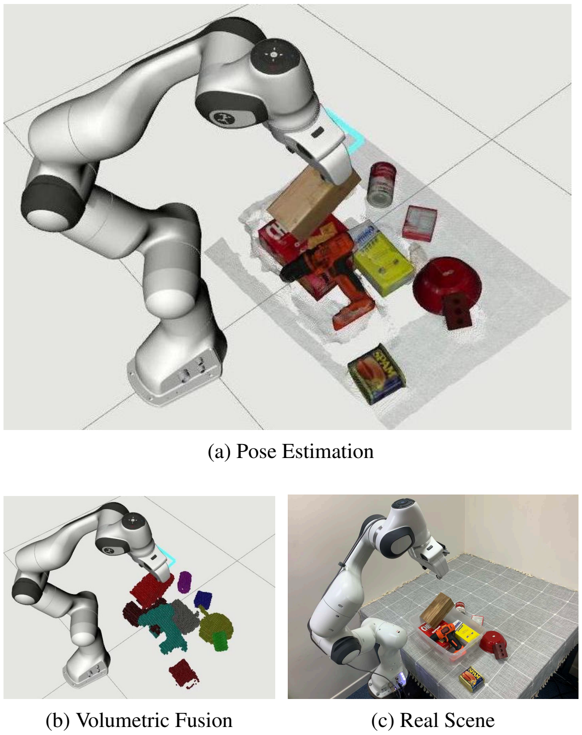
MoreFusion: Multi-object Reasoning for 6D Pose Estimation from Volumetric Fusion

Kentaro Wada, Edgar Sucar, Stephen James, Daniel Lenton, Andrew J. Davison
Dyson Robotics Laboratory, Imperial College London
{k.wada18, e.sucar18, slj12, djl11, a.davison}@imperial.ac.uk

Abstract

Robots and other smart devices need efficient object-based scene representations from their on-board vision systems to reason about contact, physics and occlusion. Recognized precise object models will play an important role alongside non-parametric reconstructions of unrecognized structures. We present a system which can estimate the accurate poses of multiple known objects in contact and occlusion from real-time, embodied multi-view vision. Our approach makes 3D object pose proposals from single RGB-D views, accumulates pose estimates and non-parametric occupancy information from multiple views as the camera moves, and performs joint optimization to estimate consistent, non-intersecting poses for multiple objects in contact.

We verify the accuracy and robustness of our approach experimentally on 2 object datasets: YCB-Video, and our own challenging Cluttered YCB-Video. We demonstrate a real-time robotics application where a robot arm precisely and orderly disassembles complicated piles of objects, using only on-board RGB-D vision.


(a) Pose Estimation

(b) Volumetric Fusion

(c) Real Scene

Figure 1: **MoreFusion** produces accurate 6D object pose predictions by explicitly reasoning about occupied and free space via a volumetric map. We demonstrate the system in a real-time robot grasping application.

1. Introduction

Robots and other smart devices that aim to perform complex tasks such as precise manipulation in cluttered environments need to capture information from their cameras that enables reasoning about contact, physics and occlusion among objects. While it has been shown that some short-term tasks can be accomplished using end-to-end learned models that connect sensing to action, we believe that extended and multi-stage tasks can greatly benefit from persistent 3D scene representations.

Even when the object elements of a scene have known models, inferring the configurations of many objects that are mutually occluding and in contact is challenging even with state-of-the-art detectors. In this paper we present a vision system that can tackle this problem, producing a persistent 3D multi-object representation in real-time from the multi-view images of a single moving RGB-D camera. Our system has four main components, as highlighted in Figure 2: 1) 2D object detection is fed to object-level fusion to make volumetric occupancy map of objects. 2) A pose prediction network that uses RGB-D data and the surrounding occupancy grid makes 3D object pose estimates. 3) Collision-based pose refinement jointly optimizes the poses of multiple objects with differentiable collision checking. 4) The intermediate volumetric representation of objects are replaced with information-rich CAD models.

Our system takes full advantage of depth information and multiple views to estimate mutually consistent object

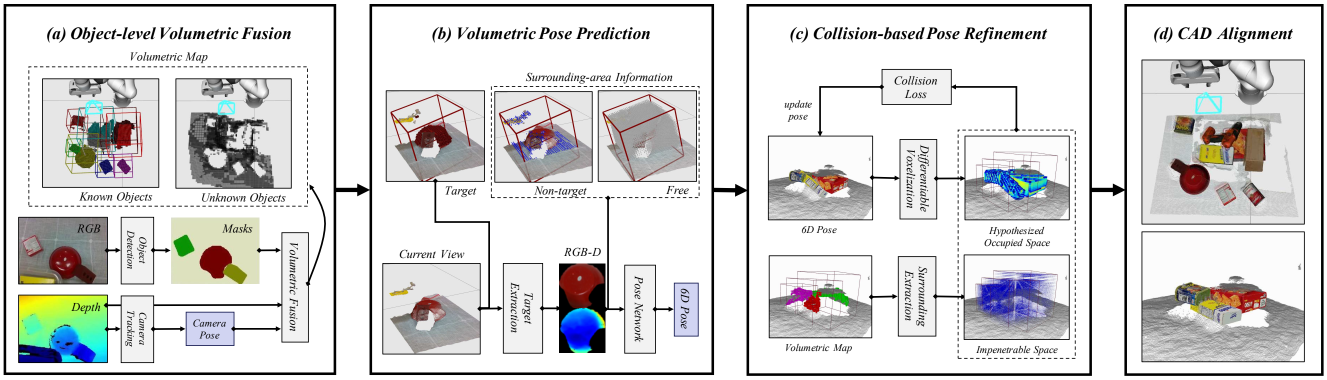


Figure 2: **Our 6D pose estimation system.** Object segmentation masks from RGB images are fused into a volumetric map , which denotes both occupied and free space (a). This volumetric map is used along with RGB-D data of a target object crop to make an initial 6D pose prediction (b). This pose is then refined via differentiable collision checking (c) and then used as part of a CAD alignment stage to enriches the volumetric map (d).

