Intra-Logistics with Integrated Automatic Deployment: Safe and Scalable Fleets in Shared Spaces

H2020-ICT-2016-2017
Grant agreement no: 732737

DELIVERABLE 6.4

Perception system for detecting boxes and wrapping

Due date: month 24 (December 2018)
Deliverable type: R
Lead beneficiary: ORU

Dissemination Level: PUBLIC

Main author: Todor Stoyanov and Martin Magnusson (ORU)

1 Introduction

In this deliverable we outline the development of a perception system within Task T6.4 of the ILIAD project, targeted at the detection of plastic wrapping and goods placed on pallets. We describe several distinct components developed and evaluated within the project, outline the integration of said components within the ILIAD framework, and identify directions for future improvement of the overall system.

This deliverable is structured around the two tasks investigated: identification of the presence and configuration of goods stored on pallets is discussed in Section 2; detecting the presence of plastic wrapping of pallets with goods is discussed in Section 3. This deliverable summarises the efforts pursued within task T6.4, focusing on perception relevant for autonomous manipulation and handling of goods. Some of the problems discussed here are specific instantiations of the more general semantic mapping problem, which is the subject of task T1.4. The main difference is that in T6.4 we specifically need to consider tight requirements on accuracy, governed by the manipulation tasks. In this deliverable we focus on methods targeted on manipulation, in which the scale of the semantic maps may be sacrificed for higher accuracy.

Our overarching aim within task T6.4 is thus to develop, evaluate, and deploy a set of specific perception methods that provide input to the other manipulation-related tasks within WP6.

2 Object pose estimation

Visual object recognition is an extensively studied problem that has benefited extensively from the recent surge in artificial neural network research. Within ILIAD we are interested in a more specific instance of object recognition than the commonly investigated generic object recognition task.

In particular, we are interested in detecting the relative position and orientation of goods, with respect to either a global reference system, or a robot centric one. The primary reasoning for that requirement is that object detection in ILIAD serves as an input to the goods handling pipeline: we are interested in identifying poses with respect to a coordinate system that can be related precisely to the base of the robot arms used.

In addition to strict requirements on object pose accuracy, the scenario addressed here poses additional challenges: varying illumination conditions and low illumination in shelves, low-texture and symmetric objects, clutter, occlusions, and tightly stacked objects. As such, the methods developed within ILIAD make use of depth data to augment visual information. A possible simplyfing assumption is knowledge of the type of products we need to detect: as our target scenarios feature pallets stacked with homogeneous products, and a warehouse management system aware of the location and product types of each pallet. While this assumption is reasonable for the ILIAD usecase, we have not as yet imposed it, in an effort to keep the developed methods more generally applicable.

To address the problem of object pose estimation we developed two distinct systems. The first system is based on the VoxNet [18] architecture, and is a purely geometric object pose estimation algorithm. That system, along with evaluations carried out on ILIAD scenarios is briefly described in Section 2.1. A purely depth-based recognition system however struggles in cases where geometric texture is low: as in for example when uniform goods are stacked very closely together. In addition, the method evaluated, like most state of the art in the field, offers only single-shot predictions: i .e., based on a single view of the scene. To counter these limitations, we developed a system for object detection and fusion of multiple single view predictions into a consistent semantic model (outlined in Section 2.2). The system takes into account both appearance and depth information,

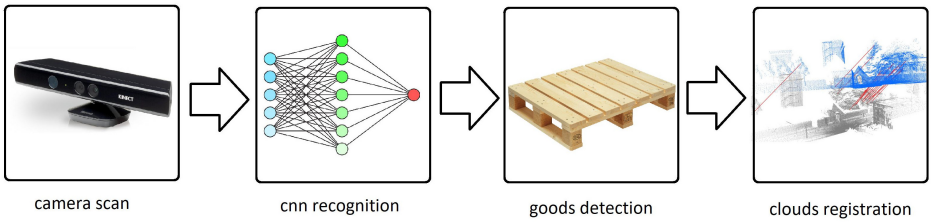


Figure 1: An overview of a volumetric obejct recognition system. The pipeline starts by acquiring a model of the environment by means of a moving RGB-D sensor. The model is reconstructed uisng the SDFTracker algorithm [6]. Next, a 3D Convolutional Neural Network classifies regions based on their likelihood of containing target objects. After a filtering step, object candidate locations are used to seed a local registration procedure that determines the poses of detected goods.

as well as cues provided by a semantic segmentation algorithm, to produce a semantic map of medium-scale environments. Finally, in Section 2.3 we deploy an object pose estimation method that uses the semantic model as a base and integrates pose predictions from a sequence of scene observations to produce accurate object pose estimation.

2.1 Single-shot object detection

In this section, we briefly outline a volumetric object recognition system developed as an MSc project in collaboration between Pisa University and Örebro University. An overview of the system design is shown in Figure 1. The overall reasoning behind the proposed pipeline is to perform object recognition and detection on complete fused workspace models, instead of relying on single-view information. Prior work [5] has demonstrated that classical computer vision (local keypoint detection / description) performs better on multi-view fused information than on single-view or even noise-filtered single-view data.

Several recent object recognition frameworks have proposed methods for transferring Convolutional Neural Networks (CNNs) to the domain of 3D data. Notably, the VoxNet [18] architecture proposes a CNN architecture with three-dimensional convolutional filters that operate on ordered occupancy grid data, while PointNet [21] extends the proposed approach to unorganized point cloud data. Due to their applicability to range data, both architectures have seen wide applicability in the robotics community (e .g., Wang et al. [28] and Zaganidis et al. [32]). In this section, we adapt and deploy a variant of VoxNet that operates on truncated signed distance fields (TSDFs), instead of on occupancy maps. The overall object recognition and pose estimation pipeline proposed proceeds in four phases:

Scanning phase: In this step the SDFTracker algorithm [6] is used to simultaneously track the pose of the camera and reconstruct a TSDF model of the workspace. The obtained model then serves as an input to the next phase.

Recognition phase: A sliding window of fixed size is convolved with the workspace model. For each window we run an instance of the modified VoxNet classification CNN that results in a confidence vector. The confidences for each object type and background class are then stored in a volumetric grid and passed on to the next stage. The procedure is illustrated in Figure 2: the dark red shaded region indicates the extent of the TSDF model, while the gray shaded window shows the extent of a single sliding window. At each subsequent step the window is moved with a stride of half its size.

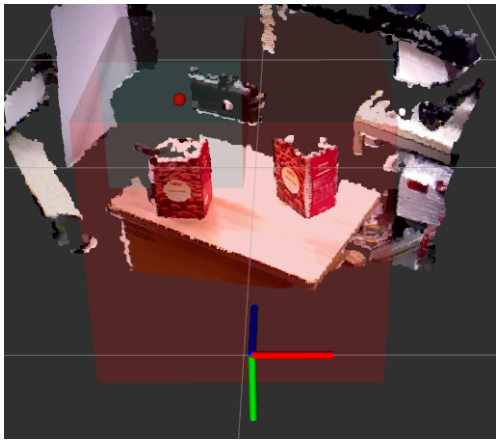


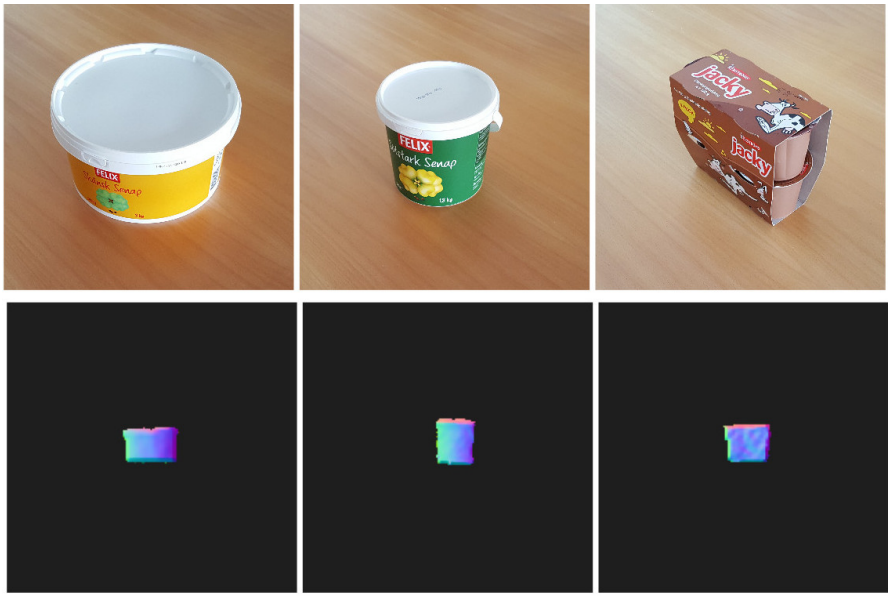
Figure 2: An illustration from one of the test scenarios for the proposed pipeline. The current point cloud from the RGB-D sensor is shown, with an overlay that encodes the extent of the TSDF model (red) and the initial position of one sliding filter (gray), centered around the location of the red dot in the picture.

