Article

The International Journal of Robotics Research
1–22

DOI: 10.1177/0278364919846551
journals.sagepub.com/home/ijr
SAGE

Physics-based scene-level reasoning for object pose estimation in clutter

Chaitanya Mitash, Abdeslam Boularias and Kostas Bekris

Abstract

This paper focuses on vision-based pose estimation for multiple rigid objects placed in clutter, especially in cases involving occlusions and objects resting on each other. Progress has been achieved recently in object recognition given advancements in deep learning. Nevertheless, such tools typically require a large amount of training data and significant manual effort to label objects. This limits their applicability in robotics, where solutions must scale to a large number of objects and variety of conditions. Moreover, the combinatorial nature of the scenes that could arise from the placement of multiple objects is difficult to capture in the training dataset. Thus, the learned models might not produce the desired level of precision required for tasks, such as robotic manipulation. This work proposes an autonomous process for pose estimation that spans from data generation to scene-level reasoning and self-learning. In particular, the proposed framework first generates a labeled dataset for training a convolutional neural network (CNN) for object detection in clutter. These detections are used to guide a scene-level optimization process, which considers the interactions between the different objects present in the clutter to output pose estimates of high precision. Furthermore, confident estimates are used to label online real images from multiple views and re-train the process in a self-learning pipeline. Experimental results indicate that this process is quickly able to identify in cluttered scenes physically consistent object poses that are more precise than those found by reasoning over individual instances of objects. Furthermore, the quality of pose estimates increases over time given the self-learning process.

Keywords

Object detection, 6D pose estimation, robot perception, convolutional neural networks, Monte Carlo tree search, lifelong learning

1. Introduction

A critical capability of a robot is to be able to identify the six-degree-of-freedom (6-DoF) poses of objects in their surroundings so as to be able to manipulate them. Many environments, however, contain cluttered scenes, where objects are placed in complex arrangements, and can only be partially observed from the robot's viewpoint due to occlusions. An example of such a setup exists in current day warehouses, where robots are being deployed for tasks such as picking from bins, packing, and sorting. Recently, deep learning methods, such as those employing convolutional neural networks (CNNs), have become popular for object detection (Redmon et al., 2016; Ren et al., 2015) and pose estimation (Kehl et al., 2017; Xiang et al., 2018), outperforming alternatives in object recognition benchmarks. These desirable results are typically obtained by training CNNs using datasets that involve a very large number of labeled images. However, these datasets need to be collected in a way that captures the intricacies of the environment the robot is deployed in, such as lighting conditions, occlusions, and self-occlusions, in clutter.

The recent Amazon Picking Challenge (APC) (Correll et al., 2016) has reinforced this realization and has led into the development of datasets specifically for the detection of objects inside cluttered shelving units (Rennie et al., 2016; Singh et al., 2014; Zeng et al., 2017). These datasets are created either with human annotation or by incrementally placing one object in the scene and using foreground masking. An increasingly popular approach to avoid manual labeling is to use synthetic datasets generated by rendering 3D CAD models of objects with different viewpoints. Synthetic datasets have been used to train CNNs for object

Computer Science Department, Rutgers University, Piscataway, NJ, USA

Corresponding author:
Chaitanya Mitash, Computer Science Department, Rutgers University, 617 Bowser Road, Piscataway, NJ 08854, USA.
Email: cm1074@rutgers.edu

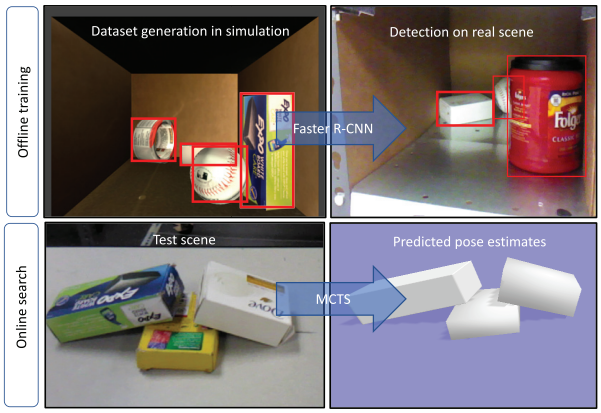

Fig. 1. Top: A physically-realistic dataset is generated and is used to train a CNN for object detection. Bottom: In the online phase, pose estimation is performed via a Monte Carlo tree search process, which performs scene-level reasoning to output physically realistic pose estimates of higher accuracy.

detection (Peng et al., 2015) and viewpoint estimation (Su et al., 2015). One major challenge in using synthetic data is the inherent difference between virtual training examples and real testing data. For this reason, there is considerable interest in studying the impact of texture, lighting, and shape to address this disparity (Sun and Saenko, 2014). Another issue with synthetic images generated from rendering engines is that they display objects in poses that are not necessarily physically realistic. Moreover, occlusions are usually treated in a rather naive manner, i.e., by applying cropping, or pasting rectangular patches, which again results in unrealistic scenes (Movshovitz-Attias et al., 2016; Peng et al., 2015; Su et al., 2015).

This motivates the development of a synthetic dataset that could capture the known parameters of the environment and generate data accordingly. It should also be able to avoid overfitting to the unknown parameters. The first key idea presented in this work is the use of a physics engine in a synthetic dataset generation pipeline. The physics engine defines environmental constraints on object placement, which naturally capture in the training set, the distribution of object poses that can realistically appear during testing. Furthermore, a physics engine is a very convenient tool to parameterize the unknown scene features, such as illumination. Randomization over such parameters is very effective in avoiding overfitting to synthetic textures of objects.

Even after detecting all the objects in a given image using a trained CNN, the problem of estimating the 6-DoF poses of the objects in the 3D workspace involves geometric reasoning regarding the position and orientation of the detected objects. Very recently end-to-end learning for 6-DoF pose estimation was proposed by Kehl et al. (2017) and Xiang et al. (2018). These methods predict the approximate 6-DoF poses and are often followed by an online local optimization process in the form of iterative closest points (ICP) (Besl and McKay, 1992). Other solutions that have been developed use a CNN for object segmentation (Hernandez et al., 2016; Zeng et al., 2017) followed by a 3D model alignment step using point cloud registration techniques (Besl and McKay, 1992; Mellado et al., 2014). The quality of the pose estimate, however, can still suffer due to over-reliance on the learned models.

The second key observation of this work is to treat individual object predictions with some level of uncertainty and perform a global, scene-level optimization process that takes object interactions into account. This information arises from physical properties, such as respecting gravity and friction as well as the requirement that objects do not penetrate one another. Through this physical reasoning, which is achieved by incorporating physics simulations, the resulting pose estimates for the objects are of improved accuracy and by default consistent. In this way, they can be directly used in the context of manipulation planning framework.

Once the system has access to an object detector and a pose estimation process, it could already be deployed for the desired application. However, as the system performs its task, it gets access to data in the operation domain which it did not have access to initially. This data could be very useful in further improving the performance of the system. Nevertheless, this data is not labeled. This motivates the need for automatically labeling real images and adding them to the existing synthetic dataset.

Overall, the current work has two contributions. Algorithmically, this work proposes a Monte Carlo tree search (MCTS) based optimization process for scene-level reasoning with physics-based priors. The MCTS-based algorithm searches over the cartesian product of individual object pose candidates to find the optimal scene hypothesis with respect to a score defined in terms of similarity of rendered hypothesized scenes with the input data. The search performs constrained local optimization for each of these candidate object poses via physics correction and ICP. This helps in pruning a large search space and, thus, quickly achieving accurate pose estimates. Second, this work provides a complete pipeline for 6-DoF pose estimation of objects placed in clutter. The main components of the proposed pipeline include the following.

- **A physics simulation tool**, which uses scene information, such as the placement of a resting surface (e.g., tabletop, shelf, etc.), object models, and camera calibration to set up an environment for generating training data. The tool performs physics simulation to place objects at realistic configurations and renders images of scenes to automatically generate a synthetic dataset to train an object detector. This tool exploits the known environmental constraints and randomizes the unknown parameters to generate a dataset, which captures, to a good extent, the properties of clutter.
- **A self-learning process**, which employs a robotic manipulator to autonomously collect multi-view images of real scenes and to label them automatically using the object detector trained with the above physics-based

simulation tool. The key insight behind this system is the fact that the robot can often find a good viewing angle that allows the detector to accurately label the object and estimate its pose. The object's predicted pose is then used to label images of the same scene taken from more difficult and occluded views. The transformations between different views are known because they are obtained by moving the robotic manipulator.

The proposed pipeline is evaluated over four challenging datasets, namely, the *Shelf&Tote* dataset (Zeng et al., 2017), *Linemod* (Hinterstoisser et al., 2012), *Linemod-Occluded* (Brachmann et al., 2014), and the *Extended Rutgers RGBD dataset*, which was collected by the authors and labeled to test the applicability of the physics-based scene-level reasoning process.

On the *Shelf&Tote* dataset, the proposed pipeline, which bootstraps pose estimation with a synthetic dataset outperforms state-of-the-art systems that have access to labeled real images. *Extended Rutgers RGBD dataset* was collected to reflect different levels of physical dependencies between objects. Evaluation over this dataset shows that the MCTS-based reasoning could quickly identify physically realistic accurate poses for complex setups where approaches that consider individual object instances fail to provide a good solution. Finally, the entire pipeline was evaluated on *Linemod* and *Linemod-Occluded* according to a recently published benchmark on pose estimation (Hodan et al., 2018) and the proposed approach outperforms several state-of-the-art techniques on this task.

This paper is an integration of two conference articles by the same authors into a complete framework for object pose estimation (Mitash et al., 2017, 2018). It expands upon the technical details provided in the aforementioned articles and provides additional examples as well as evaluations. In particular, it contributes a comprehensive process for object pose estimation, which starts with the generation of training data in physics-based simulation, followed by the steps of congruent set matching to generate object pose hypothesis, pose clustering to reduce the cardinality of the hypothesis set, and a scene-level optimization process to get accurate pose estimates. It also discusses how the obtained pose estimates can be used in a self-learning process to reduce the domain gap that might exist between the simulated training data and real test scenes. The dataset and code for the entire pipeline have been made publicly available (at `http://www.physimpose.com`).

2. Related work

This section discusses the different methodologies for object pose estimation and their relation to the current work.

2.1. Local point descriptors

One popular approach to pose estimation is to match feature points between textured 3D models and images (Collet et al., 2011; Lowe, 1999; Rothganger et al., 2006). This requires, however, textured objects and good lighting conditions, which has instead motivated the use of range data. Some range-based techniques compute correspondences between local point descriptors on the observed scene and on the object CAD models. Once correspondences are established, robust detectors such as generalized Hough transform (Ballard, 1981) or random sample consensus (RANSAC) (Fischler and Bolles, 1981a) are used to compute the rigid transform that is consistent with the majority of correspondences. Several local descriptors are available (Aldoma et al., 2012a), such as signature of histograms of orientations (SHOT) (Tombari et al., 2010), fast point feature histogram (FPFH) (Rusu et al., 2009), and Spin Images (Johnson and Hebert, 1999). There has also been work on improving the efficiency of RANSAC and Hough transform (Papazov and Burschka, 2010; Tombari and Di Stefano, 2010). Feature-based approaches can be extended to multi-view object recognition (Pillai and Leonard, 2015) and pose estimation (Erkent et al., 2016) so as to increase accuracy relative to single frame estimates. This family of methods depends on local surface information, which is sensitive to the resolution and quality of sensor and model data. The features are often parameterized by the area of influence, which is not trivial to decide. Smaller area could lead to less discriminative features between different surfaces on the object, while a larger area could result in sensitivity to occlusion and noise.

2.2. Oriented point pair features

One proposed way to counter these limitations is to use *oriented point pair features* (Drost et al., 2010) so as to create a global object model in the form of a map that stores the model points that exhibit each feature. This map can then be used to match the features in the scene and to obtain the object pose through a fast voting scheme. This idea was later extended to incorporate color (Choi and Christensen, 2012), geometric edge information (Drost and Ilic, 2012), and visibility context (Kim and Medioni, 2011). Recently, point pair features were used for segmenting the scene into several clusters, where each cluster generates a separate pose hypothesis (Birdal and Ilic, 2015). The votes are weighted based on the probability of visibility of model points. Recent work (Hinterstoisser et al., 2016) used a sampling strategy for scene points by reasoning about the size of the object model. The approach modifies the voting scheme to accommodate sensor noise by also voting in the neighboring bins. Point pair features have been criticized in some occasions for performance loss in the presence of background clutter, sensor noise, and also due to their quadratic computational complexity.

