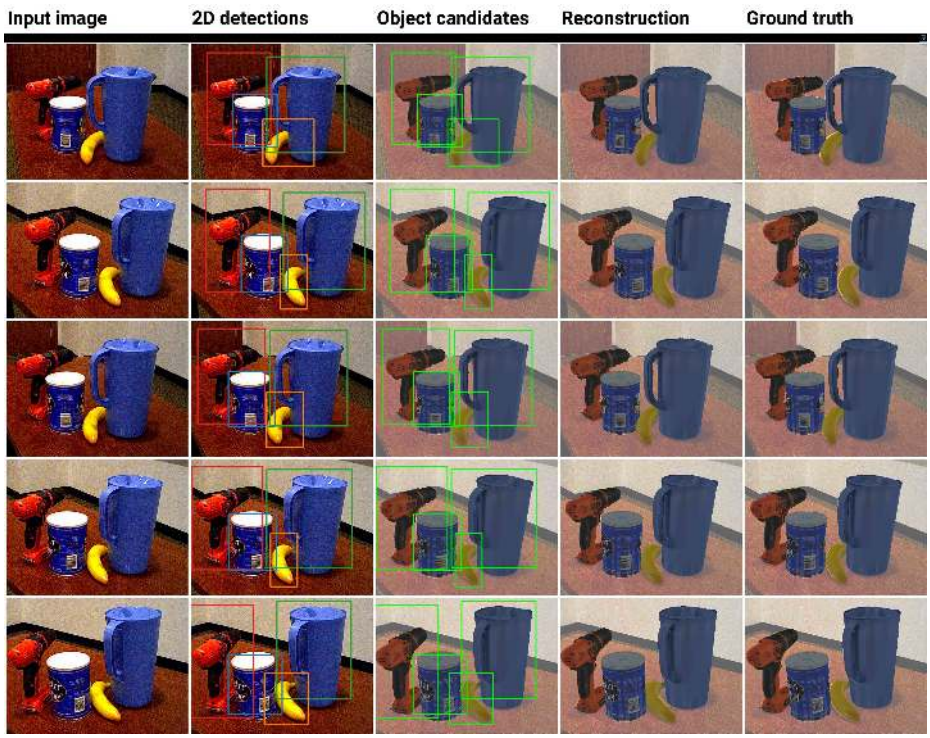
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 125.

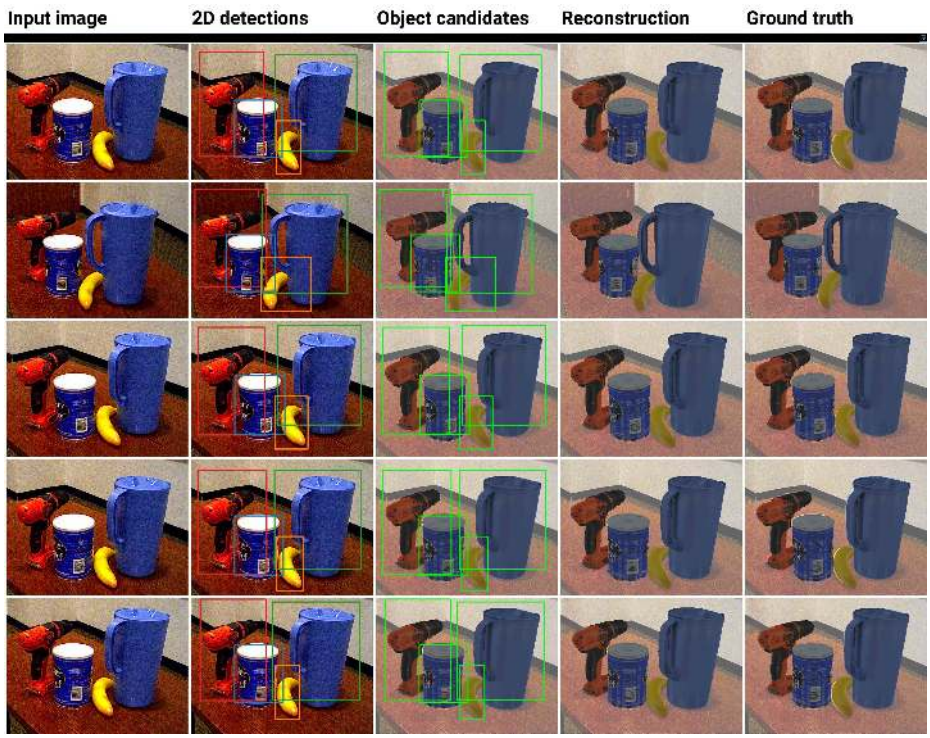
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 126.