Filtering phase: We pass a non-max suppression filter over the output from the previous stage and identify object candidate locations.

Refinement phase: If the confidence score is above a pre-set threshold, each candidate location serves as an initial guess of the pose of an object instance. We then use this initial guess to run a local registration refinement, using point-to-plane ICP between an object model from a database and the TSDF model. If the ICP fitness score (i .e., the overlap between the object model and the workspace model) is above a second threshold, we report the detected object instance and its pose.

The CNN architecture is based directly on VoxNet, with minor modifications. We borrow the original architecture, but add batch normalisation and dropout regularisation to improve training performance. We also rescale the TSDF values in the interval $[0, 1]$. The network is trained on a set of partial observations of a sub-set of objects from the ILIAD scenario, with example RGB and TSDF views shown in Figure 3. We train using the Adam Optimizer and employ early stopping. Initial tests of the system in simplified conditions showed promising results, with an average accuracy of the CNN object recognition phase of 81 %.

The full perception system has been integrated at the experimental setup with the ILIAD manipulator system at Pisa University and undergone application-based testing. The results from these tests indicate that while the system is capable of detecting well separated objects with good accuracy, closely stacked scenarios present difficulties, resulting in few and inaccurate detections. This limitation stems from the original system design that relies on a high amount of geometric features, which are absent in case of tightly packed uniform goods, and can only be mitigated by the inclusion of additional sensing modalities. The work presented in the following two sections is an attempt to move in that direction and provide more reliable detection of objects in cluttered scenes.



(a) Big senap box (b) Small senap box (c) Pudding box

Figure 3: An illustration of three objects used in the initial tests of the recognition system, along with rendered views of their associated TSDF models.

2.2 Semantic 3D map for manipulation

2.2.1 Motivation

In this section, we propose a 3D mapping system to produce highly accurate object-aware semantic scene reconstruction. Our work benefits from incorporating state of the art RGB-D SLAM and deep-learning-based instance segmentation techniques [9, 29]. Unlike previous related works [19, 20, 23], which solely use semantic information for data fusion, we employ rich segmentation information from CNNs to increase the robustness of camera tracking through a joint cost function wherein all given information is used: the depth, RGB image, and segmentation information. We also develop a CNN architecture beyond the original Mask R-CNN [23] to input RGB image and output adaptive weights for the cost function used in the sensor pose estimation process. In contrast to existing approaches that update the probabilities for all elements (surfels or voxels) 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 pixel-wise probability to the class. An example of a semantic object map reconstructed by our system is shown in Figure 4. Details of the work presented in this section is also available in Hoang et al. [11].

2.2.2 Methodology

Our pipeline is composed of three main components as illustrated in Figure 5. The input RGB-D data is processed through a semantic instance segmentation stage, followed by a frame-to-model alignment stage, and finally a model fusion stage. In the following, we summarise the key elements of our method.

Segmentation: Produce object masks with semantic labels using our CNN architecture.

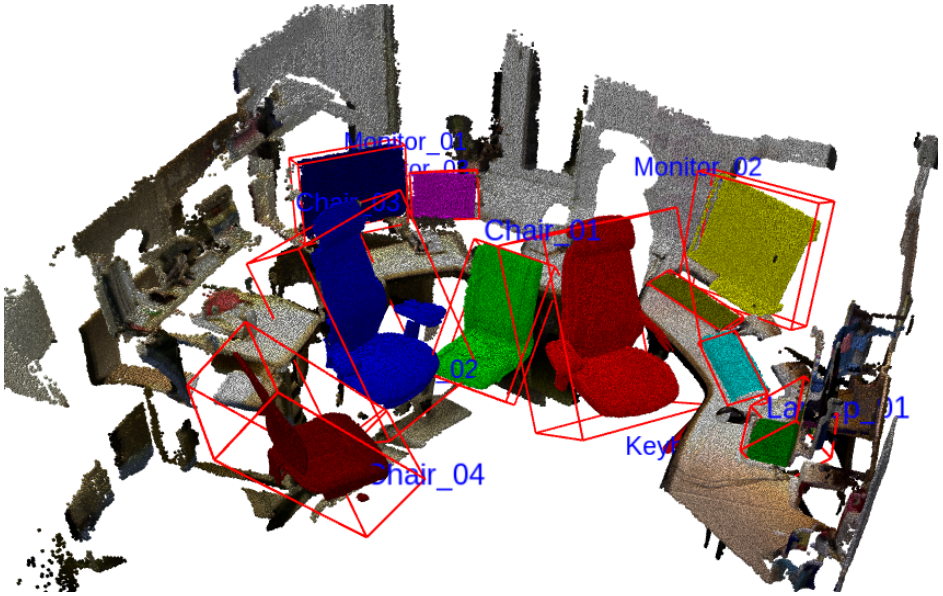


Figure 4: An instance-aware semantic 3D map of an office environment produced by the proposed semantic mapping system.

The developed architecture also predicts weights for the joint cost function for camera tracking.

Tracking: Estimate camera pose within the ElasticFusion pipeline using our proposed joint cost function. We combine the cost functions of geometric, photometric, and semantic estimates in a weighted sum. The adaptive weights are generated by the segmentation process above.

Fusion and Segmentation Improvement: Fuse segmentation information into 3D map using our instance-based semantic fusion. To improve segmentation accuracy, misclassified regions are corrected by two criteria which reply on a sequence of CNN predictions.

Segmentation Network In our framework, we employ an end-to-end CNN framework, Mask R-CNN, 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, which is extracted by RoIPool. Note that 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 [16]. 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.

To integrate deep-learning based segmentation and classification into our system, we extend Mask R-CNN to identify object outlines at the pixel level while simultaneously generating an adaptive weight used in camera pose tracking stage as shown in Figure 6. A fourth banch is added to our CNN framwork, which shares computation of feature

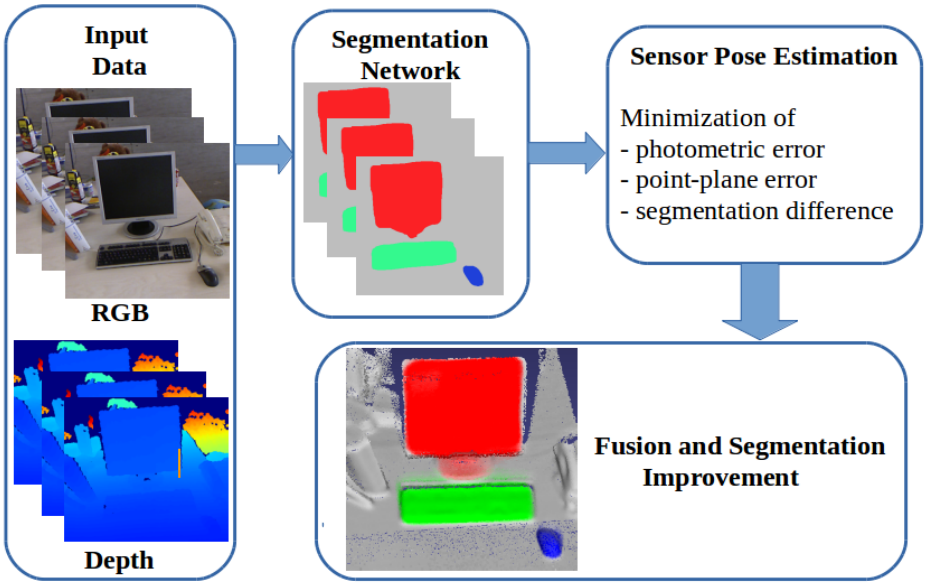


Figure 5: Flow of the proposed framework for object-level semantic mapping. The segmentation network first yields masks and probabilities specified for each category. Then the output of the segmentation stage along with the depth map and RGB frame are used for camera pose estimation. Finally, input data and semantic information are fused into the 3D map based on a transformation matrix estimated from the previous stage.