poses. The initial rough volumetric reconstruction is upgraded to precise CAD model fitting when models can be confidently aligned without intersecting with other objects. This visual capability to infer the poses of multiple objects with occlusion and contact enables robotic planning for pick-and-place in a cluttered scene *e.g*. removing obstacle objects for picking the target red box in Figure 1.

In summary, the main contributions of this paper are:

- **Pose prediction with surrounding spatial awareness**, in which the prediction network receives occupancy grid as an impenetrable space of the object;
- **Joint optimization of multi-object poses**, in which the scene configuration with multiple objects is evaluated and updated with differentiable collision check;
- **Full integration of fusion and 6D pose** as a real-time system, in which the object-level volumetric map is exploited for incremental and accurate pose estimation.

2. Related Work

Template-based methods [15, 26, 12, 11, 13, 24] are one of the earliest approaches to pose estimation. Traditionally, these methods involve generating templates by collecting images of the object (or a 3D model) from varying viewpoints in an offline training stage and then scanning the template across an image to find the best match using a distance measure. These methods are sensitive to clutter, occlusions, and lighting conditions, leading to a number of false positives, which in turn requires greater post processing. **Sparse feature-based** methods have been a popular alternative to template-based methods for a number of years [16, 21, 22]. These methods are concerned with extracting scale invaraint points of interest from images, describing them with local descriptors, such as SIFT [17] or SURF [1], and then storing them in a database to be later matched with at test time to obtain a pose estimate using a method such as RANSAC [8]. This processing pipeline can be seen in manipulation tasks, such as MOPED [5]. With the increase in affordable RGB-D cameras, **dense** methods have become increasingly popular for object and pose recognition [7, 25, 3]. These methods involve construction of a 3D point-cloud of a target object, and then matching this with a stored model using popular algorithms such as Iterative Closest Point (ICP) [2].

The use of deep neural networks is now prevalent in the field of 6D pose estimation. PoseCNN [29] was one of the first works to train an end-to-end system to predict an initial 6D object poses directly from RGB images, which is then refined using depth-based ICP. Recent RGB-D-based system are PointFusion [31] and DenseFusion [28], which individually process the two sensor modalities (CNNs for RGB, PointNet [23] for point-cloud), and then fuse them to extract pixel-wise dense feature embeddings. Our work is most closely related to these RGB-D and learning-based approaches with deep neural networks. In contrast to the point-cloud-based and target-object-focused approach in the prior work, we process the geometry using more structured volumetric representation with the geometry information surrounding the target object.

3. MoreFusion

Our system estimates the 6D pose of a set of known objects given RGB-D images of a cluttered scene. We represent 6D poses as a homogeneous transformation matrix $p \in SE(3)$, and denote a pose as $p = [\mathtt{R}|\mathbf{t}]$, where $\mathtt{R} \in SO(3)$ is the rotation and $\mathbf{t} \in \mathcal{R}^3$ is the translation.

Our system, summarized in Figure 2, can be divided into four key stages. (1) An **object-level volumetric fusion** stage which combines the object instances masks produced from an object detection along with depth measurement and camera tracking component to produce a volumetric map

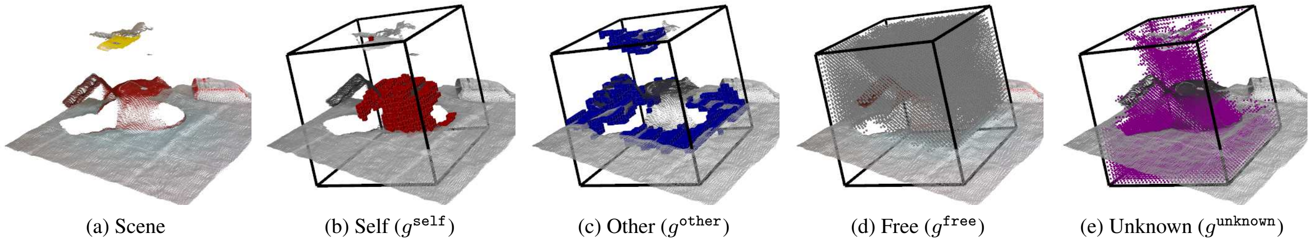

(a) Scene (b) Self ($g^{\texttt{self}}$) (c) Other ($g^{\texttt{other}}$) (d) Free ($g^{\texttt{free}}$) (e) Unknown ($g^{\texttt{unknown}}$)

Figure 3: **Surrounding spatial information.** These figures show the occupancy grid (32 × 32 × 32 voxels) of the red bowl. The free (d) and unknown (e) grids are visualized with points instead of cubes for visibilities.

known and unkown objects. (2) A **volumetric pose prediction** stage which uses the surrounding information from the volumetric map along with the RGB-D masks to produce an initial pose prediction for each of the objects. (3) A **collision-based pose refinement** stage that jointly optimizes the pose of multiple objects via gradient descent by using differentiable collision checking between object CAD models and occupied space from the volumetric map. (4) A **CAD alignment** stage that replaces the intermediate representation of each object with a CAD model, containing compact and rich information. In the following sections, we expend further on each of these stages.