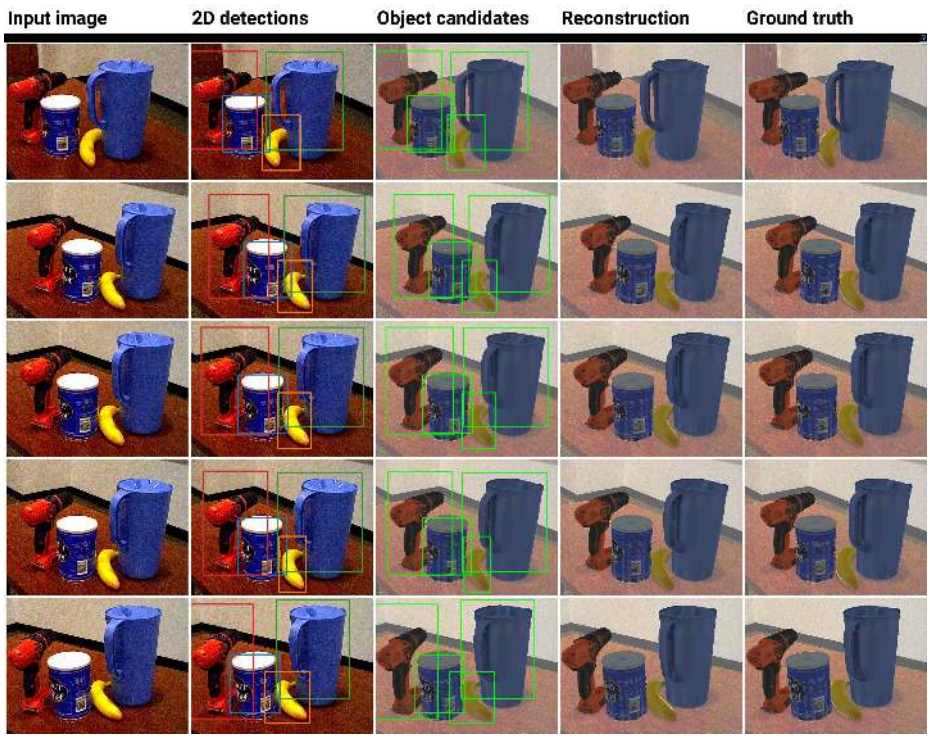
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 127.

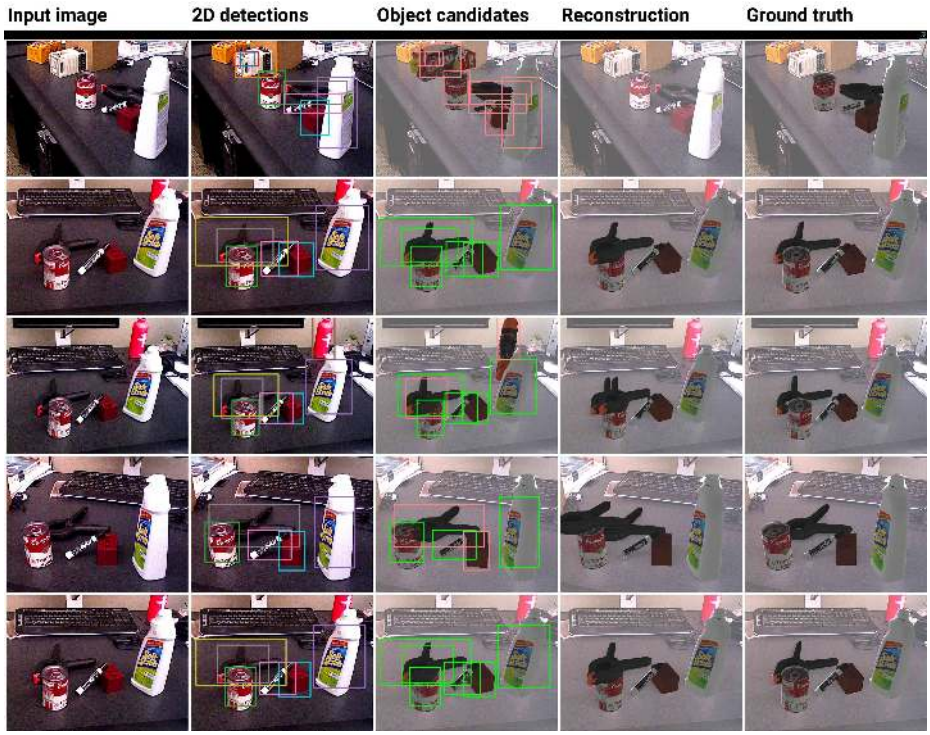
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 128.

