# Localization in a Semantic Map via Bounding Box Information and Feature Points

Sarthak Pathak[1], Irem Uygur[1], Lin Shize[2], Renato Miyagusuku[3], Alessandro Moro[1], Atsushi Yamashita[1], and Hajime Asama[1]

***Abstract*— Mobile service robots often operate in human environments such as corridors, offices, classrooms, homes, etc. In order to function properly, they need to be aware of their 6 Degree of Freedom (6 DoF) location. In addition, it is important that they possess semantic information i.e. knowledge of the types and positions of objects around them. In this method, we propose a method which obtains all of the above information directly. This method operates by using a camera as a "semantic sensor". The robot obtains the direction of objects such as doors, windows, tables, etc. around itself in 2D camera images by detecting bounding boxes. It then uses these object locations to localize itself within a floor map of the environment, which is typically available for most indoor environments.**

**However, bounding box information is highly unstable due to the various changes in lighting, pose, size, etc. Hence, we also semantically tag feature points on detected objects and use them in our Monte-Carlo based localization framework. This increases the robustness and accuracy of our approach, as is demonstrated by experiments.**

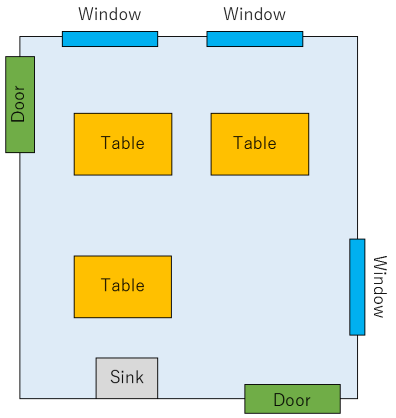


Fig. 1. Example of a semantic map i.e. a floor plan. This is a floor plan of a typical indoor environment. In our work, we try to localize inside of such a map based on camera-based object detection.

## I. Introduction

Due to the advances in sensor technology and deep learning, it is now possible to make robots that understand context and perform intelligent actions in homes, offices, and other indoor environments. However, there are two basic requirements for such systems which remain unsatisfactory. An intelligent mobile robot needs to be aware of 1. its location in the environment 2. its location with respect to objects in its surroundings, referred to as "semantic" information. Together, we refer to these as "semantic localization", which is aimed for in this research. We make use of an existing environmental map consisting of a floor plan with various objects marked on it. Such a floor plan, shown in Figure 1, with the object layout is commonly available for most indoor environments. Semantic localization is absolutely crucial in order to enable the robot to navigate its surroundings and interact with various objects in its surroundings. A simple use-case scenario for such a system could be a robot needs to navigate towards and open a door to fetch an object in another room.

In this research, we make use of semantic information directly in a map-based localization framework to provide semantic localization. To obtain semantic information, we make use of a 360 degree camera which can see in all directions. Convolutional Neural Networks (CNN) are used to extract semantic information from 360 degree images in the form of bounding boxes. The frame-to-frame location of the camera inside an environmental map is tracked using bounding box information, aided by an inertial measurement unit (IMU). Since bounding box information is jittery and unstable, we combine it with frame-to-frame feature point information in order to provide local anchoring and smoothing. This results in lower localization error.

The rest of this paper is divided as follows. The next section explains previous research on semantic localization. Next, we provide a brief overview of the proposed approach, and the reasons for adopting it. The section after that goes explains, in detail, the proposed method. Finally, experiments are conducted to evaluate its performance and the paper is concluded.

## II. Previous Research

The straightforward approach to achieve semantic localization is to use Radio Frequency ID (RFID) tags and other beacons to tag various objects [1]. However, these methods require costly prior installation and are difficult/costly to maintain over time. Moreover, it is tedious to perform this process for every new environment. Using wireless networks, as done in [2] requires no such installation and maintenance, as wireless networks are available inside most buildings. However, the accuracy is low, and this requires prior access of the environment for learning wireless signal strength patterns, as it is a fingerprinting technique.

Another straightforward method is to divide semantic localization into its two respective steps. Semantic information

The author affiliations are as follows:
[1]The University of Tokyo
[2]Tsinghua University
[3]Utsunomiya University

could be obtained with the help of a camera using Convolutional Neural Networks (CNN) [3], [4]. These approaches typically provide a 2D bounding box for each object detected inside an image. By moving the camera, these objects could be triangulated to 3D locations around the robot. Finally, these objects could be overlaid on an existing environment map along with the robot location, providing semantic localization. However, this approach involves multiple steps of 3D triangulation of objects and map construction. Each of these steps involve errors and any of them failing would create issues in the final result. Moreover, the object map also needs to be registered to the pre-existing environmental map, which is not a trivial problem. It is far more prudent to use the existing environmental map and match the sensor observations directly to the map for localization.