2.3. Template matching and coordinate regression

Another category of methods for pose estimation is based on *template matching*, such as *Linemod* (Hinterstoisser

et al., 2012) and variants such as Hodaň et al. (2015). This method is based on viewpoint sampling around a 3D CAD model and building templates for each viewpoint based on color gradient and surface normals, which are later matched to compute object pose. GPU-based implementations help to speed-up computation (Cao et al., 2016). Other popular approaches (Brachmann et al., 2014; Krull et al., 2015; Tejani et al., 2014) are based on learning to predict 3D object coordinates in the local model frame. A recent effort (Michel et al., 2017) performed geometric validation on these predictions using a conditional random field. The performance of these approaches can be highly dependent on the predictions of three-dimensional object coordinates from the random forest, which are not trivial to train. Template matching approaches, on the other hand, often fail to reason about occlusions.

2.4. Deep learning

The success of deep learning on problems related to object detection and semantic segmentation (Long et al., 2015; Ren et al., 2015) has motivated their use for aspects of pose estimation or for the development of direct pipelines for pose estimation (Xiang et al., 2018). Inspired by the applicability of CNNs for descriptor learning of RGB-D views (Wohlhart and Lepetit, 2015), recent work (Kehl et al., 2016) has demonstrated deep learning of descriptive features from local RGB-D patches used to create 6D pose hypotheses. Similarly, CNNs have been used to detect semantic keypoints to estimate the 6- DoF pose consistent with the keypoints (Pavlakos et al., 2017). Deep learning has also been integrated in a principled way with a global search for the discovery of 3- DoF poses of multiple objects (Narayanan and Likhachev, 2016). There are also other data-driven approaches for identifying features for object recognition (Bo et al., 2014). The success of these approaches often depends on representative labeled training data. In addition, these methods are often followed by a local optimization process, such as ICP, which is not always sufficient for fixing the errors and ambiguities in predictions. The current work leverages the success of deep learning in the task of object segmentation. It considers, however, the uncertainties in individual object predictions to guide a global optimization process to estimate poses.

2.5. Registration methods

Many recent pose estimation techniques (Hernandez et al., 2016; Mitash et al., 2018; Zeng et al., 2017) integrate CNNs for segmentation with pointset registration techniques (Mellado et al., 2014). Popular local registration approaches are Iterative Closest Points (ICP) (Besl and McKay, 1992) and its variants (Bouazix et al., 2013; Mitra et al., 2004; Rusinkiewicz and Levoy, 2001; Segal et al., 2009; Srivatsan et al., 2017), which typically require a good initialization. Otherwise, registration requires finding the best aligning rigid transform over the 6-DoF space of all possible transforms, which are uniquely determined by three pairs of (non-degenerate) corresponding points. A popular strategy is to invoke RANSAC to find aligned triplets of point pairs (Irani and Raghavan, 1996) but suffers from a frequently observable worst case $O(n^3)$ complexity in the number n of data samples, which has motivated many extensions (Cheng et al., 2013; Gelfand et al., 2005). The 4PCS algorithm (Aiger et al., 2008) achieved $O(n^2)$ output-sensitive complexity using four congruent point basis instead of three. This method was extended to Super4PCS (Mellado et al., 2014), which achieves $O(n)$ output-sensitive complexity. The accuracy of these methods, however, highly depends on the predictions from the object detector.

2.6. SLAM

SLAM (Durrant-Whyte and Bailey, 2006; Thrun et al., 2005) is a popular problem in robotics which deals with constructing the map of an unknown environment with a sensor mounted on a robot while simultaneously keeping track of the location of the robot in the world frame. Recently, a popular strategy in this field is object-based SLAM (McCormac et al., 2018; Salas-Moreno et al., 2013), which performs object pose estimation and tracking using a depth sensor and uses the relative configurations of the objects to reason about the location of the camera. Several such approaches also make use of synthetic datasets with simulated camera trajectories to learn semantic information for indoor scenes (McCormac et al., 2017). There have also been efforts (Stein and Roy, 2018) at bridging the domain gap between these synthetic scenes and sensor-acquired images for semantic labeling via image translation techniques such as CycleGAN (Zhu et al., 2017). Although these work have a notion of using synthetic dataset for learning semantic scene segmentation and of applying scene-level constraints, this problem space is quite different from that of object pose estimation in cluttered scenarios.

2.7. Global scene-level reasoning

A popular approach to resolve conflicts arising from local reasoning is to generate object pose candidates and perform a hypothesis verification (HV) step (Akizuki and Hashimoto, 2016; Aldoma et al., 2012b, 2013). The hypotheses generation in most cases occurs using a variant of RANSAC (Fischler and Bolles, 1981b; Mellado et al., 2014). One of this method's drawbacks is that the generated hypotheses might already be conflicted due to errors in object segmentation and thus performing an optimization over this might not be very useful. A recently proposed method reasons globally about the hypotheses generation process (Michel et al., 2017). Nevertheless, this requires explicit training for pixel-wise predictions. Another approach to counter these drawbacks corresponds to an exhaustive but informed search to find the best scene

hypothesis over a discrete set of object placement configurations (Narayanan and Likhachev, 2016). A tree search formulation as described above was defined to effectively search in 3-DoF space. It is not easy, however, to apply the method for 6-DoF pose estimation due to scalability and resolution issues.

This work shows that by training an object detector with an autonomous clutter-aware process, it is possible to generate a set of object candidate poses by a fast global point cloud registration method, which only has local geometric conflicts. Generating candidate poses in this manner and then applying a search process, which constrains each object expansion to other object placements, leads to significant improvements in the final pose estimation results.

3. Problem setup

This work considers the problem of estimating the 6D poses of N known objects $\{O_1, \ldots, O_N\}$ in a scene, captured by an RGB-D camera. Knowledge of the following elements is assumed.

- Geometric models are given as textured triangular meshes $\{M_1, \ldots, M_N\}$ for all objects present in the scene. Mass of objects are kept constant across all objects and friction, and linear and angular damping coefficients for objects are set to maximum within the simulator.
- Triangular mesh and pose T_{rs} for the resting surface of the objects, such as a shelf or a table in a global reference frame.
- The intrinsic and extrinsic parameters K, T_{cam} for the camera.

The estimated poses are returned as a set of rigid-body transformations $\{T_1, \ldots, T_N\}$, where each $T_i = (t_i, R_i)$ captures the translation $t_i \in R^3$ and rotation $R_i \in SO(3)$ of object model M_i in a globally defined reference frame.

4. Approach

This section presents the proposed approach for object recognition and pose estimation. It first describes how a dataset of labeled images could be generated autonomously to train a CNN for object detection. It then outlines how the detection output of CNNs is used in a search process to obtain 6-DoF pose estimates of multiple objects present in the scene. Finally, it describes a self-learning pipeline that uses the pose estimation output to label real images from multiple views and re-train the detector to improve its performance.

4.1. Generating the training dataset

The first component of the proposed framework physically simulates scenes containing target objects and generates images of the corresponding scenes using the parameters of a known camera. This is used to generate a synthetic dataset for training a CNN -based object detector. The pipeline for this process is depicted in Figure 2.

The dataset generation process mimics a real-world setup involving a sensing system for robotic manipulation, where a camera is mounted on a robotic arm. The robot is placed in front of a surface for object placement (resting surface), such as a shelf-bin or table-top, which contains the objects. In such a setup, forward kinematics can be used to provide the 6-DoF pose T_{cam} of the camera. Furthermore, a camera calibration process provides the intrinsic parameters of the camera K. The pose of the resting surface T_{rs} relative to the robot is determined by a RANSAC-based estimation process (Fischler and Bolles, 1981b). For instance, for the shelf depicted in Figure 2, such a pose estimation process was implemented by computing the edges and planes on the retrieved depth data and matching them against the known geometry of the shelf.

Given the above information as input, the method aims to render and automatically label several images in simulation as discussed in Algorithm 1.

The algorithm simulates a scene by first selecting randomly a set of objects O from the list of available object models $M_{1:N}$ (line 3). The initial pose of an object is provided by function INITIAL_RANDOM_POSES (line 4), which samples uniformly at random along the x and y axes from the range $(\frac{-dim_i}{2}, \frac{dim_i}{2})$, where dim_i is the dimension of the resting surface along the ith axis. The initial position along the z axis is fixed and can be adjusted to either simulate dropping or placing. The initial orientation is sampled appropriately in SO(3). Then, function PHYSICS_SIM is called (line 5), which physically simulates the objects and allows them to fall due to gravity, bounce, and collide with each other as well as with the resting surface. Any interpenetrations among objects or with the surface are treated by the physics engine. The final poses of the objects $\mathbf{P}^O_{final}$, when they stabilize, resemble real-world poses. Gravity, friction coefficients, and mass parameters are set at similar values globally and damping parameters are set to the maximum to promote fast stabilization.

The environment lighting and point light sources are varied with respect to location, intensity, and color for each rendering (line 6). Simulating various indoor lighting sources helps to avoid over-fitting, which makes the training set more robust to different testing scenarios. Once lighting conditions are chosen, the simulated scene is rendered from multiple views using the pre-defined camera poses (line 6). The rendering function RENDER requires the set of stabilized object poses T^O_{final}, the camera viewpoint as well as the selected lighting conditions and intrinsic camera parameters (line 7). Finally, perspective projection is applied to obtain 2D bounding box labels for each object in the scene with function PROJECT (line 8). The overlapping portion of the bounding boxes for the object that is further away from the camera is pruned.

The generated synthetic dataset is used to train an object detector based on Faster-RCNN (Ren et al., 2015), which utilizes a deep VGG network architecture (Simonyan and

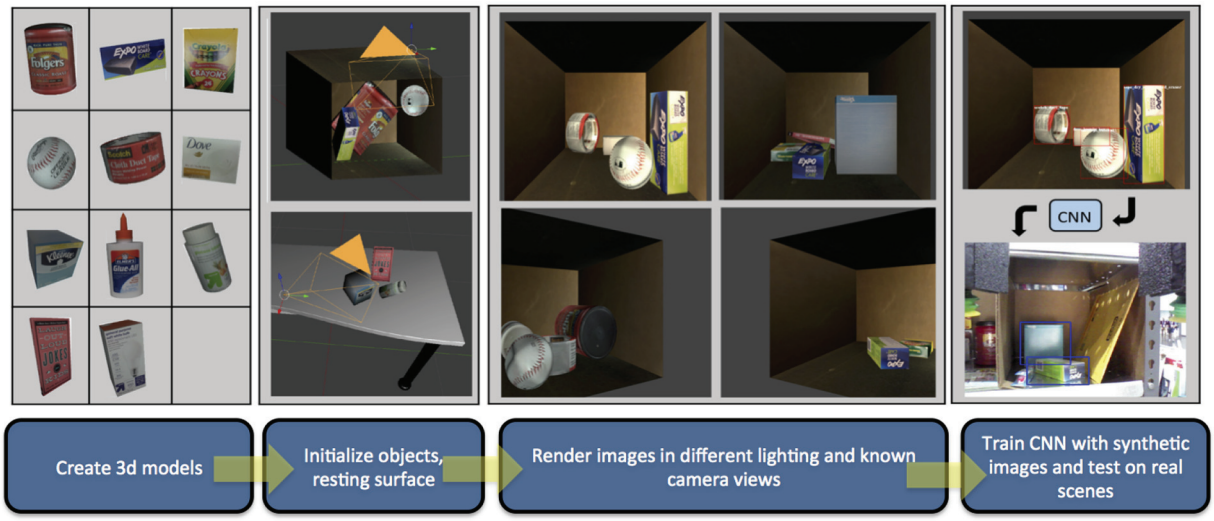

Fig. 2. Pipeline for physics aware simulation. The 3D CAD models are generated and loaded in a calibrated environment on the simulator. A subset of the objects is chosen for generating a scene. Objects are physically simulated until they settle on the resting surface under the effect of gravity. The scenes are rendered from known camera poses. Perspective projection is used to compute 2D bounding boxes for each object. The labeled scenes are used to train a Faster-RCNN object detector (Ren et al., 2015), which is tested on a real-world setup.

