IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 30, 2020, accepted October 19, 2020, date of publication October 28, 2020,
date of current version November 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034537

RoomSLAM: Simultaneous Localization and Mapping With Objects and Indoor Layout Structure

ISMAIL RUSLI[1], BAMBANG R. TRILAKSONO[1,2], (Member, IEEE), AND WIDYAWARDANA ADIPRAWITA[1,2], (Member, IEEE)

[1]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung 40132, Indonesia
[2]University Center of Excellence on Artificial Intelligence for Vision, Natural Language Processing and Big Data Analytics (U-CoE AI-VLB), Institut Teknologi Bandung, Bandung 40132, Indonesia

Corresponding author: Ismail Rusli (ismailrusli@students.itb.ac.id)

This work was supported in part by the Ministry of Research and Technology and the Ministry of Education and Culture of Indonesia through the WCU 2020 Program managed by the Institut Teknologi Bandung. The work of Ismail Rusli was supported in part by the Telkom University.

ABSTRACT This article presents RoomSLAM, a Simultaneous Localization and Mapping (SLAM) method for mobile robots in indoor environments where environments are modeled by points and quadrilaterals in 2D space. Points represent positions of semantic objects whereas quadrilaterals approximate the structural layout of the environment, namely rooms. The benefit of such modeling is threefold. Firstly, rooms are a logical way to partition a graph in large-scale SLAM. Secondly, rooms and objects reduce search space in data association. Lastly, the model contains a higher level of semantic, which is beneficial to autonomous robots whenever inter-room navigation is needed. The method was evaluated with two public datasets and the results were compared to those of ORBSLAM and RGBDSLAM.

INDEX TERMS Indoor mapping, mobile robot, RGBD sensor, semantic, SLAM.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) is a set of techniques to deal with correlated uncertainties in sensor movements and sensor readings in an unknown environment to simultaneously map the environment and track the sensor poses. Typically, it consists of a technique to estimate model parameters from noisy data and a technique to associate two sensor readings from different timestamps (data association, loop-closure detection).

There are several kinds of maps generated by SLAM and it depends on the sensor used. For example, an occupancy grid map is usually generated by SLAM that uses laser rangefinders whereas a sparse point map is generated by SLAM that uses cameras. Recently, SLAM is also able to generate 3D dense reconstruction of an environment with the advance in RGBD sensors and computer hardware [1], [2].

Sparse point map and dense reconstruction are useful models. However, they do not contain semantic information of environments. These models are cumbersome for robots that are projected to work in human environments. Therefore, SLAM researches are now heading toward semantic SLAM, i.e. SLAM that generates maps with semantic entities.

Notable results in semantic SLAM are [3]–[8] and [9]. Similarities between these works are that all maps resemble point-based maps with semantic objects or walls are now taking place of points. Although this is sufficient, having a higher degree of semantic is always desirable. For example, in an indoor environment, if a robot only maps objects, it will see the map as a collection of objects. In reality, objects are clustered spatially, e.g. by rooms. Certainly, this spatial information is advantageous for the robot's autonomy.

This article describes a method of semantic SLAM using objects and walls as a model of an environment. Each object or wall plays a single landmark in the environment, whereas a collection of walls are combined into quadrilaterals to approximate rooms. During the SLAM loop, data association and parameter estimation are localized within this room, which makes SLAM efficient and guarantees map scalability. Rooms are also utilized to detect potential loop-closures. Specifically, contributions of the method are as follows.

The associate editor coordinating the review of this manuscript and approving it for publication was Shafiqul Islam.

TABLE 1. Semantic entities used in recent works in semantic SLAM.

SLAM	Objects	Walls
SLAM++ [3]	✓	×
Object-SLAM [4]	✓	×
[5]	✓	✓
QuadricSLAM [6]	✓	×
[7], CubeSLAM [8]	✓	✓
[9], [16]	×	✓

1) Usage of rooms to reduce search space in data association and to find potential loop-closures.
2) Usage of rooms as a semantic sub mapping strategy to deal with SLAM scalability.
3) Generate a map with objects and floorplan-like structure of indoor environments.

The rest of the paper is organized as follows. Section II describes related works to the presented method. Section III and Section IV gives detail of the method followed by its evaluation in MIT Stata Center Dataset and TUM RGBD Dataset. Finally, Section V concludes this article and conveys some plans for future work.

II. RELATED WORKS

The problem of decomposing a single image into semantic entities, such as walls, ceilings, floors, and object instances is known as image understanding. The decomposition could be either in the form of 2D boxes in image space or 3D bounding boxes and structural layout in 3D space. Works such as [10]–[14], and [15] have shown to have great accuracy in inferring semantic entities from an image. However, these works do not run in real-time which render them unsuitable for mobile robotic applications.

Recently, works in Simultaneous Localization and Mapping (SLAM) have moved toward semantic SLAM. Instead of using geometric features, such as points or lines, semantic SLAM uses semantic entities, e.g. objects and walls, as elements of a map. Table 1 shows semantic entities used in recent works in semantic SLAM.

SLAM++ [3] uses a hand-held RGBD camera to scan an environment that is cluttered with repetitive domain-specific objects, e.g. chairs and tables. A Point-Pair Feature (PPF) based object detector then performs object detection and object localization while the motion of the camera is tracked using ICP-based camera-model tracking. All parameters, i.e. camera and object poses are estimated using graph optimization with additional ground-plane constraints for objects.

Object-SLAM [4] uses a standard salient feature matching method for object detection. They combine points and objects as landmarks and estimate SLAM parameters with standard graph optimization.

The recent popularity of deep learning-based object detection and recognition is adopted for SLAM in [5]–[8]. Hosseinzadeh *et al.* [5] use two kinds of landmark, i.e. objects which are represented by dual quadrics and planes which represent any plan regions in an environment, e.g. table surfaces or walls. To detect objects, they use YOLOv3 [17], and to detect planes they use PlaneNet [18]. To estimate SLAM parameters, they perform graph optimization with additional constraints i.e. plane-plane constraints from Manhattan assumptions, point-plane constraints, and supporting/tangency constraints. Similarly, [6] also uses YOLOv3 as an object detector and represents objects as dual quadrics.

Yang and Scherer [7] use a monocular camera to map objects and walls. First, they generate proposals of indoor structure layout by projecting detected ground-wall edges to 3D space. They use YOLOv3 to detect objects and project 2D bounding boxes from YOLOv3 to 3D space using the method in [8]. Generated objects and plane proposals are then supplied to CRF based inference engine to get the best proposal for single image understanding.

Shariati *et al.* [9] exploit the Manhattan structure of indoor environments to estimate their structural layouts and robot trajectories. They use a method known as entropy compass to get the axis-aligned planar fragments from RGBD data. Once salient axes are known, they label each pixel in the depth map by its orientation according to a small patch orientation of its surrounding. The pixels are then grouped to create layout segments, which are typically walls, floors, and ceilings. Combined with robot poses data from visual-inertial odometry, they build a factor graph with factors correspond to either pose-pose constraints or layout segment measurement constraints. The latter is simply one-dimensional distance measurement from robot to layout segments.

Shariati et. al. model each wall as a single entity in [9]. However, in their subsequent work [16], they also model a higher level of semantic from layout segments, i.e. rooms. This is by far the closest concept to that of the method presented in this article. However, [16] generates model of rooms after it gets the result from SLAM. Therefore, they do not exploit the concept of rooms for the benefit of SLAM.

The work that shows the benefit of room concept in SLAM is that of Salas *et al.* [12], although it is not categorized as semantic SLAM. In their work, Salas et. al. show at least two benefits of using the room concept in SLAM. Firstly, it splits the map into sub-maps which will ease the burden of computation. Secondly, it prevents the SLAM system to make incorrect inferences whenever severe occlusions occur, e.g. when a robot leaves a room or turning around a corner. The main difference with RoomSLAM is that in [12], rooms itself are not part of SLAM whereas, in RoomSLAM, rooms (at least in the form of walls) are part of the system and are used for example in data association.

III. RoomSLAM

This section starts with some mathematical notations and definitions used throughout the paper. After that, the method of RoomSLAM is briefly overviewed in Section III-B. Details of components of RoomSLAM are then described in Section III-C (object detector), Section III-D (wall detector), Section III-E (robot motion model), Section III-F (room detection/creation), Section III-G (data association),

Section III-H (loop-closure detection), and finally Section III-I (optimization).