Other methods try to extrapolate maps to 3D and match them with 3D sensors such as RGB-D cameras [5] [6]. This is typically done in accordance to the Manhattan World Assumption [7]. However, this involves unnecessary use of 3D point clouds and raises the computational complexity. Further, objects not fitting the Manhattan World Assumption can cause issues. Another set of methods tries to use features that are easily observable by the camera. [8] tried to use 3D line features and matched them with 2D lines observable from a 360 degree camera. However, it is difficult to achieve semantic localization with this approach, and it is further difficult to construct these 3D line maps.

Instead, it is advantageous to directly locate objects within sensor measurements and use them as landmarks to localize on the environmental map. [9] and [10] did so by using distance and bearing measurements. However, this requires the use of a depth sensor, which have a low field of view and high power requirements. We find that using a 360 degree camera, with a wide field of view, is crucial for accuracy as it allows for the maximum number of visible objects. It is not so trivial to obtain distance measurements from a monocular camera.

Unlike them, [11] and [12] used the bearing angle (i.e. the 2D object location inside camera images) of uniquely detected objects to localize around them. [13] also attempted semantic localization by directly using the object type and bearing angle as sensor measurements. These approaches used objects detected inside images as the anchor to calculate the bearing angle. However, object detection (especially with bounding boxes) is often unstable. Detections are often missed and the bounding boxes usually shift from frame-to-frame due to changes in illumination, viewpoint, distance, etc. In the proposed approach, we aim to solve this problem by combining detected objects with feature points to stabilize object detections.

## III. Approach

As concluded in the previous sections, it is essential for robots to self-localize and be aware of objects around them, and the best way to achieve this is by using the objects themselves, detected inside camera frames, as anchors. Object detection inside image frames is easily achieved by

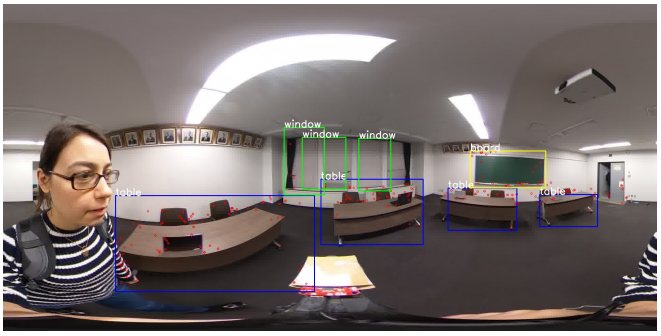


Fig. 2. Example showing descriptive bounding box information and non-descriptive feature point information.

Convolutional Neural Networks (CNN) such as YOLO (You Only Look Once) [4], [3], which detects several objects in the form of bounding boxes. This is the most lightweight approach for doing so, and runs in real-time on common processors.

The centers of bounding boxes can be taken as anchors to find their bearing angles from the camera. However, bounding boxes depend on various factors such as object viewpoint, illumination, and object size. As the robot moves through the environment, these factors change, causing bounding box centers to drift and jitter. This can lead to a drop in the accuracy of semantic localization approaches which utilize these.

On the other hand, feature points such as A-KAZE [14], are very stable against changes in viewpoint, illumination, and size. However, these are not descriptive enough to be used for semantic localization.

Therefore, we propose a combination of the two in order to provide stable inputs for semantic localization. Along with descriptive bounding box information, we detect non-descriptive feature points inside bounding boxes and tag them with object labels.

An example of this is shown below in Fig. 2.

While bounding boxes provide larger context and semantic information, they are unstable. Meanwhile, feature points can be tracked accurately, providing stable local anchors during frame-to-frame camera motion. We use feature points to calculate object velocities in image frames and combine them with bounding box detections in a Kalman filter to update the object tracking. As will be shown in experiments towards the end of this paper, this results in lesser errors and more stable detection.

## IV. Proposed Method

This section explains the approach, in detail. We begin with a basic overview of what our method entails.

### A. Overview

As mentioned, our method makes use of semantic information directly to localize a robot inside a semantic map i.e. a floor plan. We make use of a 360 degree camera to extract semantic information about the environment to