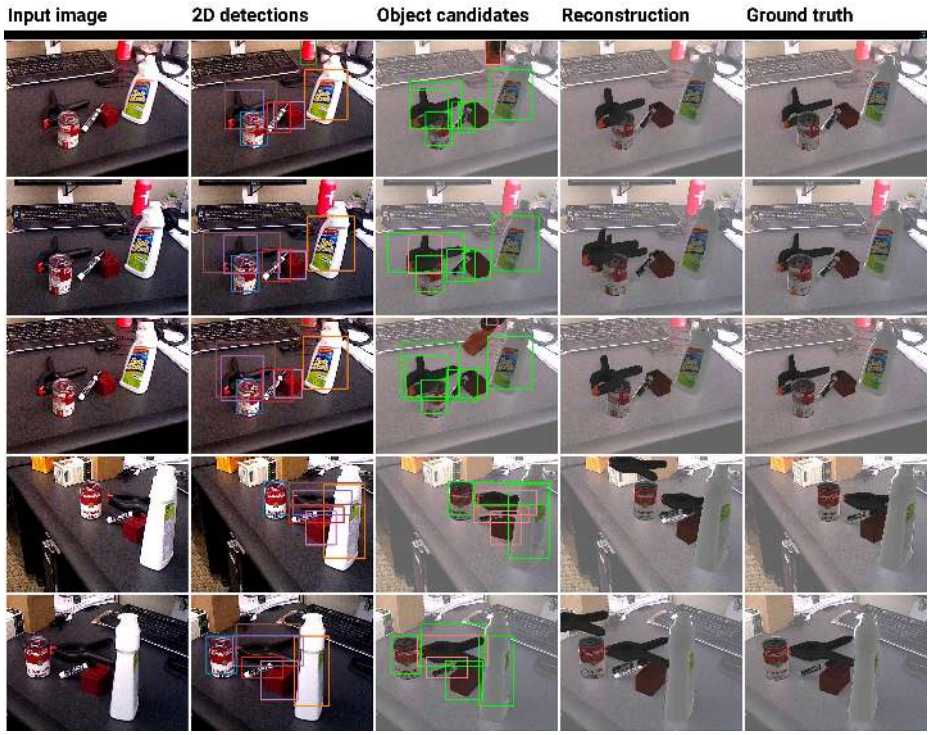
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 129.

