A 2D-3D Object Detection System for Updating Building Information Models with Mobile Robots

Max Ferguson
Stanford University
maxferg@stanford.edu

Kincho Law
Stanford University
law@stanford.edu

Abstract

Automation in facility management and construction could significantly improve efficiency and productivity of the building industry. However, for robots and autonomous systems to operate effectively in dynamic and unstructured environments such as construction sites, they must be able to infer or obtain a semantic model of the environment. We propose a system for identifying common objects in a worksite and automatically adding them to an existing geometric model of the environment. The system is composed of two components: A novel 2D-3D object detection network designed to detect and localize a worksite objects, and a multi-object Kalman filter tracking system used to filter false-positive detections. The proposed system is validated using data collected with a purpose-built mobile robot. The mobile robot captures images of the worksite with an RGB-D camera and uses the proposed system to spawn newly placed objects in an existing geometric model of the worksite environment. The annotated RGB-D images are made publicly available to accelerate future research in this field.

1. Introduction

The growing prevalence of automated data-capture systems and semi-autonomous robots is likely to have a positive impact on efficiency and productivity in the construction and facility management sectors. However, for automated systems to work safely and efficiently in a complex environment like a worksite, they must have access to, or be capable of generating a semantic-rich model of the environment. More specifically, these automated systems must understand *what* types of physical objects are near them, and *where* those objects are positioned in 3D space. To enable safe automation, it is essential that these systems are able to operate correctly in both static and dynamic environments. For example, a construction robot operating in a narrow hallway may have to simultaneously infer the position of static objects, such as construction site tools, and dynamic objects, like people. Figure 1 shows how our proposed system is applied to the task of automatically spawning detected objects in an existing geometric model of the environment.

Creating and maintaining a dynamic model of the building environment remains a difficult challenge in computer science, robotics and civil engineering. Recent advances in Lidar, depth-sensing [12], visual odometry [24], and related technologies [6, 16] have made it possible to obtain a static representation of large-scale spaces with relative ease. In most cases, this static representation exists as a point cloud or polygon mesh. Research efforts in computer vision have focused on extracting semantic information from these static representations with encouraging results [1, 2]. However, we have not yet seen any comprehensive examples where these techniques are used to parse point cloud information from a mobile robot on a frame-by-frame basis, and automatically update a large-scale geometric model.

Parsing point-cloud data from a mobile robot introduces a number of challenges which are not encountered when using pre-collected indoor datasets:

- **Partial Availability of Information:** At a given time, information can only be observed through sensors on the mobile robot. Many existing methods can only be used effectively once a point-cloud is available for the entire space.
- **Moving Objects:** The majority of indoor scene understanding datasets are presented under the assumption that the environment remains static during data collection. This assumption is unreasonable in situations where semantic parsing is used to enable safe automation in worksites or modern facilities.
- **Localization Error and Motion Blur:** Point cloud data collected from a moving mobile robot tends to be noisier than that collected from a stationary RGB-D camera or Lidar sensor. Hence, semantic parsing systems designed for mobile robots need to be designed with additional robustness in mind.

Figure 1. Generating large-scale dynamic spatial information models using mobile robots.

These issues signify the necessity of adopting new approaches to semantic parsing of point cloud data from mobile robots. This paper introduces a methodology that, given an RGB-D image from a moving mobile robot, identifies objects of interest in that image, estimates the location of each object, and continues to refine that estimate of the object location as more data becomes available.

One unique property of our system is the method by which we predict the position of an object in 3D space. In particular, we leverage a state-of-the-art 2D instance segmentation network to generate region proposals in 2D image space. Each 2D region proposal is then projected into 3D space using depth information from the RGB-D camera. We then combine features from the 2D and 3D representations to refine the predicted size and position of the object.

Another key property of the method is using a Kalman filter tracker to track the position of detected objects across multiple RGB-D frames. This allows us to refine the position estimates of stationary objects upon receiving additional frames from the RGB-D camera. It also provides a good way to filter false-positive detections. As an added benefit, the system naturally supports real-time position estimation of moving objects, like people.

The following points summarize the main contributions of this paper:

1) We propose Object R-CNN, a new 2D-3D object detection algorithm that can simultaneously classify common objects in a worksite, estimate their size, and predict their position relative to the camera.

2) We show that a multi-object Kalman filter-based tracking system can be used to track objects in the mobile-robot field of view, refine position estimations, and filter false-positive detections.

3) We collected an RGB-D dataset using a purpose-built SLAM-enabled mobile robot, and annotated objects in the dataset with 2D segmentation masks and 3D position and size. The dataset and annotations are made publicly available at http://worksite.stanford.edu.