A. NOTATIONS AND DEFINITIONS

The i-th robot pose, $\mathbf{r}_i^0$, relative to a global frame of reference is described by 3 parameters, i.e. its position (x_i^0, y_i^0) and its orientation (α_i^0) in a 2D space. The global frame of reference coincides with the initial robot pose, which is denoted $\mathbf{r}_0^0$.

$$\mathbf{r}_i^0 = (x_i^0, y_i^0, \alpha_i^0)$$

To simplify notations, 0 superscript is dropped and all variables are assumed to be measured with respect to the global frame of reference unless otherwise stated, e.g. $\mathbf{r}_i \equiv \mathbf{r}_i^0$.

In calculation, $\mathbf{r}_i$ is represented by an element of SE(2).

$$\mathbf{r}_i = \begin{bmatrix} \cos\alpha_i & -\sin\alpha_i & x_i \\ \sin\alpha_i & \cos\alpha_i & y_i \\ o & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ 0 & 1 \end{bmatrix}$$

with $\mathbf{R}_i$ is a 2D rotation matrix and $\mathbf{t}_i = (x_i\ y_i)^T$ is a 2D translation vector.

The i-th object pose is represented by its 2D position

$$\mathbf{o}_i = (o_{xi}, o_{yi})$$

Each object has an attribute c_i, which indicates its class of object, e.g. "chair" and "monitor".

The i-th wall is represented by two parameters, i.e., its perpendicular distance (ρ_i) to the origin and its orientation (θ_i) with respect to the horizontal axis of the global frame of reference.

$$\mathbf{w}_i = (\rho_i, \theta_i)$$

Fig. 1 shows a wall j with respect to a frame of reference that coincides with $\mathbf{r}_i$. Each wall does not contain information about edges. Therefore, each wall is an infinite line.

Each wall either has a type (t_j) of virtual or real. A virtual wall is a wall that is generated hypothetically to make a room. There is no measurement associated with this wall. Meanwhile, a real wall is a wall that has been measured. A virtual wall becomes real when a measurement is associated with it.

Measurement $\mathbf{z}$ of a robot pose, an object, and a wall are represented by the same parameters as those in state-space respectively.

$$\begin{aligned} \mathbf{z}_{r|j} &= (x_j, y_j, \alpha_j) \\ \mathbf{z}^i_{o|j} &= (o^i_{xj}, o^i_{yj}) \\ \mathbf{z}^i_{w|j} &= (\rho^i_j, \theta^i_j) \end{aligned}$$

For each object measurement $\mathbf{z}^i_{o|j}$, there is attached a class attribute, z_j.

As in its state-space counterpart, $\mathbf{z}_{r|j}$ is represented by an element of SE(2).

Covariance matrices for robots, objects, and walls are denoted by $\Sigma_{r|i}$, $\Sigma_{o|j}$, and $\Sigma_{w|k}$ respectively. Covariance matrices for robots, objects, and walls measurements are denoted by $\mathbf{Q}_{r|i}$, $\mathbf{Q}^i_{o|j}$, and $\mathbf{Q}^i_{w|k}$ respectively.

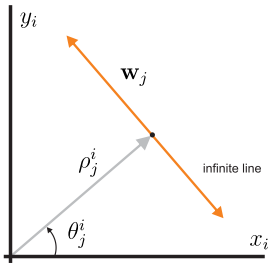

FIGURE 1. Wall parameters.

An i-th room, $\mathcal{R}_i$, is defined by a set of 4 walls that form a quadrilateral. A collection of all rooms is denoted by R.

$$\mathcal{R}_i = \{\mathbf{w}_j\}_{j=1}^4$$

B. SYSTEM OVERVIEW

Modern SLAM systems typically consist of two modules run in parallel, namely front-end and back-end. In RoomSLAM, the front-end is responsible for walls and objects detection, rooms detection/creation, and data association. Meanwhile, the back-end is responsible for state estimation through graph optimization. Fig. 2 shows a block diagram of RoomSLAM.

For each timestep i, RoomSLAM receives data from two synchronized sources. Firstly, it receives RGB and depth images from an RGBD sensor. Secondly, it receives a position and orientation of the robot $(\mathbf{z}_{r|i})$ from a wheel or visual odometry.

An RGB image is used by YOLOv3 [17] to detect objects in the surrounding of the current robot pose. A depth image is first converted to a point cloud and then used to infer the position of the objects in the 3D world. The same point cloud is also used to extract walls.

Initially, RoomSLAM creates a quadrilateral room based on current detected walls. If there is no wall detected, a rectangular room from four virtual walls is created instead. In subsequent moves, each time a wall measurement is associated with a virtual wall, the wall becomes real.

During exploration, a robot is detected to leave the current room when it crosses a real wall of the room. Whenever this happens, the algorithm will search for a possible revisitation of previous rooms or otherwise creates a new room. On the other hand, the robot is not considered to leave the current room when it crosses a virtual wall. Instead, the virtual wall will be pushed away from the robot to make the room bigger. Therefore, during the SLAM operation, a room is allowed to deform either because it is still formed by virtual walls or due to the result of a graph optimization in the back-end.

Room is a key part of RoomSLAM. During the front-end loop, RoomSLAM will search for associations of object and wall measurements with objects and walls in the current room. In the back-end, RoomSLAM also only optimizes subgraph associated with objects and walls in the current room.

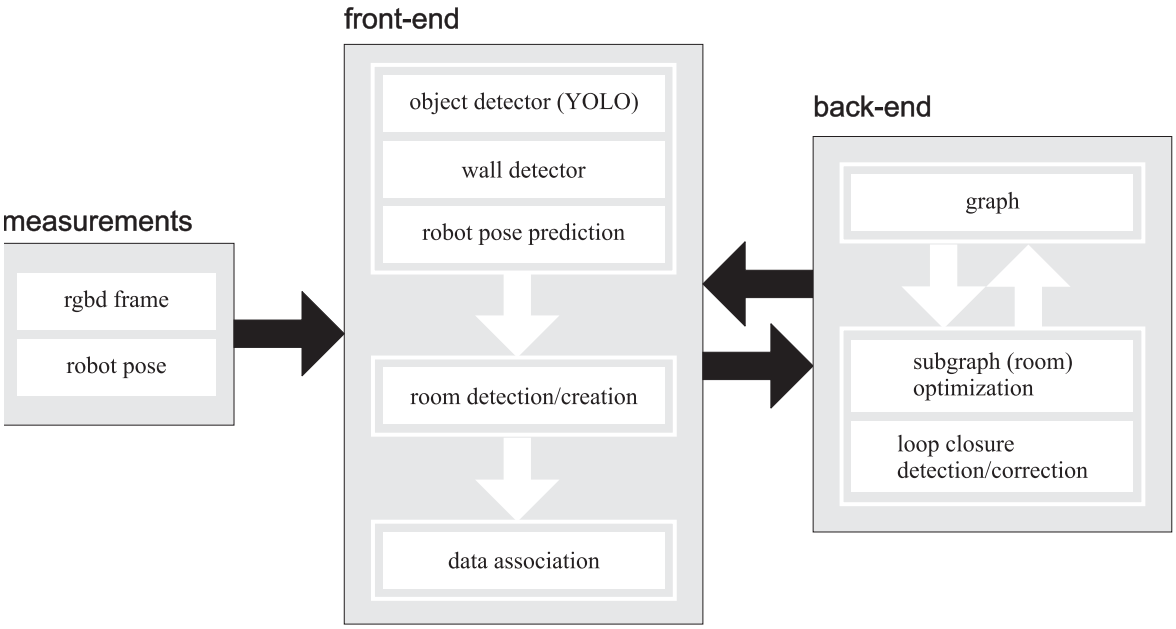

FIGURE 2. Block diagram of RoomSLAM.

TABLE 2. Classes of object detected by YOLOv3 in the MIT Stata Center dataset and the TUM RGBD dataset.

backpack	bench	bicycle	book
bottle	bowl	cellphone	chair
clock	cup	dining table	keyboard
laptop	microwave	pottedplane	refrigerator
remote	sink	sofa	sports ball
teddy bear	toilet	tvmonitor	vase

Whenever available, RoomSLAM also searches for room-to-room similarity for loop-closure detection. However, for the current implementation, it only applies to corridors.

C. OBJECT DETECTOR

RoomSLAM uses YOLOv3 [17] and its pre-trained model as an object detector. The model is trained to detect 80 classes of indoor and outdoor objects. The output from YOLOv3 is bounding boxes of objects in image space. Table 2 shows some objects that were detected in the MIT Stata Center dataset and the TUM RGBD dataset.

