Robotics and Autonomous Systems 133 (2020) 103632

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Object-RPE: Dense 3D reconstruction and pose estimation with convolutional neural networks

Dinh-Cuong Hoang *, Achim J. Lilienthal, Todor Stoyanov

Centre for Applied Autonomous Sensor Systems (AASS), Orebro University, Sweden

ARTICLE INFO

Article history:
Available online 27 August 2020

Keywords:
Object pose estimation
3D reconstruction
Semantic mapping
3D registration

ABSTRACT

We present an approach for recognizing objects present in a scene and estimating their full pose by means of an accurate 3D instance-aware semantic reconstruction. Our framework couples convolutional neural networks (CNNs) and a state-of-the-art dense Simultaneous Localization and Mapping (SLAM) system, ElasticFusion (Whelan et al., 2016), to achieve both high-quality semantic reconstruction as well as robust 6D pose estimation for relevant objects. We leverage the pipeline of ElasticFusion as a backbone, and propose a joint geometric and photometric error function with per-pixel adaptive weights. While the main trend in CNN-based 6D pose estimation has been to infer object's position and orientation from single views of the scene, our approach explores performing pose estimation from multiple viewpoints, under the conjecture that combining multiple predictions can improve the robustness of an object detection system. The resulting system is capable of producing high-quality instance-aware semantic reconstructions of room-sized environments, as well as accurately detecting objects and their 6D poses. The developed method has been verified through extensive experiments on different datasets. Experimental results confirmed that the proposed system achieves improvements over state-of-the-art methods in terms of surface reconstruction and object pose prediction. Our code and video are available at https://sites.google.com/view/object-rpe.

1. Introduction

Simultaneous localization and mapping (SLAM) is a crucial enabling technology for autonomous robots. With the increasing availability of RGB-D sensors, research on visual SLAM has made giant strides in development [1–3]. These approaches achieve dense surface reconstruction of complex and arbitrary indoor scenes while maintaining real-time performance through implementations on highly parallelized hardware. However, the purely geometric map of the environment produced by classical SLAM systems is not sufficient to enable robots to reason about and manipulate their surroundings. Thus, the inclusion of rich semantic information and 6D poses of object instances within a dense map is useful for robots to effectively operate and interact with objects.

Beyond classical SLAM systems that solely provide a purely geometric map, the idea of a system that generates a dense map in which object instances are semantically annotated has attracted substantial interest in the research community [4–6]. Semantic 3D maps are important for robotic scene understanding, planning and interaction. In the case of robotic manipulation, providing accurate object poses together with semantic information are crucial for robots that have to manipulate the objects around them in diverse ways.

To obtain the 6D pose of objects, many approaches were introduced in the past [7–9]. However, because of the complexity of object shapes, measurement noise and presence of occlusions, these approaches are not robust enough in real applications. Recent work has attempted to leverage the power of deep CNNs to solve this nontrivial problem [10–12]. These techniques demonstrate a significant improvement of the accuracy of 6D object pose estimation on some popular datasets such as YCB-Video or LineMOD. Even so, due to the limitation of single-view-based pose estimation, the existing solutions generally do not perform well in cluttered environments and under large occlusions.

This paper extends our previous work [13], in which we developed a system for 6D object pose estimation that benefits from the use of an instance-aware semantic mapping system and from combining multiple predictions. Our prior work relies on a robust camera tracking method that combines adaptively weighted photometric, geometric and semantic cost terms in a single objective function. In [13] these adaptive weights are chosen on a per-image basis, while ideally they should be different for each pixel, as certain regions in the image can contain varying amounts of structure and color. Therefore, in order to improve the performance of camera tracking, in this paper we propose

* Corresponding author.
E-mail addresses: Cuong.Hoang@oru.se (D. Hoang), Achim.Lilienthal@oru.se (A.J. Lilienthal), Todor.Stoyanov@oru.se (T. Stoyanov).

https://doi.org/10.1016/j.robot.2020.103632
0921-8890/

a registration cost function with per-pixel adaptive weights. We also provide validation of the proposed algorithms on more diverse datasets. Regarding object pose estimation, intuitively by combining pose predictions from multiple camera views, the accuracy of the estimated 3D object pose can be improved. Based on this, our framework deploys simultaneously a 3D mapping algorithm to reconstruct a semantic model of the environment, and an incremental 6D object pose recovery algorithm that carries out predictions using the reconstructed model. We demonstrate that we can exploit multiple viewpoints around the same object to achieve robust and stable 6D pose estimation in the presence of heavy clutter and occlusion.

In summary, this paper contributes with:

- An instance-aware semantic mapping system that is capable of producing accurate semantic maps of room-sized environments. We improve segmentation accuracy by correcting misclassified regions using two proposed criteria which rely on location information and pixel-wise probability of the class.
- A registration cost function combining geometric and appearance cues weighted adaptively. We achieve reliable camera tracking and state-of-the-art surface reconstruction.
- A method that can be used to accurately predict the pose of objects under partial occlusion. We demonstrate that by integrating deep learning-based pose prediction into our semantic mapping system we are able to address the challenges posed by missing information due to clutter, self-occlusions, and bad reflections.

2. Related work

2.1. Dense RGB-D reconstruction

During the last years, many different mapping systems were developed in order to get high-quality reconstructions in real-time using an RGB-D camera [1,2,14–17]. Most of these approaches have a very similar processing pipeline. In the first stage, noise reduction and outlier removal are applied to the raw depth measurements and then vertex maps are generated. Additional information such as normals also might be extracted from the depth image. In the next step, the sensor pose is estimated in a frame-to-frame or frame-to-model fashion by minimizing a cost function. Finally, the surface measurements are integrated into the global scene model based on the camera pose determined in the previous stage. ElasticFusion [1] and BundleFusion [17] demonstrated that they can achieve fast and robust mapping and tracking in large environments. In this paper, we leverage the pipeline of ElasticFusion as a backbone (BundleFusion is an alternative to ElasticFusion). We propose a joint geometric and photometric error function with per-pixel adaptive weights. The weights are estimated based on textureness assessment.

2.2. Dense semantic reconstruction

Several recent works [18–21] have utilized semantic segmentation CNN architectures to obtain semantically labeled dense scene reconstruction. SemanticFusion [20] employs the real-time dense visual SLAM system ElasticFusion to provide a reliable camera pose tracking and a globally consistent map of fused surfels. In addition, the method utilizes a Bayesian update scheme to keep track of the semantic class probability distribution for each surfel and to update those probabilities based on the CNN's predictions. Similar work in [21] developed an efficient and scalable method for incrementally building a dense, semantically annotated 3D map in real-time. The authors additionally propose an efficient CNN-based semantic segmentation by refining the geometric edges on frame-wise segmentation. Both works in [20,21] illustrated that their systems do not only produce a useful semantic 3D map, but also result in an improvement in the 2D semantic labeling. However, since the above systems only consider class labels, they are unaware of object instances. To build a more meaningful map, instance-aware semantic mapping was introduced in [5,22–24]. The methods integrate deep learning-based instance segmentation and classification into a SLAM system. The resulting systems are capable of producing accurate semantic maps of room-sized environments, as well as reconstructing highly detailed object-level models. Most related to ours is the work of Runz et al. MaskFusion [6], which is able to recognize, segment, and assign semantic class labels to different objects in the scene, while tracking and reconstructing them. The 3D geometry of each object is represented as a set of surfels. MaskFusion takes advantage of combining the outputs of Mask R-CNN [25] and a geometry-based segmentation algorithm, to increase the accuracy of the object boundaries in the object masks. The authors showed that MaskFusion can be used to implement novel augmented reality applications or perform common robotics tasks.