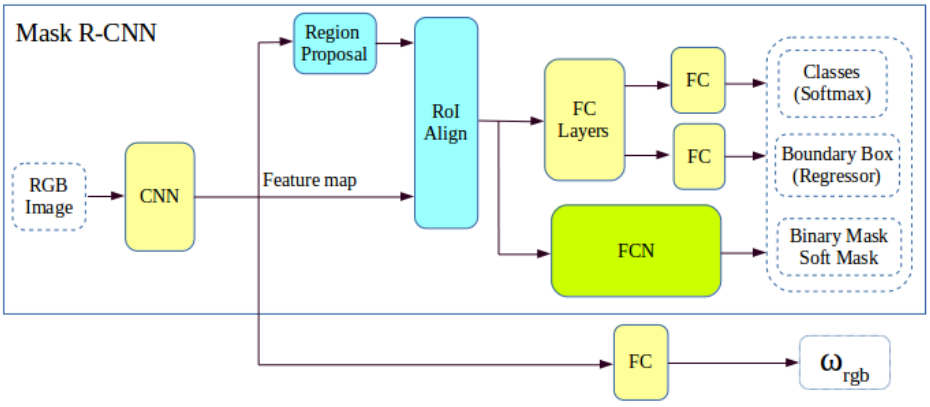


Figure 6: CNN architecture. Extending Mask R-CNN to predict masks and class probabilities while simultaneously yielding an adaptive weight for camera tracking.

maps with Mask R-CNN branches and outputs the weight by a fully connected layer. In general, the network consists of a backbone CNN, a region proposal network (RPN), a ROI classifier, a bounding Box Regressor, a mask branch, and a camera tracking weight estimator. The CNN backbone is a standard convolutional neural network that is used for extracting a feature map. This convolutional feature map not only becomes the input for the other stages of Mask R-CNN, but also shares computation with our extended branch for adaptive weight estimation. Therefore, the developed network receives an RGB image, and returns a set of per pixel class probabilities and weights used in the cost function in the subsequent alignment stage. The weight estimation is treated as a classification problem where the target is a binary decision whether or not the given RGB image should be used in the registration process. In other words, we aim to train our weight predicting model as a binary classifier, where one class signifies that the RGB image contains useful information for the subsequent registration process, while the other class indicates the converse. The probability predicted from classification model is considered as an adaptive weight for our joint cost function for camera pose estimation.

Camera Tracking To perform camera tracking, our object-oriented mapping system maintains a fused surfel-based model of the environment (similar to the model used by ElasticFusion [29]). Here we borrow and extend the notation proposed in the original ElasticFusion paper. The model is represented by a cloud of surfels $\mathscr{M}^s$, where each surfel consists of of a position $p \in \mathbb{R}^3$, normal $n \in \mathbb{R}^3$, colour $c \in \mathbb{N}^3$, weight $w \in \mathbb{R}$, radius $r \in \mathbb{R}$, initialisation timestamp t_0 and last updated timestamp t. In addition, each surfel also stores an object instance label $l_{\mathbf{o}} \in N$. Each object instance $\mathbf{o}$ is associated with a discrete probability distribution over potential class labels, $P(l_{\mathbf{o}} = l_i)$ over the set of class labels, $l_i \in \mathscr{L}$.

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 $d : \Omega \to R$. 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 $\tilde{u}$ the homogeneous form of u. The perspective projection of a 3D point $p = [x, y, z]^\top$ is defined as $u = \pi(K\,p)$, where $\pi(p) = (x/z, y/z)$. In the following, we describe our proposed approach for combined ICP pose estimation.

Our approach aims to estimate a sensor pose that minimises the cost over a combination of the global point-to-plane energy, photometric error and semantic difference. We wish to minimise a joint optimisation objective:

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

where E_{icp}, $\omega_{\text{rgb}} E_{\text{rgb}}$, and $\omega_{\text{sem}} E_{\text{sem}}$ are the geometric, photometric and semantic error terms respectively. The photometric error function is weighted by a factor predicted from our the CNN. The weight for semantic error is defined as $\omega_{\text{sem}} = N_m / N_u$, where N_m is the number of non-background pixels and N_u is the number of pixels per frame.

The details of first two terms in Equation (1) can be found in Whelan et al. [29]. E_{icp} is the point-to-plane error metric in which the object of minimisation is the sum of the squared distance between a point from a live surface measurement and the tangent plane at its correspondence point from the model prediction. The cost function performs well in environments with high geometric texture, however tracking failures can occur in case there are not enough features to fully constrain all 6 DOFs of the camera pose. For instance, if the measured points are located on planar surfaces then the point-to-plane error metric will fail to register successive views. This is because there will be no mechanism to guarantee that a global minimum can be reached by shifting source points to target points in the direction perpendicular to the normals. Steinbrücker et al. [24] used

the color information to overcome this. $\omega_{\text{rgb}} E_{\text{rgb}}$ is the cost over the photometric error between the current color image and the predicted a model color from the last frame.

A key distinction between our approach and ElasticFusion is that instead of only estimating the camera pose via geometric and photometric data, we additionally employ semantic information to perform frame-to-frame tracking. The cost we wish to minimise depends on the difference in predicted likelihood values between the label probability maps. To simplify minimising the cost function, we only take the probability of the most likely class on each pixelwise probability vector $Q(u,P) = \max P(l_i)$. We denote values of $Q(u,P)$ over a given image as semantic probability map. So based on this simplification, the semantic probability error can be formulated as:

$$E_{\text{sem}} = \sum_{u \in \Omega} (Q(u, P_t) - Q(\Psi(\hat{\xi}, u), P_{t-1}))^2 \tag{2}$$

In words, P_t and P_{t-1} are per-pixel independent probability distributions over the class labels from the frame at time step t and $t-1$ respectively. The vector $\Psi(\hat{\xi}, u)$ is the warped pixel and defined according to the incremental transformation $\hat{\xi}$:

$$\Psi(\hat{\xi}, u) = \pi(K \exp(\hat{\xi}) T p(u, D_t)) \tag{3}$$

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

Fusion and Segmentation Improvement Given a frame RGB-D at time step t, each mask M from Mask R-CNN must be connected to 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 percent overlap between the mask M and a predicted mask $\hat{M}$ for object instance $\mathbf{o}$ is computed as $\mathbb{U}(M, \hat{M}) = (M \cap \hat{M})/\hat{M}$. Then the mask M is mapped to object instance $\mathbf{o}$ which has the predicted mask $\hat{M}$ with largest overlap, where $\mathbb{U}(M, \hat{M}) > 0.3$.

Unlike existing work [19, 20, 23] where each element in a 3D map (e .g., surfel or TSDF maps) stores a probability distribution over all classes, 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, means of a recursive Bayesian update is used in [10]. However, this scheme often results in an overly confident class probability distribution that contains scores unsuitable for ranking in object detection [19]. In order to make the distribution become more even, we update the class probability by simple averaging:

$$P(l_i | I_{1,\ldots,t}) = \frac{1}{t} \sum_{j=1}^{t} (p_j | I_t) \tag{4}$$

Moreover, existing work only store the probability distribution over the class labels, and miss the background probability from the binary mask branch. Conversely, we enrich segmentation information on each surfel by adding the probability to account for background/object predictions. To that end, each surfel in our 3D map has a non-background probability attribute p_o.

As presented in He et al. [9] the binary mask branch first generates an $m \times m$ floating-number mask which is then resized to the RoI size, and binarised 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 enables us to update all the surfels with the corresponding probability as follows:

$$p_{\mathbf{o}} = \frac{1}{t}\sum_{j=1}^{t} p_j(I_t) \tag{5}$$

Despite the power and flexibility of Mask R-CNN, it usually misclassified 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 a 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 non-background probability of surfels within an n frames 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 two criteria: (i) its non-background probability is greater than 0.4; (ii) there is at least one object pixel inside its 6-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. Here, we found $n = 10$ and $\sigma_{\text{object}} = 10$ provide good performance.

2.2.3 Evaluation

We have evaluated our system by performing experiments on the TUM [25] and YCB video [31] datasets. These experiments are aimed at evaluating both trajectory estimation and surface reconstruction accuracy. A comparison against most related works is also performed here.

For all tests, we run our system on a standard desktop PC running 64-bit Ubuntu 16.04 Linux with an Intel Core i7-4770K 3.5 GHz and a nVidia GeForce GTX 1080 Ti 6 GB GPU. Our pipeline is implemented in Python with Tensorflow 1.6 for segmentation and C++ with CUDA for mapping. The input is standard 640×480 resolution RGB-D video.

To train our CNNs, We start with a weights file that has been trained on the ImageNet dataset [7] with a ResNet-101 [8] backbone. We finetune layers of Mask R-CNN on the COCO dataset with 10 common object classes in indoor environments (backpack, chair, keyboard, laptop, monitor, computer mouse, cell phone, sink, refrigerator, microwave) and on a portion of the YCB video data set not used in the evaluations. To train the weight estimator branch, we split SceneNN dataset [13] into two groups based on camera pose ground truth and trajectory estimation of ElasticFusion using only photometric error.