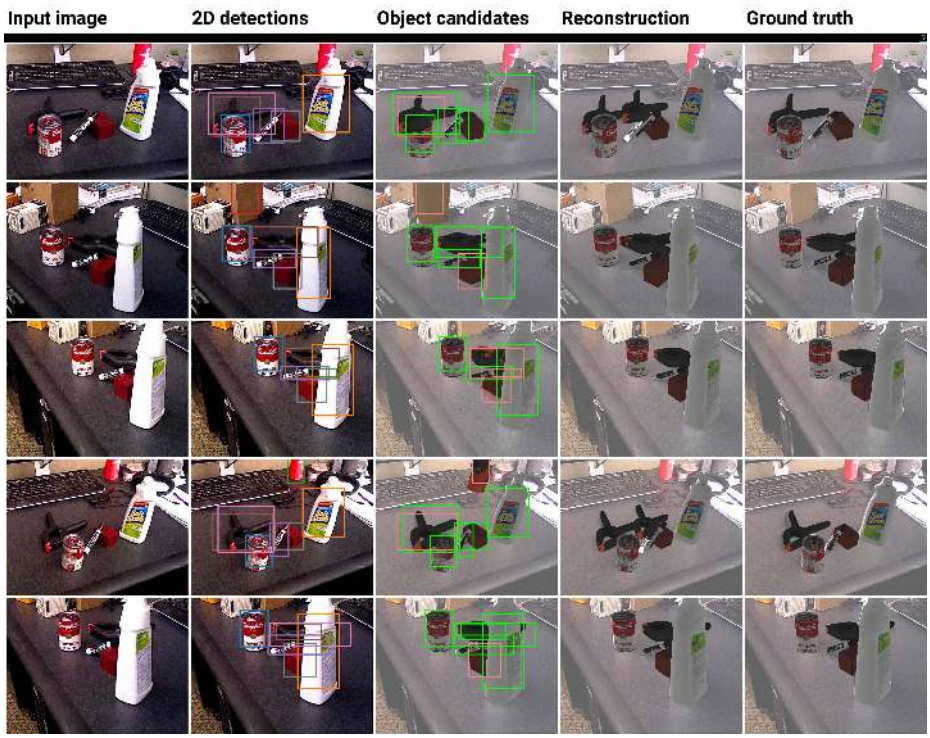
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 130.

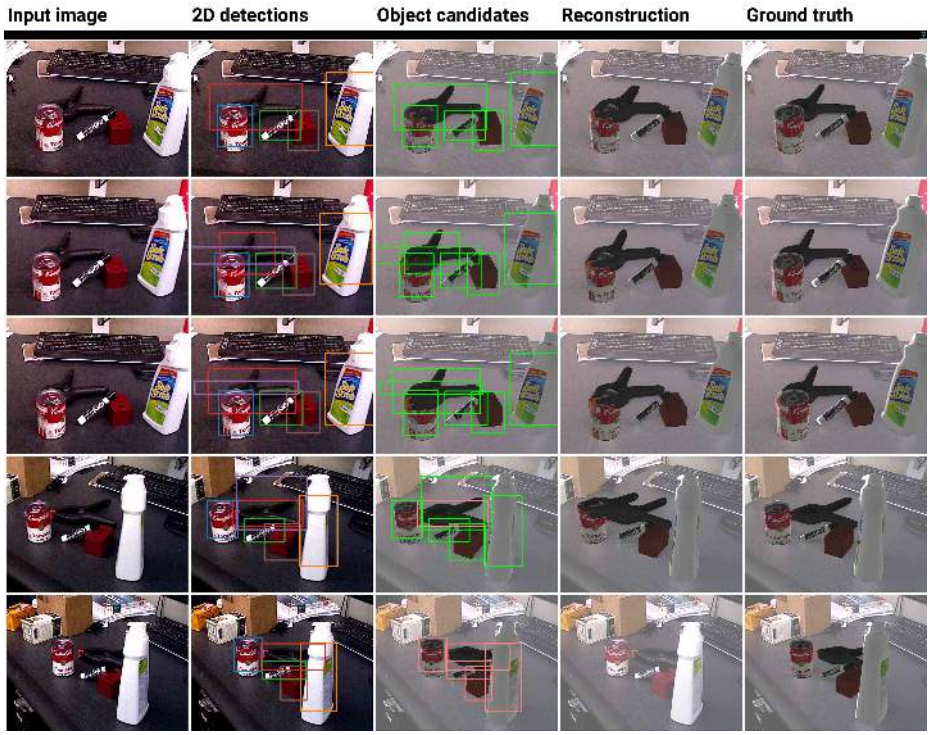
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 131.