To get a 2D position of an object, the corresponding 2D box in image space is projected into a 3D space to form a cuboid. In Fig. 3, left top blue box in image space is projected into a point cloud space (lower blue box). The blue box in the point cloud space becomes the front face of a cuboid. A centroid is then calculated from the point cloud contained in the cuboid and the 2D position of the object is just a projection of the centroid to the xy-plane (floor). Measurement model for an object is shown in (1).

$$\mathbf{o}_j^i = \mathbf{g}(\mathbf{o}_j, \mathbf{r}_i) = \mathbf{R}_i^{-1}(\alpha_i) \times \mathbf{o}_j + \mathbf{t}_i^{-1}$$
$$\text{with } \mathbf{t}_i^{-1} = \begin{bmatrix} -x_i \cos\alpha_i - y_i \sin\alpha_i \\ x_i \sin\alpha_i - y_i \cos\alpha_i \end{bmatrix} \quad (1)$$

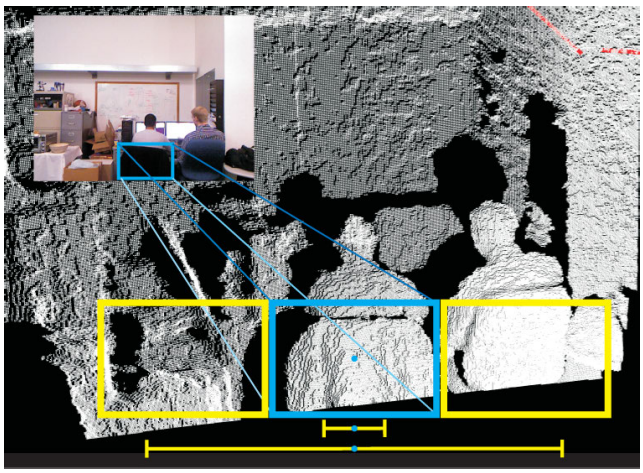

FIGURE 3. Two standard deviations for an object measurement. If the object is treated as a point (blue dot), its position uncertainty is likely to be in the area of the blue box (short yellow line in the lower part of the image represents standard deviation). Else, if the object is treated as a rigid body, its position uncertainty is likely to be outside of the blue box (yellow boxes are possible positions of the object).

Treating an object as a point needs careful consideration. On the one hand, as a point, position uncertainty is confined within the cuboid. This is because the variance of the point is calculated from the distribution of the point cloud within the cuboid. On the other hand, as a rigid body, position uncertainty is likely to span in the order of the object's dimensions. For example, in Fig. 3, yellow boxes are possible positions of the object. This is important, especially in data association. Therefore, RoomSLAM uses two covariance matrices for object measurements, i.e. the point covariance in graph optimization and the rigid body covariance in data association.

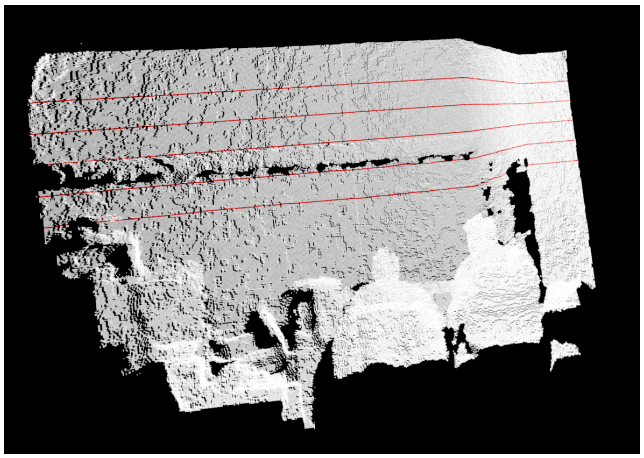

FIGURE 4. Red lines are lines fitted to the upper rows of the point cloud data.

D. WALL DETECTOR

RoomSLAM uses a simple algorithm to detect walls [19]. It samples several rows from an organized point cloud data and applies line-fitting (RANSAC) to each of them. The rows are chosen arbitrarily although the upper rows are preferable to avoid occlusion from objects (Fig. 4).

For each sample, RANSAC typically returns none to two lines (lines that are too short are discarded). Wall detector then groups lines from all samples according to parameter proximity. A group with more than one line is considered representing a wall. Parameters of the wall are calculated by averaging lines parameters in the group.

Measurement model for a wall is shown in (2). The equation relates $\mathbf{w}_j$ with $\mathbf{w}_j^i$. The derivation is in Appendix.

$$\mathbf{w}_j^i = \mathbf{h}(\mathbf{w}_j, \mathbf{r}_i) = \begin{bmatrix} \rho_j - x_i \cos\theta_j - y_i \sin\theta_j \\ \theta_j - \alpha_i \end{bmatrix} \tag{2}$$

E. ROBOT MOTION MODEL

Robot motion data come from a wheel or visual odometry in the form of the current robot position and heading with respect to the global frame of reference (first robot pose). For each two consecutive robot motion data, the motion model is calculated by a black-box motion model [20], as shown in (3).

$$\begin{aligned} \mathbf{r}_i &= \mathbf{f}(\mathbf{r}_{i-1}, \mathbf{u}_i) = \mathbf{r}_{i-1} \cdot \mathbf{u}_i \\ &= \mathbf{r}_{i-1} \cdot (\mathbf{z}_{r|i-1}^{-1} \cdot \mathbf{z}_{r|i}) \end{aligned} \tag{3}$$

where $(\cdot)$ is a matrix multiplication operator and $\mathbf{u}$ denotes an action that causes the robot to move from step $i-1$ to i.

F. ROOM DETECTION/CREATION

RoomSLAM starts with an empty map and the robot is in the origin of the global frame of reference. If the robot measures some walls, a room is created according to one of the measured wall (Fig. 5[a]). Otherwise, a rectangular room from 4 virtual walls is created.

A wall is principally an infinite line. However, as a part of a room, the wall will be seen as a line segment due to intersection with other walls.

As mentioned earlier in Section III-B, a robot is detected to leave a room when it crosses a real wall of the room (Fig. 5[d]). Algorithm `DetectRoom` in the following gives detail of the process.

```
DetectRoom (R_i, r_j, r_{j-1}, R):
path = f(r_{j-1}, r_j)
for all w_m ∈ R_i do
  if path intersect w_m then
    if t_m = virtual then
      ρ_m^{j-1} ← ρ_m^{j-1} + constant
      update R_i
      return R_i
    else
      for all R_n ∈ R do
        if r_j is inside R_n then
          return R_n
        end if
      end for
      R_ℓ = createRoom(r_j, w_m)
      return R_ℓ
    end if
  end if
end for
return R_i
```

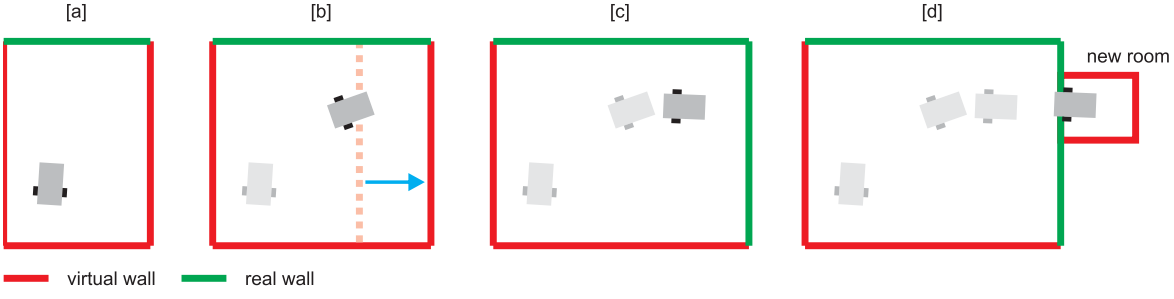

FIGURE 5. [a] A robot is in a room with only one real wall. [b] The robot crosses a virtual wall and the wall is pushed away to create a larger room. [c] The virtual wall in [b] is associated with a measurement and becomes real. [d] The robot crosses the real wall and leaves the room to create a new room or enter a previously visited room.

In line 1, a path is created from two consecutive robot poses. The path is then checked against all walls in the current room to find intersections. If the path intersects a virtual wall (line 4), then distance (ρ_m^{j-1}) to the wall is increased (line 6) by a constant value (Fig. 5[b]) and the algorithm returns the current room. On the other hand, if the path intersects a real wall, then the current robot pose is checked against all rooms (line 10). If the current robot pose is in one of the rooms, the algorithm returns that room. Otherwise, the algorithm creates a new room (line 15) and returns it. The following is the algorithm to create a new room.