Camera Pose Tracking We compare the trajectory estimation performance of our system to two most related works MaskFusion and Fusion++ on the widely used RGB-D benchmark of [25]. This benchmark 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. To evaluate the error in

Table 1: Comparison of ATE RMSE on RGB-D SLAM benchmark. All units given are in metres.

	PCL-KinFu	Kintinuous	Elastic Fusion	Mask Fusion	Fusion++ Fusion++	Proposed
fr1_desk	0.073	0.037	**0.020**	0.034	0.049	0.022
fr1_room	0.187	0.075	0.068	0.153	0.235	**0.065**
fr1_desk2	0.102	0.071	**0.048**	0.093	0.153	0.056
fr1_360	-	0.116	**0.108**	0.157	-	0.126
fr1_teddy	-	0.132	**0.083**	0.129	-	0.095
fr2_desk	0.103	**0.034**	0.071	0.108	0.114	0.083
fr2_xyz	0.077	0.029	**0.011**	0.041	0.020	0.025
fr2_rpy	-	0.018	0.015	0.076	-	**0.012**
fr3_long_office	0.086	0.030	0.017	0.102	0.108	0.085
fr3_large_cabinet	-	0.144	0.099	0.133	-	**0.052**

Table 2: Comparison of surface reconstruction accuracy results on the YCB objects (mm).

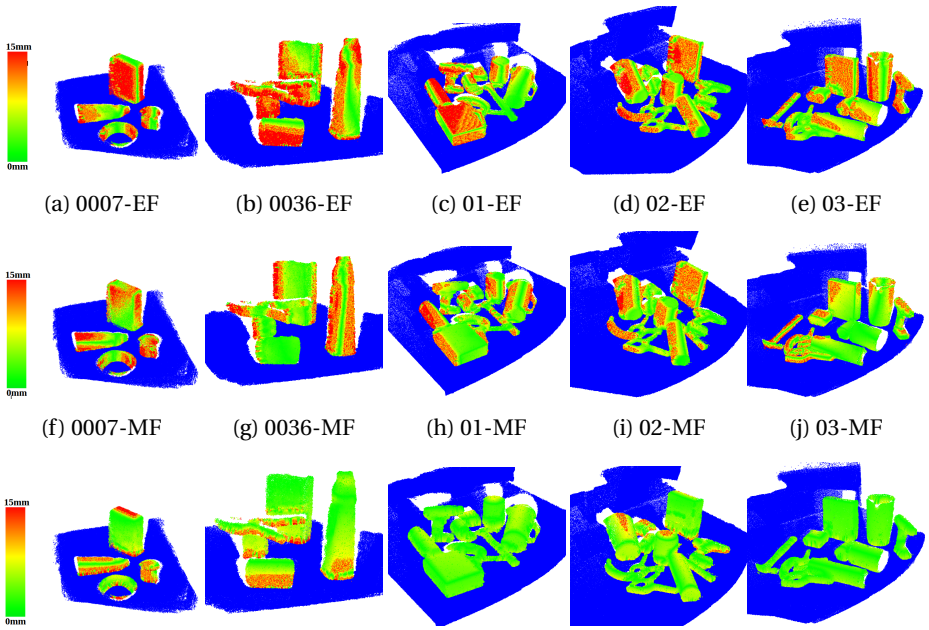
	ElasticFusion	MaskFusion	Our System
YCB video 0007	9.6	7.3	**6.5**
YCB video 0036	8.1	6.4	**5.7**
YCB video 0072	10.1	9.4	**8.7**
Our sequence 01	7.1	6.7	**3.7**
Our sequence 02	7.3	6.6	**4.1**
Our sequence 03	7.5	6.2	**3.4**

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 [25].

Table 1 shows the results. From these we can see the performance of our system is comparable to state of the art classical approaches, and outperforms both MaskFusion and Fusion++. Results for Fusion++ are taken from the respective publication as presented by the authors, and values for MaskFusion are calculated from MaskFusion implementation. While the original ElasticFusion algorithm still obtains the best overall ATE performance, the results of our approach are comparable. Despite this relative similarity in the average trajectories, our approach performs better in reconstructing the relevant object-scale detail, as discussed in the next sub-section.

Reconstruction It should be noted that a good performance on a camera trajectory benchmark does not always imply a high quality surface reconstruction. We have evaluated our system by performing experiments on the Yale-CMU-Berkeley (YCB) Object and Model set [4]. We finetuned our network on the training set of the YCB-Video dataset. It contains 92 real video sequences for 21 object instances. 89 videos along with 80,000 synthetic images are used for training. We evaluate on the remaining test videos from the original data set, as well as on three video sequences we acquired independently in scenes featuring a larger number of objects and more complex camera trajectories.

In order to evaluate surface reconstruction quality, we compare the object models obtained through our approach to the ground truth YCB object models. Note that the



(a) 0007-EF (b) 0036-EF (c) 01-EF (d) 02-EF (e) 03-EF

(f) 0007-MF (g) 0036-MF (h) 01-MF (i) 02-MF (j) 03-MF

(k) 0007-Proposed (l) 0036-Proposed (m) 01-Proposed (n) 02-Proposed (o) 03-Proposed

Figure 7: Heat maps showing reconstruction error of ElasticFusion (EF), MaskFusion (MF), and our proposed system on the remaining test videos from The YCB-video dataset (0007, 0036, 0072) and three video sequences (01, 02, 03) we acquired independently in scenes featuring a larger number of objects and more complex camera trajectories.

ground truth object models are only used here to compute evaluation metrics, unlike in prior works like SLAM++ which use them within the tracking framework. For every object present in the scene, we first register the reconstructed model M to the ground truth model G. 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.

Table 2 and Figure 7 show the mean reconstruction error over the six sequences produced by our system, MaskFusion and ElasticFusion. Our method consistently results in the lowest reconstruction errors over all datasets. From this comparison it is evident that our approach benefits greatly from the use of the proposed joint cost function with adaptive weights. Interestingly we observe an increase in accuracy is achieved when more segmented objects appeared in the reconstructed environment, suggesting that our framework makes efficient use of the available semantic information to improve surface reconstruction quality. In other words, when the number of objects of interest increases the semantic probability map becomes more textured, which leads to a better reconstruction performance. We also note that in our aproach all surfels on objects of interest are always *active*, while ElasticFusion segments these surfels into *inactive* if they have not been observed for a period of time ∂_t. This means that object surfels are updated all the time. As a results, our framework is able to produce a highly accurate object-oriented semantic map.

Segmentation Accuracy Evaluation In this section, we show on the YCB video dataset that our system leads to an improvement in the 2D instance labelling over the baseline single frame predictions generated by Mask-RCNN. Our 2D masks are obtained by reprojecting

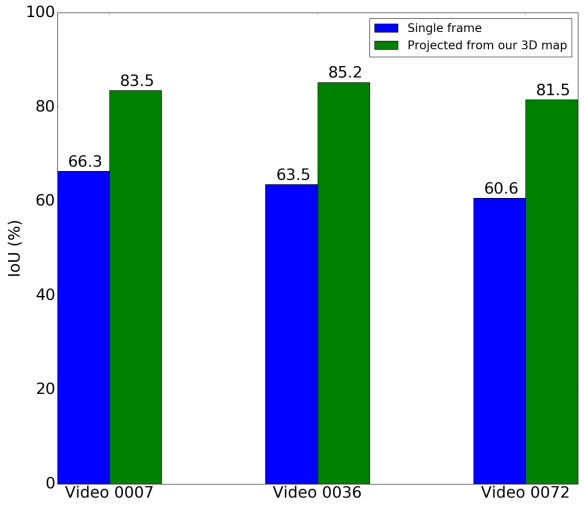


Figure 8: Results of segmentation accuracy evaluation on YCB videos.

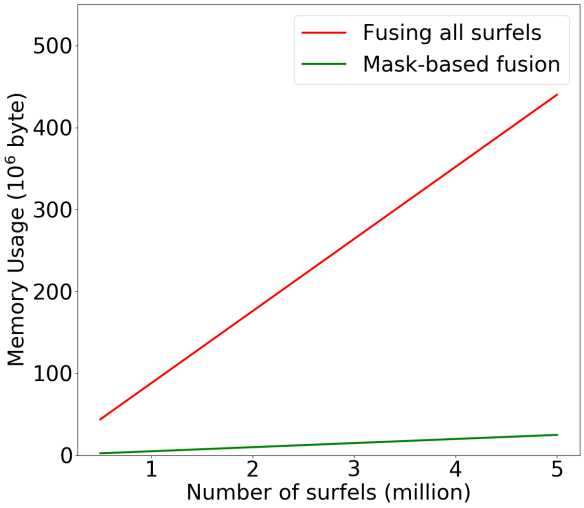


Figure 9: Memory usage for storing class probabilities.

the reconstructed 3D model. We use the Intersection over Union (IoU) metric for this evaluation, which measures the number of pixels common between the grounth-truth and prediction masks divided by the total number of pixels present across both masks. The results of this evaluation are shown in Figure 8. We observe that the segmenation performance improved, on average, from 63.5% for a single frame to 83.4% when projecting the predictions from the 3D map.