The paper is organized as follows: An overview of related work is provided in Section 2. A detailed description of the 2D-3D object recognition system is provided in Section 3. Section 4 introduces our multi-object tracking system. Section 5 explains how data is collected to validate the proposed algorithms using a custom-built mobile robot, and provides quantitative results. The paper is concluded with a brief discussion and conclusion.

2. Related work

This section provides an overview of the related works, with particular emphasis on the following main points that differentiate our approach. Specifically, (1) we detect objects of interest using 2D images and project those detections into 3D space, whereas most existing approaches detect objects in either 2D space or 3D space; (2) we combine features from the 2D image and 3D point cloud to refine the predicted size and position of each object; and (3) our system supports detection and localization of stationary and moving objects.

Object Detection and Scene Understanding: Motivated by the potential applications of scene understanding, research efforts in computer vision have been directed towards the problem of automatically extracting semantic information from 2D images [11] and 3D point clouds [1, 2]. With recent technological advances, 2D object detection algorithms are now able to identify a diverse set of objects with a high level of accuracy. Additionally, some of the leading 2D object detection algorithms are fast enough to operate in real-time [27]. However, despite recent advances, 3D object detectors are still limited in accuracy, inference speed, and the ability to recognize a diverse set of objects. A recent survey of indoor scene reconstruction and understanding is provided in [23].

The system described in this work is most closely related with systems that address the problem of 3D object detection. The VoxNet architecture addresses this problem by utilizing a 3D CNN, and achieves relatively good per-

formance for many 3D object detection tasks [21]. Our object detection network shares many similarities with the methods proposed in [5] and [31], which combine 2D and 3D information to generate object proposals. The current state-of-the-art systems include Frustum PointNets, which generate region proposals using a 2D image detector [26]. Unlike our system, Frustum PointNets do not utilize features from the 2D image to classify objects. Hence, it has difficulty identifying small or distant objects using comparatively sparse point cloud data.

Automated Data Collection: There has been some interest in using Unmanned Aerial Vehicles (UAVs) or Unmanned Ground Vehicles (UGVs) to automatically gather information about the building environment [10, 25]. However, the computer vision systems so far tend to be incomplete, often lacking the ability to accurately classify or localize objects of interest [3, 9, 22].

SLAM: The notion of automatically creating geometric environment maps has been explored extensively by the robotics community. Perhaps the most noteworthy technique is Simultaneous Localization and Mapping (SLAM), which aims to construct a map of an unknown environment while simultaneously keeping track of an agent's location within it [32]. Related task in robotics include visual-SLAM and structure from motion (SfM), which generate a static geometric representation of the environment using a consecutive series of RGB images. While effective for reconstructing scene geometry, these techniques do not directly address the problem of scene understanding.

3. 2D-3D Object Detection System

Inspired by the recent progress in 2D instance segmentation, we propose a new computer vision system to simultaneously identify objects in 2D images and localize them in 3D space. The proposed system, referred to as Object R-CNN, works by projecting 2D region proposals, into a 3D point cloud captured with an RGB-D camera.

Object R-CNN is an extension of the 2D instance segmentation framework, Mask R-CNN, which has obtained state-of-the-art accuracy in a number of instance segmentation challenges [11]. In our system, Mask R-CNN is first used to detect objects of interest in the RGB image. Each detection is projected into 3D space to form a 3D region proposal. Features are cropped from the point cloud using this region proposal and fed into a 3D CNN. Features from Mask R-CNN and the 3D CNN are concatenated and used to generate a refined estimate of the size and position of the object. This conceptually simple extension of Mask R-CNN allows 3D object detection to be performed on RGB-D images, while achieving the high level of accuracy normally only reached by 2D object detectors.

3.1. Mask R-CNN

We now briefly introduce the Mask R-CNN framework [11]. Mask R-CNN consists of two stages. The first stage, called the Region Proposal Network (RPN), proposes candidate object bounding boxes, referred to as regions of interest (RoIs). The second stage, extracts features from each RoI and performs classification, bounding-box regression, and instance segmentation in parallel. Each component in Mask R-CNN is briefly discussed as follows:

Backbone: In Mask R-CNN, a deep CNN is used to convert RGB images into a featurized representation. A number of common neural network architectures have been used as the backbone for Mask R-CNN, including variants of the VGG [30] and ResNet [28] architectures. To better detect objects of different scale, a feature pyramid network (FPN) can be used as the backbone [18]. Mask R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, but otherwise the rest of the approach is similar to vanilla ResNet.