1: `createRoom` $(\mathbf{r}_i, \mathbf{w}_j)$:
2: $\mathcal{R}_k \leftarrow \{\mathbf{w}_j\}$
3: **for** $\ell \leftarrow 1, 3$ **do**
4: create new virtual wall $\mathbf{w}_\ell$
5: $\rho_\ell^i \leftarrow \max(\text{constant}, \rho_j^i)$
6: $\mathbf{w}_\ell^i \leftarrow (\rho_\ell^i, \theta_j^i + \ell \times \pi/2)$
7: $\mathcal{R}_k \leftarrow \mathcal{R}_k \cup \mathbf{w}_\ell$
8: **end for**
9: **return** $\mathcal{R}_k$

A new room is initialized using a real wall ($\mathbf{w}_j$) and 3 virtual walls ($\mathbf{w}_\ell$ in line 4). The virtual walls are oriented such that the four walls create a rectangular room (line 6). The current robot pose should be in this newly created room and the distance of each virtual wall to the robot (ρ_ℓ^i) depends on the distance between the real wall and the robot (ρ_j^i). However, to prevent the algorithm creates a too-small room, ρ_ℓ^i is taken to be the maximum between ρ_j^i and a constant value (line 5).

G. DATA ASSOCIATION

Data association is a process of finding associations between current measurements and all previous measurements. RoomSLAM needs to do two data associations, i.e. objects associations and walls associations. For objects associations, two simple criteria are used, i.e. Squared Mahalanobis Distance (SMD) and object's class. For walls associations, due to unknown wall measurement covariance, Euclidean distance is used.

1) OBJECTS ASSOCIATIONS

For each object, $\mathbf{z}_{o|k}^i$ of class c_ℓ, that is measured from the current robot pose, RoomSLAM calculates its SMD, $(D_{o|jk}^i)^2$, to every object of the *same class*, $\mathbf{o}_j^i$, in the current room where $\mathbf{o}_j^i$ is given by (1).

$$(D_{o|jk}^i)^2 = (\mathbf{o}_j^i - \mathbf{z}_{o|k}^i)^T (\mathbf{S}_{jk}^i)^{-1} (\mathbf{o}_j^i - \mathbf{z}_{o|k}^i) \tag{4}$$

with $\mathbf{S}_{jk}^i$ is

$$\begin{aligned}\mathbf{S}_{jk}^i &= \frac{\partial \mathbf{o}_j^i}{\partial \mathbf{o}} \Sigma_{o|j} \frac{\partial \mathbf{o}_j^{i\,T}}{\partial \mathbf{o}} + \mathbf{Q}_{o|k}^i \\ &= \mathbf{R}_i^{-1} \Sigma_{o|j} (\mathbf{R}_i^{-1})^T + \mathbf{Q}_{o|k}^i \end{aligned} \tag{5}$$

All objects in the current room with SMDs lie within 95% confidence interval of chi-square distribution are candidates for associations. However, to avoid ambiguity whenever there is more than one object lie within the interval, the one with the smallest SMD is picked as the best candidate.

$$(D_{o|jk}^i)^2 < \chi_{1,0.05}^2$$

As mentioned earlier in Section III-C, $\mathbf{Q}_{o|k}^i$ in (5) is an object measurement covariance when the object is treated as a rigid body. The covariance is taken to be a diagonal matrix, i.e. no correlation between x and y positions, and the variances are set to be twice the dimensions of the object. The dimensions of the object itself are set to twice the standard deviation when the object is treated as a point.

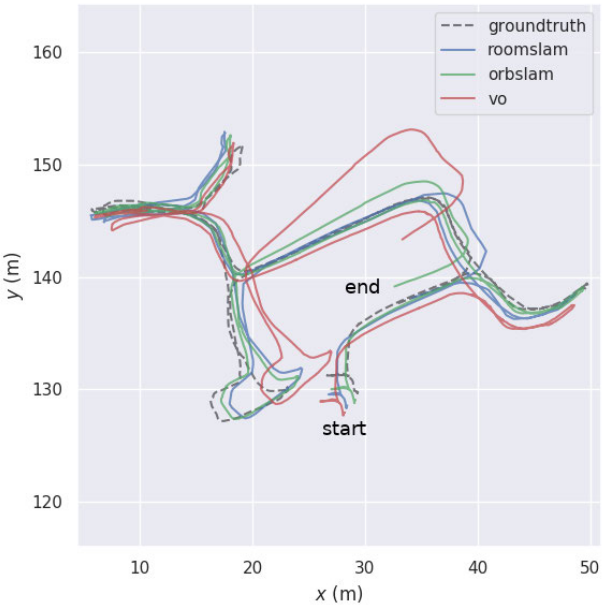

FIGURE 6. Estimated trajectories of sequence 2012-04-06-11-15-29.

2) WALLS ASSOCIATIONS

Euclidean distance is used to find an association between a wall measurement and a wall already on the map. However, because both the measurement and the wall are parameterized in the polar coordinate system, RoomSLAM treats each parameter separately. Mathematically, for each wall measurement $\mathbf{z}_{w|j}^i = (\mathbf{z}_{r|j}^i, \mathbf{z}_{\theta|j}^i)$, RoomSLAM calculates an Euclidean distance of each parameter to every wall, $\mathbf{w}_k^i = (r_k^i, \theta_k^i)$, in the current room.

$$D_{jk}^i = \begin{bmatrix} D_{r|jk}^i \\ D_{\theta|jk}^i \end{bmatrix} = \begin{bmatrix} |\mathbf{z}_{r|j}^i - r_k^i| \\ |\mathbf{z}_{\theta|j}^i - \theta_k^i| \end{bmatrix} \tag{6}$$

where $\mathbf{w}_k^i = (r_k^i, \theta_k^i)$ is from (2).

A measurement is associated with a wall if both $D_{r|jk}^i < \tau_r$ and $D_{\theta|jk}^i < \tau_\theta$ where τ_r and τ_θ are thresholds with values of 0.2 and 25° respectively.

H. LOOP-CLOSURE DETECTION

One of the benefits of the room model is the ability to exploit similarities in rooms to detect a loop-closure. However, in the

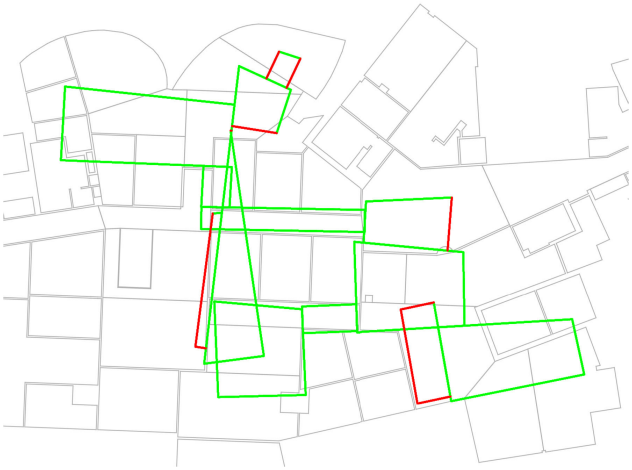

FIGURE 7. Floorplan-like map estimated with RoomSLAM for sequence 2012-04-06-11-15-29. Red lines are virtual walls and green lines are real walls. Global orientation is adjusted manually. The room appears triangular in the center of the figure is a quadrilateral. The missing side is a short virtual wall on the upper side of the room.

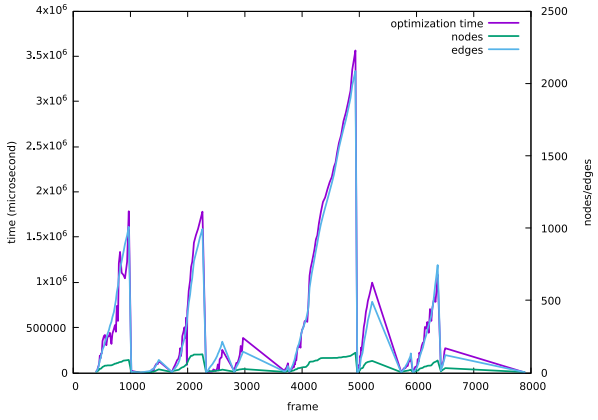

FIGURE 8. The number of nodes, edges, and optimization time during robot exploration in sequence 2012-04-06-11-15-29. The algorithm maintains a graph for each room which makes the back-end never optimize large graphs.

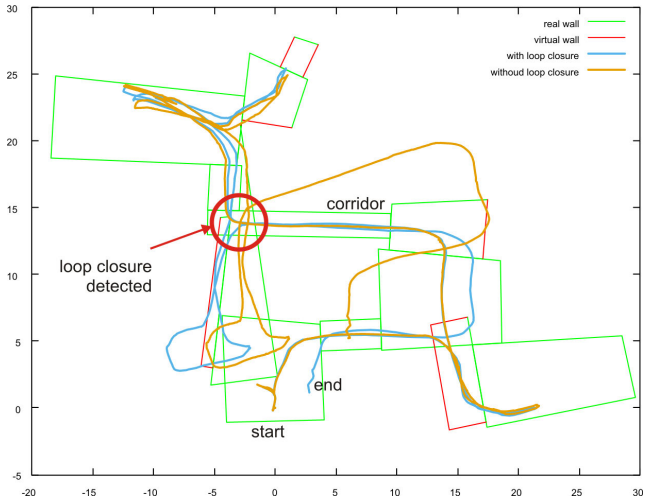

FIGURE 9. Loop closure in corridor in sequence 2012-04-06-11-15-29 (frame of reference is shifted and rotated).