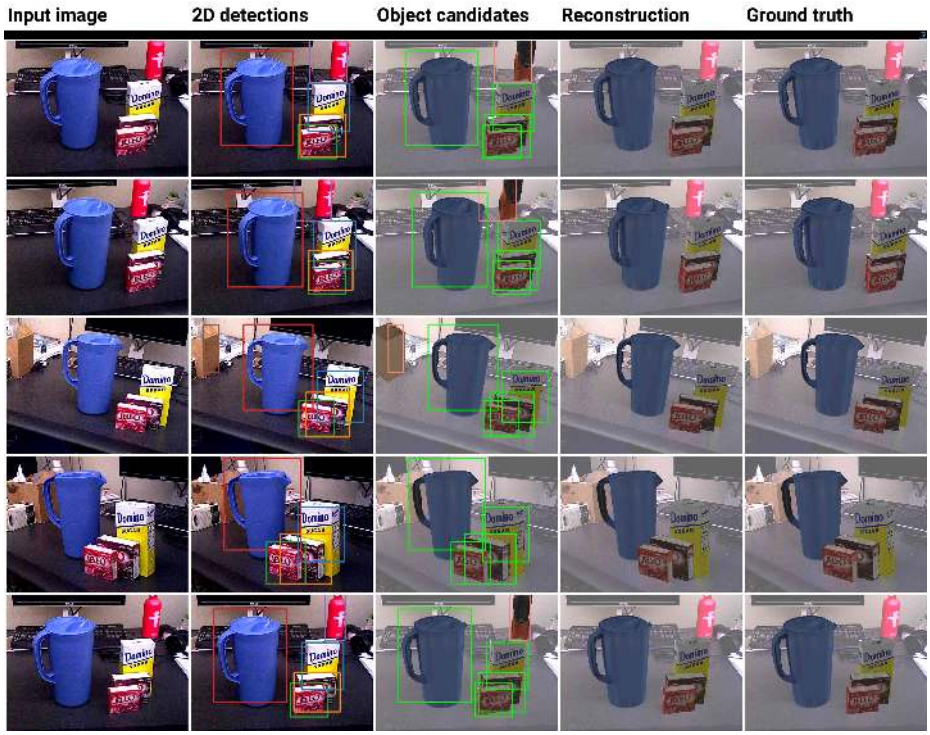
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 132.

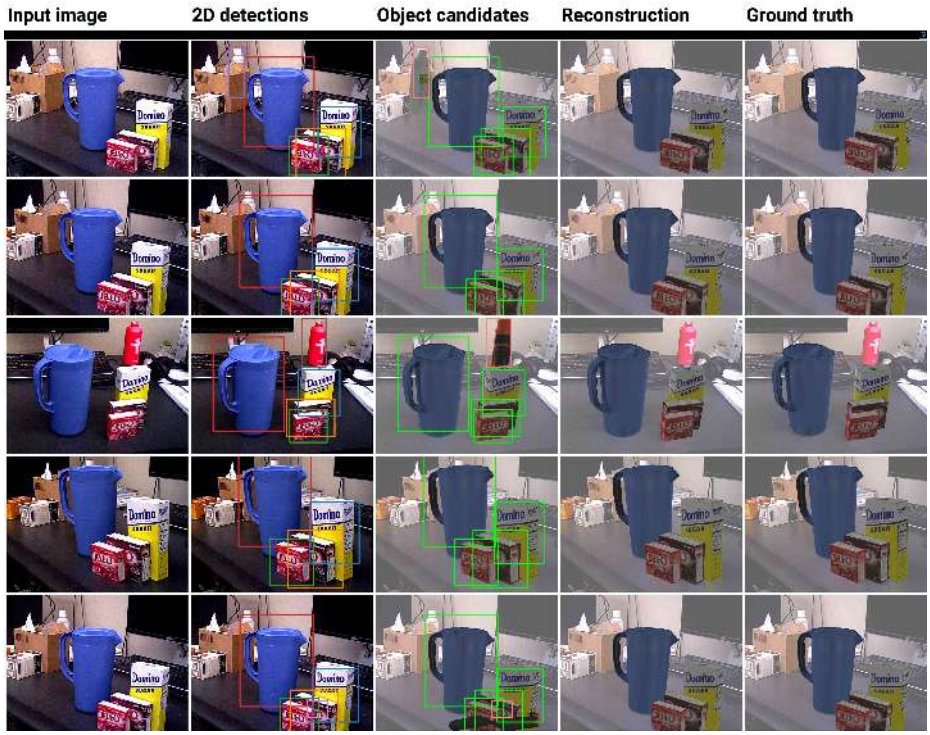
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 133.