Region Proposal Network: The RPN takes a feature map of any size as input and outputs a set of rectangular object proposals. To generate region proposals, a small CNN is convolved with a featurized representation of the image. The input to this small CNN is an $n \times n$ spatial window of the feature map from the CNN backbone. The RPN has two sibling output layers: a box-classification layer (*cls*) and a box-regression layer (*reg*). The *cls* layer outputs objectness scores that estimate the probability of object and not object for each candidate bounding box. The *bbox* layer outputs coordinate adjustments for each of the boxes. At the end of the region proposal stage, the top q candidate boxes are selected by objectness score as the RoIs.

Region-based Detector: In the second stage of Mask R-CNN, each RoI is assigned a class and adjusted bounding box coordinates by the region-based detector (RBD). The input to the RBD is cropped from the CNN backbone, according to the predicted size and location of the RoI. An RoIAlign layer is used to convert the input to a fixed-size feature map. The RBD produces two output vectors and a segmentation mask tensor: The first vector contains probability estimates for each of the k object classes plus a catch-all background class; the second vector encodes refined bounding-box positions; the segmentation mask tensor encodes k binary masks of resolution $m \times m$, one for each of the k classes. Figure 3 shows the output of Mask R-CNN when trained on our worksite object dataset.

3.2. Object R-CNN

We now describe how the Mask R-CNN framework is extended to support localization of objects in 3D space. The 2D bounding boxes from Mask R-CNN are projected into 3D space using depth information in the RGB-D image, generating a set of region proposals in 3D space. Each

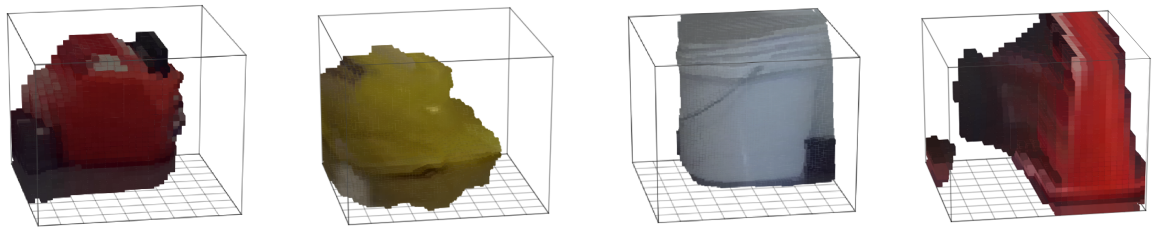

Figure 2. Examples of voxel encodings of common worksite objects. From the left (a) a gas container, (b) a hard hat, (c) a bucket and (d) a safety cone.

Figure 3. Two-dimensional image detections using Mask R-CNN after training on our worksite object dataset

region proposal is featurized using a 3D CNN. Features are concatenated with those from RoIAlign and used to predict the size and location of objects in 3D space.

Three-dimensional Anchors: RGB-D images describe a set of points in 3D space, each with an RGB color value. Rectangular region proposals from Mask R-CNN are projected into this 3D space, to form a frustrum. We define a 3D anchor A, as a cubic region of space at depth Z_A which ideally contains the target object. We estimate the depth of the target object, using the median point depth inside the frustum. The side length H_A of A is estimated by projecting the 2D bounding box to depth Z_A, in 3D space:

$$H_A = \max\left(Z_A \frac{w_{bbox}}{f_x}, Z_A \frac{h_{bbox}}{f_y}\right), \tag{1}$$

where w_{bbox} and h_{bbox} are respectively the width and height of the bounding box in pixels, and f_x and f_y are the focal lengths of the camera in the horizontal and vertical directions, respectively. The center of A is obtained by projecting the center of the bounding box to depth Z_A:

$$X_A = \frac{Z_A}{f_x}(x_{bbox} - c_x), \tag{2}$$

$$Y_A = \frac{Z_A}{f_y}(y_{bbox} - c_y), \tag{3}$$

where (X_A, Y_A, Z_A) is the center of A in 3D space, and (x_{bbox}, y_{bbox}) is the center of the bounding box in the image coordinate system. Parameters c_x and c_y are the intrinsic camera parameters, which are obtained during the camera calibration process.