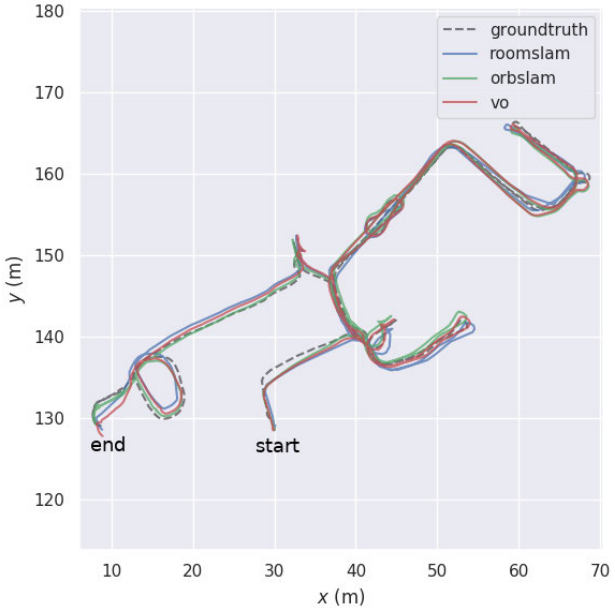

FIGURE 10. Estimated trajectories of sequence 2012-04-06-11-28-12.

current implementation, RoomSLAM only exploits similarities in corridors. A room is considered a corridor if two of their walls are parallel or near parallel with a distance less than 2.5 m. Two corridors are the same if their Euclidean distance and the difference in their orientations are below certain thresholds.

I. OPTIMIZATION

Optimization runs in the background, parallel to the front-end module in RoomSLAM. Every robot pose, object pose, and wall are estimated in a graph optimization process. There are two kinds of optimization, i.e. room optimization and loop-closure optimization. In principle, both optimizations are having the same objective functions. The difference is that room optimization is applied to nodes within a room whereas loop-closure optimization is applied to nodes across multiple rooms. Equation (7) shows the objective function for the optimization process in RoomSLAM.

$$\{X^*\}_{\mathcal{R}} = \underset{\{X\}_{\mathcal{R}}}{\operatorname{argmin}} \sum_{i,j,k} (\mathbf{e}^2_{r|i} + (\mathbf{e}^i_{o|j})^2 + (\mathbf{e}^i_{w|k})^2) \tag{7}$$

where $\{X\}_{\mathcal{R}}$ means all $\mathbf{r}$, $\mathbf{o}$, and $\mathbf{w}$ in room $\mathcal{R}$ and

$$\begin{aligned}
\mathbf{e}^2_{r|i} &= (\mathbf{r}_i - \mathbf{z}_{r|i})^T \mathbf{Q}^{-1}_{r|i} (\mathbf{r}_i - \mathbf{z}_{r|i}) \\
(\mathbf{e}^i_{o|j})^2 &= (\mathbf{o}^i_j - \mathbf{z}^i_{o|j})^T (\mathbf{Q}^i_{o|j})^{-1} (\mathbf{o}^i_j - \mathbf{z}^i_{o|j}) \\
(\mathbf{e}^i_{w|k})^2 &= (\mathbf{w}^i_k - \mathbf{z}^i_{w|k})^T (\mathbf{Q}^i_{w|k})^{-1} (\mathbf{w}^i_k - \mathbf{z}^i_{w|k})
\end{aligned}$$

The optimization module is implemented using the g2o framework [21].

IV. EXPERIMENTAL EVALUATIONS

RoomSLAM is evaluated with two datasets, i.e TUM RGBD dataset [22] and MIT Stata Center dataset [23]. TUM dataset

TABLE 3. Root Mean Squared Error (RMSE) in trajectories estimation of sequences from MIT and TUM dataset by RoomSLAM, ORBSLAM, and RGBDSLAM. All figures are in meter.

	RoomSLAM	ORBSLAM	RGBDSLAM
MIT 2012-04-06-11-15-29	1.40	**1.06**	6.76
MIT 2012-04-06-11-28-12	**0.92**	1.3	15.86
MIT 2012-01-25-12-33-29	**1.35**	1.47	13.86
fr2/pioneer_360	**0.1**	0.85	0.25
fr2/pioneer_slam	**0.17**	0.77	0.36
fr2/pioneer_slam2	**0.36**	0.48	1.0
fr2/pioneer_slam3	**0.19**	0.64	0.32

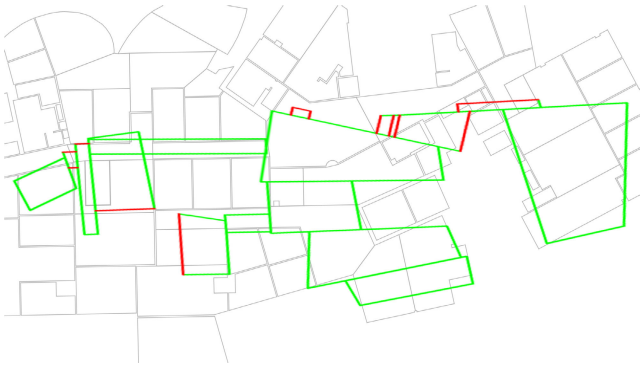

FIGURE 11. Floorplan-like map of sequence 2012-04-06-11-28-12.

FIGURE 13. Floorplan-like map of sequence 2012-01-25-12-33-29.

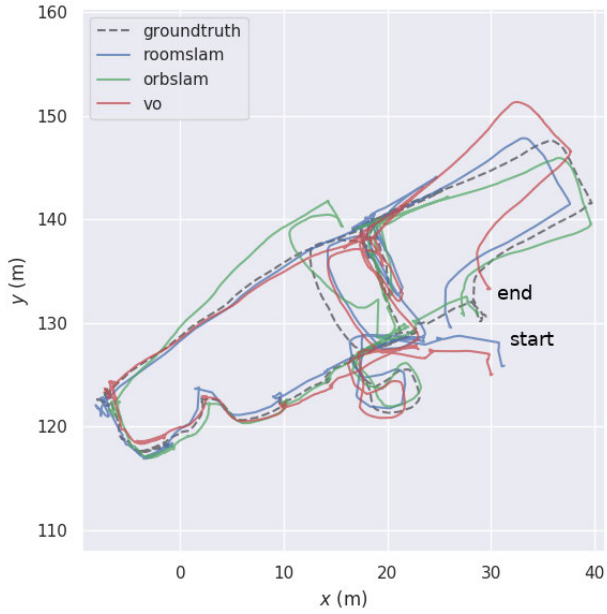

FIGURE 12. Estimated trajectories of sequence 2012-01-25-12-33-29.

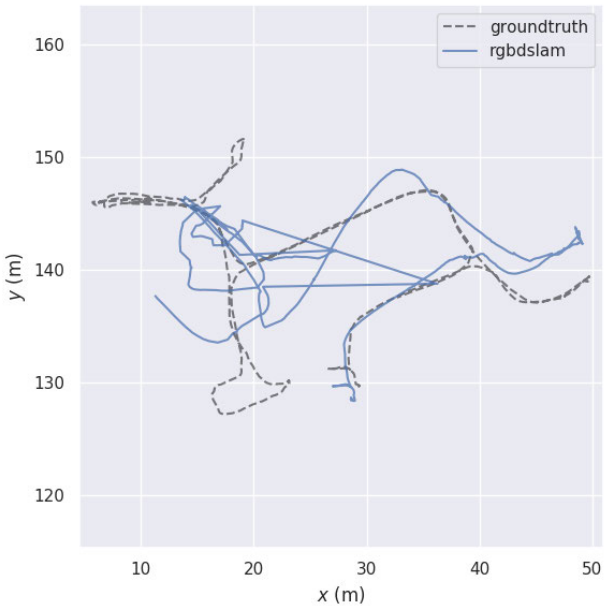

FIGURE 14. RGBDSLAM performed well at the beginning of the trajectory but then failed in the end.

is chosen because it is widely used in SLAM community as a means of benchmarking and MIT dataset is chosen because it is a challenging indoor dataset for the following reasons.

1) It has an irregular structural layout;
2) It has large spaces that make an RGBD sensor frequently failed to reach space boundaries (walls);
3) It has walls with large glass windows; and
4) It is a highly cluttered environment.