Algorithm 1: PHYSIM_CNN($T_{cam}, T_{rs}, K, M_{1:N}$)

```
//T_cam: set of camera poses for rendering
//T_rs: pose of the resting surface
//K: intrinsic camera parameters
//M_1:N: mesh models for all N objects
dataset ←∅
while (|dataset| < desired size) do
    O ← a random subset of objects from M_1:N;
    T^O_init ← INITIAL_RANDOM_POSES(O);
     //random initial pose within a specified domain is assigned to each object in O
    T^O_final ← PHYSICS_SIM(T^O_init, T_rs, O);
     //physics simulation is performed to obtain the final, physically consistent object pose
     for objects in O
    Light ← PICK_LIGHTING_CONDITIONS();
    foreach (view ∈ T_cam) do
        image ← RENDER(T^O_final, view, K, Light);
        {labels, bboxs} ← PROJECT(T^O_final, view);
         //set of object poses T^O_final is used to generate bounding-boxes in all views
        dataset ← dataset ∪ (image, labels, bboxs);
Train FASTER-RCNN with the generated dataset;
```

Zisserman, 2015). The dataset generation module has been implemented using the Blender API, which internally uses the Bullet physics engine and has been publicly shared (`https://github.com/cmitash/physim-dataset-generator`).

A critical requirement for learning with synthetic data as discussed above is the need for modeling the domain in the simulation. The precision with which the geometry and texture of the objects and support surface need to be modeled depends on the set of objects to be detected. If the objects have very different geometries, a noisy modeling of the shape using surface reconstruction technique such as KinectFusion (Izadi et al., 2011) is good enough for the recognition task, such as in the *Linemod* dataset (Hinterstoisser et al., 2012). If there are multiple objects with similar geometry, accuracy in modeling the texture and color is more critical to achieving a good performance, for example in the *Shelf&Tote* dataset (Zeng et al., 2017). Other physical properties such as mass and friction coefficients of objects have been kept as constant over all objects for the scope of this work while object material properties and parameters corresponding to the illumination of the environment have been randomized within a wide domain.

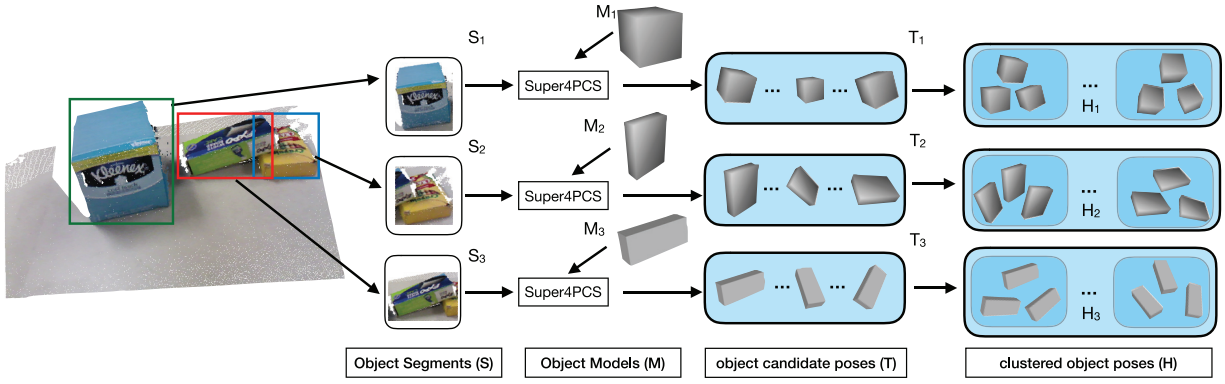

Fig. 3. The process of hypotheses generation for objects present in the scene. The process starts with extracting object segments $S_{1:3}$ using Faster-RCNN, followed by a congruent set matching process to compute a set of possible model transformations ($T_{1:3}$) that correspond to the respective segments. These transformations are then clustered to produce object- specific hypotheses sets ($H_{1:3}$).

Algorithm 2: GEN_HYPOTHESIS(RGB, depth,$M_{1:N}$)

```
//Given an RGB-D image and a set of object models M_{1:N}, GEN_HYPOTHESIS generates pose candidates h_O
for each object O.
H ← {h_O = ∅, ∀O ∈ M_{1:N}};
foreach object O in the scene do
    bbox_O ← RCNN_DETECT(RGB, O);
    //bounding box is detected for object O using the trained Faster-RCNN detector.
    P_O ← GET_3DPOINTS(bbox_O, depth);
    //3D point cloud of object O is extracted from the depth image according to bbox
    T_O ← CONGRUENT_SET_MATCHING(M_O, P_O);
    //a set of pose candidates is generated as illustrated in Figure 4
    {cluster_tr, center_tr} ← KMEANS_tr(T_O);
    //candidate poses are clustered according to their translation vectors using the KMeans
    algorithm.
    foreach cluster C in cluster_tr do
        {cluster_rot, center_rot} ← K-KMEANS_tr(C);
        //candidate poses within cluster C are further clustered according to their Euler angles
        using Kernel-KMeans.
        h_O ← h_O ∪ (center_tr, center_rot);
    H ← H ∪ h_O;
return H;
```

4.2. Pose estimation

The next component of the system is a method for 6-DoF pose estimation. It proceeds by:

1. generating a set of pose hypotheses based on the detections from the previously trained detector for each object present in the scene; and
2. searching efficiently over the set of joint hypotheses for the most globally consistent solution.

Global consistency is quantitatively evaluated by a score function. The score function measures the similarity between the actual observed depth image and a rendering of the objects in simulation using their hypothesized poses. The hypothesized poses are adapted during the search process, so as to correspond to poses where the objects are placed in a physically realistic and stable configuration according to a physics engine that simulates rigid object dynamics.

4.2.1. Hypothesis generation. Some of the desired properties for a set of 6D pose hypotheses are as follows:

- informed and diverse enough such that the optimal solution is either already contained in the set or a close enough hypothesis exists so that a local optimization process can fine-tune it and return a good result;
- limited in size, as evaluating the dependencies among the hypotheses set for different objects can lead to a combinatorial explosion of possible joint poses and significantly affect the computational efficiency;
- does not require extensive training.

This work considers all of these properties while generating the hypothesis set. The pseudocode for hypothesis generation is presented in Algorithm 2.

The detector trained with the autonomous training process proposed in the previous section is used to extract bounding-box ($\texttt{bbox}_O$) for each object O in the scene.

Table 1. Evaluating the quality of the hypotheses set returned by Super4CPS with respect to different metrics.

Metric for selection	Mean Rotation error	Mean Translation error
[All hypotheses] max. LCP score	11.16°	1.5 cm
[All hypotheses] min. rotation error from ground truth	2.11°	2.2 cm
[All hypotheses] min. translation error from ground truth	16.33°	0.4 cm
[Clustered hypotheses] min. rotation error from ground truth	5.67°	2.5 cm
[Clustered hypotheses] min. translation error from ground truth	20.95°	1.7 cm

This, in turn, gives a segment P_O of the 3D point cloud. Segment P_O is a subset of the point cloud of the scene and contains points from the visible part of the object O. Segment P_O frequently contains some points from nearby objects because the bounding box does not perfectly match the shape of the object.

The received point set P_O is then matched to the object model M_O in the subroutine CONGRUENT_SET_MATCHING in Algorithm 2 to generate pose candidates for the object. This module, inspired by the Super4PCS (Mellado et al., 2014) algorithm, iteratively samples a set of four co-planar points from P_O called the base and finds sets of four points on the model which are congruent under rigid transformation to the base. Each pair of congruent sets gives a pose hypothesis. The matching process is depicted in Figure 4. The fact that the distances, angles, and ratios of the intersection of line segments are maintained over a rigid transform is used to come up with an efficient linear time algorithm for finding the congruent sets. The time complexity of this process is $O(n+m+k)$, where n is the number of points on the sampled object model, m is the number of point pairs on the model which are at the same distance as a point pair on the sampled base, and k is the number of the congruent sets found corresponding to the sampled base. As opposed to RANSAC (Fischler and Bolles, 1981b), which has a time complexity of $O(n^3)$ for matching a set of three points to all triplets of points on the model, this process better exploits the geometric constraints from rigid transformations and efficiently produces a relatively small set of pose candidates.

Nevertheless, Super4PCS evaluates each of these transformations to find the one that achieves the best alignment according to the largest common pointset (LCP) metric. This returned transformation, however, is not necessarily the optimal object pose as the point cloud segment extracted via the detection process could include parts of other objects or due to lack of visible surface might not be informative enough to compute the correct solution. This motivates the consideration of other possible transformations for the objects, which can be evaluated in terms of scene-level consistency.

Thus, the proposed process retains a set of possible transformations T_O computed using congruent set matching within a given time budget t_o. It is interesting to consider the quality of the hypotheses set returned by the above process by measuring the error between the returned pose

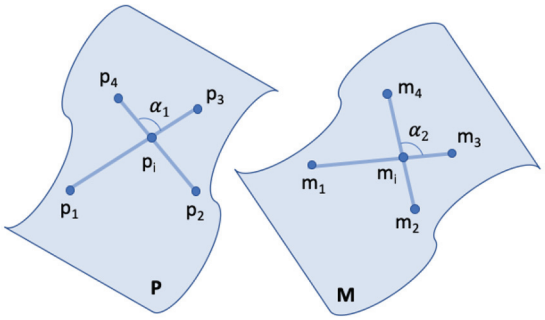

Fig. 4. The `congruent-set-matching` process which finds sets of four points on the scene and on the object model that are congruent under rigid transformation. All the point pairs on the model M with distances similar to $|p_1 - p_3|$ and $|p_2 - p_4|$ on the sampled base, can be found in linear time using an efficient technique as described in Mellado et al. (2014). Then four-point congruent sets are found by evaluating these point pairs based on other invariances such as ratios and angles.

hypotheses and the ground truth. For this purpose, a validation dataset containing 90 object poses was used. Specifically, in each hypothesis set, the pose hypothesis that has the minimum error in terms of rotation is selected as well as the one with the minimum translation error. The mean errors for these candidates over the dataset are listed in Table 1. The results positively indicate the presence of hypotheses close to the true solution. Specifically, the candidate with the minimum rotation error seems almost perfect in the rotation and not very far even with respect to translation. Nevertheless, this hypothesis set contained approximately 20,000 elements. It is intractable to evaluate scene-level dependencies for that many hypotheses per object as the combined hypotheses set over multiple objects grows exponentially in size.

4.2.2. Clustering of hypotheses. To reduce the cardinality of the hypotheses sets returned by the subroutine CONGRUENT_SET_MATCHING in Algorithm 2, this work proposes to cluster the 6D poses in each set T_O, given a distance metric. Computing distances between object poses, which are defined in SE(3), in a computationally efficient

manner is not trivial (Zhang et al., 2007). This challenge is further complicated if one would like to consider the symmetry of the geometric models, so that two different poses that result in the same occupied volume given the object's symmetry would get a distance of zero.

To address this issue, a two-level hierarchical clustering approach is followed. The first level involves computing clusters of the pose set in the space of translations (i.e., the clustering occurs in $\mathbb{R}^3$ by using the Euclidean distance and ignoring the object orientations) using a *K*-means process (Arthur and Vassilvitskii, 2007) to obtain a smaller set of cluster representatives $cluster_{tr}$. In the second level, the poses that are assigned to the same clusters are further clustered based on a distance computed in the SO(3) space that is specific to the object model, i.e., by considering only the orientation of the corresponding pose. The second clustering step uses a *kernel K-means* approach (Dhillon et al., 2004), where the cluster representative is found by minimizing the sum of kernel distances to every other point in the cluster. This process can be computationally expensive but returns cluster centers that nicely represent the accuracy of the hypotheses set. By using this clustering method, the size of the hypotheses set can be reduced down from 20,000 rigid transforms in T_O to 25 object pose hypotheses in h_O for each object in the scene. The two bottom rows of Table 1 evaluate the quality of the cluster representatives in the hypotheses set. This evaluation indicates that the clustering process returns hypotheses as cluster representatives that are still close to the true solution. In this way, it provides an effective way of reducing the size of the hypotheses set without sacrificing its diversity.