3.1. Object-level Volumetric Fusion

Building a volumetric map is the first stage of our pose estimation system, which allows the system to gradually increase the knowledge about the scene until having confidence about object poses in the scene. For this object-level volumetric fusion stage, we build a pipeline similar to [18, 27, 30], combining RGB-D camera tracking, object detection, and volumetric mapping of detected objects.

RGB-D Camera Tracking Given that the camera is mounted on the end of a robotic arm, we are able to retrieve the accurate pose of the cameras using forward kinematics and a well-calibrated camera. However, to also allow this to be used with a hand-held camera, we adopt the sparse SLAM framework ORB-SLAM2 [20] for camera tracking. Unlike its monocular predecessor [19], ORB-SLAM2 tracks camera pose in metric space, which is crucial for the volumetric mapping.

Object Detection Following the prior work [18, 30], RGB images are passed to Mask-RCNN[9] which produce 2D instance masks.

Volumetric Mapping of Detected Objects We use octree-based occupancy mapping, OctoMap[14], for the volumetric mapping. By using octree structure, OctoMap can quickly retrieve the voxel from queried points, which is critical for both updating the occupancy value from depth measurements and checking occupancy value when use in the later (pose prediction and refinement) components of the pipeline.

We build this volumetric map for each detected objects including unknown (background) objects. In order to track the objects that are already initialized, we use the intersect over union of the detected mask in the current frame and rendered mask current reconstruction following prior work [18, 30]. For objects that are already initialized, we fuse new depth measurements to the volumetric map of that object, and a new volumetric map is initialized when it finds a new object when moving the camera. This object-level reconstruction enables to use volumetric representation of objects as an intermediate representation before the model alignment by pose estimation.

3.2. Volumetric Pose Prediction

Our system retrieves surrounding information from the volumetric map to incorporate spatial awareness of the area surrounding a target object into pose prediction. In this section, we describe how this surrounding information is represented and used in pose prediction.

3.2.1 Occupancy Grids as Surrounding Information

Each target object (o_m) for pose prediction carries its own volumetric occupancy grid. The voxels that make up this grid can be in one of the following states: (1) Space occupied by the object itself ($g^{\texttt{self}}$) from the target object reconstruction. (2) Space occupied by other objects ($g^{\texttt{other}}$) from reconstruction of the surrounding objects. (3) Free space ($g^{\texttt{free}}$) identified by depth measurement. (4) Unknown space ($g^{\texttt{unknown}}$) unobserved by mapping because of occlusion and sensor range limit (Figure 3).

Ideally, the bounding box of surrounding information should cover the whole area of the target object even if it is occluded. This means the bounding box size should change depending on the target object size. Since we need to use fixed voxel dimension for network prediction (e.g., 32 × 32 × 32), we use different voxel size for each object computed from the object model size (diagonal of the bounding box divided by the voxel dimension).

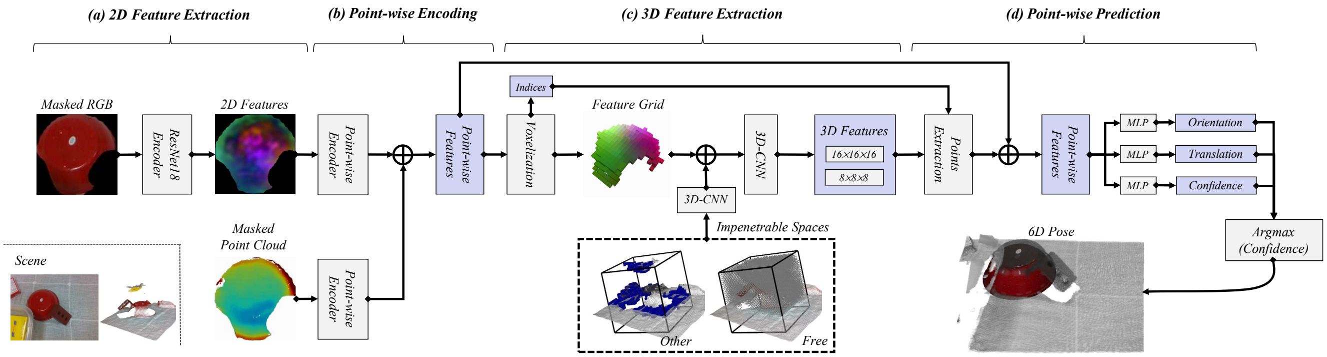


Figure 4: **Network architecture,** which performs pose prediction using masked RGB-D of the target object with its surrounding information as a occupancy grid.