In TUM dataset, RoomSLAM uses wheel odometry as a source for robot motion data whereas, in MIT dataset, RoomSLAM uses visual odometry from ORBSLAM.

Results of RoomSLAM evaluation in mentioned datasets are compared to those of ORBSLAM [24] and RGBDSLAM [25] which are considered state-of-the-art. Trajectory comparisons between methods and groundtruth are done with the help of the Umeyama alignment method [26] whereas map comparisons are done by manually overlaying the resulted

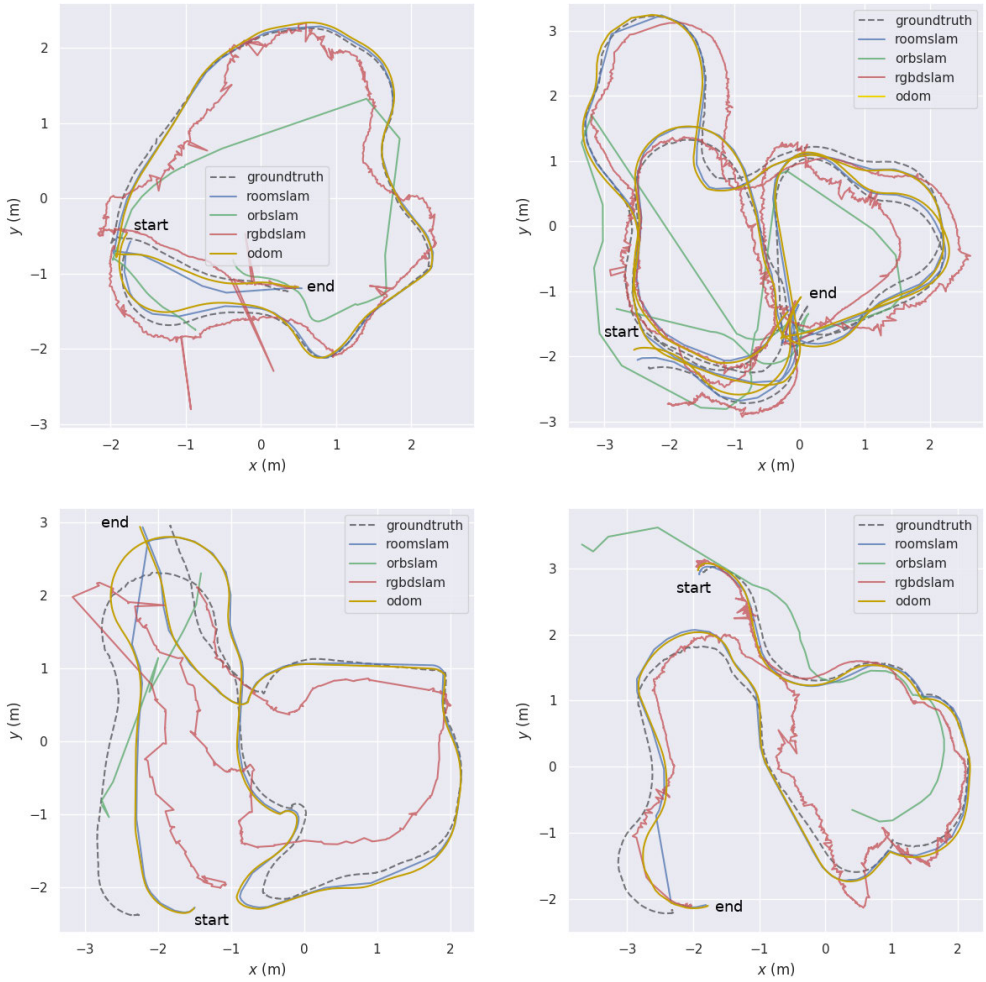

FIGURE 15. Estimated trajectories of TUM SLAM dataset. Left-top: fr2/pioneer_360, right-top: fr2/pioneer_slam, left-bottom: fr2/pioneer_slam2, right-bottom: fr2/pioneer_slam3.

map with groundtruth (only for MIT dataset). Table 3 shows the evaluation results.

A. MIT STATA CENTER DATASET

1) SEQUENCE 2012-04-06-11-15-29

This sequence covers an area of 50 m × 25 m where the robot travels a 230 m long trajectory in 11 minutes. In this sequence, the robot makes an important close loop that occurs in a corridor. RoomSLAM exploits the fact that corridor detection is reliable and uses it to make a significant trajectory correction.

The estimated trajectory compared to those of ORBSLAM, visual odometry, and groundtruth is shown in Fig. 6. The estimated floorplan-like map is shown in Fig. 7.

The map is not accurately model the structural layout of the environment. There are misaligned walls and overlapped rooms. Although this needs improvement, RoomSLAM still makes benefit out of the model.

Rooms are space-partitioning components. RoomSLAM uses it to limit the size of the graph in the back-end so it never grows unbounded. Fig. 8 shows the number of nodes, edges, and optimization time during robot exploration in sequence

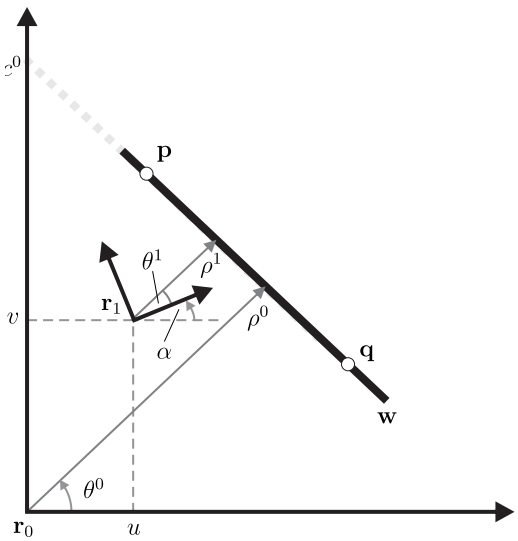

FIGURE 16. Derivation of wall measurement model.

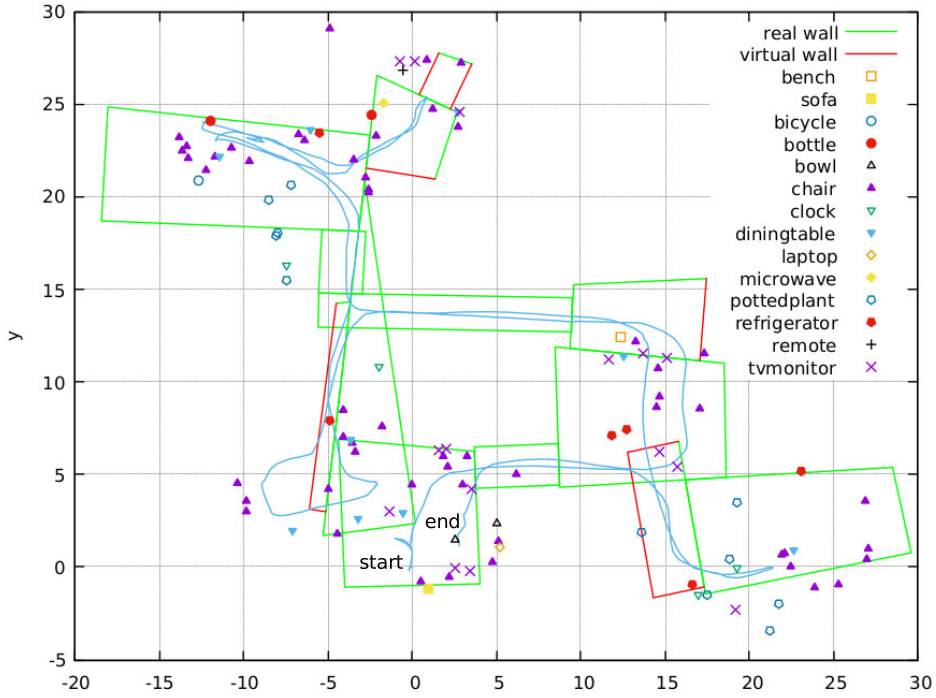

FIGURE 17. Result of RoomSLAM for sequence 2012-04-06-11-15-29.