Taking advantage of instance-aware semantic mapping, in this work we demonstrate that our proposed object pose estimator can benefit from the use of accurate masks generated by the mapping system. Our work differs from the above methods as the developed system is able to provide an instance-aware semantic map along with 6D poses of objects. The proposed approach increases the robustness of sensor tracking through an objective function with per-pixel adaptive weights. Instead of updating probabilities for all elements in the 3D map, we reduce the space complexity by a more efficient strategy based on instance labels. In addition to the highly accurate semantic scene reconstruction, we correct misclassified regions using two proposed criteria which rely on location information and the pixel-wise probability of the class.

2.3. Object pose estimation

In recent years, CNN architectures have been extended to the object pose estimation task [10–12]. SingleShotPose [11] simultaneously detects an object in an RGB image and predicts its 6D pose without requiring multiple stages or having to examine multiple hypotheses. It is end-to-end trainable and only needs the 3D bounding box of the object shape for training. This method is able to deal with textureless objects, however, it fails to estimate object poses under large occlusions. To handle occlusions better, the PoseCNN architecture [10] employs semantic labeling which provides richer information about the objects. PoseCNN recovers the 3D translation of an object by localizing its center in the image and estimating the 3D center distance from the camera. The 3D rotation of the object is estimated by regressing convolutional features to a quaternion representation. In addition, in order to handle symmetric objects, the authors introduce ShapeMatch-Loss, a new loss function that focuses on matching the 3D shape of an object. The results show that this loss function produces superior estimation for objects with shape symmetries. However, this approach requires Iterative Closest Point (ICP) for refinement which is prohibitively slow for real-time applications. To solve this problem, Wang et al. proposed DenseFusion [12] which is approximately 200x faster than PoseCNN-ICP and outperforms previous approaches on two datasets, YCB-Video and LineMOD. The key technique of DenseFusion is that it extracts features from the color and depth images and fuses RGB values and point clouds at the per-pixel level. This per-pixel fusion scheme enables the model to explicitly reason about the local appearance and geometry information, which is essential to handle

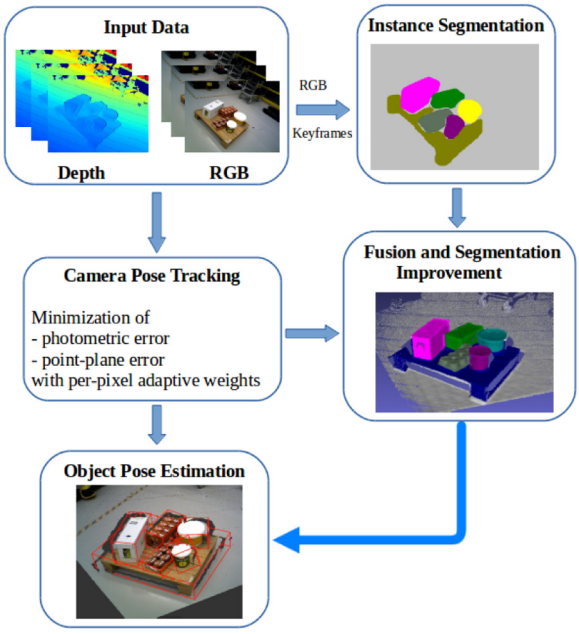


Fig. 1. Overview of the proposed system. In the main thread, input data is utilized for camera pose tracking. In a separate thread, RGB keyframes are processed by an instance segmentation framework (Mask R-CNN [25]). Then depth, color and semantic information are fused into the 3D map based on the transformation matrix estimated from the camera tracking stage. The last component is a 6D object pose estimator that output the pose of objects from multiple viewpoints.

occlusions between objects. In addition, an end-to-end iterative pose refinement procedure is proposed to further improve pose estimation while achieving near real-time inference. Although DenseFusion has achieved impressive results, like other single-view-based methods it suffers significantly from the ambiguity of object appearance and occlusions in cluttered scenes, which are very common in practice. In addition, since DenseFusion relies on segmentation results for pose prediction, its accuracy highly depends on the performance of the segmentation framework used. As in pose estimation networks, if the input to a segmentation network contains an occluder, the occlusion significantly influences the network output. In this paper, while exploiting the advantages of the DenseFusion framework, we replace its segmentation network by our semantic mapping system that provides a high-quality segmentation mask for each instance. We address the problem of the ambiguity of object appearance and occlusion by combining predictions using RGB-D images from multiple viewpoints.

3. Methodology

Our pipeline is illustrated in Fig. 1. Firstly, input data is utilized for camera pose tracking. In a separate thread, RGB keyframes are processed by an instance segmentation framework (Mask R-CNN) and the detections are filtered and matched to the existing instances in the 3D map. When no match occurs, new object instances are created. Then using the estimated camera pose and instance masks, the dense 3D geometry of the map or model is updated by fusing the points labeled in the fusion stage. The last component is a 6D object pose estimator that output the pose of objects by combining predictions from single-view-based predictions. In the following, we summarize the key elements of our method.

Instance Segmentation: The network takes in RGB images and extracts instance masks labeled with object class, which serve as input to the subsequent registration and fusion stages.

Camera Pose Tracking: Estimate camera poses within the ElasticFusion pipeline using a joint cost function that combines the cost functions of geometric and photometric estimates in an adaptively weighted sum.

Data Fusion: Our 3D map representation is an unordered list of surfels similar to [1]. The surfel map is updated by merging the newly available RGB-D frame into the existing models. In addition, segmentation information is fused into the map using our instance-based semantic fusion scheme. To improve segmentation accuracy, misclassified regions are corrected by two criteria which rely on a sequence of CNN predictions.

Object Pose Estimation: First, we employ DenseFusion that operates on object instances from single views to predict object poses. Instead of using depth and color frames captured by the camera, we use the surfel-splatted predicted depth map and the color image of the model from the previous pose estimate for DenseFusion. The predicted poses are then used as a measurement update in a Kalman filter to estimate optimal 6D pose of objects.

3.1. Instance segmentation

We employ an end-to-end CNN framework, Mask R-CNN [25] for generating a high-quality segmentation mask for each instance. Mask R-CNN has three outputs for each candidate object, a class label, a bounding box offset, and a mask. Its procedure consists of two stages. In the first stage, candidate object bounding boxes are proposed by a Region Proposal Network (RPN). In the second stage, classification, bounding-box regression, and mask prediction are performed in parallel on each small feature map. To speed up inference and improve accuracy, the mask branch is applied to the highest scoring 100 detection boxes after running the box prediction. The mask branch predicts a binary mask from each RoI using an FCN architecture [26]. The binary mask is a single $m \times m$ output regardless of class, which is generated by binarizing the floating-number mask or soft mask at a threshold of 0.5. Output of Mask-RCNN including class probabilities and masks are then used in data fusion stage. In our previous work [13], we extended Mask R-CNN to also regress an RGB image confidence weight for use in the registration step. However, producing confidence weights from every frame using the additional branch in Mask-RCNN is computationally intense, limiting the suitability of the overall system in real-time applications. In addition, the confidence weights are chosen on a per-image basis, while ideally they should be different for each pixel, as certain regions in the image can contain varying amounts of structure and color. To address the limitations of the prior work, in this paper we remove the registration weight prediction branch and propose a registration cost function with per-pixel adaptive weights as described in Section 3.2

3.2. Camera pose tracking

To perform camera tracking, our mapping system maintains a fused surfel-based model of the environment (similar to the model used by ElasticFusion [1]). Here we borrow and extend the notation proposed in the original ElasticFusion paper. The model is represented by a cloud of surfels $\mathcal{M}^s$, where each surfel consists of a position $p \in \mathbb{R}^3$, normal $n \in \mathbb{R}^3$, color $c \in \mathbb{N}^3$, initialization timestamp t_0 and last updated timestamp t. In addition Object-RPE maps each element of the 3D map (surfel) to a pair $(l_s, \mathbf{o}_s) \in \mathcal{L} \times \mathbb{N}$, where l_s represents the semantic class of surfel s and $\mathbf{o}_s$

represents its object instance id. $\mathcal{L}$ is a predetermined set of L semantic classes encoded by $\mathcal{L} := \{0, \ldots, L-1\}$.