3.2.2 Pose Prediction Network Architecture

The initial 6D pose of each object is predicted via a deep neural network (summarized in Figure 4) that accepts both the occupancy grid information described in §3.2.1 and masked RGB-D images. The architecture can be categorized into 4 core components: (1) 2D feature extraction from RGB from a ResNet; (2) Point-wise encoding of RGB features and point cloud; (3) Voxelization of the point-wise features followed by 3D-CNNs; (4) Point-wise pose prediction from both 2D and 3D features.

2D Feature Extraction from RGB Even when depth measurement are available, RGB images can still carry vital sensor information for precise pose prediction. Because of the color and texture detail in RGB images, this can be an especially strong signal for pose prediction of highly-textured and asymmetric objects.

Following [28, 31], we use ResNet18 [10] with succeeding upsampling layers [32] to extract RGB features from masked images. Though both prior methods [28, 31] used cropped images of objects with a bounding box, we used masked images which makes the network invariant to changes in background appearance, and also encourages it to focus on retrieving surrounding information using the occupancy grid.

Point-wise Encoding of RGB Features and Point Cloud Similarly to [28], both the RGB features and extracted point-cloud points (using the target object mask) are encoded via several fully connected layers to produce point-wise features, which are then concatenated.

Voxelization and 3D-CNN Processing From these point-wise features we build a feature grid (with the same dimensions as the occupancy grid), which will be combined with the occupancy grid extracted from the volumetric fusion. The concatenated voxel grid is processed by 3D-CNNs to extract hierarchical 3D features reducing voxel dimension and increasing the channel size. We process the original grid (voxel dimension: 32) with 2-strided convolutions to have hierarchical features (voxel dimension: 16, 8).

An important design choice in this pipeline is to perform 2D feature extraction before voxelization, instead of directly applying 3D feature extraction on the voxel grid of raw RGB pixel values. Though 3D CNNs and 2D CNNs have similar behaviour when processing RGB-D input, it is hard to use a 3D CNN on a high resolution grid unlike a 2D image, and also the voxelized grid can have more missing points than an RGB image because of sensor noise in the depth image.

Point-wise Pose Prediction from 2D-3D Features To combine the 2D and 3D features for pose prediction, we extract points from the 3D feature grid that corresponds to the point-wise 2D features with triliner interpolation. These 3D and 2D features are concatenated as point-wise feature vectors for the pose prediction, from which we predict both the pose and confidence as in [28].

3.2.3 Training the Pose Prediction Network

Training Loss For point-wise pose prediction, we follow DenseFusion [28] for training loss which is extended version of the model alignment loss from PoseCNN [29]. For each pixel-wise prediction, this loss computes average distance of corresponding points of the object model transformed with ground truth and predicted pose (pose loss).

Let $[\mathtt{R}|\mathbf{t}]$ be ground truth pose, $[\hat{\mathtt{R}}_i|\hat{\mathbf{t}}_i]$ be i-th point-wise prediction of the pose, and $p_q \in X$ be the point sampled from the object model. This pose loss is formulated as:

$$L_i = \frac{1}{|X|} \sum_q ||(\mathtt{R}p_q + \mathbf{t}) - (\hat{\mathtt{R}}_i p_q + \hat{\mathbf{t}}_i)||. \tag{1}$$

For symmetric objects, which have ambiguity for the correspondence in object model, nearest neighbor of transformed point is used as the correspondence (symmetric pose loss):

$$L_i = \frac{1}{|X|} \sum_q \min_{p_{q'} \in X} ||(\mathtt{R}p_q + \mathbf{t}) - (\hat{\mathtt{R}}_i p_{q'} + \hat{\mathbf{t}}_i)||. \tag{2}$$

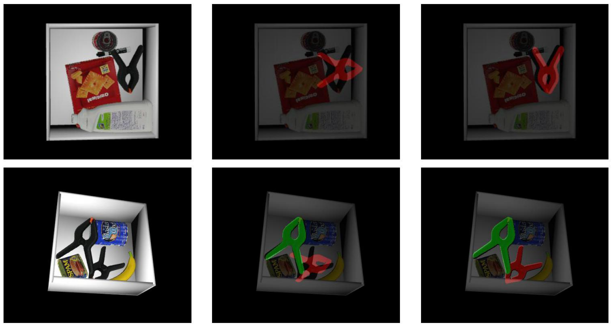
The confidence of the pose prediction is trained with these pose loss in an unsupervised way. Let N be number of pixel-wise predictions and c_i be the i-th predicted confidence. The final training loss L is formulated as:

$$L = \frac{1}{N} \sum_i (L_i c_i - \lambda \log(c_i)), \tag{3}$$

where λ is the regularization scaling factor (we use $\lambda = 0.015$ following [28]).

Local Minima in Symmetric Pose Loss Though the symmetric pose loss is designed to handle symmetric objects using nearest neighbour search, we found that this loss is prone to be stuck to local minima compared to the standard pose loss, which uses 1-to-1 ground truth correspondence in the object model. Figure 5b shows the examples where the symmetric pose loss has a problem with the local minima with the non-convex shaped object.

For this issue, we introduce warm-up stage with standard pose loss (*e.g.* 1 epoch) during training before switching to symmetric pose loss. This training strategy with warm-up allows the network first to be optimized for the pose prediction without local minima problem though ignoring symmetries, and then to be optimized considering the symmetries, which gives much better results for pose estimation of complex-shaped symmetric objects (Figure 5c).