Thus far, we have defined a region in 3D space, A, which we believe contains the object of interest. We use this 3D region proposal to crop features from the point cloud. More specifically, the anchor A is divided into a predetermined number of voxels. Each voxel is assigned an RGBA color based on its position, using a method similar to space carving [17]. More specifically, the position of each voxel in 3D space is projected back onto the 2D RGB-D image using a pinhole camera model. If the depth of the voxel is less than the depth indicated by the depth channel of the RGB-D image, the voxel is colored black and assigned an alpha (opacity) value of 0. Otherwise, the voxel is assigned an alpha value of 1 and colored to match the corresponding pixel. Once constructed, the voxel representation encodes a colored representation of each target object in 3D space. This encoding contains valuable spatial information for accurately predicting the size and position of the object, as shown in Figure 2.

The fixed-size voxel representation is featurized using a simple 3D CNN, similar to VoxNet [21]. In a 3D CNN, each convolutional filter has height, width and depth dimensions. This allows the network to process 3D spatial relationships, in much the same way as a classical CNN can capture 2D spatial relationships. 3D max-pooling layers are used to reduce the dimensionality of the 3D representation. After multiple convolution and pooling layers, the representation is flattened into a single-dimension array and concatenated with features from ROIAlign. The resulting representation contains 3D features from the voxel representation and 2D features from the Mask R-CNN backbone. The full architecture is shown in Figure 4. The architecture of the 3D CNN is provided in the supplementary [7].

3.3. Loss Functions

Object R-CNN is trained in a similar manner to Mask R-CNN. The Mask R-CNN framework defines loss functions for classification L_{cls}, bounding box regression L_{bbox} and image segmentation L_{mask}. We propose two additional loss

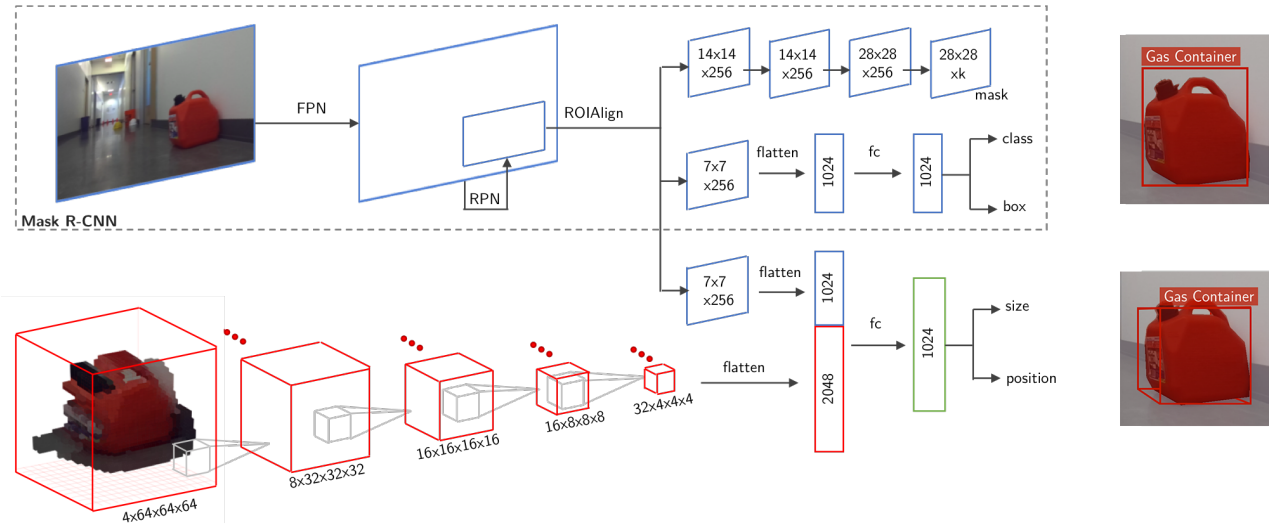

Figure 4. Neural network architecture of the proposed object detection network. Numbers denote spatial resolution and channels. In the case where a tensor has more than one dimension, the channel dimension is always the last. Arrows denote either 2D convolution, fully connected (fc) layers, or flattening operations. All 2D convolution layers are 3×3 with stride 2.

functions, one for 3D position prediction and one for 3D size prediction. The position of the object in 3D space is encoded as follows:

$$\phi_{pos}(B_A; A) = \frac{1}{H_A} \begin{bmatrix} X_B - X_A \\ Y_B - Y_A \\ Z_B - Z_A \end{bmatrix} \tag{4}$$

where (X_B, Y_B, Z_B) is the geometric centroid of the box B. Similarly, the size of each bounding box B in 3D space is encoded as follows:

$$\phi_{size}(B_A; A) = \frac{1}{H_A} \begin{bmatrix} W_B \\ H_B \\ D_B \end{bmatrix} \tag{5}$$