Run-time Performance and Memory Usage Our current system does not run in real time because of heavy computation in instance segmentation. Our CNN requires 350 ms, while camera pose estimation, fusion and segmentation require a further 70 ms on a typical sample of RGB-D SLAM benchmark [25].

We compared our mask-based fusion method with other approaches [19, 20, 23] which assign class probabilities to each element of the 3D map rather than to each mask.

The memory usage of the proposed method is significantly reduced compared to the conventional approach over all samples as shown in Figure 9. The average memory usage of the proposed method is 5.7 % of those conventional approaches.

2.3 Pose Estimation

2.3.1 Motivation

In this section we describe Object-RPE (Reconstruction and Pose Estimation): a system for online object pose estimation that builds on top of the high-quality instance-aware semantic 3D map introduced in Section 2.2 and extends it to produce a complete instance-aware semantic reconstruction and 6D object pose estimation framework. The work benefits from integrating a state-of-the-art deep learning-based pose estimation technique [28] into our 3D scene reconstruction system. 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 recovering 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, as illustrated by the example shown in Figure 10. The work presented in this section will be presented at the ECMR conference [12].

2.3.2 Methodology

The proposed pipeline is illustrated in Figure 11. While Section 2.2 presented our approach for segmentation, registration, and fusion [11], this section presents a 6D object pose estimator that exploits multiple views of the same instance and our high-quality semantic map to accurately predict the pose of an object under heavy occlusion.

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 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, Y, 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, 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 [28]. The image embedding network consists of a ResNet-18 encoder followed by 4 up-sampling layers as a decoder. The PointNet 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

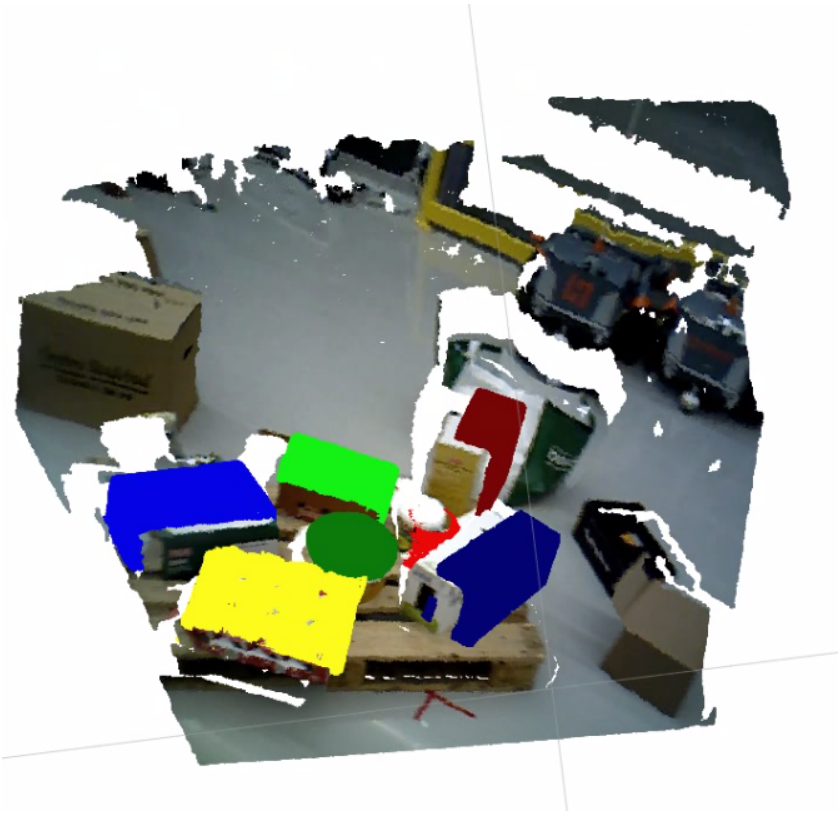


Figure 10: Example showing a colored 3D map with object models re-projected in the scene, using the object pose provided by the proposed system. Note that these poses have not been post-processed with ICP.

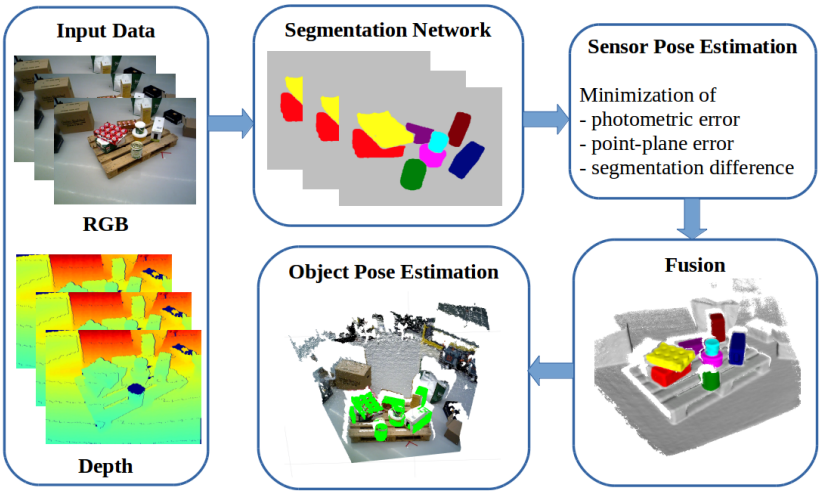


Figure 11: Overview of the proposed system for pose estimation.

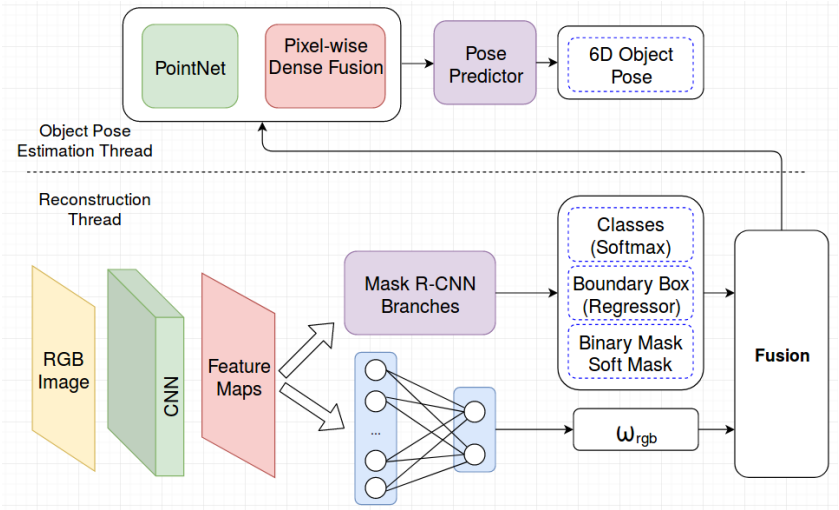


Figure 12: CNN architectue. Extending Mask R-CNN to predict masks and classes probabilities while simultaneously yielding an adaptive weight for camera tracking. DenseFusion uses the predicted model depth map and predicted model masks to output object pose predictions.

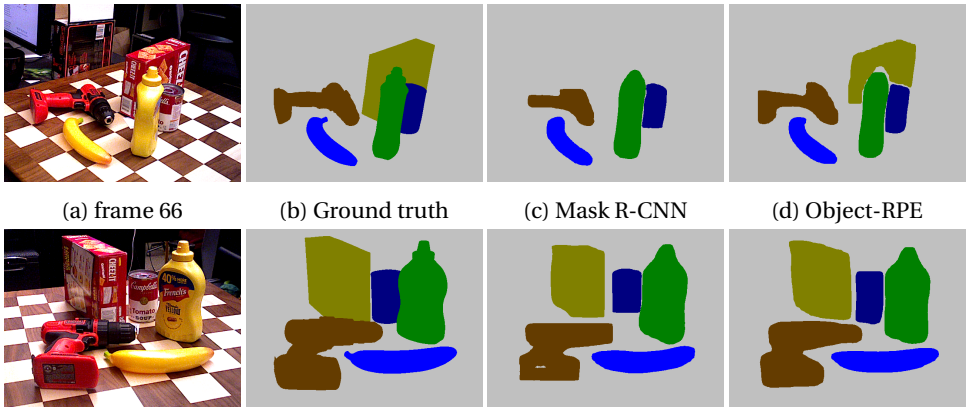
crop is fed into a fully convolutional network and then each pixel is mapped to a color feature embedding. For the point cloud, a PointNet-like architecture is utilized to extract geometric features. Having generated features, the next step combines both embeddings and outputs the estimation of the 6D 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 Figure 13 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, we expect that our object pose estimation method benefits from the use of the 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:

$$\mathbf{x} = [X \quad Y \quad Z \quad \phi \quad \varphi \quad \psi]^\top \tag{6}$$

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.