4.2.3. Search. Once the hypotheses set is built for each object in the scene, the task reduces to finding the object poses that lie in the physically consistent neighborhood of the pose candidates and best explain the overall observed scene. In particular, given:

- the observed depth image I_D;
- the number of objects in the scene $\mathtt{N}$;
- a set of 3D mesh models for these objects $\mathtt{M}_{1:N}$;
- and the sets of 6D transformation hypotheses for the objects $h_{1:N}$ (output of Algorithm 2);

the problem is to search in the hypotheses sets for an *N*-tuple of poses $T_{1:N}$ so that $T_i \in f(h_i)$, i.e., one pose per object. The set $T_{1:N}$ should maximize a global score computed by comparing the observed depth image with the rendered image $R(T_{1:N})$ of object models placed at the corresponding poses $T_{1:N}$. Here, f is the constrained local optimization of the object pose $\mathtt{h}_i$ based on physical consistency with respect to the other objects in the scene and also the fact that the same points in the scene point cloud cannot be explained by multiple objects simultaneously. Then, the global optimization score is defined as

$$C(I_D, T_{1:N}) = \sum_{p \in P} Sim(R(T_{1:N})[p], I_D[p])$$

where p is a pixel (i,j) of a depth image, $R(T_{1:N})[p]$ is the depth of pixel p in the rendered depth image, $I_D[p]$ is the depth of pixel p in the observed depth image, $P = \{p | R(T_{1:N})[p] \neq 0 \text{ or } I_D[p] \neq 0)\}$ and

$$Sim(R(T_{1:N})[p], I_D[p]) = \begin{cases} 1, & \text{if } |R(T)[p] - I_D[p]| < \epsilon \\ 0, & \text{otherwise} \end{cases}$$

for a predefined precision threshold ε. Therefore, score $\mathtt{C}$ counts the number of non-zero pixels p that have a similar depth in the observed image I_D and in the rendered image $\mathtt{R}$ within an ε threshold. Thus, overall the objective is to find:

$$T^*_{1:N} = \arg \max_{T_{1:N} \in f(h_{1:N})} C(I_D, R(T_{1:N}))$$

At this point, a combinatorial optimization problem arises so as to identify $T^*_{1:N}$, which is approached with a tree search process. A state in the search tree corresponds to a subset of objects in the scene and their corresponding poses. The root state s_0 is a null assignment of poses. A state s_d at depth d is a placement of d objects at specific poses selected from the hypotheses sets, i.e., $\mathrm{s}_d = \{(M_i, T_i),\ i = 1 : d\}$ where T_i is the pose chosen for object M_i, which is assigned to a tree depth i. The goal of the tree search is to find a state at depth $\mathtt{N}$, which contains a pose assignment for all objects in the scene and maximizes the above-mentioned rendering score. Algorithm 3 describes the expansion of a state in the tree search process towards this objective.

The EXPAND routine takes as input the state s_d at tree depth d, the point cloud segment corresponding to the next object to be placed, P_{d+1}, and the pose hypothesis T_{d+1} for the next object to be placed M_{d+1}. Lines 3 and 4 of the

Algorithm 3: EXPAND$(s_d, (M_{d+1}, T_{d+1}), P_{d+1})$

```
//s_d: state at depth d (pose assignment for the first d objects)
//(M_{d+1}, T_{d+1}): mesh model and pose hypothesis for the (d+1)th object
//P_{d+1}: point cloud segment for (d+1)th object
1. if d = N then
2.     return NULL;
       //maximum depth of tree is reached
3. foreach (M_O, T_O) ∈ s_d do
4.     P_{d+1} ← P_{d+1} - POINTS_EXPLAINED(P_{d+1}, M_O, T_O);
       //remove points from P_{d+1} already assigned to an object M_O at T_O in s_d
5. T_{d+1} ← TRIMMED_ICP((M_{d+1}, T_{d+1}), P_{d+1});
       //pose is locally refined using trimmed-ICP
6. T_{d+1} ← PHYSICS_SIM((M_{d+1}, T_{d+1}), s_d);
       //pose is locally refined based on physics simulation
7. s_{d+1} ← s_d ∪ (M_{d+1}, T_{d+1});
8. return s_{d+1};
```

Algorithm 4: SEARCH

```
    //M_{1:N}: mesh models for all N objects
    //P_{1:N}: point cloud segments for all N objects
    //h_{1:N}: pose candidate sets for all N objects
1.  Function MCTS (M_{1:N}, P_{1:N}, h_{1:N})
2.      S←∅
3.      L_{1:K} ← GET_DEPENDENCY(P_{1:N});
        //L_{1:K} is a partition of N objects into K subsets where objects belonging to different subsets
            are physically independent of each other
4.      foreach L ∈ L_{1:K} do
5.          s_0 ← ∅
6.          best_render_score ← 0;
7.          best_state ← s_0
8.          while search_time < t_th do
                //t_th is a pre-defined time budget.
9.              s_i ← SELECT(s_0, M_{1:N}, P_{1:N}, h_{1:N});
                //s_i is the next state to be expanded based on UCB
10.             {s_N, R} ← RANDOM_POLICY(s_i, M_{1:N}, P_{1:N}, h_{1:N});
                //s_N is the state obtained by randomly selecting poses for all unplaced objects, i.e., not
                in s_i. R is the rendered score for s_N
11.             if R>best_render_score then
12.                 best_render_score ← R;
13.                 best_state ← s_R;
14.             BACKUP_REWARD(s_i, R);
                //R is used to update estimated costs of all states s along the path from s_i to the root node.
15.         S ← T ∪ best_state;
16.     return S;
        //S is a set of object poses for N objects
```

algorithm iterate over the objects already placed in state s_d and remove points explained by these object placements from the point cloud segment of the next object to be placed. This step helps in achieving much better segmentation, which is utilized by the local optimization step of Trimmed ICP (Chetverikov et al., 2002) in line 5. The poses of objects in state s_d physically constrain the pose of the new object to be placed. For this reason, a rigid-body physics simulation is performed in line 6. The physics simulation is initialized by inserting the new object into the scene at pose T_{d+1}, while the previously inserted objects in the current search branch are stationary in the poses $T_{1:d}$. A physics engine is used to ensure that the newly placed object attains a physically realistic configuration (stable and no penetration) with respect to other objects and the table under the effect of gravity. After a fixed number of simulation steps, the new pose T_{d+1} of the object is appended to the previous state to get the successor state s_{d+1}.

The above primitive is used to search over the tree of possible object poses. The objective is to exploit the contextual ordering of object placements given information from physics and occlusion. This does not allow an additive rendering score to be defined over the search depth as in previous work (Narayanan and Likhachev, 2016), which demands the object placement to not occlude any part of the already placed objects. Instead, this work proposes to use a heuristic search approach based on MCTS utilizing the *upper confidence bound* (UCB) formulation (Kocsis and Szepesvári, 2006) to trade off exploration and exploitation in the expansion process. The pseudocode for the search is presented in Algorithms 4 and 5.

To effectively utilize the constrained expansion of states, an order of object placements needs to be considered. This information is encoded in a *dependency graph*, which is a directed acyclic graph that provides a partial ordering of object placements but also encodes the interdependency of objects. An example of a *dependency graph* structure is presented in Figure 5. The vertices of the dependency graph correspond to the objects in the observed scene. Simple rules are established to compute this graph based on the detected segments $\mathrm{P}_{1:N}$ for objects $O_{1:N}$.

- A directed edge connects object O_i to object O_j if the x–y projection of P_i in the world frame intersects with the x–y projection of P_j and the z-coordinate (negative gravity direction) of the centroid for P_j is greater than that of P_i.
- A directed edge connects object O_i to object O_j if the detected bounding-box of O_i intersects with that of O_j and the z-coordinate of the centroid of P_j in camera frame (normal to the camera) is greater than that of P_i.

The information regarding the independence of objects helps to significantly speed up the search as the independent objects are then evaluated in different search trees and prevent exponential growth of the tree. This results in K ordered list of objects, $\mathrm{L}_{1:K}$ coming from the module GET_DEPENDENCY of Algorithm 5, each of which is passed to an independent tree search process for pose computation.

Algorithm 5: SEARCH MODULES

```
1.  Function SELECT (s, M_{1:N}, P_{1:N}, h_{1:N})
      //s: state of the search tree
      //M_{1:N}: mesh models for all N objects
      //P_{1:N}: point cloud segments for all N objects
      //h_{1:N}: pose candidate sets for all N objects
2.    while depth(s) < N do
3.      if s has unexpanded child then
4.        d ← depth(s);
5.        T_{d+1} ← NEXT_POSE_HYPOTHESIS(h_d);
          //T_{d+1} is the next pose candidate for (d+1)th object that has not already been expanded for state s.
6.        return EXPAND(s, (M_{d+1}, T_{d+1}), P_{d+1});
          //appends the pose T_{d+1} to state s
7.      else
8.        Return best child s according to UCB Equation (1);
9.    return s;
10. Function RANDOM_POLICY (s, M_{1:N}, P_{1:N}, h_{1:N})
      //s: state of the search tree
      //M_{1:N}: mesh models for all N objects
      //P_{1:N}: point cloud segments for all N objects
      //h_{1:N}: pose candidate sets for all N objects
11.   while depth(s) < N do
12.     d ← depth(s);
13.     T_{d+1} ← GET_RANDOM_HYPOTHESIS(h_{d+1});
        //T_{d+1} is a random pose assigned to the (d+1)th object.
14.     s ← EXPAND(s, (M_{d+1}, T_{d+1}), P_{d+1});
        //appends the pose T_{d+1} to state s.
15.   return {s, render(s)};
16. Function BACKUP_REWARD (s, R)
      //s: state of the search tree
      //R: render score for the state s
17.   while s ≠ NULL do
18.     n(s) ← n(s) + 1;
19.     h(s) ← h(s) + R;
        //number of expansions and estimated cost of state s is updated
20.     s ← parent(s);
```

The MCTS proceeds by selecting the first unexpanded node starting from the root state. The selection of the next state to be expanded takes place based on a reward associated with each state. The reward is the mean of the rendering score received at any leaf node in the state's subtree along with a penalty based on the number of times this subtree has been expanded relative to its parent. This is the UCB formulation (Kocsis and Szepesvári, 2006). Formally, given a state s of the search tree, the next state to be expanded is selected as

$$s = \underset{s' \in succ(s)}{\arg\max} \frac{h(s')}{n(s')} + \alpha \sqrt{\frac{2log(n(s))}{n(s')}} \quad (1)$$

where $h(s)$ is the estimated score for state s, $n(s)$ is the number of times the subtree rooted at the state s has been expanded, and α is the parameter that controls the trade-off between exploration and exploitation in the search process. The selected state is then expanded by using a RANDOM_POLICY, which in this case is picking a random object pose hypothesis for each of the succeeding objects while performing the constrained local optimization at each step. The output of this policy is the final rendering score of the generated scene hypotheses. This reward is then backpropagated in the step BACKUP_REWARD to all preceding nodes. Thus, the search is guided to the part of the tree, which gets a good rendering score but also explores other portions, which have not been expanded enough (controlled by the parameter α). Figure 5 visualizes these steps of the MCTS pipeline.

5. Self-learning

Given access to an object detector and a pose estimation process trained with the physics-based simulator, the self-learning pipeline labels real-world images with a robust multi-view pose estimation. This is based on the idea that the detector performs well on some views, while it might be imprecise or fail in other views. Aggregating 3D data over the confident detections and with access to the knowledge of the environment, a 3D segment can be extracted for each object instance in the scene. This process, combined with the fact that 3D models of objects are available, makes it highly likely to estimate correct 6-DoF poses of objects given enough views and search time. The results of pose estimation are then projected back to the multiple views and used to label real images. These examples are very effective to reduce the confusion in the classifier for novel views. The process also autonomously reconfigures the scene using manipulation actions to apply the labeling process iteratively over time in different scenes, thus generating a labeled dataset which is used to re-train the object detector. The pipeline of the process is presented in Figure 6 and the pseudocode is provided in Algorithm 6.

A robotic arm is used to move the sensor to different pre-defined camera configurations T_{cam} and capture RGB and depth images of the scene (lines 2–3). The PRACSYS motion planning library (Kimmel et al., 2012; Littlefield et al., 2015) was used to control the robot in the accompanying implementation.

The detector trained using physics-aware simulation is then used to extract bounding boxes corresponding to each object in the scene (line 7). There might exist a bias in simulation either with respect to texture or poses, which can lead to imprecise bounding boxes or complete failure in certain views. For the detection to be considered for further processing, a threshold is considered on the confidence value returned by RCNN (line 8).