where W_B, H_B and D_B are the width, height and depth of the box B. The prediction of the position of each object in 3D space is expressed as a regression problem. The position-based loss for A is expressed as a function of the predicted position encoding $f_{pos}(\mathcal{I}; A, \theta)$ and ground truth position encoding $\phi_{pos}(B_A; A)$:

$$L_{pos}(A, \mathcal{I}; \theta) = \ell_{Huber}\left(\phi_{pos}(B_A; A) - f_{pos}(\mathcal{I}; A, \theta)\right), \tag{6}$$

where $\mathcal{I}$ is the RGB-D image, θ is the neural network parameters, and ℓ_{Huber} is the Huber loss function [14]. The size-based loss for A is expressed as a function of the predicted size encoding $f_{size}(\mathcal{I}; A, \theta)$ and ground truth size encoding $\phi_{size}(B_A; A)$:

$$L_{size}(A, \mathcal{I}; \theta) = \ell_{Huber}\left(\phi_{size}(B_A; A) - f_{size}(\mathcal{I}; A, \theta)\right), \tag{7}$$

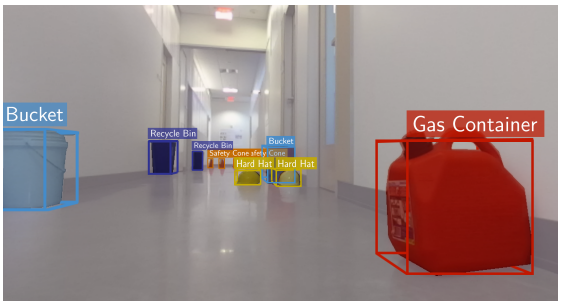

Figure 5. Three-dimensional object detections using Object R-CNN after training on our worksite object dataset. Near and far objects are correctly classified.

The loss for each anchor A is defined as the weighted sum of L_{cls}, L_{bbox}, L_{mask}, L_{pos} and L_{size}. The loss for each RGB-D image is defined as the average loss across positive anchors. An example of the output from Object R-CNN is shown in Figure 5.

4. Object Tracking and Detection Filtering

The output from CNN-based object detectors, including Object R-CNN, tends to be somewhat noisy, in that they generate false-positive detections with non-trivial probability. In this section, we describe a multi-object tracking system, that tracks the position of detected objects, allowing false-positive detections to be filtered out.

4.1. Position Estimation

A Kalman filter is created to maintain an estimate of the position of each object in the mobile robot field of view (FOV). In the proposed system, one Kalman filter is assigned to each object being tracked. Without loss of generality, assume that there is a single object in the FOV of the mobile robot at time t. Let $P_t \in \mathbb{R}^3$ denote the true position of the object centroid in the global coordinate system. Similarly, let $\dot{P}_t \in \mathbb{R}^3$ represent the true velocity of the object in the same coordinate system. The object can be modeled as a discrete time system with true state x:

$$x_t = \begin{bmatrix} P_t \\ \dot{P}_t \end{bmatrix} \tag{8}$$

It is assumed that changes to the state are governed by a state transition model F with process noise ϵ:

$$x_t = Fx_{t-1} + \epsilon_{t-1}, \tag{9}$$

where F is chosen to enforce constant-velocity as defined in [4]. In the case of moving objects, such as people, the process noise accounts for changes in state not explained by the constant velocity model. The true state x is not directly measurable, but is related to the observed position of the object $z \in \mathbb{R}^3$ as:

$$z_t = Hx_t + v_t, \tag{10}$$

where H is the observation model and v_t is the observation noise. The observation noise accounts for noise induced during the robot localization, object detection, and depth estimation. As the mobile robot is unable to directly observe the velocity of objects, the observation matrix is defined as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

The Kalman filter computes the optimal state estimate $\hat{x}$ by recursively combining previous estimates with new observations. It consists of two phases: (1) predict, where the optimal state $\hat{x}_{t|t-1}$ prior to observing z_t is computed; and (2) update, where the optimal posterior state $\hat{x}_{t|t}$ after observing z_t is computed. Additionally, it computes the prior error covariance $\Sigma_{t|t-1} = \text{cov}(x_t - \hat{x}_{t|t-1})$ and posterior estimate error covariance $\Sigma_{t|t} = \text{cov}(x_t - \hat{x}_{t|t})$. The prediction step is:

$$\hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1}, \tag{12}$$

$$\Sigma_{t|t-1} = F\Sigma_{t-1|t-1}F^T + Q, \tag{13}$$

and the update step is:

$$K_t = \Sigma_{t|t-1}H^T(H\Sigma_{t|t-1}H^T + R)^{-1}, \tag{14}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H\hat{x}_{t|t-1}), \tag{15}$$