(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

Figure 13: Examples of masks generated by Mask R-CNN and produced by reprojecting the current scene model.

Prediction:

$$\hat{\mathbf{x}}_t^- = \hat{\mathbf{x}}_{t-1} \tag{7}$$

$$P_t^- = P_{t-1} \tag{8}$$

Measurement update:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- \oplus K_t(z_t \ominus \hat{\mathbf{x}}_t^-) \tag{9}$$

$$K_t = P_t^-(R_t + P_t^-)^{-1} \tag{10}$$

$$P_t = (I_{6\times 6} - K_t)P_t^- \tag{11}$$

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

$$R_t = \mu I_{6\times 6} \tag{12}$$

where μ is the average distance of all segmented object points from the corresponding 3D model points transformed according to the estimated pose.

2.3.3 Evaluation

We evaluated our system on the YCB-Video [31] dataset and on a newly collected warehouse object dataset, using products from Orkla Foods. 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 tested on the other 5 videos. Our experiments are aimed at evaluating both 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 standard desktop PC running 64-bit Ubuntu 16.04 Linux with an Intel Core i7-4770K 3.5 GHz and a nVidia GeForce GTX 1080 Ti 6 GB 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

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

(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

Figure 14: The set of 11 objects in the warehouse object dataset.

from RGB-D video with a resolution of 640x480 pixels. The DenseFusion networks were trained for 200 epochs with a batchsize of 8. Adam [15] was used as the optimizer with learning rate set to 0.0001.

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 warehouse application of robotics as well as to thoroughly evaluate our method, we collected an RGB-D video dataset of 11 objects as shown Figure 14, 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 instance segmentation masks generated using the LabelFusion framework [17], 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 Figure 15. 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 classical black-white chessboard and the OpenCV library. In order to track the camera pose through the motion capture system, we attached four spherical markers on the sensor. In addition, another

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

(a) (b) (c)

Figure 15: 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.

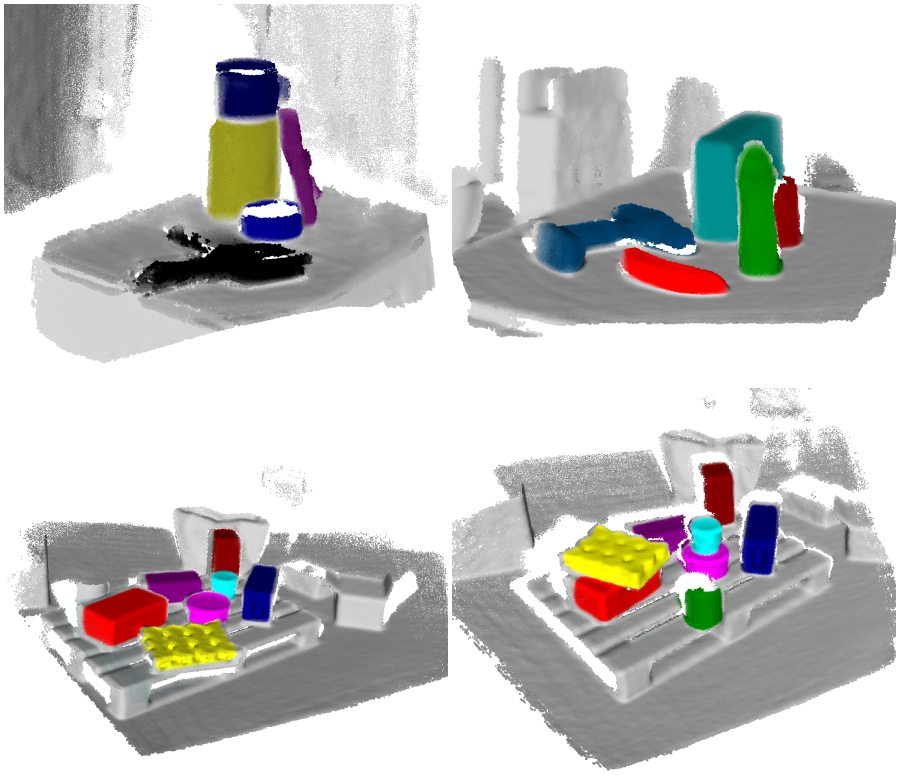


Figure 16: Examples of 3D object-aware semantic maps from the YCB-Video dataset (above) and the warehouse object dataset (below).

four markers were also placed on the outer corners of a calibration checkerboard. By detecting these markers, we were able to estimate the transformation between the pose from the motion capture system and the optical frame of the RGB-D camera.

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 registration techniques [17]. Next, we project every vertex from M onto G and compute the distance

Table 3: Comparison of surface reconstruction and pose estimation accuracy results on the YCB objects.

	Reconstruction (mm)		6D Pose Estimation				
	ElasticFusion	Object-RPE	DenseFusion (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**	6.1	97.2	97.7	97.9	97.9	**98.3**
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**	6.4	96.6	96.8	96.8	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	**5.9**	92.5	93.4	91.1	91.0	**95.6**
MEAN	6.1	5.3	93.0	93.6	93.7	94.1	95.9

Table 4: Comparison of surface reconstruction and pose estimation accuracy results on the warehouse objects.

	Reconstruction (mm)		6D Pose Estimation				
	ElasticFusion	Object-RPE	DenseFusion (DF)	DF-PM	DF-PM-PD	DF-PM-PD-PC	Object-RPE
001_frasvaf_box	8.3	**6.2**	60.5	63.2	64.1	65.4	**68.7**
002_small_jacky box	7.4	**6.9**	61.3	66.3	65.1	66.2	**69.8**
003_jacky_box	6.6	**5.8**	59.4	65.4	66.5	68.3	**73.2**
004_skansk_can	7.9	**7.7**	63.4	66.7	67.5	67.8	**68.3**
005_sotstark_can	7.3	**5.9**	58.6	62.4	65.3	66.2	**69.5**
006_onos_can	8.1	**6.9**	60.1	63.4	65.7	66.1	**70.4**
007_risi_frutti_box	5.3	**4.2**	59.7	64.1	63.2	63.5	**67.7**
008_pauluns_box	5.8	**5.3**	58.6	62.4	65.9	66.6	**70.2**
009_tomatpure	7.4	**6.2**	63.1	65.6	66.3	67.3	**73.1**
010_pallet	11.7	**10.5**	62.3	64.5	64.6	66.3	**67.4**
011_half_pallet	12.5	**11.4**	58.9	64.1	63.1	63.4	**68.5**
MEAN	8.0	7.0	60.5	64.4	65.3	66.1	69.7

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 summarised in Table 3 and Table 4. Qualitative results are shown in Figure 16. We can see that our reconstruction system significantly outperforms the baseline. While ElasticFusion results in the lowest reconstruction errors on two YCB objects (006_mustard_bottle and 011_banana), our approach achieves the best performance on the remaining 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 object-oriented semantic map.

Pose Estimation Results We use the average closest point distance (ADD-S) metric [28, 31] for evaluation. We report the area under the ADD-S curve (AUC) following PoseCNN [31] and DenseFusion [28]. The maximum threshold is set to 10 cm. 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 3 and 4 present a detailed evaluation for all the 21 objects in the YCB-Video dataset and 11 objects in the warehouse dataset. 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 observe that in all cases combining information from multiple views improved the accuracy of the pose estimation over the original DensFusion. We see an improvement of 2.3 % over the baseline single frame method with Object-RPE, from 93.6 % to 95.9 % for the YCB-Video dataset. We also observe a marked improvement, from 60.5 % for a single frame to 69.7 % with Object-RPE on the warehouse object dataset. Furthermore, we ran a number of ablations to analyse 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 both datasets (+0.6 % and +3.9 %). The performance benefit of DF-PM-PD was less clear as it resulted in a very small improvement of +0.1 % and +0.9 % over DF-PM. For DF-PM-PD-PC, performance improved additionally with +0.5 % on the YCB-Video dataset and +1.7 % on the warehouse object dataset. The remaining improvement is due to the fusion of estimates in the EKF. The run-time performance of our system is currently slower than real time because of heavy computations in the instance segmentation, with an average computational cost of 500 ms per frame.