The pixel-wise depth information *Seg3d* within the confidently detected bounding boxes *bbox* (line 9) is aggregated in a common point cloud per object $Cloud_O$ given information from multiple views (line 10). The process employs environmental knowledge to clean the aggregated

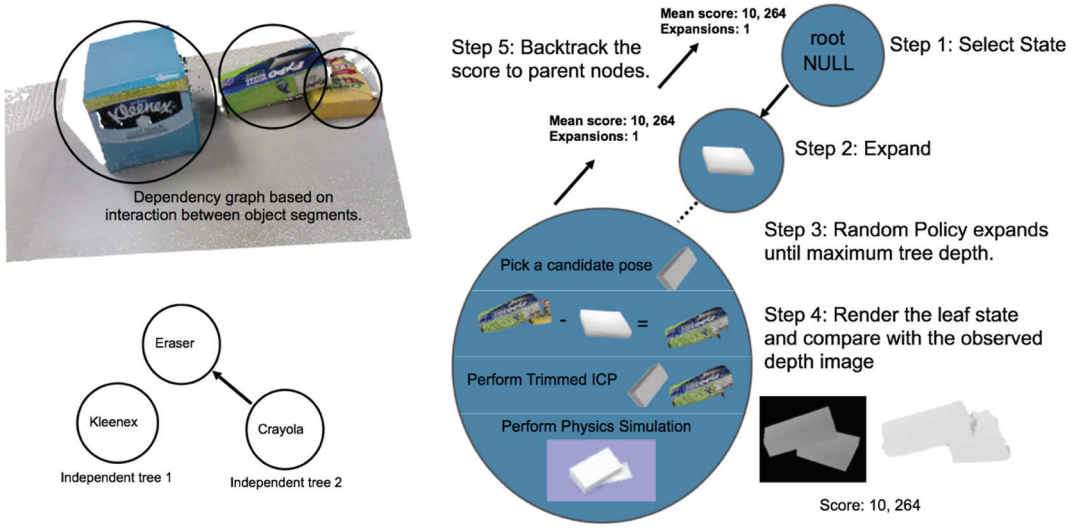

Fig. 5. Left: How a dependency graph is built based on interactions between the object segments. The Kleenex object is placed separately and does not need to be evaluated in the same tree as the others, whereas the placement of expo depends on the placement of crayola. Right: One iteration of the MCTS process. The next state to expand is selected based on the previously computed score for the states. The selected state is then evaluated by executing a random policy which keeps expanding state until all objects in the current tree are placed. Finally, a rendering of this completely reconstructed scene is compared with the observed depth image to compute a score for the state.

Algorithm 6: SELF-LEARN($dataset, T_{cam}, M_{1:N}$)

```
    //dataset: synthetic training dataset
    //T_cam: set of camera poses to collect images
    //M_1:N: mesh models for all N objects
1.  while |dataset| < desired size do
2.      foreach view ∈ T_cam do
3.          {RGB_view, D_view} ← CAPTURE(view);
            //RGB-D images are collected my moving the camera to all views in T_cam
4.      foreach object O in the scene do
5.          Cloud_O = ∅;
6.          foreach view ∈ T_cam do
7.              bbox ← SIM_DETECT(RGB_view);
                //bounding box is detected for object O in image RGB_view
8.              if conf(bbox) > ε then
9.                  Seg3d ← CONVERT3D(bbox, D_view);
10.                 Cloud_O ← Cloud_O ∪ Seg3d;
                //point cloud of object O is extracted from depth image according to bbox
11.         OUTLIER_REMOVAL(Cloud_O);
12.         T_O ← COMPUTE_6DPOSE(Cloud_O, M_O);
            //6D pose is computed given the point cloud segment and object model
13.     foreach view ∈ T_cam do
14.         {labels, bboxs} ← PROJECT(T_O, view);
            //Estimated pose T_O is used to generate bounding-boxes in all views
15.         dataset ← dataset ∪ (RGB_view, labels, bboxs);
16.     randObj ← SAMPLE_RANDOM_OBJECT(M_1:N);
17.     RECONFIGURE_OBJECT(randObj);
        //randomly selected objects are moved to pre-specified configuration.
18. Train Faster-RCNN using the expanded dataset;
```

point cloud (line 11). Points outside the resting surface bounds are removed and outlier removal is performed based on k-nearest neighbors and a uniform grid filter.

Several point cloud registration methods were tested for registering the 3D model M_O with the corresponding segmented point cloud $Cloud_O$ (line 12). This included Super4PCS (Mellado et al., 2014), fast global registration (Zhou et al., 2016), and simply using the principal component analysis (PCA) with ICP (Besl and McKay, 1992). The Super4PCS algorithm used alongside ICP was found

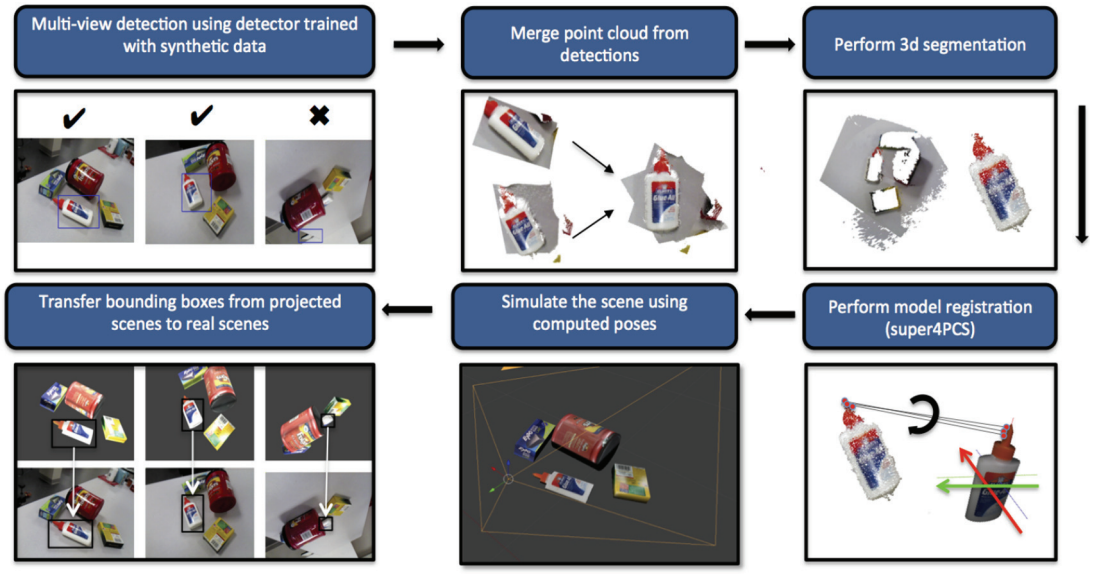

Fig. 6. Automatic self-labeling pipeline: the detector trained with simulated data is used to detect objects from multiple views. The point cloud aggregated from successful detections undergoes 3D segmentation. Then, Super4PCS (Mellado et al., 2014) is used to estimate the 6D pose of the object in the world frame. The computed poses with high confidence are projected back to the views to obtain precise labels over real images.

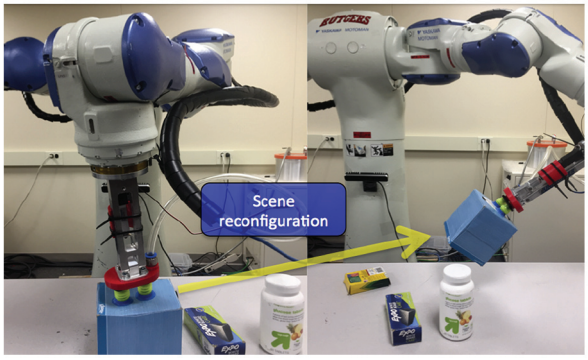

Fig. 7. Manipulator performing scene reconfiguration by moving an object from one configuration on the table to another.

to be the most applicable for the target setup as it is the most robust to outliers and returns a very natural metric for confidence evaluation. Super4PCS returns the best rigid alignment according to the LCP. The algorithm searches for the best score, using transformations obtained from four-point congruences. Thus, given enough time, it generates the optimal alignment with respect to the extracted segment.

After the 6-DoF pose is computed for each object, the scene is recreated in the simulator using object models placed at the pose T_O and projected to the known camera views (line 14). Bounding boxes are computed on the simulated setup and transferred to the real images. This gives precise bounding box labels for real images in all the views (line 15).

To further reduce manual labeling effort, an autonomous scene reconfiguration is performed (lines 16–17). The robot reconfigures the scene with a pick and place manipulation action to iteratively construct new scenes and label them, as in Figure 7. For each reconfiguration, the object to be moved is chosen randomly and the final configuration is selected from a set of pre-defined configurations in the workspace.

Finally, the Faster-RCNN network is re-trained with the expanded dataset. The factors that prevent this process from a label drift are as follows. (1) The network is re-trained with a large number of accurate synthetic data. Thus, the training is immune to some amount of label noise in the self-labeled data. (2) Only the most confident detections from multiple views are considered and a global-search-based process for pose estimation is used to obtain the estimates which are eventually used for labeling.

6. Evaluation

This section evaluates several aspects of the proposed approach. It describes the experimental setup, evaluation metrics and compares against baseline alternatives. The evaluations are performed over four datasets with different challenges in each, as described in Table 2. The *Shelf&Tote* dataset (Zeng et al., 2017) offers 148 different configurations of objects from the APC, placed in bins of a shelf with challenging conditions such as occlusions and shiny reflective surfaces of the shelf. In each scene, two to five objects are supposed to be detected. The *Extended Rutgers RGBD dataset* was created for the purpose of studying the utility of the proposed physics-based pose estimation process. RGB-D images for 42 different configurations of objects were collected and ground-truth 6-DoF poses were labeled for each object in the image. The dataset contains

Fig. 8. Images from training datasets. Left: Uniformly sampled synthetic data. Center: Training data from the *Shelf&Tote* dataset (Zeng et al., 2017). Right: Dataset generated from the proposed physics-aware simulation.

Table 2. Statistics for the test datasets.

	Shelf& Tote	Ext. Rutgers	LM	LM-O
Number of objects	11	11	8	8
Number of scenes	148	42	8	3
Number of frames	2,220	42	1,600	200
Objects/scene	2–5	3	1	8
Sensor	Intel Realsense		Microsoft Kinect	
Resolution	640×480			

Table 3. Evaluating object detection trained with synthetic data.

Method	mAP (%)
Zeng et al. (2017) (benchmark)	75%
Sampled from test data distribution	69%
Sampled from uniform distribution	31%
Physics-aware simulation	64%
Physics-aware simulation + randomized illumination	70%
Method	**mAP (%)**
Self-learning (2,000 images)	75%
Self-learning (6,000 images)	81%
Self-learning (10,000 images)	82%

the same 11 objects from the APC as the *Shelf&Tote* dataset, representing different object geometries. Each scene contains three objects to be detected and the object placement is a mix of independent object placements, objects with physical dependencies such as one stacked on/or supporting the other object and occlusions. The dataset was collected using an Intel RealSense sensor mounted over a Motoman robotic manipulator. The *Linemod* (LM) dataset (Hinterstoisser et al., 2012) is a popular dataset for evaluating pose estimation techniques. For several frames captured from different views, one object instance is labeled per scene. However, the scene has a clutter of other known and unknown objects. Brachmann et al. (2014) labeled the pose of all the known objects (eight object classes) in clutter for one such test sequence. This test sequence referred to as the *Linemod-Occluded* (LM-O) dataset has a high level of occlusions in several views.

In the first section, the synthetic data generation pipeline and the effect of self-learning is evaluated on the *Shelf&Tote* dataset. This is followed by a detailed summary of performance and accuracy of the pose estimation approach on the *Extended Rutgers RGBD dataset*. Finally, the entire pipeline is evaluated on the LM and LM-O dataset according to the recently published benchmark (Hodan et al., 2018).

6.1. Evaluating the dataset generation pipeline

The dataset generation pipeline is evaluated for the task of bounding-box object detection. A Faster-RCNN-based (Ren et al., 2015) object detector is trained with the datasets generated from different pipelines. The most likely bounding-box prediction for each of the known classes in the scene is considered and a mean average precision (mAP) is calculated. The predicted bounding-box is a true positive when the intersection-over-onion (IoU) of the predicted bounding box with the ground-truth bounding-box is greater than a threshold (set to a standard IoU value of 0.5).