The image space domain is defined as $\Omega \subset \mathbb{N}^2$, where an RGB-D frame is composed of a color map and a depth map D of depth pixels $d : \Omega \to \mathbb{R}$. We define the 3D back projection of a point $u \in \Omega$ given a depth map D as $p(u, D) = K^{-1}\tilde{u}d(u)$, where K is the camera intrinsics matrix and $\tilde{u}$ is the homogeneous form of u. The perspective projection of a 3D point $p = [x, y, z]^\top$ is defined as $u = \pi(Kp)$, where $\pi(p) = (x/z, y/z)$. Given a color image C with color $c(u) = [c_1, c_2, c_3]^\top$, the intensity value of a pixel $u \in \Omega$ is defined as $I(u, C) = (c_1 + c_2 + c_3)/3$.

We estimate an incremental transformation $\hat{\xi}$ between a newly captured RGB-D image at time t and the previous sensor pose at time $t-1$ by minimizing a joint optimization objective:

$$E_{combined} = E_{icp} + E_{rgb} \tag{1}$$

where E_{icp} and E_{rgb} are the geometric and photometric error terms respectively. The main difference between our approach and ElasticFusion is that instead of using fixed weights, we estimate per-pixel adaptive weights based on textureness assessment. To define the textureness of each depth image pixel, we assume that untextured regions are often piecewise flat and thus the amount of characteristic features is low. Under these assumptions, the idea behind our proposed cost function is to favor highly textured regions of the image.

In the term of the geometric energy E_{icp}, between the current depth map D_t and the predicted model depth map from the last frame $\hat{D}^a_{t-1}$ we aim to minimize the cost of the point-to-plane ICP registration error:

$$E_{icp} = \sum_{u \in \Omega} \lambda_{icp}(u)((v^k(u) - \exp(\hat{\xi})T v^k_t(u))n^k)^2 \tag{2}$$

where v^k_t is the back-projection of the kth vertex in the current depth frame D_t; and v^k and n^k are respectively the back-projection of the corresponding vertex in the predicted depth frame of the 3D map from the previous frame $t-1$ and its normal. T is the current estimate of the transformation from the previous camera pose to the current one. λ_{icp} is the weight computed from Eq. (3). The energy is adaptively weighted based on the local variance at u, we define it as in [27]:

$$\lambda(u) = \frac{\sigma_u^2}{\sigma_u^2 + \epsilon} \tag{3}$$

where σ_u denotes the local variance of the 5x5 patch around pixel u in the current depth image D_t, and ϵ is an empirically set constant. The higher the variance, the closer the weight is to 1. Fig. 2 shows an example of per-pixel weights for a RGB-D image.

In term of photometric energy E_{rgb}, between the live color image C^l_t and the predicted model color from the last frame $\hat{C}^a_{t-1}$ we minimize differences in brightness:

$$E_{rgb} = \sum_{u \in \Omega} \lambda_{rgb}(u)(I(u, C^l_t) - I(\Psi(\hat{\xi}, u), \hat{C}^a_{t-1}))^2 \tag{4}$$

where the weight λ_{rgb} is computed from Eq. (3) with the variance σ_u taken as the variance of a local 5×5 patch of pixels from the intensity image $I(u, C)$. The vector $\Psi(\hat{\xi}, u)$ is the warped pixel and defined according to the incremental transformation $\hat{\xi}$:

$$\Psi(\hat{\xi}, u) = \pi(K \exp(\hat{\xi})Tp(u, D_t)) \tag{5}$$

Finally, we find the transformation by minimizing the objective (1) through the Gauss–Newton non-linear least-square method with a three-level coarse-to-fine pyramid scheme.

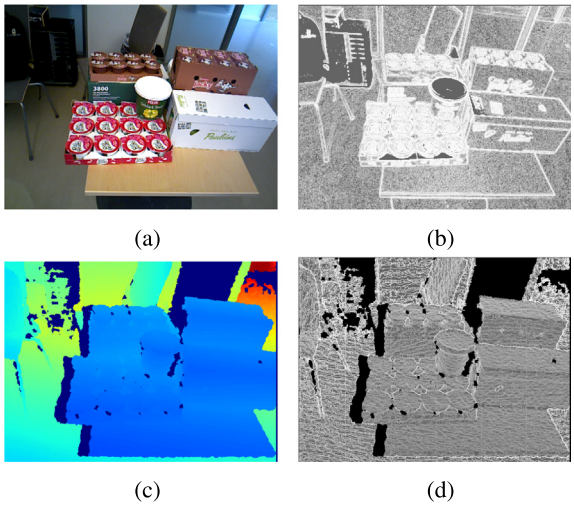


Fig. 2. Visualization of per-pixel weights computed on depth and color images: (a) color image; (b) weights on color image; (c) depth image; (d) weights on depth image.

3.3. Data association and segmentation refinement

Data association: Given an RGB-D frame at time step t, each mask M from Mask R-CNN must be associated with an instance in the 3D map. Otherwise, it will be assigned as a new instance. To find the corresponding instance, we use the tracked camera pose and existing instances in the map built at time step $t-1$ to predict binary masks via splatted rendering. The overlap percentage between the mask M and a predicted mask $\hat{M}$ for object instance $\mathbf{o}$ is computed as $\mathbb{U}(M, \hat{M}) = \frac{M \cap \hat{M}}{\hat{M}}$. Then the mask M is mapped to object instance $\mathbf{o}$ which has the predicted mask $\hat{M}$ with largest overlap, where $\mathbb{U}(M, \hat{M}) > 0.3$.

