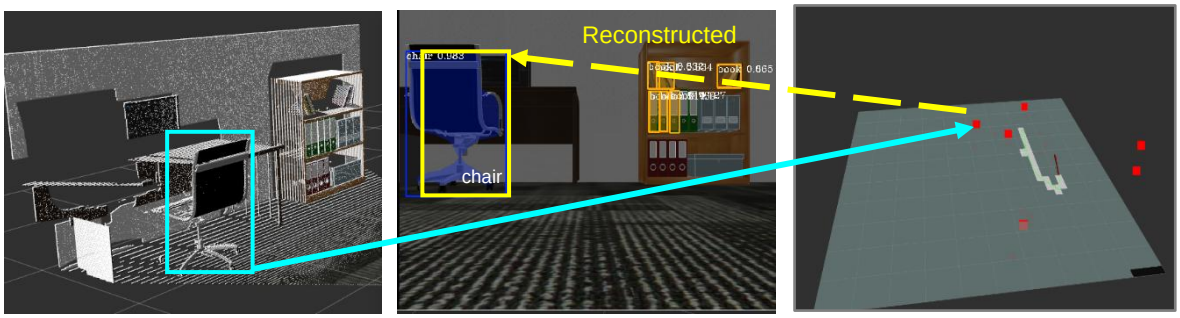
3D world coordinates to use this information during navigation. Lastly, if the mobile robot detects the same object as a different class from the previously detected class, we compare the previous class with the current one and use the objectness score to select the more accurate class label.



(a) Point cloud (b) RGB Image (c) The 3D world coordinates

Fig. 3 These figures show how our approach maintains the locations and classes of detected objects in the 3D space, and reconstructs them onto the 2D image plane. The images in (a) and (b) show a point cloud and RGB image from the RGB-D camera mounted on a mobile robot. (c) is a reconstructed 3D map in the 3D world coordinates. When the robot recognizes a chair during navigation, we obtain the point cloud set in the 2D bounding box region of the chair (the box in (a)) and convert the point cloud of the chair to a 3D location (the red point in (c)). This 3D information contains the 2D bounding box location, class probability, mean depth, etc. When the chair is observed in the field of view again, we then project the 3D location of the previously detected object onto the RGB image plane (the yellow box in (b)) using affine projection. Based on this information, we can compare which class is more accurate between the reconstructed 2D bounding box (the yellow box in (b)) and the currently detected 2D bounding box (the blue box in (b)).

To compare between different classes in the same location, we reconstruct the 2D bounding box from the saved 3D object information using an affine projection. In homogeneous coordinates, the affine projection is represented as the following:

$$p_{img} = K[R|t]P_{world}\,, \tag{1}$$

where p_{img} is $[\mathrm{x}, \mathrm{y}, 1]^{\mathrm{T}}$ in the 2D image coordinates, P_{world} is $[\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1]^{\mathrm{T}}$ in the 3D world coordinates, K is an intrinsic matrix, and [R|t] is an extrinsic matrix. We obtain the intrinsic matrix from camera calibration and the extrinsic matrix from odometry of the mobile robot. Thus, we can save information about detected objects such as their location and class regardless of the motion of the mobile robot.

2.2. CLASS COMPARISON USING AN OBJECTNESS SCORE

In this section, we introduce a novel method to determine which class to choose when an object is detected as belonging to different classes. Most DLNs train their network architectures with an image dataset containing the overall shapes of objects. Therefore, it is necessary to maintain a sufficient distance between the robot and objects for DLNs to perform accurate object detection. Using this property, we

formulate an objectness score (S_{obj}) and define an object managing algorithm (Alg. 1). S_{obj} is used to determine which object class is more accurate when one object is recognized as belonging to multiple classes. Alg. 1 computes the 2D IOU to determine whether it is the same object and uses the objectness score to select a more accurate class label.

Computer-vision-based DLNs such as Faster R-CNN (Ren 2015), Mask R-CNN (He 2017), and YOLOv3 (Redmon 2018) estimate 2D bounding boxes and classes which have class probabilities. Traditional methods (e.g., Bowman 2017 and Qi 2018) use class probabilities from DLNs to determine which class assignment to keep. Class probabilities are very simple and useful, but those often make a mistake when only a part of the object is observed along the distance. We assume that an estimated class is more accurate when the DLNs can see the entire shape of an object. Therefore, considering the distance from the object, we define an objectness score (S_{obj}) which is designed to give more weight to observations made when the distance is sufficient for the whole object shape to be observed. Here, S and obj mean a score and an object, respectively.