To study how the object pose distribution affects the training process, different techniques for synthetic data generation are evaluated. The results of experiments performed on the *Shelf&Tote* dataset are presented in Table 3.

The following is a brief discussion of the dataset generation techniques used for the comparisons.

6.1.1. Training data generated using test data distribution. The objective here is to establish an upper bound for the performance of a detector trained with simulated images. For this purpose, the object detector is trained with the knowledge of pose distribution from the test data. This process consists of estimating the density of the test data with respect to object poses using kernel density estimation, and generating training data according to this distribution. The sampled scenes were used to train a Faster-RCNN detector, which achieved an accuracy of 69%.

Table 4. Evaluating pose estimation on model learnt from self-learning process.

2D-segmentation method	3D-registration method	Mean-error rotation (deg)	Mean-error translation (m)	Success (%)
Ground-truth bounding-box	PCA + ICP	7.65	0.02	84.8
FCN (trained with Zeng et al. (2017))	PCA + ICP	17.3	0.06	54.6
FCN (trained with Zeng et al. (2017))	Super4PCS + ICP	16.8	0.06	54.2
FCN (trained with Zeng et al. (2017))	fast-global-registration	18.9	0.07	43.7
Faster-RCNN (Proposed training)	PCA + ICP	**8.50**	0.03	**79.4**
Faster-RCNN (Proposed training)	Super4PCS + ICP	8.89	**0.02**	75.0
Faster-RCNN (Proposed training)	fast-global-registration	14.4	0.03	58.9

6.1.2. Uniformly sampled synthetic data. This alternative is a popular technique for generating synthetic data. It uses 3D models of the objects to render their images from several viewpoints sampled on a spherical surface centered at the object. The background image corresponded to the APC shelf, on top of which randomly selected objects were pasted at sampled locations. This process allows to simulate occlusions and mask subtraction provides the accurate bounding boxes in these cases. The objects in these images are not guaranteed to have physically realistic poses. This method of synthetic data generation does not perform well on the target task, giving a low accuracy of 31%.

6.1.3. Generating training data with physics-aware simulation. The accuracy of 64% achieved by the proposed physics-aware simulator is close to the upper bound. By incorporating the knowledge of the camera pose, resting surface and by using physics simulation, the detector is essentially constraining the distribution of poses to resemble the one from which the test data comes.

The results discussed until now were with respect to a constant lighting condition. As the dataset grows, a dip in the performance is observed. This is expected as the detector overfits with respect to the synthetic texture, which does not mimic real lighting conditions. This is not desirable, however. To deal with this issue, the lighting conditions are varied according to the location and color of the light source. This does resolve the problem to some extent but the dataset bias still limits performance to an accuracy of 70%.

Once a detector is trained with the dataset from simulation, the self-learning pipeline is executed. It is used to automatically label training images from *Shelf&Tote* dataset. The real images are incrementally added to the simulated dataset to re-train the Faster-RCNN. This results in a performance boost of 12%. This result also outperforms the training process by Zeng et al. (2017) which uses approximately 15,000 real images labeled using background subtraction. The reason that the proposed method outperforms a large dataset of real training images is that the proposed system contains labels for objects placed in a clutter and not just single instances of objects.

The detector trained from the self-learning pipeline is also evaluated on the task of multi-view pose estimation. Table 4 compares the Faster-RCNN-based detector trained with the proposed dataset generation technique to a fully convolutional network (FCN) trained with dataset generation process from Zeng et al. (2017). Different algorithms for estimating the 6D pose is considered and the success is reported by counting the instances when the pose prediction encounters an error in translation less than 5 cm and mean error in the rotation less than 15°. It is also interesting to note that the success in pose estimation is on a par with the success achieved using ground-truth bounding boxes.

6.2. Evaluating the search-based pose estimation

In this section, the proposed MCTS-based algorithm is evaluated over the *Extended Rutgers RGBD dataset*. The scenes in the dataset express three different levels of interaction between objects, namely, independent object placement where an object is physically independent of the rest of objects, two-object dependencies where an object depends on another, and three-object dependencies where an object depends on two other objects.

The evaluation is performed by computing the error in translation, which is the Euclidean distance of an object's center compared with its ground-truth center (in centimeters). The error in rotation is computed by first transforming the computed rotation to the frame attached to the object at ground truth. The rotation error is the average of the roll, pitch, and yaw angles (in degrees) of the transformation between the returned rotation and the ground-truth one, while taking into account the object's symmetries, which may allow multiple correct answers. The results provide the mean of the errors of all the objects in the dataset.

The evaluation was first performed for methods that reason about one object at a time, i.e., methods that do not perform any scene-level reasoning. These approaches trust the segments returned by the object segmentation module and perform model matching followed by local refinement to compute object poses. The results of performing pose estimation over the collected dataset with some of these techniques are presented in Table 5. Zeng et al. (2017) developed a system for pose estimation towards the APC 2016. The system uses a FCN to get pixel-level

Table 5. Comparing our approach with different pose estimation techniques.

Method	No dependencies		2-object dependencies		3-object dependencies		All	
	Rot. Err.	Trans. Err.	Rot. Err.	Trans. Err.	Rot. Err.	Trans. Err.	Rot. Err.	Trans. Err.
APC-Vision-Toolbox	15.5°	3.4 cm	26.3°	5.5 cm	17.5°	5.0 cm	21.2°	4.8 cm
Faster-RCNN + Super4PCS + ICP	2.4°	0.8 cm	14.8°	1.7 cm	12.1°	2.1 cm	10.5°	1.5 cm
PHYSIM-Heuristic (depth + LCP)	2.8°	1.1 cm	5.8°	1.4 cm	12.5°	3.1 cm	6.3°	1.7 cm
PHYSIM-MCTS (proposed approach)	2.3°	1.1 cm	5.8°	1.2 cm	5.0°	1.8 cm	**4.6°**	**1.3 cm**

segmentation of objects in the scene, then uses PCA for pose initialization, followed by ICP to get the final object pose. This system was designed for shelf and tote environments and often relies on multiple views of the scene. Thus, the high error in pose estimates could be attributed to the low recall percentage in retrieving object segment achieved by the semantic segmentation method, which in turn resulted in the segment not having enough information to compute a unique pose estimate. The second system tested uses a Faster-RCNN-based object detector trained with the *physics-based dataset generator* as described previously. The point cloud segments extracted from the bounding box detections were used to perform pose estimation using two different approaches: (i) PCA followed by ICP and (ii) Super4PCS followed by ICP (Besl and McKay, 1992). Even though the detector succeeded in providing a high recall object segment on most occasions, in the best case the mean rotation error using local approaches was still high (10.5°). This was sometimes due to bounding boxes containing parts of other object segments, or due to occlusions. Reasoning only at a local object-level does not resolve these issues.

The proposed search framework was used to perform pose estimation on the dataset. In each scene, the dependency graph structure was used to get the order of object placement and initialize the independent search trees. Then, the object detection was performed using Faster-RCNN and congruent set matching was used to generate pose candidates, which were clustered to get 25 representatives per object. The search is performed over the combined set of object candidates and the output of the search is an anytime pose estimate based on the best rendering score. The stopping criterion for the searches was defined by a maximum number of node expansions in the tree, set to 250, where each expansion corresponds to a physics simulation with Bullet and a rendering with OpenGL, with a mean expansion time of ~0.2 s per node. The search was initially performed using a depth-first heuristic combined with the LCP score returned by the Super4PCS for the pose candidates. The results from this approach, PHYSIM-Heuristic (depth + LCP), are shown in Table 5, which indicates that it might be useful to use these heuristics if the tree depth is low (one and two object dependencies). As the number of object dependencies grows, however, one needs to perform more exploration. For three-object

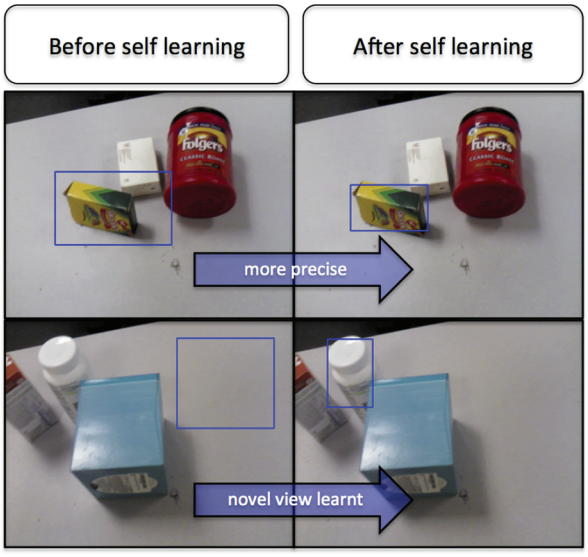

Fig. 9. Results of object detection before and after training with the self-learning process. The detector learns to predict more precise bounding boxes. It can also detect objects better from novel views.

dependencies, when using 250 expansions, this heuristic search provided poor performance. The UCT MCTS was used to perform the search, with UCBs to trade off exploration and exploitation. The exploration parameter was set to a high value ($\alpha = 5,000$), to allow the search to initially look into more branches while still preferring those that give a high rendering score. This helped in speeding up the search process significantly, and a much better solution could be reached within the same time. The plots in Figures 10 and 11 captures the anytime results from the two heuristic search approaches. Figure 12 shows some of the images from the *Extended Rutgers RGBD dataset* and the corresponding results from the UCT MCTS process.

To study the effect of training on the pose estimation process, an experiment was performed which utilizes the ground-truth segmentation of objects and performs the PHYSIM-MCTS to generate object poses. This resulted in a rotation error of 2.94° and a translation error of 0.7 cm. This is not significantly different from the results with the proposed process which indicates that the bounding-box detector trained from the autonomous dataset generation

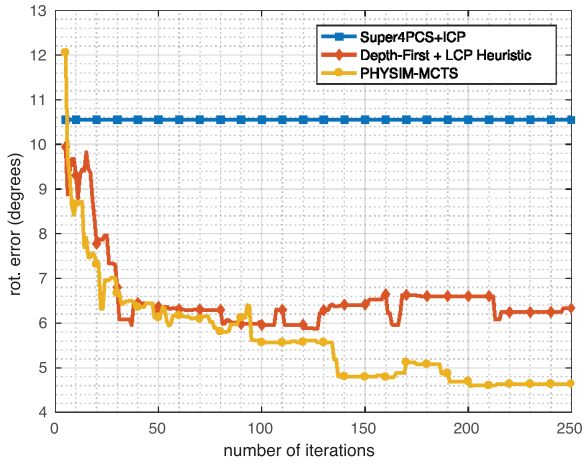

Fig. 10. Rotation error in degrees as a function of the number of iterations.

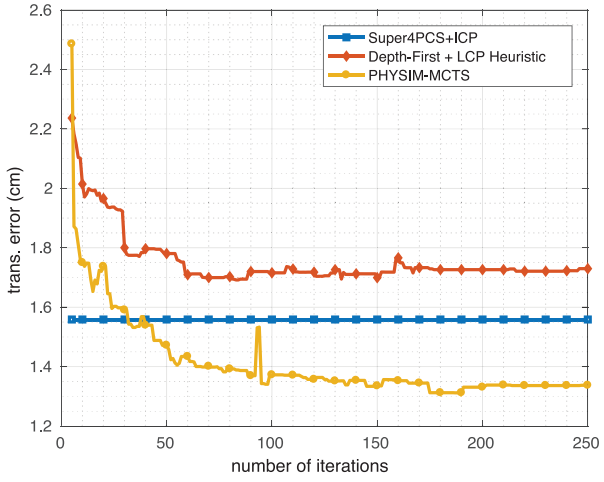

Fig. 11. Translation error in centimeters as a function of the number of iterations.

pipeline already provides enough information for this pose estimation process. Some of the reasons for failures in pose estimation when evaluated with ground-truth segmentation were found to be because of pose averaging when using cluster centers as pose representatives resulting in the failure of local optimization and when the depth sensor did not return points for some reflective surfaces.