$$\Sigma_{t|t} = (I - K_tH_t)\Sigma_{t|t-1}, \tag{16}$$

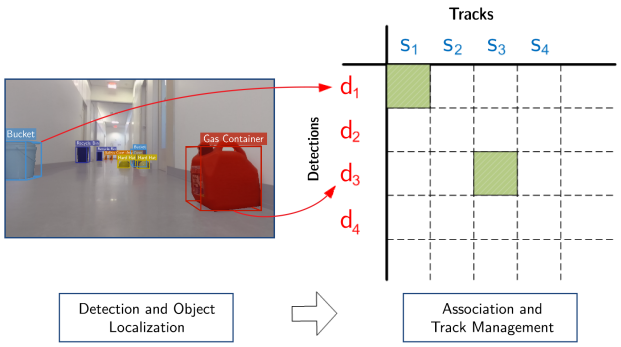

Figure 6. Association of detected objects to tracks.

where I is the identity matrix, Q is the process noise covariance and R is the measurement noise covariance. K_t is the optimal Kalman gain, which influences how much the observation impacts the state estimate. The process noise covariance and measurement noise covariance for each Kalman filter are varied based on whether the object is likely to move (people) or whether the object tends to remain stationary.

4.2. Association

Problems arise when there are more than one object in a particular frame, or when objects leave the mobile robot FOV. At each time step, detected objects are assigned to an existing Kalman filter, or a new Kalman filter is created to track the object. More formally, the mobile robot maintains a set of trackers $S = \{s_i | i = 0, 1, 2...N\}$, which encode its belief about objects in the FOV, where s_i represents a tracker. At each time step, the object detector returns a set of detections $D = \{d_j | j = 0, 1, 2...M\}$, where d_j represents a single detection. At each time step, every detection in D is assigned to a tracker in S, or in the case of an unmatched detection, a new tracker is added to S. The association process is shown diagrammatically in Figure 6.

Similarity Metric: A similarity metric is required to match detections and trackers. We use the volumetric intersection over union IoU_{vol}, to quantify the similarity between tracked objects and 3D object detections [29]. The width, height and depth of each tracked object is estimated by averaging the corresponding dimensions across previous detection assignments.

Assignment: The Hungarian algorithm [15] is used to find the assignment which maximizes the sum of IoU_{vol} of trackers and detections. A cost matrix C is defined such that its coefficient, c_{ij} is the IoU_{vol} of tracker s_i and detection d_j. In the case where the number of detections is not equal to the number of trackers, C is padded with rows or columns of zeros to make it a square matrix. The assignment process is conducted separately for each class, so as to ensure that tracked objects cannot change class.

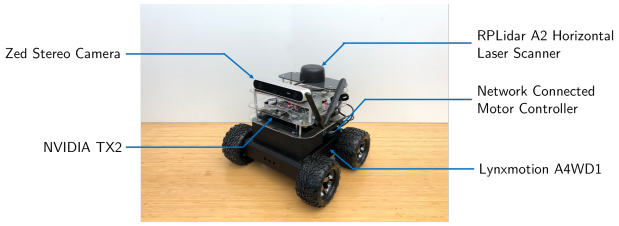

Figure 7. SLAM-enabled mobile robot.

Unmatched Detections and Trackers: Problems arise when new objects enter the FOV, tracked objects leave the FOV, or the detector produces false-positives. When an object is first detected, it is not matched with any existing track, and is referred to as unmatched detection. A new tracker is created for each unmatched detection. After an object leaves the FOV, the previously established tracker has no detection to associate with. In this scenario, the tracker is referred to as unmatched tracker. If the tracker is unmatched for more than three timesteps it is removed from the set of active trackers.

5. Experiments

In this section, we show how the proposed system can be used to automatically populate an existing model of the environment with newly spawned objects. A dataset is collected with a mobile robot, and the performance of the method is evaluated quantitatively.

5.1. Updating Large-scale Semantic-rich Models

In the field of facility and construction engineering, building information modeling (BIM) is the process of generating a semantic-rich geometric model of a building. We apply our detection and tracking system to automatically populate a building information model with newly spawned worksite objects.

Objects are added to the building information model, after being tracked for a certain number of frames. The addition process is as follows: A 3D representation of the object is selected from a repository of known CAD objects. The CAD object is resized such that it touches all edges of the predicted bounding box. The object is placed in the building information model such that the geometric center matches the location predicted by the Kalman filter. In some cases, the object will intersect with an object that already exists in the building information model. The objects are merged if they have the same class and a IoU_{vol} value larger than 0.5. Upon receiving subsequent detections, the trackers are updated, and the position of tracked objects is updated in the building information model.