To efficiently store class probabilities, we propose to assign an object instance label $\mathbf{o}$ to each surfel and then this label is associated with a discrete probability distribution over potential class labels, $P(L_\mathbf{o} = l_i)$ over the set of class labels, $l_i \in \mathbb{L}$. In consequence, we need only one probability vector for all surfels belonging to the same object entity. This makes a big difference when the number of surfels is much larger than the number of classes. To update the class probability distribution, recursive Bayesian update is used as in [28]. However, this scheme often results in an overly confident class probability distribution that contains scores unsuitable for ranking in object detection [5]. In order to make the distribution more even, we update the class probability by simple averaging:

$$P(l_i|I_{1,\ldots,t}) = \frac{1}{t}\sum_{j=1}^{t}(p_j|I_t) \tag{6}$$

Besides fusing main class probabilities, we enrich segmentation information on each surfel by adding the probability to account for background/object predictions from the binary mask branch of Mask R-CNN. To that end, each surfel in our 3D map has a non-background (object) probability attribute p_o. As presented in [25] the binary mask branch first generates an $m \times m$ floating-number mask which is then resized to the RoI size, and binarized at a threshold of 0.5. Therefore, we are able to extract a per-pixel non-background probability map with the same image size 480×640. Given the RGB-D frame at time step t, a non-background probability $p_\mathbf{o}(I_t)$ is assigned to each pixel. Camera tracking and the 3D back projection introduced in Section 3.2

enables us to update all the surfels with the corresponding probability as following:

$$p_{\mathbf{o}} = \frac{1}{t}\sum_{j=1}^{t} p_j(I_t) \tag{7}$$

Segmentation Improvement: Despite the power and flexibility of Mask R-CNN, it frequently misclassifies object boundary regions as background. In other words, the detailed structures of an object are often lost or smoothed. Thus, there is still much room for improvement in segmentation. We observe that many of the pixels in the misclassified regions have non-background probability just slightly smaller than 0.5, while the soft probabilities mask for real background pixel is often far below the threshold. Based on this observation, we expect to achieve a more accurate object-aware semantic scene reconstruction by considering the non-background probability of surfels within a n frame sequence. With this goal, each possible surfel s ($0.4 < p_{\mathbf{o}} < 0.5$) is associated with a confidence $\vartheta(s)$. If a surfel is identified for the first time, its associated confidence is initialized to zero. Then, when a new frame arrives, we increment the confidence $\vartheta(s) \leftarrow \vartheta(s) + 1$ only if the corresponding pixel of that surfel satisfies 2 criteria: (i) its non-background probability is greater than 0.4; (ii) there is at least one object pixel inside its 8-neighborhood. After n frames, if the confidence $\vartheta(s)$ exceeds the threshold σ_{object}, we assign surfel s to the closest instance. Otherwise, $\vartheta(s)$ is reset to zero.

3.4. Multi-view object pose estimation

Given an RGB-D frame sequence, the task of 6D object pose estimation is to estimate the rigid transformation from the object coordinate system $\mathcal{O}$ to a global coordinate system $\mathcal{G}$. We assume that the 3D model of the object is available and the object coordinate system is defined in the 3D space of the model. The rigid transformation consists of a 3D rotation $R(\omega, \varphi, \psi)$ and a 3D translation $T(X, Y, Z)$. The translation T is the coordinate of the origin of $\mathcal{O}$ in the global coordinate frame $\mathcal{G}$, and R specifies the rotation angles around the X-axis, Y-axis, and Z-axis of the object coordinate system $\mathcal{O}$.

Our approach outputs the object poses with respect to the global coordinate system by combining predictions from different viewpoints. For each frame at time t, we apply DenseFusion to masks back-projected from the current 3D map. The estimated object poses are then transferred to the global coordinate system $\mathcal{G}$ and serve as measurement inputs for an extended Kalman filter (EKF) based pose update stage.

Single-view based prediction: In order to estimate the pose of each object in the scene from single views with respect to the local camera coordinate system, we apply DenseFusion to masks back-projected from the current 3D map. The network architecture and hyperparameters are similar as introduced in the original paper [12]. The image embedding network consists of a ResNet-18 encoder followed by 4 up-sampling layers as a decoder. The PointNet-like architecture is a multi-layer perceptron (MLP) followed by an average-pooling reduction function. The iterative pose refinement module consists of 4 fully connected layers that directly output the pose residual from the global dense feature. For each object instance mask, a 3D point cloud is computed from the predicted model depth pixels and an RGB image region is cropped by the bounding box of the mask from the predicted model color image. First, the image crop is fed into a fully convolutional network and then each pixel is mapped to a color feature embedding. For the point cloud, a PointNet-like architecture is utilized to extract geometric features. Having generated features, the next step combines both embeddings and outputs the estimation of the 6D pose of the object using a pixel-wise fusion network. Finally, the pose estimation results are improved by a neural network-based iterative refinement module. A key distinction between our approach and DenseFusion is that instead of directly operating on masks from the segmentation network, we use predicted 2D masks that are obtained by reprojecting the current scene model. As illustrated in Fig. 3 our semantic mapping system leads to an improvement in the 2D instance labeling over the baseline single frame predictions generated by Mask R-CNN. As a result, our object pose estimation method benefits from the use of more accurate segmentation results.

Object pose update: For each frame at time t, the estimates obtained by DenseFusion and camera motions from the registration stage are used to compute the pose of each object instance with respect to the global coordinate system $\mathcal{G}$. The pose is then used as a measurement update in a Kalman filter to estimate an optimal 6D pose of the object. Since we assume that the measured scene is static over the reconstruction period, the object's motion model is constant. The state vector of the EKF combines the estimates of translation and rotation:

$$\mathbf{x} = [X \quad Y \quad Z \quad \phi \quad \varphi \quad \psi]^\top \tag{8}$$

Let x_t be the state at time t, $\hat{\mathbf{x}}_t^-$ denote the predicted state estimate and P_t^- denote predicted error covariance at time t given the knowledge of the process and measurement at the end of step $t-1$, and let $\hat{\mathbf{x}}_t$ be the updated state estimate at time t given the pose estimated by DenseFusion z_t. The EKF consists of two stages: prediction and measurement update (correction) as follows.

Prediction:

$$\hat{\mathbf{x}}_t^- = \hat{\mathbf{x}}_{t-1} \tag{9}$$

$$P_t^- = P_{t-1} \tag{10}$$

Measurement update:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- \oplus K_t(z_t \ominus \hat{\mathbf{x}}_t^-) \tag{11}$$

$$K_t = P_t^-(P_t^m + P_t^-)^{-1} \tag{12}$$

$$P_t = (I_{6\times 6} - K_t)P_t^- \tag{13}$$

Here, $\ominus$ and $\oplus$ are the pose composition operators. K_t is the Kalman gain update. The 6×6 matrix P_t^m is measurement noise covariance, computed as:

$$P_t^m = \mu I_{6\times 6} \tag{14}$$

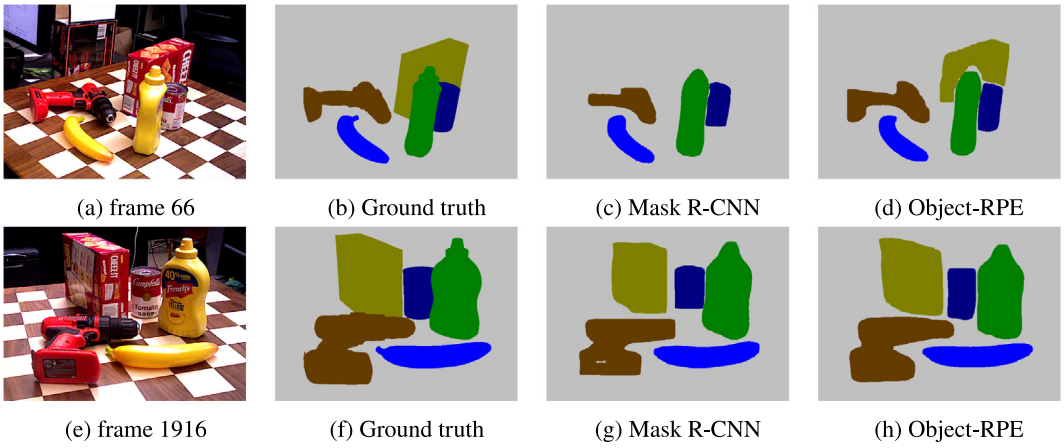
where μ is the mean distance from measured object points to its 3D model transformed according to the estimated pose. The measured object points are computed from depth and mask back-projected from the current 3D map.