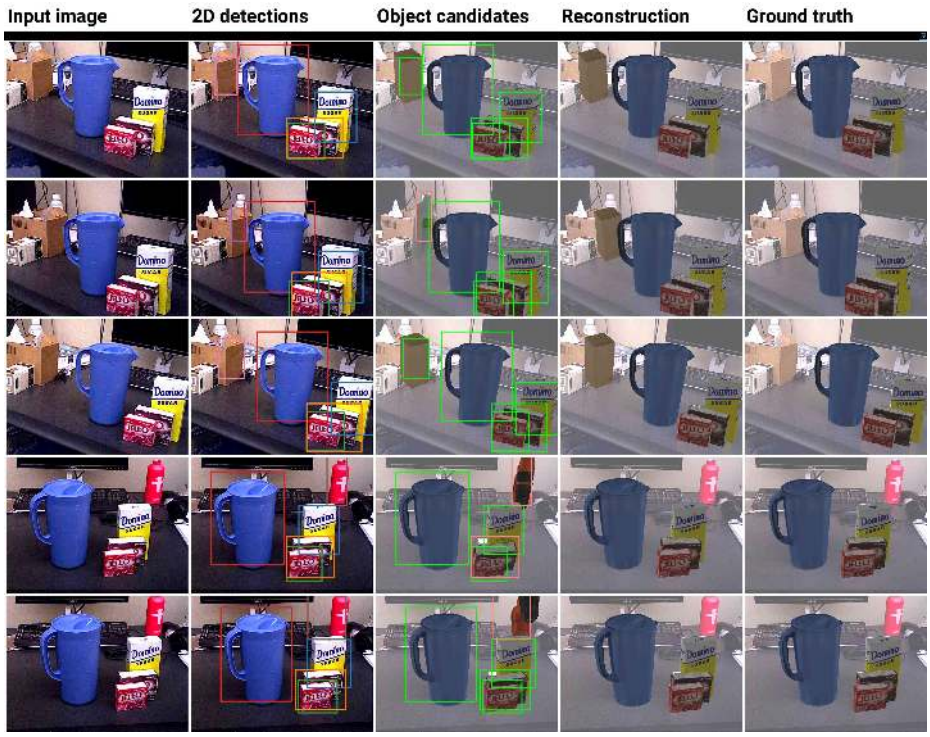
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 134.

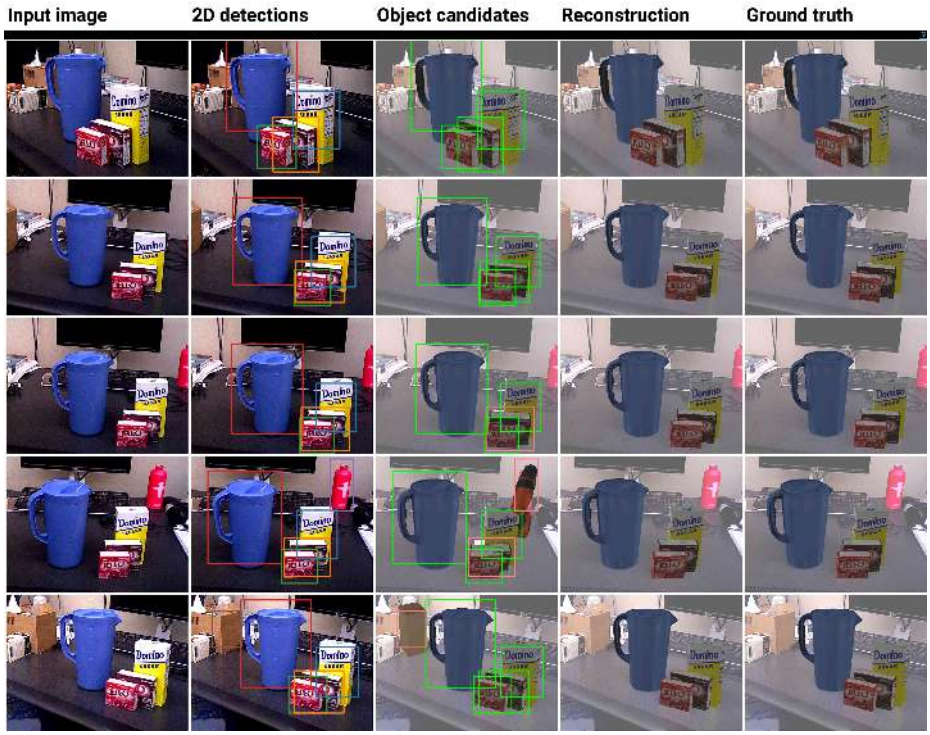
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 135.