Objectness Score (S_{obj}). The objectness score is defined as the following:

$$S_{obj} = \alpha S_{poc} + (1-\alpha) S_{\mathrm{dept\,h}}, \tag{3}$$

where S_{poc} is a class probability estimated by a DLN such as Mask R-CNN, $S_{\mathrm{dept\,h}}$ is the normalized distance score between the camera and an object, and α is a relative weight between S_{poc} and $S_{\mathrm{dept\,h}}$. Intuitively speaking, we give a higher score when we have a high class probability from detecting an object that is also located a sufficient distance from the camera.

In more detail, if the distance between the robot and an object is too great, the object looks like a dot. This problem causes DLNs to misinterpret the class of the detected object. Thus, we define $S_{\mathrm{dept\,h}}$ by using field-of-view information from the RGB-D camera to quantify a sufficient distance for recognizing the entire shape of the object. To realize this idea, we define $S_{\mathrm{dept\,h}}$ as a normalized distance with a value between 0 and 1 using min-max normalization:

$$\mathrm{S}_{\mathrm{dept\,h}} = \frac{d_i - \min(d)}{\max(d) - \min(d)}, \tag{2}$$

where d_i is the distance between a robot and the i-th detected object, and $\min(d)$ and $\max(d)$ are pre-defined values for the minimum and maximum distance from the RGB-D camera mounted on the robot for observing objects in a sufficient distance range; in our test, $\min(d)$ and $\max(d)$ are set to be 0.8 m and 3 m respectively.

We now explain how our algorithm (Alg. 1) works. Alg. 1 selects a more accurate object through the objectness score and stores the detected object information such as a 3D position, 2D bounding box, class, and objectness score in the list of detected objects (P_{obj}) during navigation. When the list of detected objects P_{obj} is empty and an object is detected, the detected object is stored in P_{obj} (Line: 1). Then, if a new object

o_{new} is detected, we search the k-nearest neighbor objects (ρ_1, ρ_2, …, ρ_k) of o_{new} from P_{obj} using *SearchKNN(·)* (Line: 6), which returns the k-nearest neighbor objects (k=3). Subsequently, we check whether o_{new} and any of the k-nearest neighbor objects share the same position through *Check2DIOU(·)* (Line: 8), which returns *true* if the 2D IOU (intersection over union) between o_{new} and ρ_i is larger than a threshold value (0.9), where ρ_i is the i-th nearest neighbor object. If *Check2DIOU(·)* is *true*, we consider o_{new} and ρ_i to be in the same location. Thus, if the objectness score of o_{new} ($S_{o_{new}}$) is larger than the objectness score of the existing object ρ_i (S_{ρ_i}), ρ_i is replaced by o_{new} because o_{new} has more accurate object information than ρ_i according to the objectness score.

In addition, if o_{new} does not overlap with any one of the k-nearest neighbor objects (ρ_1, ρ_2, …, ρ_k), o_{new} is newly inserted in P_{obj} (Line: 12, 13).

Algorithm 1 Class selection algorithm

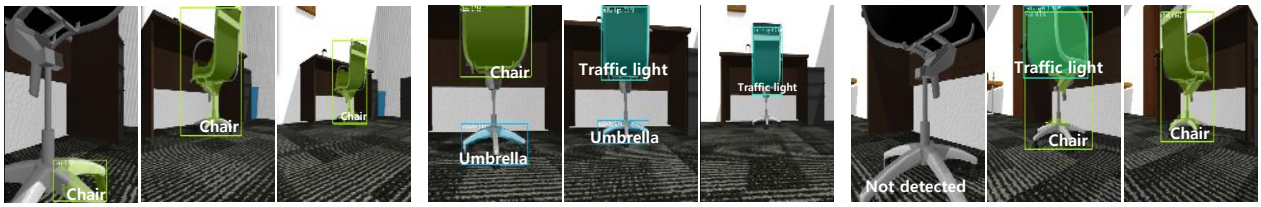
Input: A newly detected object o_{new},
A list of detected objects P_{obj}
Output: Updated P_{obj}

1: **if** P_{obj} is empty **then**
2: *Insert* o_{new} into P_{obj};
3: **end if**
4: $\rho_i \leftarrow$*SearchKNN(*o_{new}*)*;
5: *Calculate* an objectness score of o_{new}, $S_{o_{new}}$;
6: **while** ρ_i is exist **do**:
7: *Calculate* an objectness score of ρ_i, S_{ρ_i};
8: **if** *Check2DIOU(*o_{new}, ρ_i*)* **and** $S_{o_{new}}$ is larger than S_{ρ_i} **then**
9: *Replace* ρ_i with o_{new};
10: **end if**
11: **end while**
12: **if** o_{new} is not the same location as all ρ_i **then**
13: *Insert* o_{new} into P_{obj};
14: **end if**