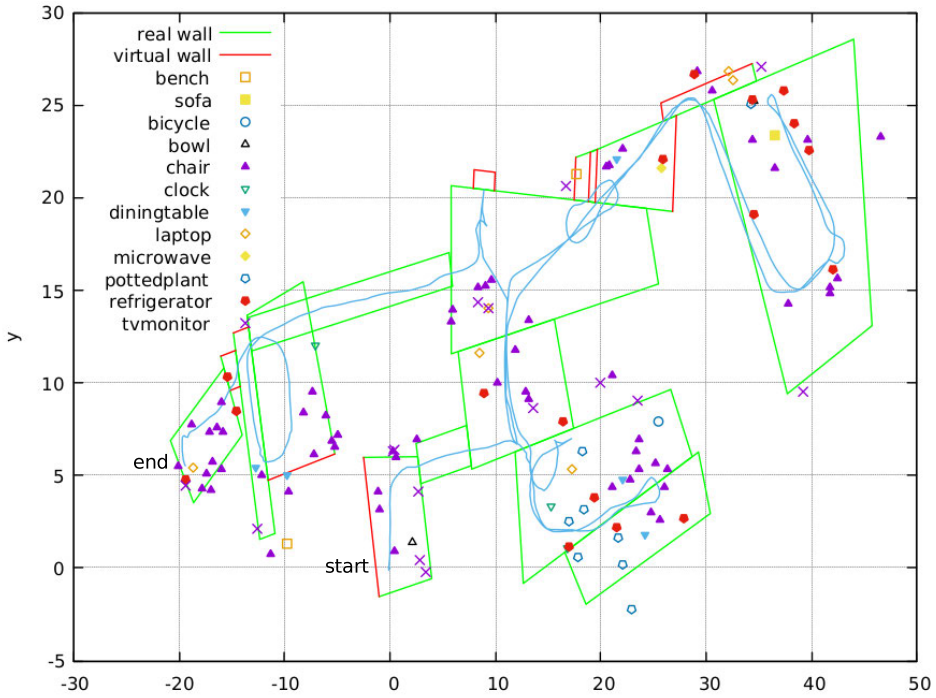

FIGURE 18. Result of RoomSLAM for sequence 2012-04-06-11-28-12.

2012-04-06-11-15-29. As shown in the figure, the number of nodes and edges always jumps back to a low number which happens when the robot leaves a room.

Another benefit of rooms is that RoomSLAM can search for room-to-room matches to find a close loop. Fig. 9 shows a loop-closure that occurs in a corridor which is relatively easy and reliable to detect. A corridor is detected when the robot measures two walls oriented to the left and right of the robot. Typically, the two walls are narrowly separated. This makes those walls are in a good area of sensing of RGBD sensors. RoomSLAM detects that the robot revisits the corridor when its pose is within or near the area of the corridor.

The complete estimation of sequence 2012-04-06-11-15-29, i.e. trajectory, walls (rooms), and objects are shown in Fig. 17.

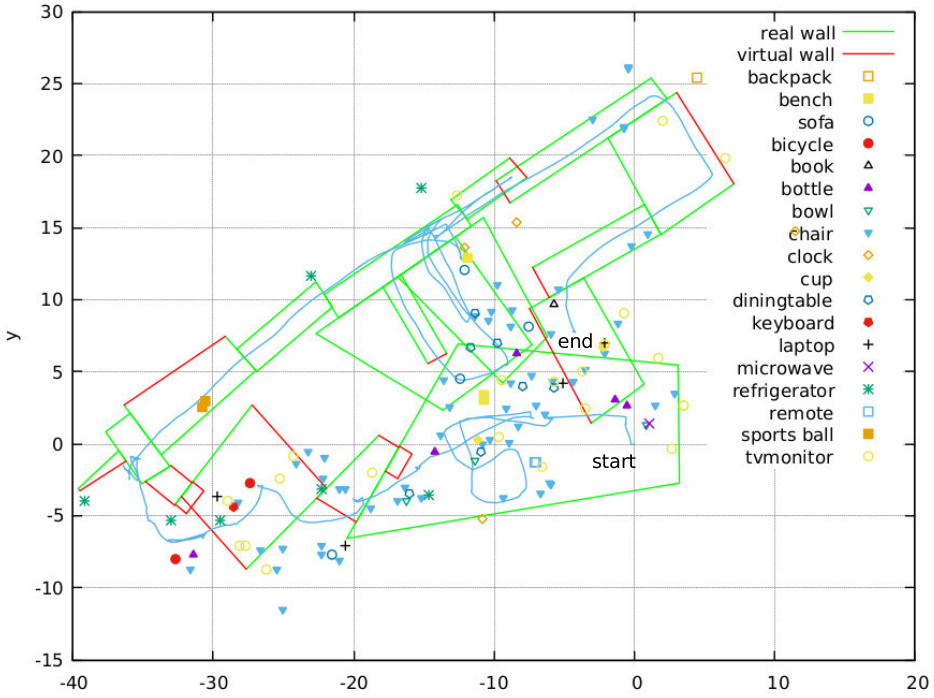

FIGURE 19. Result of RoomSLAM for sequence 2012-01-25-12-33-29.

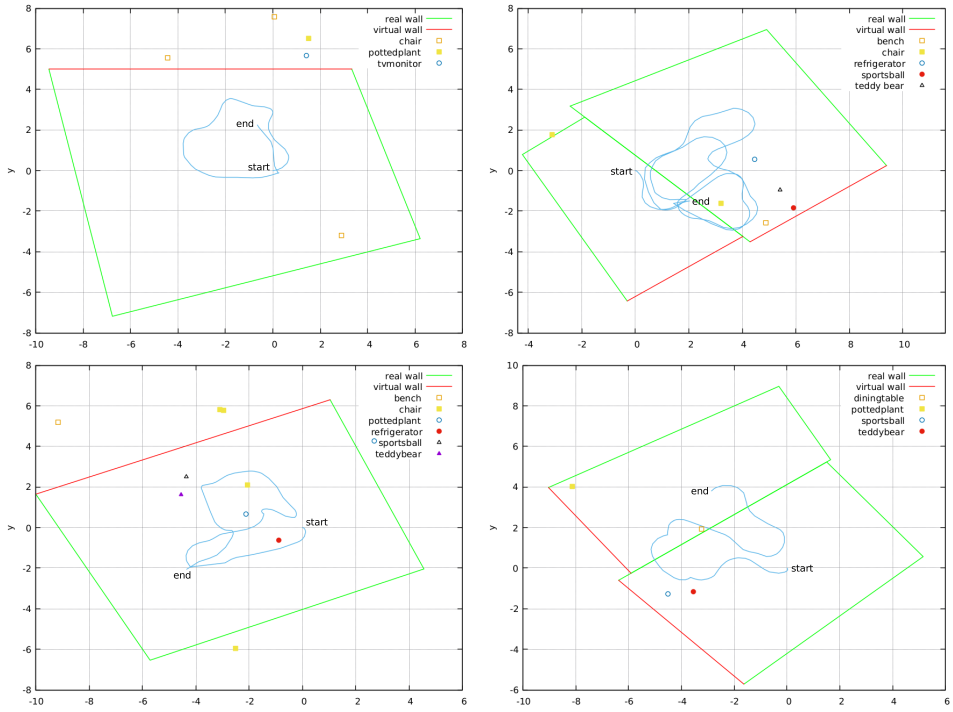

FIGURE 20. Results of RoomSLAM for sequences in TUM dataset. Clockwise from top-left: fr2/pioneer_360, fr2/pioneer_slam, fr2/pioneer_slam3, and fr2/pioneer_slam2. RoomSLAM falsely approximates structural layout in sequence fr2/pioneer_slam and fr2/pioneer_slam3 into two rooms. This is because there is a panelboard in the middle of the room which is detected as a wall.

2) SEQUENCE 2012-04-06-11-28-12

In this sequence, the robot starts in the same place as that of the previous sequence. However, the robot explores further to the right side area of the building which has a different main orientation from that of the center area. This adds complexity to room detection and creation algorithm.

The sequence covers an area of 70 m $\times$ 30 m and the robot runs along a 620 m trajectory in 23 minutes. However,

RoomSLAM was unable to finish the trajectory because of visual odometry module in ORBSLAM failed to track ORB features once during exploration and unsucceeded to recover from it. This happened when the robot was close to a white wall where the detector failed to detect ORB features. The algorithm itself failed to recover because the close loop detection module, which was responsible for relocalization was turned off.

The estimated trajectory of this sequence is shown in Fig. 10 and the floorplan-like map is shown in Fig. 11. The complete estimation is shown in Fig. 18.

3) SEQUENCE 2012-01-25-12-33-29

This is a very challenging sequence because the robot has many pure rotational moves. The sequence covers an area of 50 m × 35 m and the robot travels along a 270 m path in 13 minutes. The estimated trajectory is shown in Fig. 12 and the floorplan-like map is shown in Fig. 13. The complete estimation is in Fig. 19.

4) RGBDSLAM IN MIT DATASET

RGBDSLAM [25] is one of the state-of-the-art in SLAM. However, it did not perform well in MIT dataset. As shown in Fig. 14, RGBDSLAM showed a good estimation at the beginning but failed in the rest of the trajectory.