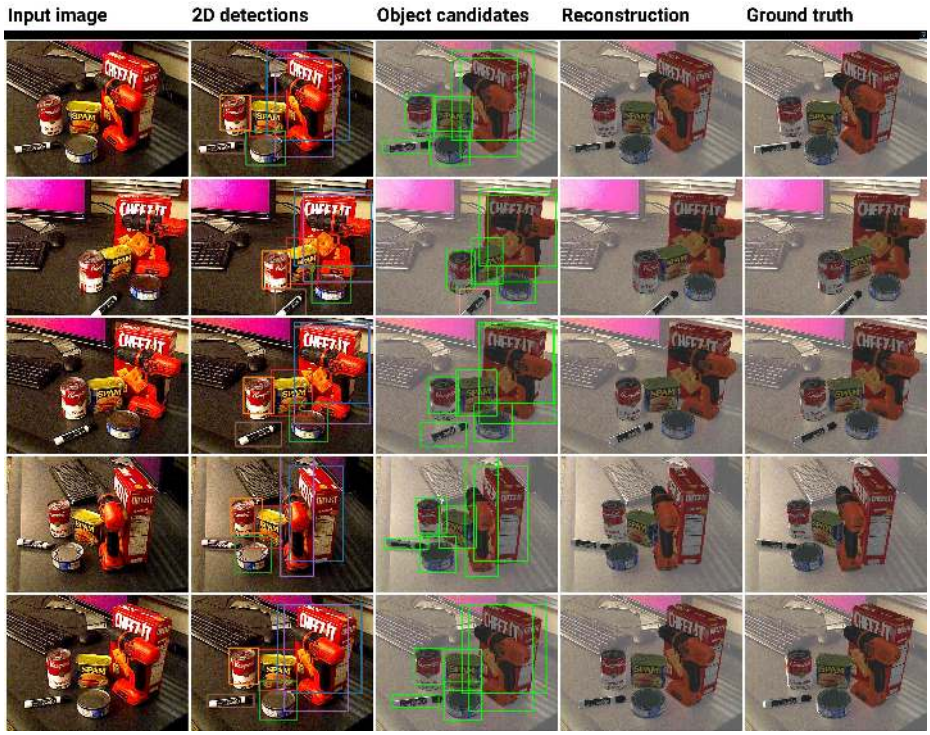
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 136.