4. Experiments

In this section, we evaluate the proposed system through extensive experiments on four datasets: TUM RGB-D dataset [29], YCB-Video dataset [10], SceneNN [30] and a newly collected warehouse object dataset. The TUM RGB-D dataset was used for evaluation of the tracking and mapping component of our framework, while the remaining three datasets were used for evaluation of the semantic mapping and pose retrieval components. Note that due to the disjoint object categories present in the three datasets, both Mask-RCNN and DenseFusion were trained independently for each dataset. For evaluation on the SceneNN dataset we used 75 scenes for training and 20 scenes for testing. The YCB-Video dataset was split into 80 videos for training and the remaining 12 videos for testing. For the warehouse object dataset, the system was trained on 15 videos and

Table 1
Comparison of absolute trajectory error RMS [m]/relative orientation error RMS [deg] as indicated in [29] on the warehouse dataset and TUM RGB-D dataset. ElasticFusion (EF); MaskFusion (MF); Ours (fixed λ_{icp}): our proposed registration using a fixed weight for geometric energy and per-pixel adaptive weights for photometric energy; Ours (fixed λ_{rgb}): our proposed registration using a fixed weight for photometric energy and per-pixel adaptive weights for geometric energy; Object-RPE: our proposed registration using per-pixel adaptive weights for both geometric energy and photometric energy.

	EF	MF	Ours (fixed λ_{icp})	Ours (fixed λ_{rgb})	Object-RPE
freiburg1_desk	0.020/1.625	0.034/2.487	0.019/1.393	0.018/1.245	**0.017**/**0.996**
freiburg1_room	0.068/2.045	0.153/2.342	**0.065**/1.542	0.066/1.623	**0.065**/**1.325**
freiburg1_teddy	0.083/1.743	0.129/1.897	0.080/1.540	0.080/1.365	**0.079**/**1.206**
freiburg2_desk	0.071/0.918	0.108/1.549	0.071/0.883	**0.070**/0.887	**0.070**/**0.885**
freiburg2_xyz	0.011/0.477	0.041/0.977	**0.009**/0.406	0.010/0.412	**0.009**/**0.399**
freiburg3_large_cabinet	0.099/2.138	0.133/2.455	0.060/1.351	0.065/1.486	**0.052**/**1.210**
warehouse_01	0.025/1.529	0.026/1.982	0.023/1.332	**0.021**/1.210	**0.021**/**1.101**
warehouse_02	0.031/1.870	0.040/2.654	0.028/1.657	0.029/1.669	**0.027**/**1.554**
warehouse_03	0.036/2.331	0.043/2.765	0.034/1.877	0.030/1.743	**0.029**/**1.521**
warehouse_04	0.022/1.644	0.031/2.382	0.021/1.660	0.018/1.563	**0.016**/**1.316**
warehouse_05	0.045/1.954	0.055/2.378	0.037/1.651	0.033/1.546	**0.032**/**1.442**
warehouse_06	0.028/1.980	0.033/2.121	0.026/1.971	**0.025**/1.667	**0.025**/**1.550**



(a) frame 66 (b) Ground truth (c) Mask R-CNN (d) Object-RPE

(e) frame 1916 (f) Ground truth (g) Mask R-CNN (h) Object-RPE

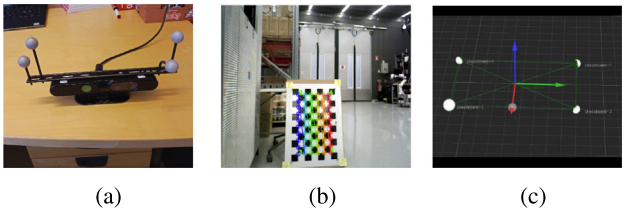
Fig. 3. Examples of masks generated by Mask R-CNN and produced by reprojecting the current scene model.

(a) Waffle (b) Jacky (c) Skansk (d) Sotstark

(e) Onos (f) Risi Frutti (g) Pauluns (h) Tomatpure

(i) Small Jacky (j) Pallet (k) Half Pallet

Fig. 4. The set of 11 objects in the warehouse object dataset.



(a) (b) (c)

Fig. 5. We collected a dataset for the evaluation of reconstruction and pose estimation systems in a typical warehouse using (a) a hand-held ASUS Xtion PRO LIVE sensor. Calibration parameters were found by using (b) a chessboard and (c) reflective markers detected by the motion capture system.

tested on the other 5 videos. Our experiments are aimed at evaluating trajectory estimation, surface reconstruction and 6D object pose estimation accuracy. A comparison against the most closely related works is also performed here.

For all tests, we ran our system on a desktop PC running 64-bit Ubuntu 16.04 Linux with an Intel(R) Xeon(R) E-2176G CPU 3.70 GHz and an Nvidia GeForce RTX 2080 Ti 10GB GPU. Our pipeline is implemented in C++ with CUDA for RGB-D image registration. The Mask R-CNN and DenseFusion codes are based on the publicly available implementations by Matterport[1] and Wang.[2] In all of the presented experimental setups, results are generated from RGB-D video with a resolution of 640 × 480 pixels. The DenseFusion networks were trained for 200 epochs with a batch size of 8. Adam [31] was used as the optimizer with a learning rate set to 0.0001.

1 https://github.com/matterport/Mask_RCNN.
2 https://github.com/j96w/DenseFusion.