(a) Scene (b) Symmetric pose loss (c) With loss warm-up

Figure 5: **Avoiding local minima with loss warm-up.** Our loss warm-up (c) gives much better pose estimation for complex-shaped (*e.g.* non-convex) symmetric objects, for which symmetric pose loss (b) is prone to local minima.

3.3. Collision-based Pose Refinement

In the previous section, we showed how we combine image-based object detections, RGB-D data and volumetric estimates of the shapes of nearby objects to make per-object pose predictions from a network forward pass. This can often give good initial pose estimates, but not necessarily a mutually consistent set of estimates for objects which are in close contact with each other. In this section we therefore introduce a test-time pose refinement module that can jointly optimize the poses of multiple objects.

For joint optimization, we introduce differentiable collision checking, composing of occupancy voxelization of the object CAD model and an intersection loss between occupancy grids. As both are differentiable, it allows us to optimize object poses using gradient descent with optimized batch operation on a GPU.

Differentiable Occupancy Voxelization The average voxelization of feature vectors mentioned in §3.2.2 uses feature vectors using points and is differentiable with respect to the feature vector. In contrast, the occupancy voxelization needs to be differentiable with respect to the points. This means the values of each voxel in the occupancy grid must be a function of the points, which has been transformed by estimated object pose.

Let p_q be a point, s be the voxel size, and l be the origin of the voxel (*i.e.* left bottom corner of the voxel grid). We can transform the point into voxel coordinate with:

$$u_q = (p_q - l)/s. \tag{4}$$

For each voxel v_k we compute the distance δ against the point:

$$\delta_{qk} = ||u_q - v_k||. \tag{5}$$

We decide the occupancy value based proportional to the distance from nearest point, resulting in the occupancy value o_k of k-th voxel being computed as:

$$\delta_k = \min(\delta^t, \min_q(\delta_{qk})) \tag{6}$$

$$o_k = 1 - (\delta_k / \delta^t), \tag{7}$$

where δ^t is the distance threshold.

Occupancy Voxelization for a Target Object This differentiable occupancy voxelization gives occupancy grids from object model and hypothesized object pose. For a target object o_m, the points sampled from its CAD model p_q are transformed with the hypothesized pose $(\mathtt{R}_m|\mathbf{t}_m)$: $p_q^{\mathtt{T}} = \mathtt{R}_m \mathbf{x}_q + \mathbf{t}_m$, from which the occupancy value is computed. The point is uniformly sampled from the CAD model (including internal part), and gives a hypothesized occupancy grid of the target object $g_m^{\mathtt{target}}$.

Similarly, we perform this voxelization with the surrounding objects $\widetilde{o}_n$. Unlike the target object voxelization, surrounding objects $\widetilde{o}_n$ are voxelized in the voxel coordinate of the target: $u_q^{\widetilde{o}} = (p_q^{\widetilde{o}} - l_o)/s_o$ where l_o is the occupancy grid origin of the target object and s_o is its voxel size. This gives the hypothesized occupancy grids of surrounding objects of the target object: $g_n^{\mathtt{nontarget}}$.

Intersection Loss for Collision Check The occupancy voxelization gives the hypothesized occupied space of the target $g_m^{\mathtt{target}}$ (m-th object in the scene) and surrounding objects $g_n^{\texttt{nontarget}}$. The occupancy grids of surrounding objects are built in the voxel coordinate (center, voxel size) of the target object and aggregated with element-wise max:

$$g_m^{\texttt{nontarget}} = \max_n g_n^{\texttt{nontarget}}. \tag{8}$$

This gives a single impenetrable occupancy grid, where the target object pose is penalized with intersection. In addition to the impenetrable occupancy grid from the pose hypothesis of surrounding objects, we also use the occupancy information from the volumetric fusion: occupied space including background objects $g_m^{\texttt{other}}$, free space $g_m^{\texttt{free}}$ (Figure 3), as additional impenetrable area: $g_m^{\texttt{impen}} = g_m^{\texttt{other}} \cup g_m^{\texttt{free}}$. The collision penalty loss $L_i^{\texttt{c}-}$ is computed as the intersection between hypothesized occupied space of the target and the impenetrable surrounding grid:

$$g_m^{\texttt{target}-} = \max_k (g_m^{\texttt{nontarget}}, g_m^{\texttt{impen}}) \tag{9}$$

$$L_m^{\texttt{c}+} = (g_m^{\texttt{target}} \odot g_m^{\texttt{target}-}))/\sum_k g_m^{\texttt{target}}, \tag{10}$$

where $\odot$ is element-wise multiplication.

Though this loss correctly penalizes the collision among the target and surrounding objects, optimizing for this alone is not enough, as it does not take into account the visible surface constraint of the target object $g_m^{\texttt{self}}$. The other term in the loss is the intersection between the hypothesized occupied space of the target and with this grid $L_m^{\texttt{c}+}$, to encourage the surface intersection between object pose hypothesis and volumetric reconstruction:

$$L_m^{\texttt{c}-} = (g_m^{\texttt{target}} \odot g_m^{\texttt{self}})/\sum_k g_m^{\texttt{self}}. \tag{11}$$

We compute these collision and surface alignment losses for N number of objects with the batch operation on GPU, and sum them as the total loss L:

$$L = \frac{1}{N}\sum_m (L_m^{\texttt{c}+} - L_m^{\texttt{c}-}). \tag{12}$$

This loss is minimized with gradient descent allowing us to jointly optimize the pose hypothesis of multiple objects.