Object class	Train Set	Test Set
Safety Cone	1423	285
Hard Hat	908	181
Bucket	854	169
Recycle Bin	843	170
Safety Gloves	612	122
Gas Canister	481	83
Person	502	104
Total	5623	1114

Table 1. Number of annotated objects in the worksite dataset.

5.2. Dataset

A dataset is collected and annotated to validate the proposed computer vision system. The dataset is collected using a purpose-built SLAM-enabled mobile robot. The mobile robot uses a Zed RGB-D camera for collection of RGB-D images, and an RPLIDAR A2 horizontal laser scanner for localization. A photo of the mobile robot is shown in Figure 7. RGB-D images are collected at 30 Hz. The position and orientation of the mobile robot is estimated at 120 Hz using a high-quality SLAM algorithm [13].

The RGB-D images are annotated with 2D segmentation masks and 3D bounding boxes, for a select set of object classes. A total of 5623 objects are annotated across 1014 RGB-D images to form a training dataset. A further 1114 objects across 205 images are annotated to form the test set. A summary of the dataset is provided in Table 1.

5.3. Implementation Details

Object R-CNN is trained on the aforementioned dataset. The neural network weights for Mask R-CNN are initialized using pretrained weights generated using the Microsoft Common Objects in Context (COCO) dataset [19]. The remaining weights are initialized randomly using the Xavier initialization method [8]. The model is trained by minimizing the total loss $L(\mathcal{I};\theta)$ across all images in the training dataset, using stochastic gradient descent with momentum. When tracking objects, a minimum of three detections is required before an object is added to the building information model. If a tracker is unmatched for three frames, the tracker is removed from the active set of trackers.

5.4. Main Results

The system is evaluated on both 2D bounding box prediction accuracy and 3D reconstruction accuracy. The 11-point average precision (AP) metric is used as a metric for quantifying the 2D bounding box accuracy [20]. A prediction is considered correct if the IoU_{area} of the predicted bounding box and the ground truth is larger than 0.5. The AP for each object class is provided in Table 2.

The system is also evaluated on 3D object detection, us-

Object class	Mask R-CNN	Object R-CNN
Safety Cone	0.95	0.95
Hard Hat	0.83	0.84
Bucket	0.84	0.84
Recycle Bin	0.86	0.85
Safety Gloves	0.79	0.81
Gas Canister	0.87	0.85
Person	0.86	0.86
mAP	0.87	0.87

Table 2. Average precision for 2D bounding box prediction on the worksite object test dataset.

Object class	Baseline [5]	Object R-CNN
Safety Cone	0.77	0.88
Hard Hat	0.60	0.76
Bucket	0.68	0.65
Recycle Bin	0.79	0.80
Safety Gloves	0.65	0.80
Gas Canister	0.70	0.81
Person	0.69	0.75
mAP	0.71	0.79

Table 3. Average precision for 3D bounding box prediction on the worksite object test dataset.

ing the worksite object dataset. The method proposed in [5] is used as a baseline. Both systems are trained on the worksite object train set and evaluated on the worksite object test set. A prediction is considered correct if the IoU_{vol} of the predicted bounding box and the ground truth is larger than 0.5. The results are presented in Table 3.

Thus far, we have shown that the proposed system can accurately detect and localize objects in 2D and 3D space. We also test the ability of the system to accurately spawn new objects in the building information model. The annotated test dataset is used to form an accurate representation of the object positions in the building. The proposed system is used to reconstruct scenes in the test dataset. A detection are considered correct if the predicted position matches the ground truth with an IoU_{vol} greater than 0.5. As the object tracking system does not produce a confidence score in the position of objects, it is difficult to use the AP metric to measure the performance of the entire system. Instead, the precision and recall are reported directly. Moving objects are ignored. To further understand the characteristics of the system we remove the tracking and merging components and test the system again. The results are provided in Table 4.

Test System	Precision	Recall	F1
Proposed system	**0.90**	0.87	**0.89**
Without tracking	0.34	0.94	0.50
Without tracking/merging	0.11	**0.95**	0.20

Table 4. Scene reconstruction ablation study. The proposed system is used to spawn new objects in known geometric model of the building. The scene reconstruction accuracy is compared to that from a system without the tracking and/or merging components.

6. Discussion

The proposed system enables the dynamic reconstruction of worksite scenes using RGB-D images from a mobile robot. The recall of the proposed system is higher when the tracking component is removed, as fewer correct detections are incorrectly filtered out. However, the precision is significantly reduced when the tracking system is removed, due to a much larger number of false-positive detections.