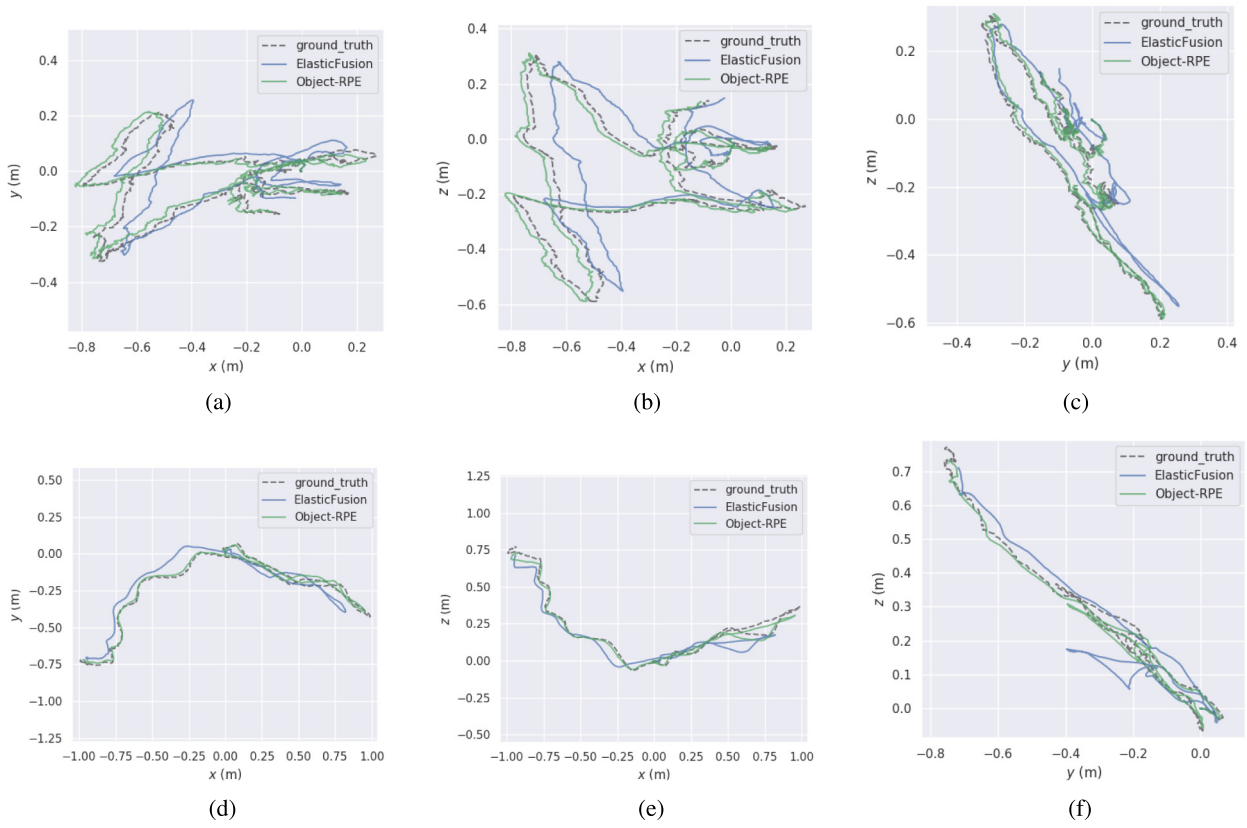


Fig. 6. The result trajectories estimated by ElasticFusion and Object-RPE compared to the ground truth of two videos in the warehouse dataset. Ground truth and camera trajectories projected to 2D: (a–c) video 1, (d–f) video 2.

4.1. The warehouse object dataset

Unlike scenes recorded in the YCB-Video dataset or other publicly available datasets, warehouse environments pose more complex problems, including low illumination inside shelves, low-texture and symmetric objects, clutter, and occlusions. To advance applications of robotics as well as to thoroughly evaluate our method, we collected an RGB-D video dataset of 11 objects as shown in Fig. 4, which is focused on the challenges in detecting warehouse object poses using an RGB-D sensor. The dataset consists of over 20,000 RGB-D images extracted from 20 videos captured by an ASUS Xtion PRO Live sensor, the 6D poses of the objects and ground truth instance segmentation masks manually generated using the LabelFusion framework [32], as well as camera trajectories from a motion capture system developed by Qualisys.[3] Calibration is required for both the RGB-D sensor and motion capture system shown in Fig. 5. We calibrated the motion capture system using the Qualisys Track Manager (QTM) software. For RGB-D camera calibration, the intrinsic camera parameters were estimated using the classical black-white chessboard and the OpenCV library. For extrinsic calibration, four markers were placed on the outer corners of the checkerboard as in [29]. We also attached four spherical markers on the sensor. Similar to [29], we were able to estimate the transformation between the pose from the motion capture system and the optical frame of the RGB-D camera.

4.2. Trajectory estimation

We compare the trajectory estimation performance of our Object-RPE to the state-of-the-art mapping system ElasticFusion and the most related work MaskFusion on the warehouse dataset and the widely used TUM RGB-D dataset [29]. This benchmark [29] is one of the most popular datasets for the evaluation of RGB-D SLAM systems. The dataset covers a large variety of scenes and camera motions and provides sequences for debugging with slow motions as well as longer trajectories with and without loop closures. Each sequence contains the color and depth images, as well as the ground-truth trajectory from the motion capture system. The benchmark does not contain ground-truth data for instance segmentation and object pose estimation. The set of objects in the scene is also not known. Thus, we did not train Mask R-CNN and DenseFusion on this dataset. Similar to [6], we used pre-trained weights for the MS COCO dataset to run Mask R-CNN for MaskFusion. To evaluate the error in the estimated trajectory by comparing it with the ground-truth, we adopt the absolute trajectory error (ATE) root-mean-square error metric (RMSE) as proposed in [29].

Table 1 shows the results. The best quantities are marked in bold. We performed an ablation study and computed the trajectory errors for our approach where we kept either the photometric or geometric error terms fixed. We note that the full version of our approach relying on adaptive weights (last column of Table 1) consistently results in the lowest observed trajectory errors across all datasets. A visualization of trajectories by running ElasticFusion and Object-RPE on two videos in the warehouse dataset is shown in Fig. 6.

4.3. Reconstruction results

In order to evaluate surface reconstruction quality, we compare the reconstructed model of each object to its ground truth 3D model. For every object present in the scene, we first register the reconstructed model M to the ground truth model G by a user interface that utilizes human input to assist traditional

[3] https://www.qualisys.com.

Table 2
Comparison of surface reconstruction error and pose estimation accuracy results on the YCB objects. ElasticFusion (EF), DenseFusion (DF).

	Reconstruction (mm)		6D pose estimation				
	EF	Object-RPE	DF	DF-PM	DF-PM-PD	DF-PM-PD-PC	Object-RPE
002_master_chef_can	5.7	**4.5**	96.4	96.8	96.5	97.0	**97.6**
003_cracker_box	5.2	**4.8**	95.5	96.2	96.2	96.9	**97.3**
004_sugar_box	7.2	**5.3**	97.5	97.4	97.0	97.2	**98.1**
005_tomato_soup_can	6.4	**5.7**	94.6	94.7	95.2	95.6	**96.8**
006_mustard_bottle	5.2	**5.0**	97.2	97.9	98.0	98.0	**98.5**
007_tuna_fish_can	6.8	**5.4**	96.6	97.1	97.4	98.1	**98.5**
008_pudding_box	5.6	**4.3**	96.5	97.3	97.1	97.6	**98.4**
009_gelatin_box	5.5	**4.9**	98.1	98.0	98.2	98.4	**99.0**
010_potted_meat_can	7.4	**6.3**	91.3	92.2	92.5	92.9	**94.7**
011_banana	6.2	**5.8**	96.6	97.2	97.2	97.4	**97.9**
019_pitcher_base	5.8	**4.9**	97.1	97.5	97.9	98.2	**99.3**
021_bleach_cleanser	5.4	**4.2**	95.8	96.5	95.9	96.3	**97.6**
024_bowl	8.8	**7.4**	88.2	89.5	90.3	90.8	**93.7**
025_mug	5.2	**5.4**	97.1	96.8	97.3	97.5	**99.1**
035_power_drill	5.8	**5.1**	96.0	96.6	96.8	96.8	**98.1**
036_wood_block	7.4	**6.7**	89.7	90.3	90.6	91.2	**95.7**
037_scissors	5.5	**5.1**	95.2	96.2	96.2	96.2	**97.9**
040_large_marker	6.1	**3.4**	97.5	98.1	97.9	97.6	**98.5**
051_large_clamp	4.6	**3.9**	72.9	76.3	77.1	77.8	**82.5**
052_extra_large_clamp	6.2	**4.6**	69.8	71.2	72.5	73.6	**78.9**
061_foam_brick	6.2	**5.7**	92.5	93.7	91.5	91.6	**95.9**
MEAN	6.1	5.2	93.0	93.7	93.8	94.1	96.0

Table 3
Comparison of surface reconstruction error and pose estimation accuracy results on the warehouse objects. ElasticFusion (EF), DenseFusion (DF).