3.4. CAD Alignment

After performing the pose estimation and refinement, we spawn object CAD models into the map once there are enough agreements on the poses estimated in different views. To compare the object poses estimated in the different camera coordinate, we first transform those poses into the world coordinate using the tracked camera pose in camera tracking module (§3.1). Those transformed object poses are compared using the pose loss, which we also use for training the pose prediction network (§3.2.3). For the recent N pose hypothesis, we compute the pose loss for each pair, which gives $N(N-1)$ pose loss: L_i $(1 \leq i \leq N(N-1))$. We count how many pose losses are under the threshold (L^t): $M = \mathbf{count}[\![L_i < L^t]\!]$. When M reaches a threshold, we initialize the object with that agreed pose.

4. Experiments

In this section, we first evaluate how well the pose prediction (§4.2) and refinement (§4.3) performs on 6D pose estimation datasets. We then demonstrate the system running on a robotic pick-and-place task(§4.4).

4.1. Experimental Settings

Dataset We evaluate our pose estimation components using 21 classes of YCB objects [4] used in YCB-Video dataset [29]. YCB-Video dataset has been commonly used for evaluation of 6D pose estimation in prior work, however, since all of the scenes are table-top, this dataset is limited in terms of the variety of object orientations and occlusions.

To make the evaluation possible with heavy occlusions and arbitrary orientations, we built our own synthetic dataset: Cluttered YCB (Figure 6). We used a physics simulator [6] to place object models with feasible configurations from random poses. This dataset has 1200 scenes ($\texttt{train} : \texttt{val} = 5 : 1$) and 15 camera frames for each.

Metric We used the same metric as prior work [29, 28], which evaluates the average distance of corresponding points: ADD, ADD-S. ADD uses ground truth and ADD-S uses nearest neighbours as correspondence with transforming the model with the ground truth and estimated pose. These distances are computed for each object pose in the dataset, and plotted with the error threshold in x-axis and the accuracy in the y-axis. The metric is the area under the curve (AUC) using 10cm as maximum threshold for x-axis.

4.2. Evaluation of Pose Prediction

Baseline Model We used DenseFusion [28] as a baseline model. For fair comparison with our proposed model, we reimplemented DenseFusion and trained with the same settings (*e.g.* data augmentation, normalization, loss).

Table 1 shows the pose prediction result on YCB-Video dataset using the detection mask of [29], where *DenseFusion* is the official GitHub implementation [1] and *DenseFusion** is our version, which includes the warm-up loss (§3.2.3) and the centralization of input point cloud (analogues to the voxelization step in our model). We find that the addition of the two added components leads to big performance improvements. In the following evaluations, we use DenseFusion* as the baseline model.

[1] https://github.com/j96w/DenseFusion

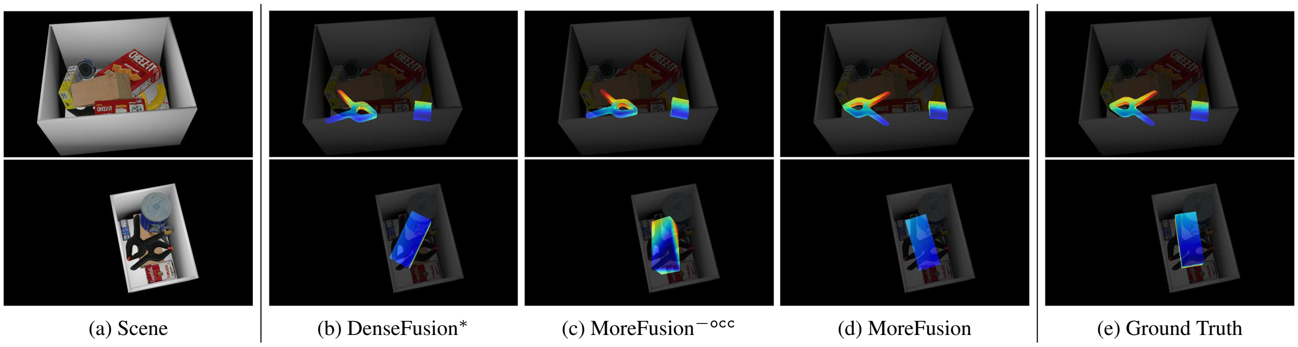

(a) Scene (b) DenseFusion* (c) MoreFusion$^{-\text{occ}}$ (d) MoreFusion (e) Ground Truth

Figure 6: **Pose prediction with severe occlusions**. Our proposed model (MoreFusion) performs consistent pose prediction with surroundings, where the baseline (DenseFusion*) and the variant without occupancy information (Morefusion$^{-\text{occ}}$) fails.

Table 1: **Baseline model results** on YCB-Video dataset, where DenseFusion is the official implementation and DenseFusion* is our reimplemented version.