The proposed method is not without limitations. As the 3D object position is calculated relative to the mobile robot position, small errors in mobile robot localization directly influence the accuracy of object localization. The system is also computationally intensive to train and evaluate, due to the dependence on two large 2D CNN's and one relatively large 3D CNN. Our implementation was limited to 1.1 frames per second at inference. Finally, our system does not predict the orientation of objects.

There are many opportunities for future work in this domain. The collection and of a larger dataset would enable a more diverse range of objects to be detected. Future work could also focus on extracting semantic information about objects in the building, such as color or texture.

7. Summary and Conclusion

This paper presented a computer vision system that enables common worksite objects to be automatically detected and added to a building information model, using RGB-D images from a mobile robot. The paper introduced Object R-CNN, a new computer vision system for joint 2D-3D object detection, and a Kalman-filter system for tracking objects in the mobile robot field of view. We demonstrated that the system can accurately detect and localize objects in 2D and 3D space. Finally, we showed that the proposed tracking system can be used to filter false-positive detections, resulting in improved detection accuracy.

Acknowledgment

This research is partially supported by the Center for Integrated Facility Engineering at Stanford University. The first author is also supported by the John A. Blume fellowship.

References

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3D semantic parsing of large-scale indoor spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016.

[3] K. Asadi and K. Han. Real-time image-to-BIM registration using perspective alignment for automated construction monitoring. In *Construction Research Congress*, 2018.

[4] Y. Bar-Shalom and K. Birmiwal. Variable dimension filter for maneuvering target tracking. *IEEE Transactions on Aerospace and Electronic Systems*, 18(5):621–629, 1982.

[5] Z. Deng and L. J. Latecki. Amodal detection of 3D objects: Inferring 3D bounding boxes from 2D ones in RGB-depth images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[6] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3D visual SLAM with a hand-held RGB-D camera. In *The RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, pages 1–15, 2011.

[7] M. Ferguson and K. H. Law. Supplementary material: A 2D-3D object detection system for updating building information models with mobile robots, Accessed November 20, 2018. Available at: http://worksite.stanford.edu/supplementary.pdf.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

[9] H. Hamledari, B. McCabe, and S. Davari. Automated computer vision-based detection of components of under-construction indoor partitions. *Automation in Construction*, 74:78–94, 2017.

[10] H. Hamledari, E. Rezazadeh Azar, and B. McCabe. IFC-based development of as-built and as-is BIMs using construction and facility inspection data: Site-to-BIM data transfer automation. *Journal of Computing in Civil Engineering*, 32(2):04017075, 2017.

[11] K. He, G. Gkioxari, P. Dollr, and R. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[13] W. Hess, D. Kohler, H. Rapp, and D. Andor. Real-time loop closure in 2D LIDAR SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278, 2016.

[14] P. J. Huber. *Robust Statistics*. John Wiley & Sons, 2005.

[15] R. Jonker and T. Volgenant. Improving the Hungarian assignment algorithm. *Operations Research Letters*, 5(4):171–175, 1986.

[16] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2100–2106, 2013.

[17] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[18] T.-Y. Lin, P. Dollr, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, page 3, 2017.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014.

[20] C. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[21] D. Maturana and S. Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[22] B. McCabe, H. Hamledari, A. Shahi, P. Zangeneh, and E. R. Azar. Roles, benefits, and challenges of using UAVs for indoor smart construction applications. In *ASCE International Workshop on Computing in Civil Engineering*, pages 349–357. American Society of Civil Engineers, 2017.

[23] M. Naseer, S. H. Khan, and F. Porikli. Indoor scene understanding in 2.5/3D: A survey. *arXiv preprint arXiv:1803.03352*, 2018.

[24] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages I–I, 2004.

[25] J. Park, Y. K. Cho, and D. Martinez. A BIM and UWB integrated mobile robot navigation system for indoor position tracking applications. *Journal of Construction Engineering and Management*, 6(2):30–39, 2016.

[26] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum PointNets for 3D object detection from RGB-D data. *arXiv preprint arXiv:1711.08488*, 2017.

[27] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

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

[29] Z. Ren and E. B. Sudderth. Three-dimensional object detection and layout prediction using clouds of oriented gradients. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1525–1533, 2016.

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

[31] S. Song and J. Xiao. Deep sliding shapes for amodal 3D object detection in RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016.

[32] S. Thrun and J. J. Leonard. Simultaneous localization and mapping. In *Handbook of Robotics*, pages 871–889. Springer, 2008.