B. TUM BENCHMARK DATASET

RoomSLAM uses four sequences from RGBDSLAM Dataset and Benchmark from the Technical University of Munich (TUM) [22] as another dataset for evaluation. Those sequences are from robot SLAM category, i.e., fr2/pioneer_360, fr2/pioneer_slam, fr2/pioneer_slam2, and fr2/pioneer_slam3. The quantitative results are shown in Table 3 and trajectories comparison are shown in Fig. 15.

As shown in the figures, ORBSLAM generally has poorer performances compared to those of RoomSLAM and RGBDSLAM. This is because ORBSLAM frequently lost track of points whenever the robot crossed an uneven floor and the camera bumped quickly. Mostly, ORBSLAM could recover from it but it still affected the overall performance.

V. CONCLUSION

This article presents a method for Simultaneous Localization and Mapping (SLAM) in indoor environments. The proposed method uses objects and walls as elements of the environment model. The method also combined walls into space partitioning entities, i.e. rooms, to generate a floorplan-like map model. Such a model gives benefit to SLAM as shown in the evaluation with MIT Stata Center dataset and TUM RGBD dataset. RoomSLAM made a comparable performance to that of ORBSLAM and performed better than RGBDSLAM. RoomSLAM also offers a rich model of environments which is absent in ORBSLAM and RGBDSLAM.

In the future, usage of a more accurate wall detector, e.g. deep learning-based detector is considered. A more accurate projection from 2D bounding boxes to 3D is also considered. The assumption that a room has to be quadrilateral will be dropped and a boolean operator for rooms will be utilized to avoid overlapping rooms.

APPENDIX
DERIVATION OF WALL MEASUREMENT MODEL

Consider two points, $\mathbf{p} = (p_x, p_y)$ and $\mathbf{q} = (q_x, q_y)$, that define a line $\mathbf{w}$. Relative to the global frame of reference $\mathbf{r}_0$ and robot pose $\mathbf{r}_1$, the points are related by a homogeneous transformation matrix $\mathbf{T}$.

$$\mathbf{p}^0 = \mathbf{T}_1^0 \mathbf{p}^1 \qquad \mathbf{q}^0 = \mathbf{T}_1^0 \mathbf{q}^1 \tag{8}$$

with

$$\mathbf{T}_1^0 = \begin{bmatrix} \cos\alpha & -\sin\alpha & u \\ \sin\alpha & \cos\alpha & v \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

with (u, v) is position of $\mathbf{r}_1$.

With respect to the global and local frame of reference respectively, the line $\mathbf{w}$ has equation

$$\begin{aligned} y^0 &= m^0 x^0 + c^0 \\ y^1 &= m^1 x^1 + c^1 \end{aligned} \tag{10}$$

The gradient of the line $\mathbf{w}$ could be derived from points $\mathbf{p}$ and $\mathbf{q}$

$$m^0 = \frac{q_y^0 - p_y^0}{q_x^0 - p_x^0} \qquad m^1 = \frac{q_y^1 - p_y^1}{q_x^1 - p_x^1} \tag{11}$$

also with the intercepts

$$c^0 = p_y^0 - m^0 p_x^0 \qquad c^1 = q_y^1 - m^1 q_x^1 \tag{12}$$

The relation between m^0 and m^1, c^0 and c^1 could be derived by plugging in (8) into (11),

$$m^1 = \frac{-\sin\alpha + m^0 \cos\alpha}{\cos\alpha + m^0 \sin\alpha} \tag{13}$$

$$c^1 = \frac{c^0 - v + m^0 u}{\cos\alpha + m^0 \sin\alpha} \tag{14}$$

Fig. 16 shows that

$$\rho^0 = |c^0| \sin\theta^0 \tag{15}$$

This is a general equation relating ρ to c. Therefore, a similar equation is also applied in the local frame of reference.

$$\rho^1 = |c^1| \sin\theta^1 \tag{16}$$

The orientation of line $\mathbf{w}$ with respect to the global frame of reference is

$$\begin{aligned} \theta^0 &= \arctan(-1/m^0) \\ \Rightarrow m^0 &= -\frac{\cos\theta^0}{\sin\theta^0} \end{aligned} \tag{17}$$

Plugging in (17) to (14)

$$c^1 = \frac{\rho^0 - v\sin\theta^0 - u\cos\theta^0}{\sin(\theta^0 - \alpha)} \tag{18}$$

Finally, plug in (18) to (16), and with the fact that $\theta^0 = \theta^1 + \alpha$

$$\rho^1 = \rho^0 - v \sin\theta^0 - u \cos\theta^0$$

■

REFERENCES

[1] R. A. Newcombe, A. Fitzgibbon, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, and S. Hodges, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th IEEE Int. Symp. Mixed Augmented Reality*, Oct. 2011, pp. 127–136.

[2] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision—ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 834–849.

[3] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 1352–1359.

[4] D. Gálvez-López, M. Salas, J. D. Tardós, and J. M. M. Montiel, "Real-time monocular object SLAM," *Robot. Auto. Syst.*, vol. 75, pp. 435–449, Jan. 2016.

[5] M. Hosseinzadeh, K. Li, Y. Latif, and I. Reid, "Real-time monocular object-model aware sparse SLAM," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 7123–7129.

[6] L. Nicholson, M. Milford, and N. Sunderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM," *IEEE Robot. Autom. Lett.*, vol. 4, no. 1, pp. 1–8, Jan. 2019.

[7] S. Yang and S. Scherer, "Monocular object and plane SLAM in structured environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3145–3152, Oct. 2019.

[8] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[9] A. Shariati, B. Pfrommer, and C. J. Taylor, "Simultaneous localization and layout model selection in manhattan worlds," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 950–957, Apr. 2019.

[10] S. Dasgupta, K. Fang, K. Chen, and S. Savarese, "DeLay: Robust spatial layout estimation for cluttered indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 616–624.

[11] Y. Ren, S. Li, C. Chen, and C.-C. J. Kuo, "A coarse-to-fine indoor layout estimation (CFILE) method," in *Computer Vision—ACCV*, S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham, Switzerland: Springer, 2017, pp. 36–51.

[12] M. Salas, W. Hussain, A. Concha, L. Montano, J. Civera, and J. M. M. Montiel, "Layout aware visual tracking and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 149–156.

[13] A. Geiger and C. Wang, "Joint 3D object and layout inference from a single RGB-D image," in *Pattern Recognition*, J. Gall, P. Gehler, and B. Leibe, Eds. Cham, Switzerland: Springer, 2015, pp. 183–195.

[14] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun, "Box in the box: Joint 3D layout and object reasoning from single images," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 353–360.

[15] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Efficient structured prediction for 3D indoor scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2815–2822.

[16] A. Shariati, B. Pfrommer, and C. J. Taylor, "Predictive and semantic layout estimation for robotic applications in manhattan worlds," 2018, *arXiv:1811.07442*. [Online]. Available: http://arxiv.org/abs/1811.07442

[17] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[18] C. Liu, J. Yang, D. Ceylan, E. Yumer, and Y. Furukawa, "PlaneNet: Piece-wise planar reconstruction from a single RGB image," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2579–2588.

[19] B. R. Trilaksono and W. Adiprawita, "Mapping walls of indoor environment using moving RGB-D sensor," in *Proc. 6th Int. Conf. Inf. Commun. Technol. (ICoICT)*, May 2018, pp. 337–342.

[20] J.-A. Fernandez-Madrigal and J. L. B. Claraco, *Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods*. Hershey, PA, USA: IGI Global, 2012.

[21] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 3607–3613.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 573–580.

[23] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The MIT stata center dataset," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1695–1699, Dec. 2013.

[24] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[25] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Sep. 2014.

[26] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 4, pp. 376–380, Apr. 1991.

ISMAIL RUSLI received the bachelor's degree in physics and the master's degree in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, where he is currently pursuing the Ph.D. degree in electrical engineering.

His research interests include robotics, computer vision, operating systems, and artificial intelligence.

BAMBANG R. TRILAKSONO (Member, IEEE) received the degree in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, and the master's and Ph.D. degrees in electrical engineering from Waseda University, Tokyo, Japan.

He is currently a Professor with the Control and Computer System Research Group, School of Electrical Engineering and Informatics, ITB. His research interests include control systems, robotics, and artificial intelligence.

WIDYAWARDANA ADIPRAWITA (Member, IEEE) received the degree (Hons.) in electrical engineering, the master's degree in informatics engineering, and the Ph.D. degree (Hons.) in electrical engineering from the Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 1997, 2000, and 2011, respectively.

He is currently an Assistant Professor with the School of Electrical Engineering and Informatics, ITB. He has written more than 50 articles published in international publications. His research interests include embedded systems, robotics, computer vision, and artificial intelligence.