Model	ADD(-S)	ADD-S
DenseFusion	83.9	90.9
DenseFusion*	89.1	93.3

Results We compared the proposed model (Morefusion) with the baseline model (DenseFusion*). For fair comparison, both models predict object poses in a single-view, where Morefusion only uses occupancy information from the single-view depth observation. We trained models using combined dataset of Cluttered-YCB and YCB-Video dataset and tested separately with ground truth masks. The result (Table 2, Figure 6) shows that Morefusion consistently predicts better poses with volumetric CNN and surrounding occupancy information. Larger improvement is performed on heavily occluded objects (visibility<30%).

Table 2: **Pose prediction comparison**, where the models are trained with the combined dataset and tested separately.

Model	Test Dataset	ADD(-S)	ADD-S
DenseFusion*	YCB-Video	88.4	94.9
MoreFusion		**91.0**	**95.7**
DenseFusion*	Cluttered YCB	81.7	91.7
MoreFusion		**83.4**	**92.3**
DenseFusion*	Cluttered YCB (visibility$^{<0.3}$)	59.7	83.8
MoreFusion		**63.5**	**85.1**

Table 3: **Effect of occupancy information** tested on Cluttered-YCB dataset with the model trained in Table 2.

Model	ADD(-S)	ADD-S
DenseFusion*	81.7	91.7
MoreFusion$^{-\text{occ}}$	82.5	91.7
MoreFusion	83.4	92.3
MoreFusion$^{+\text{target}^-}$	84.7	93.3
MoreFusion$^{+\text{target}^- +\text{bg}}$	85.5	93.8

To evaluate the effect of surrounding occupancy as input, we tested the trained model (MoreFusion) feeding different level of occupancy information: discarding the occupancy information from the single-view observation *-occ*; full reconstruction of non-target objects $+target^-$; full reconstruction of background objects *+bg*. Table 3 shows that the model gives better prediction as giving more and more occupancy information, which is very common in our incremental and multi-view object mapping system. This comparison also shows that even without occupancy information (Morefusion$^{-\text{occ}}$) our model performs better than DenseFusion* purely because of the 3D-CNNs architecture.

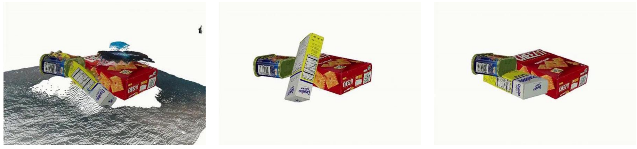
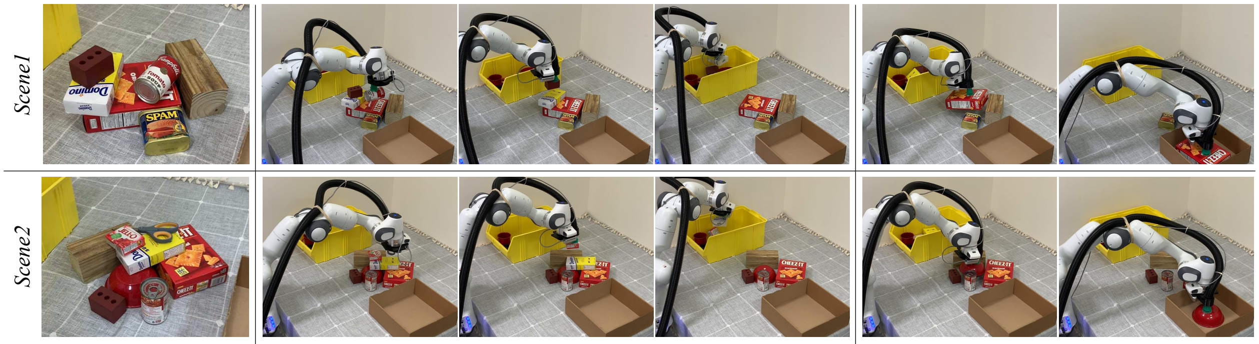

(a) No Refinement (b) ICP Refinement (c) ICC Refinement

Figure 7: **Pose refinement from intersecting object poses**, where we compare the proposed Iterative Collision Check (ICC) against Iterative Closest Point (ICP).

4.3. Evaluation of Pose Refinement

We evaluate our pose refinement, *Iterative Collision Check* (ICC), against point-to-point Iterative Closest Point (ICP) [2]. Since ICP only uses masked point cloud of the target object without any reasoning with surrounding objects, the comparison of ICC with ICP allows us to evaluate how well and in what case the surrounding-object geometry used in ICC helps pose refinement in particular.

Figure 7 shows a typical example where the pose prediction has object-to-object intersections because of less visibility of the object (e.g., yellow box). ICC refines object poses to better configurations than ICP by using the constraints from nearby objects and free-space reconstructions.



Initial state *Moving the distractor objects* *Picking the target object*

Figure 8: **Targeted pick-and-place demonstration**, where the robot must move the obstructing objects to the container, pick the target object, and then place it in the cardboard box.

For quantitaive evaluation, we used Cluttered YCB-Video dataset with pose estimate refined from initial pose prediction MoreFusion in Table 2. Figure 9 shows how the metric varies with different visibility on the dataset, in which the combination of the two methods (+ICC+ICP) gives consistently better pose than the others. With small occlusions (visibility $>= 40\%$), ICC does not perform as well as ICP because of the discrimination by the voxelization (we use 32 dimensional voxel grid). However, results are at their best with the combination of the two optimization, where ICC resolves collisions in discritized space and then ICP aligns surfaces more precisely.