3 Plastic detection

One complication that needs to be addressed for deploying robots for object picking in warehouses is that pallets that enter the warehouse are often wrapped in plastic film, to prevent objects from falling out when the pallet is moved (as seen in Figure 17. The robot must be able to cut the plastic so that objects can be removed (which is addressed in T6.3), and, before it can do that, it must also determine whether or not a pallet is wrapped.

In T6.4, we have developed a method for classifying pallets as “wrapped” or “non-wrapped”, based on estimation of a parametric surface reflectance function coupled with a support-vector machine (SVM) classifier. This was in part developed as an MSc

Figure 17: (Left) Full pallet with plastic wrapping and (right) without wrapping, ready for object picking.

thesis project at Örebro University by Bastien Veyssiere from Université Toulouse III Paul Sabatier [27]. Also the BSc theses of Frans Anton [2] and Olle Back [3] at Örebro University contributed to this work. Figures 18–20 are reprinted from Veyssiere [27].

Our method uses point clouds with intensity data from a Kinect One RGB-D sensor (or, optionally, a 3D lidar). The main steps of the method are (i) point cloud segmentation to isolate the pallet and the goods on it from the background, (ii) estimation of the surface reflectance function from the backscattered intensity at each point on the object, and (iii) an SVM classifier that uses the parameters of the reflectance function to determine if the pallet is wrapped or not.

In related work, Tatoglu and Pochiraju [26] used the intensity from a lidar to estimate diffuse and specular reflectance properties, and demonstrated their approach for segmenting an outdoor scene with a statue. Classification accuracy was merely 70 %, which indicates the difficulty of classifying materials using range sensors common in the robotics community. Kerl et al. [14] presented an approach to recover the diffuse albedo of surfaces in a scene using a Kinect One camera. Exploiting the fact that this sensor provides both an IR image, with illumination from a projector on-board the sensor, a corresponding depth image, and a colour image using ambient lighting, they transfer an illumination-independent albedo estimate computed from the IR and depth images to the colour domain. In contrast to our work, only the diffuse albedo component is reconstructed, and not the full reflectance properties. Wurm et al. [30] trained an SVM classifier using the returned range and intensity values from a lidar scanner in order to segment outdoor point clouds into vegetation vs asphalt. In contrast to the work proposed here, their method can only provide a binary classification of pre-trained classes of materials, and does not aim to reproduce the full BRDF (bidirectional reflectance distribution function).

The initial segmentation can be implemented in a fairly simple way, given that we are only interested in finding pallets that are fully packed with products. Knowing the standardised size of pallets, and using the assumptions that pallets are placed at known pick slots and that ones that potentially are wrapped also have objects stacked to at least a certain minimum height, segmenting the pallets (and their products) that are under consideration for the method can be done with a simple Euclidean region-growing segmentation.

After the segmentation step, we have for each pallet a point cloud with intensity values representing the amount of light from the Kinect's projector that is backscattered into its camera. The intensity depends on the reflectance properties of the material and the incident angle of the light, but also on the distance between the sensor and the surface.

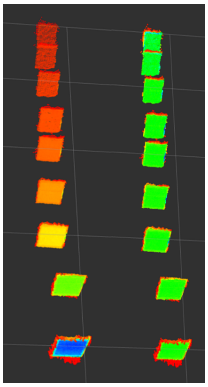


Figure 18: 3D point clouds from a Kinect One of a box at different distances. Colours show backscattered intensity. Left: default intensity values (strongly dependent on distance to surface). Right: re-scaled intensity values (constant at all distances).

The backscattered intensity that is measured for each pixel can be modelled with the so-called rendering equation

$$L^{\text{ref}}(P,\boldsymbol{\omega}_{\text{o}})=\int_{\boldsymbol{\omega}_{\text{i}}\in\mathbf{S}^2_+(P)} L(P,-\boldsymbol{\omega}_{\text{i}})f_{\text{r}}(P,\boldsymbol{\omega}_{\text{i}},\boldsymbol{\omega}_{\text{o}})(\boldsymbol{\omega}_{\text{i}}\cdot\mathbf{n}_P)\text{d}\boldsymbol{\omega}_{\text{i}}, \tag{13}$$

which describes $L^{\text{ref}}(P,\boldsymbol{\omega}_{\text{o}})$ the amount of light reflected from point P in direction $\boldsymbol{\omega}_{\text{o}}$ as a function of: $L(P,-\boldsymbol{\omega}_{\text{i}})$, the light reaching P from direction $\boldsymbol{\omega}_{\text{i}}$; $f_{\text{r}}(P,\boldsymbol{\omega}_{\text{i}},\boldsymbol{\omega}_{\text{o}})$, the reflectance function at P for the two directions $\boldsymbol{\omega}_{\text{i}}$ and $\boldsymbol{\omega}_{\text{o}}$; and $(\boldsymbol{\omega}_{\text{i}}\cdot\mathbf{n}_P)$, the incidence angle.

Since the light L measured at the sensor is attenuated proportional to distance squared, this function is strongly dependent on distance to the surface. As the classifier needs to be independent of the distance, we need to re-scale the intensity values. For this purpose we adopt the scaling function of Rodriguez-Gonzalvez et al. [22]. Figure 18 demonstrates the result of of this re-scaling step.

Given a point cloud of a single pallet, with range-compensated intensity values, we can use the measured intensity L^{ref} and incident angle $\boldsymbol{\omega}_{\text{i}}\cdot\mathbf{n}_P$ to estimate the unknown reflectance function f_{r}, which, in turn, indicates the material of the surface. The function f_{r} belongs to a class called *bidirectional reflectance distribution functions* (BRDFs). Given an incoming light direction $\boldsymbol{\omega}_{\text{i}}$ and an outgoing direction $\boldsymbol{\omega}_{\text{o}}$, a BRDF returns the ratio of light that is scattered into direction $\boldsymbol{\omega}_{\text{o}}$.

A Kinect One point cloud of a full pallet contains enough readings to make it possible to estimate the BRDF from a single image. In this case, since we use the light emitted by the sensor itself, we have $\boldsymbol{\omega}_{\text{i}}=\boldsymbol{\omega}_{\text{o}}$ for all points. After computing the normal $\mathbf{n}$ and the cosine of the angle of incidence $(\boldsymbol{\omega}_{\text{i}}\cdot\mathbf{n})$ for all points on the pallet, we fit a normalised Blinn–Phong BRDF [1] to the measured angles and intensities. Specifically, using the normalised Blinn–Phong model

$$f_{\text{BP}}=\frac{k_L}{\pi}+\frac{8+f}{8\pi}k_g(\mathbf{n}\cdot\mathbf{h})^f \tag{14}$$

we estimate the parameters k_L (amount of diffuse reflection), k_g (amount of glossy reflection) and f (surface smoothness) using COBYLA optimisation.

For this estimation, we assume that the visible surface of the pallet is homogeneous (made of the same material). However, we have found that also for mixed pallets (see Figure 19a) this procedure results in a BRDF estimate that makes it possible to determine

(a) (b)

(c) (d)

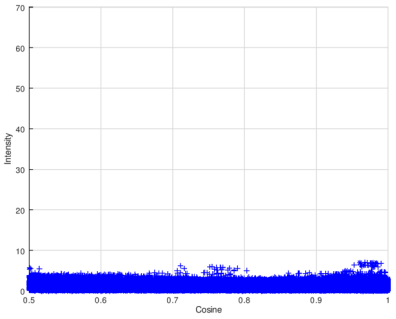
Figure 19: Test objects used for developing the plastic detector. Above: pallets with mixed products. Below: homogeneous cardboard sheet without and with plastic film.

if the pallet is wrapped or not, as indicated in Figure 20. We also qualitatively verified the reflectance data from the Kinect sensor by point measurements from a LightTec Mini-Diff instrument for reflectance characterisation (Figure 20e). Comparing Figures 20c and 20d to Figure 20e, we can see that the shape of the reflectance plot is similar for the Kinect data and the Mini-Diff reference data.