	Reconstruction (mm)		6D pose estimation				
	EF	Object-RPE	DF	DF-PM	DF-PM-PD	DF-PM-PD-PC	Object-RPE
001_frasvaf_box	8.3	**6.0**	60.5	63.5	64.6	65.9	**68.9**
002_small_jacky box	7.4	**6.5**	61.3	66.8	66.9	67.1	**70.8**
003_jacky_box	6.6	**5.7**	59.4	65.5	68.8	68.9	**73.5**
004_skansk_can	7.9	**7.5**	63.4	66.8	68.2	68.8	**68.7**
005_sotstark_can	7.3	**5.5**	58.6	62.5	65.5	66.3	**69.7**
006_onos_can	8.1	**6.6**	60.1	63.6	65.7	66.5	**70.6**
007_risi_frutti_box	5.3	**4.2**	59.7	64.5	65.2	66.1	**69.3**
008_pauluns_box	5.8	**5.3**	58.6	62.5	65.9	66.7	**70.5**
009_tomatpure	7.4	**6.1**	63.1	65.8	66.5	67.7	**73.2**
010_pallet	11.7	**10.0**	62.3	64.9	65.3	66.6	**67.8**
011_half_pallet	12.5	**10.4**	58.9	64.4	64.8	64.8	**69.4**
MEAN	8.0	6.7	60.5	64.6	66.1	66.9	69.9

Table 4
Comparison of surface reconstruction error and pose estimation accuracy results on the SceneNN objects. ElasticFusion (EF), DenseFusion (DF).

	Reconstruction (mm)		6D pose estimation				
	EF	Object-RPE	DF	DF-PM	DF-PM-PD	DF-PM-PD-PC	Object-RPE
Cabinet	9.7	**8.1**	66.7	67.1	67.5	67.5	**70.8**
Bed	10.8	**9.9**	65.2	67.4	68.2	68.3	**72.9**
Chair	8.6	**6.8**	70.5	75.2	76.3	76.5	**78.8**
Sofa	9.9	**7.2**	73.7	76.5	77.1	77.4	**78.9**
Table	7.8	**6.5**	68.4	72.2	73.3	73.3	**80.2**
Desk	11.1	**9.2**	70.1	73.4	75.7	76.6	**80.4**
Pillow	8.3	**7.2**	68.2	69.5	70.5	71.1	**77.9**
Television	8.4	**7.1**	63.8	64.9	65.1	65.5	**74.2**
Lamp	12.5	**10.6**	66.4	69.6	70.3	70.5	**73.1**
Monitor	11.3	**10.3**	72.5	77.2	78.6	78.9	**82.1**
MEAN	9.84	8.3	68.6	71.3	72.3	72.6	77.0

registration techniques [32]. Next, we project every vertex from M onto G and compute the distance between the original vertex and its projection. Finally, we calculate and report the mean distance μ_d over all model points and all objects.

The results of this evaluation on the reconstruction datasets are summarized in Tables 2–4. Qualitative results are shown in Fig. 7. We can see that our reconstruction system significantly outperforms the baseline (ElasticFusion). Our approach achieves the best performance on all objects. The results show that our reconstruction method has a clear advantage of using the proposed registration cost function. In addition, we are able to keep all surfels on object instances always *active*, while ElasticFusion has to segment these surfels into *inactive* areas if they have not been observed for a period of time ∂_t. This means that the object surfels are updated all the time. As a result, the developed system is able to produce a highly accurate instance-aware semantic map.

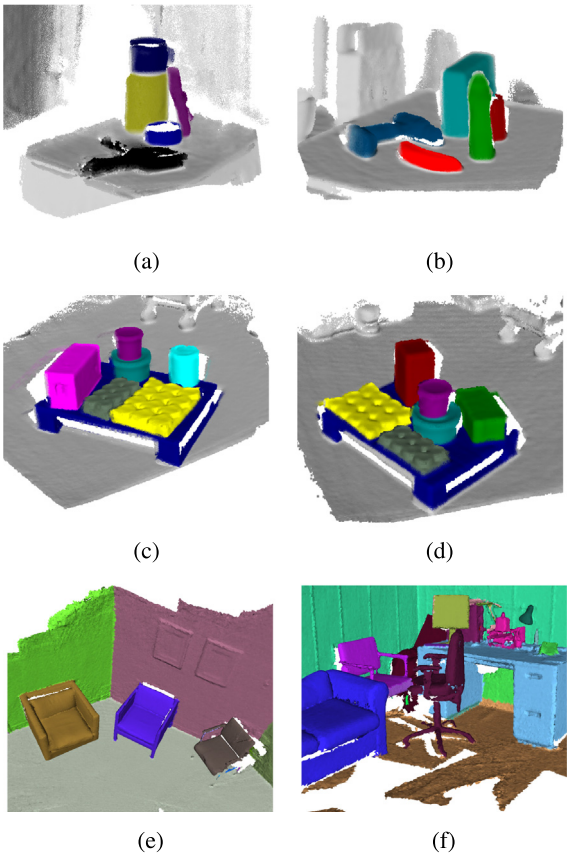


Fig. 7. Examples of 3D object-aware semantic maps from the YCB-Video dataset (a–b), the warehouse object dataset (c–d) and SceneNN dataset (e–f).

4.4. Pose estimation results

We used the average closest point distance (ADD-S) metric [10,12] for evaluation. We report the area under the ADD-S curve (AUC) following PoseCNN [10] and DenseFusion [12]. The maximum threshold was set to 10 cm as in [10] an [12]. The object pose predicted from our system at time t is a rigid transformation from the object coordinate system $\mathcal{O}$ to the global coordinate system $\mathcal{G}$. To compare with the performance of DenseFusion, we transform the object pose to the camera coordinate system using the transformation matrix estimated from the camera tracking stage. Tables 2–4 present a detailed evaluation for all the 21 objects in the YCB-Video dataset, 11 objects in the warehouse dataset and 10 selected objects in SceneNN. Object-RPE with the full use of projected mask, depth and color images from the semantic 3D map achieves superior performance compared to the baseline single frame predictions. We observed that in all cases combining information from multiple views improved the accuracy of the pose estimation over the original DenseFusion. We saw an improvement of 3.0% over the baseline single frame method with Object-RPE, from 93.0% to 96.0% for the YCB-Video dataset. We also observed a marked improvement, from 60.5% for a single frame to 69.9% with Object-RPE on the warehouse object dataset. Similarly, Object-RPE saw +8.4% improvement on the selected objects in SceneNN. Furthermore, we ran a number of ablations to analyze Object-RPE including (i) DenseFusion using projected masks (DF-PM) (ii) DenseFusion using projected masks and projected depth (DF-PM-PD) (iii) DenseFusion using projected masks, projected depth, and projected RGB image (DF-PM-PD-PC). DF-PM performed better than DenseFusion on the 3 datasets (+0.8%, +4.1% and +2.7%). The performance benefit of DF-PM-PD was less clear as it resulted in a very small improvement of +0.1%, +1.5% and +1.0% over DF-PM. For DF-PM-PD-PC, performance improved additionally with +0.4% on the YCB-Video dataset, +0.8% on the warehouse object dataset, and +0.3% on SceneNN objects. The remaining improvement is due to the fusion of estimates in the EKF.