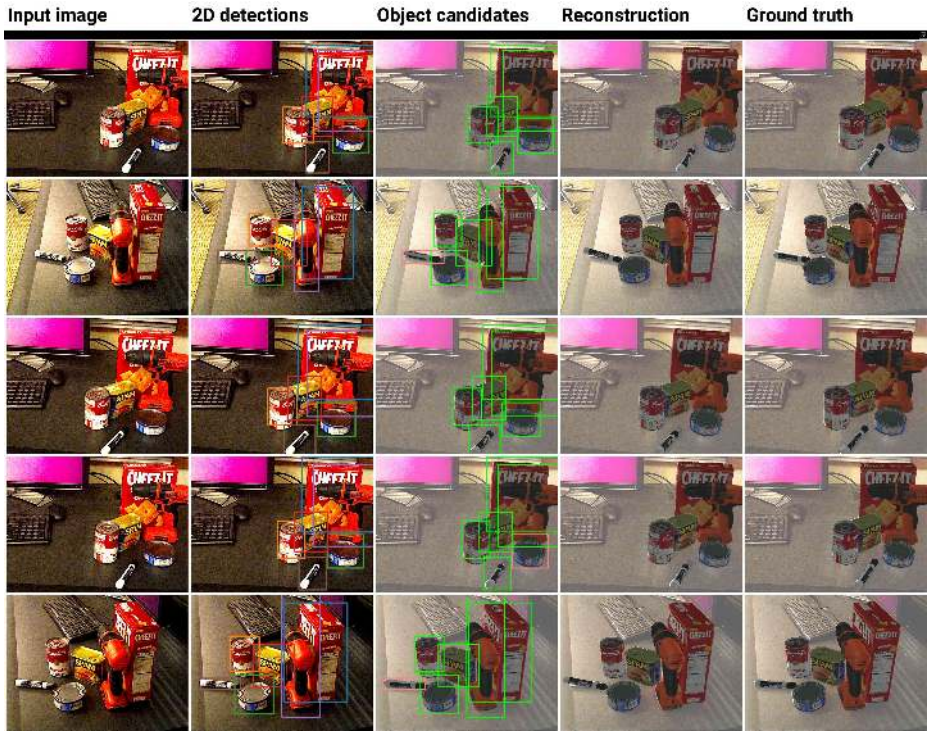
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 137.

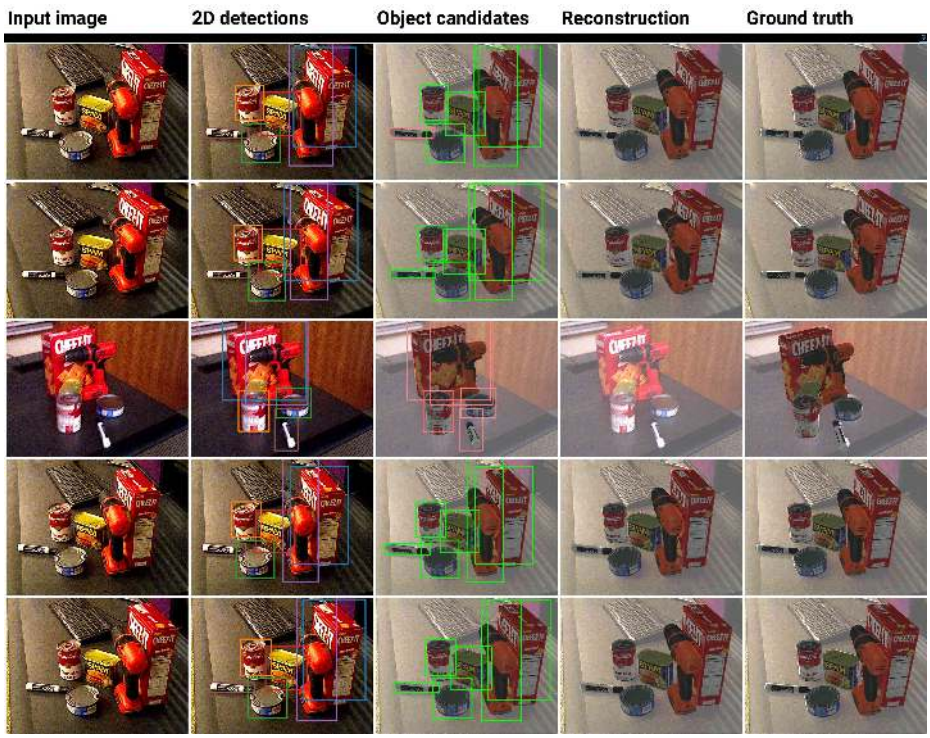
Input image
2D detections
Object candidates
Reconstruction
Ground truth

Fig. 138.