6.3. Evaluating over benchmark for pose estimation

In this section, the entire pipeline is evaluated over the *Linemod* (Hinterstoisser et al., 2012) and the *Linemod-Occluded* (Brachmann et al., 2014) datasets. Evaluation is performed according to the benchmark (Hodan et al., 2018) for eight objects as shown in Figure 13, which have corresponding ground-truth pose labels in both the datasets. The accuracy is measured in terms of the visual surface discrepancy (VSD) metric as defined in Hodan et al. (2018) with a misalignment tolerance of $\tau = 20$ mm and correctness threshold $\theta = 0.3$. Given these parameters, the error is calculated by rendering the object model at the predicted and the ground-truth pose as depth maps S and S'. These are compared with the actual depth map of the image to obtain visibility masks V and V' and the error is calculated as

$$e_{vsd} = \underset{p \in V \cap V'}{avg} \begin{cases} 0, & \text{if } p \in V \cap V' \wedge |S(p) - S'(p)| < \tau \\ 1, & \text{otherwise} \end{cases}$$

A pose is counted as correct if $e_{vsd} < \theta$. Finally the recall rate per object and over the entire dataset are presented in the Table 6.

To compare the proposed approach, first synthetic training data was generated based on the developed pipeline. Some examples of the generated images are shown in Figure 13. Overall 30,000 RGB and corresponding depth

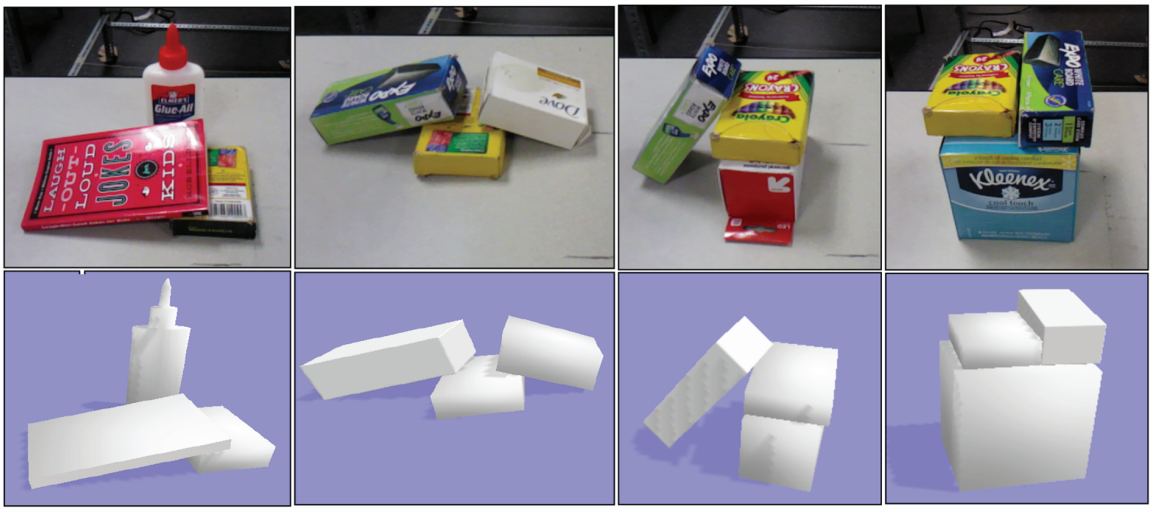

Fig. 12. Example images from *Extended Rutgers RGBD dataset* and accompanying results from the MCTS process. The results are visualized in the lightweight physics engine (Bullet), which plays an integral part in performing the local optimization in this pipeline and ensures that the returned results are physically stable configurations.

Table 6. Evaluating the performance of the proposed search process on the *Linemod* and *Linemod-Occluded* dataset according to the recent benchmark (hodan2018bop) in pose estimation.

	Linemod (recall %)								
	1	5	6	8	9	10	11	12	All
Hodaň et al. (2015)	91	91	97	69	90	97	81	79	86.9
MCTS	93	90	87	90	80	97	80	65	85.3
Vidal et al. (2018)	89	92	96	89	87	97	59	69	84.8
Drost et al. (2010)	86	93	87	92	66	96	53	67	80.0
Drost et al. (2010) (Edge)	77	98	94	96	45	94	68	66	79.8
Brachmann et al. (2016)	91	86	90	72	85	79	46	67	77.0
Hoda_n et al. (2015) (NR)	91	66	87	49	92	90	65	63	75.4
Brachmann et al. (2014)	74	88	66	81	69	66	50	75	71.1
Kehl et al. (2016)	60	79	68	68	42	91	45	42	61.9
Buch et al. (2017) (ppfh)	77	84	60	59	75	67	24	39	60.6
Buch et al. (2017) (si)	40	81	47	8	36	43	18	3	34.5
Buch et al. (2017) (ecsad)	31	66	3	0	9	49	1	0	19.9
Tejani et al. (2014)	36	1	0	11	1	70	27	0	18.3
Buch et al. (2016) (ppfh)	11	3	7	7	18	12	4	3	8.1
Buch et al. (2017) (shot)	3	9	4	3	2	10	1	0	4.0
Buch et al. (2016) (ecsad)	2	5	0	4	5	8	0	0	3.0
SL-MCTS	—	—	—	—	—	—	—	—	—
	Linemod-Occluded (recall %)								
	1	5	6	8	9	10	11	12	All
SL-MCTS	50	71	43	68	72	46	33	66	60.3
Vidal et al. (2018)	66	81	46	65	73	43	26	64	59.3
MCTS	48	59	35	78	71	48	32	65	58.4
Drost et al. (2010)	62	75	39	70	57	46	26	57	55.4
Drost et al. (2010) (Edge)	47	82	46	75	42	44	36	57	55.0
Brachmann et al. (2016)	64	65	44	68	71	3	32	61	52.0
Hodaň et al. (2015)	54	66	40	26	73	37	44	68	51.4
Brachmann et al. (2014)	50	48	27	44	60	6	30	62	41.5
Buch et al. (2017) (ppfh)	59	63	18	35	60	17	5	30	37.0
Hoda_n et al. (2015) (NR)	47	35	24	12	63	9	32	53	34.4
Kehl et al. (2016)	39	47	24	30	48	14	13	49	33.9
Buch et al. (2017) (si)	54	63	11	2	16	9	1	3	20.4
Buch et al. (2017) (ecsad)	29	29	0	0	7	8	1	0	9.6
Tejani et al. (2014)	26	2	0	1	0	0	10	0	4.5
Buch et al. (2016) (ppfh)	4	0	0	2	11	1	1	1	2.3
Buch et al. (2017) (shot)	2	7	0	0	1	1	1	0	1.5
Buch et al. (2016) (ecsad)	1	3	0	2	2	0	0	0	1.0

images are generated along with per pixel class labels. To generate this dataset, the intrinsic camera parameters, object texture, and pose of the table are kept constant. The pose of the object over the table is varied randomly over x, y position and yaw while the rest of the pose parameters are kept constant. Finally, physics simulation is applied to get a physically consistent scene which is rendered from 20 different viewpoints. The viewpoints are sampled randomly from a hemisphere of radius varying in a range similar to the test dataset. The camera sampling policy and range values are similar to those used for generating the training data in Hodan et al. (2018). Other scene parameters such as light position, light color, object material emission, background, and texture of the table are varied randomly within a pre-specified domain.

An FCN (Long et al., 2015) is trained with the generated data to obtain pixel-level classification and the output is used to guide the pose estimation process. The choice of using an FCN instead of Faster-RCNN was due to the fact that several unknown objects are present in the scene and predicting one definite location for an object in the scene would reduce the recall rate for the recognition task. In the *Linemod* dataset only one object needs to be estimated in each frame. To perform this task, first the pose of the table is computed using a RANSAC-based process and the direction of gravity is assumed to be perpendicular to the surface of the table. Then, 50 pose candidates are considered for the object based on the segmentation output and each of these are locally optimized based on physics simulation and ICP in the MCTS process. Finally, the score is computed to select the best

Fig. 13. Performing pose estimation over *Linemod* and *Linemod-Occluded* datasets. The visualizations demonstrate (left) the object models and final result of the pose estimation process on RGB-D data, (middle) the training data generated from the proposed pipeline, and (right) some instances of successful estimates as well as failure cases on the *Linemod-Occluded* dataset with high level of occlusion.

candidate. Owing to the presence of unmodeled clutter, the optimization cost cannot assume that the entire scene can be explained by the estimated pose of known objects. Thus, the optimization cost for this dataset is set so as to maximize only the alignment of the rendered depth map of the object at the predicted pose with the observed depth map. The alignment is computed with a distance threshold of 10 mm and a surface normal tolerance of 30°. The surface normal is used to avoid cases where the objects are falsely assigned to parts of large flat surfaces.

On the *Linemod-Occluded* dataset, pose for eight objects need to be estimated in every image with a high level of occlusion. Two separate tests are performed on this dataset. In the first experiment the FCN is trained with just synthetic data from the proposed pipeline and the output is used to guide the MCTS process to estimate the pose for all eight objects present in the scene. An example of the prediction is visualized in at the bottom-left of Figure 13. In the second experiment, the FCN is re-trained with additional images from the *Linemod* dataset which are labeled using the confident estimates from the pose estimation over the entire dataset and projected to all the different views. Note that in this case only the segment corresponding to one object could be extracted from each image of the *Linemod* dataset, so a mask is used during the training process to only use that small part of the image which corresponds to the object and ignores the rest. This presents only positive samples for training on real data and thus not a very significant improvement can be seen from this task. The performance corresponding to this experiment is referred to as the MCTS-SL in Table 6.

Overall, the proposed approach achieves state-of-the-art performance on both of these datasets. On the *Linemod* dataset, the proposed pipeline which is just trained on synthetic data achieves 85.3% accuracy that is just slightly below the template matching work of Hodaň et al. (2015) (86.9%) in terms of overall success. Although template matching works well in cases of less occlusion, it fails to achieve a high recall on occluded datasets. Thus, on the *Linemod-Occluded* dataset, our proposed approach achieves the highest recall rate of 60.3% when the entire pipeline is used. When the self-learning component is not used, the performance is still just slightly below the top performing method of Vidal et al. (2018).

Some examples of the successful estimates and failure conditions on this dataset are presented in Figure 13. One of the cases for failure is the presence of unmodeled objects on which the target object are physically dependent (Eggbox object in the bottom-right corner of the figure). The other failure case is that of object models getting good alignment scores with similar looking and large surfaces in the image (first three failure cases in the figure).

6.4. Limitations

One of the limitations of global reasoning, as in this approach, is the time required for computing and searching over an extensive hypotheses set. In particular, owing to the hierarchical clustering approach that was adapted to consider object specific distances, the hypotheses generation time for an object can be in the order of multiple seconds. The search process, which seemed to converge to good solutions with 150 expansions for three-object dependencies, takes approximately 30 seconds. Nevertheless, both of these processes are highly parallelizable. Future work can perform the hypotheses generation and the search with parallel computing. Another limitation of this work in the current form is the assumption that the objects are non-transparent and rigid. For transparent objects, this is due to the lack of depth data on the surface of these objects.

7. Discussion

This work provides a comprehensive framework for 6- DoF pose estimation of objects placed in clutter. It leverages the advantages of recent success in deep learning without the need for any manual effort in data collection and labeling.

It offers a novel way of performing pose estimation for objects placed in clutter by efficiently searching for the best scene explanation over the space of physically consistent scene configurations. It also provides a method to construct these sets of scene configurations by using state-of-the-art object detection and model registration techniques, which by themselves are not sufficient to give a desirable pose estimate for objects. The evaluations indicate significant performance improvement in both the tasks of object detection and pose estimation using the proposed approach. The limitations mentioned in the previous section encourage future work on fast and robust hypotheses generation and developing a method to systematically and quickly cluster object poses in SE(3), while taking into consideration the symmetries of objects. There is also a wide interest in bridging the domain gap between simulated and real images by domain randomization (Tobin et al., 2017) or with a generative learning technique (Shrivastava et al., 2017). The current work could leverage such techniques to provide an even better initialization to this process.

ORCID iDs

Chaitanya Mitash https://orcid.org/0000-0002-8547-2634
Kostas Bekris https://orcid.org/0000-0002-0675-3324

References

Aiger D, Mitra NJ and Cohen-Or D (2008) 4-points congruent sets for robust pairwise surface registration. *ACM Transactions on Graphics* 27: 85.

Akizuki S and Hashimoto M (2016) Physical reasoning for 3d object recognition using global hypothesis verification. In: *Computer Vision–ECCV 2016 Workshops*. New York: Springer, pp. 595–605.

Aldoma A, Marton ZC, Tombari F, et al. (2012a) Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation. *IEEE Robotics and Automation Magazine* 19(3): 80–91.