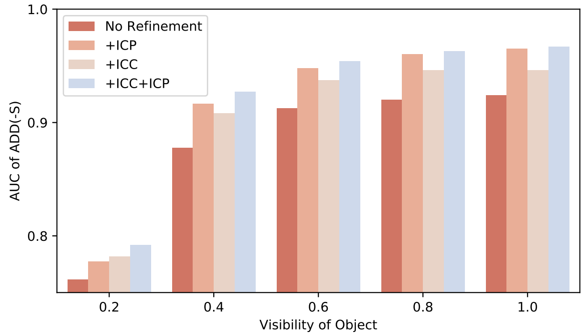


Figure 9: **Pose refinement results** on Cluttered YCB-Video, where the proposed Iterative Collision Check (ICC) gives best pose estimate combined with ICP.

4.4. Full System Demonstration

We demonstrate the capability of our full system, MoreFusion, with two demonstration: scene reconstruction, in which the system detects each known objects in the scene and aligns the pre-build object model (shown in Figure 10); and secondly, a robotic pick-and-place tasks, where the robot is requested to pick a target object from a cluttered scene with intelligently removing distractor objects to access the target object (shown in Figure 8).

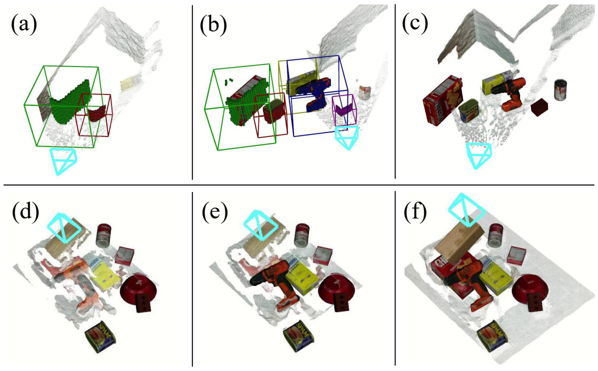


Figure 10: **Real-time full reconstruction.** Our system gradually increases the knowledge about the scene with volumetric fusion (a) and incremental CAD alignment (b) for the final reconstruction (c). The pose hypothesis of surrounding objects (*e.g.* drill, yellow box) are utilized to refine the pose predictions, to perform pose estimation of heavily occluded objects (*e.g.* red box) (d)-(f).

5. Conclusions

We have shown consistent and accurate pose estimation of objects that may be heavily occluded by and/or tightly contacting with other objects in cluttered scenes. Our real-time and incremental pose estimation system builds an object-level map that describes the full geometry of objects in the scene, which enables a robot to manipulate objects in complicated piles with intelligent of dissembling of occluding objects and oriented placing. We believe that there is still a long way to go in using known object models to make persistent models of difficult, cluttered scenes. One key future direction is to introduce physics reasoning into our optimization framework.

Acknowledgments

Research presented in this paper has been supported by Dyson Technology Ltd.

References

[1] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008. 2

[2] P. Besl and N. McKay. A method for Registration of 3D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992. 2, 7

[3] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation using 3D Object Coordinates. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. 2

[4] B. Calli, A. Singh, A. Walsman, P Srinivasa S. and, Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *International Conference on Advanced Robotics (ICAR)*, pages 510–517, 2015. 6

[5] Alvaro Collet, Manuel Martinez, and Siddhartha S Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research (IJRR)*, 30(10):1284 – 1306, 2011. 2

[6] Erwin Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 2013. 6

[7] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. 2

[8] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017. 3

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[11] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit. Gradient Response Maps for Real-Time Detection of Texture-Less Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2012. 2

[12] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolidge, N. Navab, and V. Lepetit. Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011. 2

[13] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model-Based Training, Detection and Pose Estimation of Texture-less 3D Objects in Heavily Cluttered Scenes. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2012. 2

[14] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. 3

[15] Daniel P. Huttenlocher, Gregory A. Klanderman, and William J Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(9):850–863, 1993. 2

[16] D. Lowe. Local Feature View Clustering for 3D Object Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 2

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004. 2

[18] J. McCormac, R. Clark, M. Bloesch, A. J. Davison, and S. Leutenegger. Fusion++:volumetric object-level slam. In *Proceedings of the International Conference on 3D Vision (3DV)*, 2018. 3

[19] R. Mur-Artal and J. D Tardós. ORB-SLAM: Tracking and Mapping Recognizable Features. In *Workshop on Multi View Geometry in Robotics (MVIGRO) - RSS 2014*, 2014. 3

[20] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5):1255–1262, 2017. 3

[21] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 2

[22] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 2

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 2

[24] Reyes Rios-Cabrera and Tinne Tuytelaars. Discriminatively trained templates for 3d object detection: A real time scalable approach. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013. 2

[25] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013. 2

[26] Carsten Steger. Similarity measures for occlusion, clutter, and illumination invariant object recognition. In *Joint Pattern Recognition Symposium*, 2001. 2

[27] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid. Meaningful maps with object-oriented semantic mapping. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017. 3

[28] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 4, 5, 6

[29] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018. 2, 4, 6

[30] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison, and Stefan Leutenegger. Mid-fusion: Octree-based object-level multi-instance dynamic slam. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019. 3

[31] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. PointFusion: Deep sensor fusion for 3D bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[32] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4