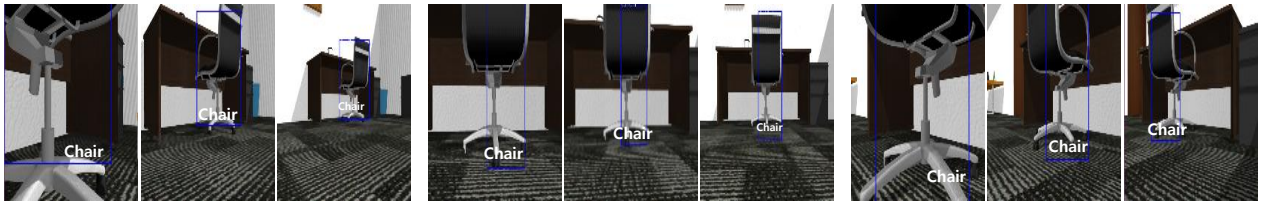
3. EXPERIMENTS

The goal of our experiment is to maintain the locations and classes of detected objects during navigation. To verify our system, we experimented in the Gazebo simulator within the Robot Operating System (ROS). We used a mobile robot based on a turtlebot model with a mounted RGB-D Kinect camera and the mobile robot moved in the indoor environment (Rasouli 2017). We set the weight $\alpha = 0.4$ in the objectness score shown in Eq. (3). Intrinsic parameters were obtained using a camera calibration technique (Zhang 2000, Khoshelham 2012) and extrinsic parameters were obtained from the pose of the camera mounted on the mobile robot. We tested our algorithm by changing the angle (-45, 0, and 45 degrees) of the mobile robot and the distance (from 0.3 to 3 m). The minimum and maximum distances of $S_{\mathrm{dept\,h}}$ were defined as 0.8 m

and 3 m, respectively. The experimental results (Fig. 3) showed that the location and class of the detected object were maintained when the mobile robot moved toward the object from different angles and distances.



(a) The results of classification with localization from Mask R-CNN



(b) The results of our algorithm

Fig. 4 These image sequences show the results of classification with localization during navigation of a mobile robot. Our method robustly identifies objects during navigation.

4. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a method for maintaining the locations and classes of detected objects using the objectness score. The objectness score determines which objects to maintain between currently and previously detected objects. Experimental results at various angles and distances showed that our method was robust for detecting objects during navigation. In future work, we will extend this approach to distinguish whether or not the maintained object is an obstacle that we need to avoid during navigation.

ACKNOWLEDGMENT

This research was supported by the SW Starlab support program (IITP-2015-0-00199) and the Ministry of Trade, Industry & Energy (MOTIE, Korea). No.10067202

REFERENCES

Bowman, S.L., Atanasov, N., Daniilidis, K., and Pappas, G.J. (2017), "Probabilistic data association for semantic SLAM," *IEEE Int. Conf. on Robotics and Automation*, ICRA.

He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017), "Mask R-CNN," *IEEE Int. Conf. on Computer Vision*, ICCV, 2961-2969.

Khoshelham, K. and Elberink, S.O. (2012), "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, Vol. **12**(2), 1437-1454.

Pirsiavash, H., Ramanan, D., and Fowlkes, C.C. (2011), "Globally-optimal greedy algorithms for tracking a variable number," *IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR.

Rasouli, A. and Tsotsos, J.K. (2017), "The effect of color space selection on detectability and discriminability of colored objects," arXiv:1702.

Redmon, J. and Farhadi, A. (2018), "YOLOv3: An incremental improvement," arXiv:1804.

Ren, S., He, K., Girshick, R., and Sun, J. (2015), "Faster R-CNN: towards real-time object Detection with region proposal networks," *Neural Information Processing Systems*, NIPS.

Song, S. and Xiao, J. (2016), "Deep sliding shapes for amodal 3D object detection in RGB-D images," *IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR.

Qi, C.R., Liu, W., Wu, C., Su, H., and Guibas, L.J. (2018), "Frustum pointnets for 3D object detection from RGB-D data," *IEEE Conf. on Computer Vision and Pattern Recognition*, CVPR.

Zhang, Z. (2000), "A flexible new technique for camera calibration," in *IEEE Transaction on Pattern Analysis and Machine Intelligence*, TPAMI, Vol. **22**, 1330-1334.