Aldoma A, Tombari F, Di Stefano L and Vincze M (2012b) A global hypotheses verification method for 3D object recognition. In: *European Conference on Computer Vision*. Berlin: Springer.

Aldoma A, Tombari F, Prankl J, Richtsfeld A, Di Stefano L and Vincze M (2013) Multimodal cue integration through hypotheses verification for RGB-d object recognition and 6DOF pose estimation. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 2104–2111.

Arthur D and Vassilvitskii S (2007) *k*-means + +: The advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, pp. 1027–1035.

Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* 13(2): 111–122.

Besl PJ and McKay ND (1992) *Method for Registration of 3D Shapes*. International Society for Optics and Photonics.

Birdal T and Ilic S (2015) Point pair features based object detection and pose estimation revisited. In: *2015 International Conference on 3D Vision (3DV)*. IEEE, pp. 527–535.

Bo L, Ren X and Fox D (2014) Learning hierarchical sparse features for RGB-(D) object recognition. *The International Journal of Robotics Research* 33(4): 581–599.

Bouazix S, Tagliasacchi A and Pauly M (2013) Sparse iterative closest point. *Computer Graphics Forum* 32(5): 1–11.

Brachmann E, Krull A, Michel F, Gumhold S, Shotton J and Rother C (2014) Learning 6D object pose estimation using 3d object coordinates. In: *European Conference on Computer Vision*. Berlin: Springer, pp. 536–551.

Brachmann E, Michel F, Krull A, et al. (2016) Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3364–3372.

Buch AG, Kiforenko L and Kraft D (2017) Rotational subgroup voting and pose clustering for robust 3D object recognition. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 4137–4145.

Buch AG, Petersen HG and Krüger N (2016) Local shape feature fusion for improved matching, pose estimation and 3D object recognition. *SpringerPlus* 5(1): 297.

Cao Z, Sheikh Y and Banerjee NK (2016) Real-time scalable 6DOF pose estimation for textureless objects. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2441–2448.

Cheng ZQ, Chen Y, Martin R, Lai YK and Wang A (2013) Supermatching: Feature matching using supersymmetric geometric constraints. In: *IEEE Transactions on Visualization and Computer Graphics* 19: 11.

Chetverikov D, Svirko D, Stepanov D and Krsek P (2002) The trimmed iterative closest point algorithm. In: *Proceedings 16th International Conference on Pattern Recognition*, Vol. 3. IEEE, pp. 545–548.

Choi C and Christensen HI (2012) 3D pose estimation of daily objects using an RGB-d camera. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3342–3349.

Collet A, Martinez M and Srinivasa S (2011) The MOPED framework: Object recognition and pose estimation for manipulation. *The International Journal of Robotics Research* 30(10): 1284–1306.

Correll N, Bekris KE, Berenson D, et al. (2016) Analysis and observations from the first Amazon Picking Challenge. *IEEE Transactions on Automation Science and Engineering* 15(1): 172–188.

Dhillon IS, Guan Y and Kulis B (2004) Kernel *k*-means: Spectral clustering and normalized cuts. In: *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM Press, pp. 551–556.

Drost B and Ilic S (2012) 3D object detection and localization using multimodal point pair features. In: *Second International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pp. 9–16.

Drost B, Ulrich M, Navab N and Ilic S (2010) Model globally, match locally: Efficient and robust 3D object recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 998–1005.

Durrant-Whyte H and Bailey T (2006) Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine* 13(2): 99–110.

Erkent O, Shukla D and Piater J (2016) Integration of probabilistic pose estimates from multiple views. In: *European Conference on Computer Vision (ECCV)*.

Fischler MA and Bolles RC (1981a) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6): 381–395.

Fischler MA and Bolles RC (1981b) Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM* 24(6): 381–395.

Gelfand N, Mitra N, Guibas L and Pottmann H (2005) Robust global registration. In: *Proceedings of the Third Eurographics Symposium on Geometry Processing*.

Hernandez C, Bharatheesha M, Ko W, et al. (2016) Team Delft's robot winner of the Amazon Picking Challenge 2016. In: *Robot World Cup*. Berlin: Springer, pp. 613–624.

Hinterstoisser S, Lepetit V, Ilic S, et al. (2012) Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: *Asian Conference on Computer Vision*. Berlin: Springer, pp. 548–562.

Hinterstoisser S, Lepetit V, Rajkumar N and Konolige K (2016) Going further with point pair features. In: *European Conference on Computer Vision*. Berlin: Springer, pp. 834–848.

Hodan T, Michel F, Brachmann E, et al. (2018) BOP: Benchmark for 6D object pose estimation. *European Conference on Computer Vision (ECCV 2018)*, pp. 19–35.

Hodaň T, Zabulis X, Lourakis M, ObdržálekŠ and Matas J (2015) Detection and fine 3D pose estimation of texture-less objects in RGB-d images. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 4421–4428.

Irani S and Raghavan P (1996) Combinatorial and experimental results for randomized point matching algorithms. In: *Proceedings of the Symposium on Computational Geometry*, pp. 68–77.

Izadi S, Kim D, Hilliges O, et al. (2011) Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. New York: ACM Press, pp. 559–568.

Johnson AE and Hebert M (1999) Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(5): 433–449.

Kehl W, Manhardt F, Tombari F, Ilic S and Navab N (2017) SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1521–1529.

Kehl W, Milletari F, Tombari F, Ilic S and Navab N (2016) Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: *European Conference on Computer Vision (ECCV)*, pp. 205–220.

Kim E and Medioni G (2011) 3D object recognition in range images using visibility context. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3800–3807.

Kimmel A, Dobson A, Littlefield Z, Krontiris A, Marble J and Bekris K (2012) PRACSYS: An extensible architecture for composing motion controllers and planners. In: *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pp. 137–148.

Kocsis L and Szepesvári C (2006) Bandit based Monte-Carlo planning. In: *ECML*, Vol. 6. Berlin: Springer, pp. 282–293.

Krull A, Brachmann E, Michel F, Ying Yang M, Gumhold S and Rother C (2015) Learning analysis-by-synthesis for 6D pose estimation in RGB-d images. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 954–962.

Littlefield Z, Krontiris A, Kimmel A, Dobson A, Shome R and Bekris KE (2015) An extensible software architecture for composing motion and task planners. In: *International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*.

Long J, Shelhamer E and Darrell T (2015) Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.

Lowe DG (1999) Object recognition from local scale-invariant features. In: *IEEE International Conference on Computer Vision (ICCV)*, Vol. 2, pp. 1150–1157.

McCormac J, Clark R, Bloesch M, Davison A and Leutenegger S (2018) Fusion++: Volumetric object-level SLAM. In: *2018 International Conference on 3D Vision (3DV)*. IEEE, pp. 32–41.

McCormac J, Handa A, Leutenegger S and Davison AJ (2017) Scenenet RGB-d: Can 5m synthetic images beat generic ImageNET pre-training on indoor segmentation. In: *Proceedings of the International Conference on Computer Vision (ICCV)*, Vol. 4.

Mellado N, Aiger D and Mitra NJ (2014) Super4PCS fast global pointcloud registration via smart indexing. *Computer Graphics Forum* 33: 205–215.

Michel F, Kirillov A, Brachmann E, et al. (2017) Global hypothesis generation for 6D object pose estimation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 462–471.

Mitash C, Bekris KE and Boularias A (2017) A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 545–551.

Mitash C, Boularias A and Bekris KE (2018) Improving 6D pose estimation of objects in clutter via physics-aware Monte Carlo tree search. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1–8.

Mitra N, Gelfand N, Pottmann H and Guibas H (2004) Registration of point cloud data from a geometric optimization perspective. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pp. 22–31.

Movshovitz-Attias Y, Kanade T and Sheikh Y (2016) How useful is photo-realistic rendering for visual learning? In: *ECCV 2016 Workshops*.

Narayanan V and Likhachev M (2016) Discriminatively-guided deliberative perception for pose estimation of multiple 3d object instances. In: *Robotics: Science and Systems*.

Papazov C and Burschka D (2010) An efficient RANSAC for 3D object recognition in noisy and occluded scenes. In: *Asian Conference on Computer Vision*. Berlin: Springer, pp. 135–148.

Pavlakos G, Zhou X, Chan A, Derpanis G and Daniilidis K (2017) 6-DOF object pose from semantic keypoints. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Peng X, Sun B, Ali K and Saenko K (2015) Learning deep object detectors from 3D models. In: *IEEE International Conference on Computer Vision*.

Pillai S and Leonard JJ (2015) Monocular SLAM supported object recognition. In: *Robotics: Science and Systems*.

Redmon J, Divvala S, Girshick R and Farhadi A (2016) You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779–788.

Ren S, He K, Girshick R and Sun J (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. In: *Advances in Neural Information Processing Systems*, pp. 91–99.

Rennie C, Shome R, Bekris KE and De Souza AF (2016) A dataset for improved RGBD-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robotics and Automation Letters* 1(2): 1179–1185.

Rothganger F, Lazebnik S, Schmid C and Ponce J (2006) 3D object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision* 66(3): 231–259.

Rusinkiewicz S and Levoy M (2001) Efficient variants of the ICP algorithm. In: *IEEE Proceedings of 3DIM*, pp. 145–152.

Rusu RB, Blodow N and Beetz M (2009) Fast point feature histograms (FPFH) for 3D registration. In: *IEEE International Conference on Robotics and Automation, 2009 (ICRA'09)*. IEEE, pp. 3212–3217.

Salas-Moreno RF, Newcombe RA, Strasdat H, Kelly PH and Davison AJ (2013) SLAM++: Simultaneous localisation and mapping at the level of objects. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359.

Segal A, Haehnel D and Thrun S (2009) Generalized-ICP. In: *Robotics: Science and Systems*, Vol. 2, p. 4.

Shrivastava A, Pfister T, Tuzel O, Susskind J, Wang W and Webb R (2017) Learning from simulated and unsupervised images through adversarial training. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 3, p. 6.

Simonyan K and Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations (ICLR)*.

Singh A, Sha J, Narayan KS, Achim T and Abbeel P (2014) Bigbird: A large-scale 3D database of object instances. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Srivatsan RA, Vagdargi P and Choset H (2017) Sparse point registration. In: *International Symposium on Robotics Research (ISRR)*.

Stein GJ and Roy N (2018) Genesis-RT: Generating synthetic images for training secondary real-world tasks. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7151–7158.

Su H, Qi CR, Li Y and Guibas LJ (2015) Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views. In: *IEEE International Conference on Computer Vision*.

Sun B and Saenko K (2014) From virtual to reality: Fast adaptation of virtual object detectors to real domains. In: *British Machine Vision Conference*.

Tejani A, Tang D, Kouskouridas R and Kim TK (2014) Latent-class Hough forests for 3D object detection and pose estimation. In: *European Conference on Computer Vision*. Berlin: Springer, pp. 462–477.

Thrun S, Burgard W and Fox D (2005) *Probabilistic Robotics*. Cambridge, MA: MIT Press.

Tobin J, Fong R, Ray A, Schneider J, Zaremba W and Abbeel P (2017) 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)*. IEEE, pp. 23–30.

Tombari F and Di Stefano L (2010) Object recognition in 3D scenes with occlusions and clutter by Hough voting. In: *2010 Fourth Pacific-Rim Symposium on Image and Video Technology (PSIVT)*. IEEE, pp. 349–355.

Tombari F, Salti S and Di Stefano L (2010) Unique signatures of histograms for local surface description. In: *European Conference on Computer Vision*. Berlin: Springer, pp. 356–369.

Vidal J, Lin CY and Mart R (2018) 6D pose estimation using an improved method based on point pair features. In: *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, pp. 405–409.

Wohlhart P and Lepetit V (2015) Learning descriptors for object recognition and 3D pose estimation. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Xiang Y, Schmidt T, Narayanan V and Fox D (2018) PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Robotics: Science and Systems (RSS)*.

Zeng A, Yu KT, Song S, et al. (2017) Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge. In: *IEEE International Conference on Robotics and Automation (ICRA)*.

Zhang L, Kim YJ and Manocha D (2007) C-DIST: Efficient distance computation for rigid and articulated models in configuration space. In: *Proceedings of the 2007 ACM symposium on Solid and physical modeling*. New York: ACM Press, pp. 159–169.

Zhou QY, Park J and Koltun K (2016) Fast global registration. *European Conference on Computer Vision*.

Zhu JY, Park T, Isola P and Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. In: *Proceedings of the International Conference on Computer Vision (ICCV)*.