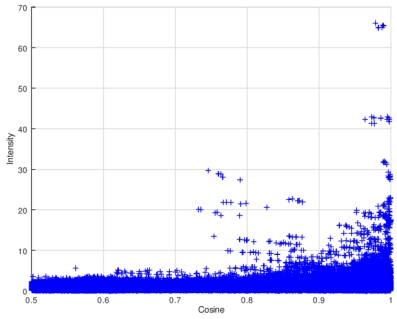
Having estimated the parameters of the BRDF for the pallet, we have trained an SVM classifier to determine if the pallet is wrapped or not. While it may be possible to achieve the same goal with deep learning rather than via a user-defined parametric function, we have chosen to adopt the method presented here because recovering the full BRDF from a surface in uncontrolled lighting settings is a goal that is worth pursuing in itself, with potential applications in, e .g., computer graphics, building information modelling (BIM), and augmented reality. The work presented in this deliverable is a first step towards that goal.

We validated the approach on the two pallets shown in Figures 19a and 19b, with and without plastic wrapping. For these tests, we used 384 Kinect One images of the mixed pallet (19a) with wrapping and 323 image without, and 89 imaged of the homogeneous pallet (19b) with wrapping and 82 images without. (We found that a χ^2 kernel gave the best results overall.) Using leave-one-out cross validation we achieved 92 % classification accuracy on the mixed pallet and 98 % accuracy on the homogeneous pallet. Training on all data from mixed pallet, we achieved 97 % accuracy on the mixed pallet and 89 % accuracy on the homogeneous pallet. This classifier that was trained on the mixed pallet has 97 % precision and 81 % recall on the homogeneous pallet.

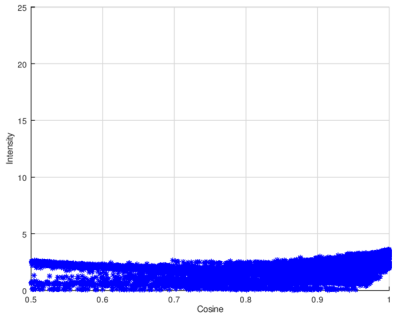
The target performance for the plastic perception system in ILIAD (as specified in the working document *"Technical requirements and performance measures (September 3, 2018)"*) is to have a binary classifier that can detect whether or not a pallet is wrapped



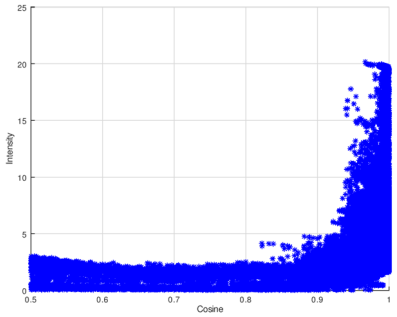
(a) Mixed pallet (Figure 19a), without wrapping.



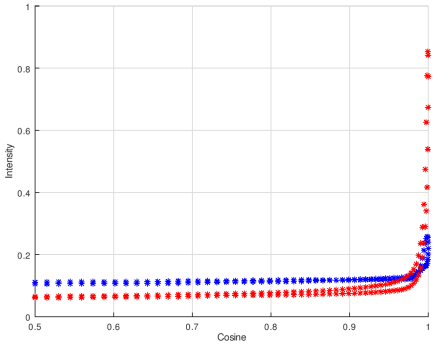
(b) Mixed pallet (Figure 19a), with wrapping.



(c) Cardboard, without wrapping (Figure 19c).



(d) Cardboard, with wrapping (Figure 19d).



(e) Reference measurement for (c)–(d). Blue: without wrapping. Red: with wrapping.

Figure 20: Reflectance measurements from the mixed pallet and cardboard sheet in Figure 19. The horizontal axis shows the cosine of the incident angle. The vertical axis shows the measured intensity. (a)–(d) Kinect One data. (e) Reference measurement with a LightTec Mini-Diff scattering instrument.

with an accuracy of $> 90\,\%$. This target has been met, based on the results shown here [2, 27]. However, we acknowledge the fact that these results are from a lab setup with limited amounts of data, and that the system should to be trained on more data, from real-world warehouses, before a confident claim on accuracy can be made. We will further evaluate the perception system with larger real-world data sets for the final Milestone 4 demonstrator.

4 Summary

In this deliverable, we have reported on the software and hardware implementation used in ILIAD for providing object poses for manipulation as well as a dense semantic 3D map of the working area. We have also reported on the software and hardware implementation of our system for detecting whether or not a pallet has wrapping that needs to be removed before picking.

The outputs of these perception modules are used as input for the manipulation modules of T6.5 and T6.3, respectively; for picking objects as well as cutting the plastic wrapping of pallets that need to be unwrapped before they can be handled.

References

[1] Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering*. 3rd. A K Peters Ltd., 2008.

[2] Frans Anton. "Mobile Robot Reflectance Acquisition to Detect Plastic Wrapping on Pallets". Bachelor's Thesis. Örebro University, June 2018.

[3] Olle Back. "Mobile Robot Reflectance Acquisition to Detect Plastic Wrapping on Pallets". Bachelor's Thesis. Örebro University, Jan. 2019.

[4] Berk Calli et al. "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols". In: *arXiv preprint arXiv:1502.03143* (2015).

[5] Daniel R Canelhas, Todor Stoyanov, and Achim J Lilienthal. "Improved local shape feature stability through dense model tracking". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3203–3209.

[6] Daniel R Canelhas, Todor Stoyanov, and Achim J Lilienthal. "SDF tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3671–3676.

[7] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.

[10] Alexander Hermans, Georgios Floros, and Bastian Leibe. "Dense 3d semantic mapping of indoor scenes from rgb-d images". In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 2631–2638.

[11] Dinh-Cuong Hoang, Todor Stoyanov, and Achim J Lilienthal. “High-quality Instance-aware Semantic 3D Map Using RGB-D Camera”. In: *arXiv preprint arXiv:1903.10782* (2019).

[12] Dinh-Cuong Hoang, Todor Stoyanov, and Achim J Lilienthal. “High-quality Instance-aware Semantic 3D Map Using RGB-D Camera”. In: *Proceedings of the European Conference on Mobile Robots (ECMR)*. 2019.

[13] Binh-Son Hua et al. “Scenenn: A scene meshes dataset with annotations”. In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 92–101.

[14] Christian Kerl, Mohamed Souiai, Jürgen Sturm, and Daniel Cremers. “Towards Illumination-Invariant 3D Reconstruction Using ToF RGB-D Cameras”. In: *2nd International Conference on 3D Vision*. 2014, pp. 39–46.

[15] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[17] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. “Label Fusion: A Pipeline for Generating Ground Truth Labels for Real RGBD Data of Cluttered Scenes”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[18] Daniel Maturana and Sebastian Scherer. “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2015, pp. 922–928.

[19] John McCormac, Ronald Clark, Michael Bloesch, Andrew J Davison, and Stefan Leutenegger. “Fusion++: Volumetric Object-Level SLAM”. In: *arXiv preprint arXiv:1808.08378* (2018).

[20] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. “Semanticfusion: Dense 3d semantic mapping with convolutional neural networks”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4628–4635.

[21] 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*. 2017, pp. 652–660.

[22] P. Rodriguez-Gonzalvez, D. Gonzalez-Aguilera, H. Gonzalez-Jorge, and D. Hernandez-Lopez. “Low-Cost Reflectance-Based Method for the Radiometric Calibration of Kinect 2”. In: *IEEE Sensors* (2016).

[23] Martin Rünz and Lourdes Agapito. “MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects”. In: *arXiv preprint arXiv:1804.09194* (2018).

[24] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. “Real-time visual odometry from dense RGB-D images”. In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 719–722.

[25] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. “A benchmark for the evaluation of RGB-D SLAM systems”. In: *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 573–580.

[26] Akin Tatoglu and Kishore Pochiraju. “Point Cloud Segmentation with LIDAR Reflection Intensity Behavior”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2012, pp. 786–790.

[27] Bastien Veyssiere. “Plastic wrapping detection application based on reflectance models for mobile robots”. Master’s Thesis. UPSSITECH, Université Toulouse III Paul Sabatier, Aug. 2018.

[28] Chen Wang et al. “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”. In: *arXiv preprint arXiv:1901.04780* (2019).

[29] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. “ElasticFusion: Real-time dense SLAM and light source estimation”. In: *The International Journal of Robotics Research* 35.14 (2016), pp. 1697–1716.

[30] Kai M. Wurm, Henrik Kretzschmar, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. “Identifying vegetation from laser data in structured outdoor environments”. In: *Robotics and Autonomous Systems* 62.5 (2014), pp. 675–684. DOI: 10.1016/j.robot.2012.10.003.

[31] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes”. In: *arXiv preprint arXiv:1711.00199* (2017).

[32] Anestis Zaganidis, Li Sun, Tom Duckett, and Grzegorz Cielniak. “Integrating deep semantic segmentation into 3-d point cloud registration”. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 2942–2949.