Table 5
Average run-time analysis of system components (ms per frame). Note that the components with ∗ process keyframes.

Component	Object-RPE
Instance Segmentation ∗	350
Registration	25
Data Fusion	15
Object Pose Estimation	40

Lastly, the running times of the individual components of Object-RPE, averaged over all evaluated sequences, are shown in Table 5. Our pipeline does not explicitly depend on Mask-RCNN, and can be configured to use a different instance segmentation backbone. The current system does not run Mask-RCNN for every frame because of heavy computation, with an average computational cost of 350 ms per frame. We instead only run instance segmentation for keyframes (1 keyframe per 10 frames). The numbers indicate that the system is capable of running at approximately 8 Hz on 640 × 480 input.

5. Conclusions

We have presented and validated a mapping system that yields high quality instance-aware semantic reconstruction while simultaneously recovering 6D poses of object instances. The main contributions of this paper is to show that (i) by combining geometric and appearance cues in an adaptively weighted sum we are able to obtain reliable camera tracking and state-of-the-art surface reconstruction and (ii) taking advantage of deep learning-based techniques and our semantic mapping system we are able to improve the performance of object pose estimation as compared to single view-based methods. We have provided an extensive evaluation on common benchmarks and our own dataset. The results confirm that Object-RPE is able to produce a high quality dense map with robust tracking. We also demonstrated that the proposed object pose estimator benefits from the use of accurate masks generated by the semantic mapping system and from combining multiple predictions based on the Kalman filter.

We believe that the instance-aware semantic mapping and object pose estimation from multi-views will open the way to new applications of intelligent autonomous robotics. As future work, to achieve real-time capabilities, we plan on investigating the optimal way to reduce the runtime requirements of the proposed system. More experiments also will be done to see how the semantic reconstruction performs in comparison with other state-of-the-art semantic mapping methods.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work has been partially funded by the European Union H2020 project ILIAD.

References

[1] T. Whelan, R.F. Salas-Moreno, B. Glocker, A.J. Davison, S. Leutenegger, Elasticfusion: Real-time dense SLAM and light source estimation, Int. J. Robot. Res. 35 (14) (2016) 1697–1716.

[2] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, A.W. Fitzgibbon, Kinectfusion: Real-time dense surface mapping and tracking, in: ISMAR, vol. 11, 2011, pp. 127–136.

[3] C. Kerl, J. Sturm, D. Cremers, Robust odometry estimation for RGB-D cameras, in: International Conference on Robotics and Automation (ICRA), IEEE, 2013, pp. 3748–3754.

[4] N. Sünderhauf, T.T. Pham, Y. Latif, M. Milford, I. Reid, Meaningful maps with object-oriented semantic mapping, in: International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 5079–5085.

[5] J. McCormac, R. Clark, M. Bloesch, A. Davison, S. Leutenegger, Fusion++: Volumetric object-level slam, in: International Conference on 3D Vision (3DV), IEEE, 2018, pp. 32–41.

[6] M. Runz, M. Buffier, L. Agapito, Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects, in: International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2018, pp. 10–20.

[7] S. Fuchs, S. Haddadin, M. Keller, S. Parusel, A. Kolb, M. Suppa, Cooperative bin-picking with time-of-flight camera and impedance controlled DLR lightweight robot III, in: International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 4862–4867.

[8] J. Corney, H. Rea, D. Clark, J. Pritchard, M. Breaks, R. MacLeod, Coarse filters for shape matching, Comput. Graph. Appl. 22 (3) (2002) 65–74.

[9] M. Germann, M.D. Breitenstein, I.K. Park, H. Pfister, Automatic pose estimation for range images on the GPU, in: Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM), IEEE, 2007, pp. 81–90.

[10] Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes, 2017, arXiv preprint arXiv:1711.00199.

[11] B. Tekin, S.N. Sinha, P. Fua, Real-time seamless single shot 6D object pose prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 292–301.

[12] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, S. Savarese, Densefusion: 6d object pose estimation by iterative dense fusion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3343–3352.

[13] D.-C. Hoang, T. Stoyanov, A.J. Lilienthal, Object-RPE: Dense 3D reconstruction and pose estimation with convolutional neural networks for warehouse robots, in: European Conference on Mobile Robots (ECMR), IEEE, 2019, pp. 1–6.

[14] F. Steinbrücker, J. Sturm, D. Cremers, Real-time visual odometry from dense RGB-D images, in: International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2011, pp. 719–722.

[15] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, J. McDonald, Robust real-time visual odometry for dense RGB-D mapping, in: International Conference on Robotics and Automation (ICRA), IEEE, 2013, pp. 5724–5731.

[16] D.R. Canelhas, T. Stoyanov, A.J. Lilienthal, SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images, in: International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 3671–3676.

[17] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, C. Theobalt, Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration, ACM Trans. Graph. 36 (4) (2017) 1.

[18] Q.-H. Pham, B.-S. Hua, T. Nguyen, S.-K. Yeung, Real-time progressive 3D semantic segmentation for indoor scenes, in: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 1089–1098.

[19] M. Antonello, D. Wolf, J. Prankl, S. Ghidoni, E. Menegatti, M. Vincze, Multi-view 3D entangled forest for semantic segmentation and mapping, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 1855–1862.

[20] J. McCormac, A. Handa, A. Davison, S. Leutenegger, Semanticfusion: Dense 3d semantic mapping with convolutional neural networks, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 4628–4635.

[21] Y. Nakajima, K. Tateno, F. Tombari, H. Saito, Fast and accurate semantic mapping through geometric-based incremental segmentation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 385–392.

[22] M. Rünz, L. Agapito, Co-fusion: Real-time segmentation, tracking and fusion of multiple objects, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 4471–4478.

[23] Y. Nakajima, H. Saito, Efficient object-oriented semantic mapping with object detector, IEEE Access 7 (2018) 3206–3213.

[24] M. Grinvald, F. Furrer, T. Novkovic, J.J. Chung, C. Cadena, R. Siegwart, J. Nieto, Volumetric instance-aware semantic mapping and 3D object discovery, IEEE Robot. Autom. Lett. 4 (3) (2019) 3037–3044.

[25] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.

[26] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3431–3440.

[27] H.-H. Vu, P. Labatut, J.-P. Pons, R. Keriven, High accuracy and visibility-consistent dense multiview stereo, IEEE Trans. Pattern Anal. Mach. Intell. 34 (5) (2011) 889–901.

[28] A. Hermans, G. Floros, B. Leibe, Dense 3d semantic mapping of indoor scenes from rgb-d images, in: International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 2631–2638.

[29] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in: International Conference on Intelligent Robots and Systems (IROS), IEEE, 2012, pp. 573–580.

[30] B.-S. Hua, Q.-H. Pham, D.T. Nguyen, M.-K. Tran, L.-F. Yu, S.-K. Yeung, Scenenn: A scene meshes dataset with annotations, in: Fourth International Conference on 3D Vision (3DV), IEEE, 2016, pp. 92–101.

[31] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[32] P. Marion, P.R. Florence, L. Manuelli, R. Tedrake, Label fusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes, in: International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 1–8.

Dinh-Cuong Hoang a doctoral candidate in Computer Science at Örebro University since December 2017. My primary research interests lie in autonomy for mobile robots, with a focus on perception algorithms for manipulation. I studied automation technology at National Taipei University of Technology and received my Masters degree in 2016.