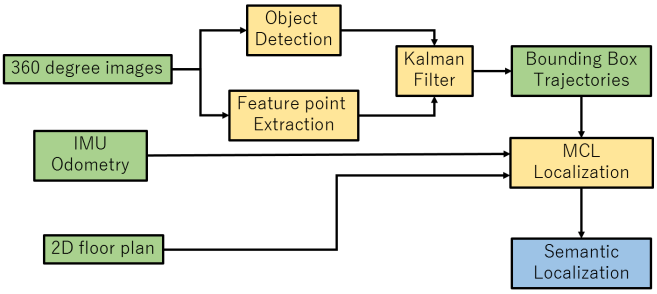


Fig. 3. System Overview

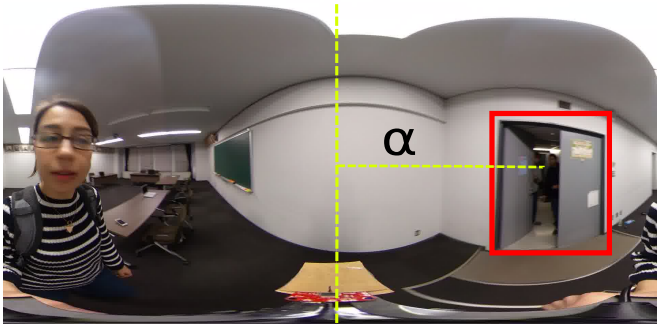


Fig. 4. Bearing angle calculation

serve as landmarks for localization. These landmarks include common objects that appear on floor plans, such as doors, windows, tables, chairs, etc. Depending on specific environments, objects can be added or removed (for e.g., for classrooms, "whiteboards" can be included). These objects are detected with the help of CNNs such as YOLO [4], [3] and bounding boxes are extracted.

Since bounding boxes are unstable and jittery, feature points are used for stabilizing them. We "tag" feature points according to the bounding box they were detected in. After tracking, the bounding box trajectory is calculated and fed into a Kalman filter for stabilization.

The stabilized bounding box centers are used in an Monte-Carlo Localization (MCL) [15] formulation to give the final result. Since MCL also requires odometry information, we make use of an Inertial Measurement Unit (IMU) as these are typically available in most robots. Fig. 3 provides an overview of the system flow.

### B. Detection of Semantic Information

First, we acquire semantic information in images via bounding box extraction. Specifically, given an image, we require the object class of each object, and its bearing angle inside the image. This is the 2D bearing angle as we aim at estimating the position $(x, y)$ and orientation $\theta$. Instead of pixel-wise semantic segmentation, which takes a heavy computational toll, our method relies on bounding box information alone, making it lightweight. In order to achieve this, we make use of TinyYOLOv2[3] . It is easy to pretrain TinyYOLOv2 for all the objects in the environment, as they are already known from the map. A 360° camera is used to capture as much of the environment as possible. Equirectangular images are output from the camera and object detection is done directly on these.

Based on each bounding box center, the bearing angle can be calculated as shown in Fig. 4. Each equirectangular image goes from 0 to 360 degrees. The center of the image is set as 0 degrees. In the equirectangular image coordinate system, we assume the center of the image to be the origin. The bearing angle for each bounding box, $\alpha$ is calculated by dividing the horizontal coordinate of its center by the width of the image and multiplying by 360. Here, $k$ is the bounding box index. Once the bearing angles for each bounding box have been detected, they need to be smoothed in order to make them stable for accurate localization.

### C. Stabilization of Semantic Information

In order to smoothen and stabilize the trajectory of each bounding box, we detect feature points over the image and track features in consecutive frames that belong to each bounding box. We make use of AKAZE[14] features as they can withstand the strong distortions present in 360 degree equirectangular images. These distortions are due to the projection of a sphere on a 2D surface and are variable depending on the height of each image pixel.

Once AKAZE features have been detected in consecutive images, we filter them in a RANSAC[16] approach. Typically, for planar images, RANSAC calculates the deviation from the epipolar equation as a euclidian distance from the epipolar plane. However, since we use 360 degree images which cannot be projected straightforwardly on a plane, they are projected on the surface of a unit sphere. Using the typical RANSAC formulation leads to low accuracy on spherical images.

Hence, in order to calculate epipolar error for RANSAC, the spherical geodesic error is used instead, in the same manner as done in [17]. The geodesic error of each feature point is the distance calculated on the surface of the unit sphere. Instead of epipolar lines, we obtain epipolar curves.

After RANSAC filtering, each inlier feature pair which lies within a bounding box is tagged with its label. If a bounding box is present in both images of the consecutive frame pair, the average velocity on the equirectangular image surface is calculated over all the features present within it to calculate the bounding box velocity $\dot{c}_k$. To be regarded as corresponding bounding boxes, we check the following three conditions: 1. Coordinates of the two boxes are close enough. 2. Sizes of the two boxes are almost the same. 3. Labels of the two boxes are exactly the same.

We use a constant velocity Kalman filter in order to update the trajectory of each bounding box based on its previous state and the bounding box velocity. Since the frame rate is high, we assume each bounding box to have a linear trajectory. This is valid due to the fact that most objects are present towards the center of the equirectangular image and move mostly horizontally. On a side note, this also makes it easier for TinyYOLOv2 to detect each bounding box.

This, for each bounding box, we update the velocity by the average velocity of all feature points present within it. A

match of all TinyYOLOv2 detections is performed to check whether new objects have entered the view of the camera. If so, a new trajectory is started. Likewise, disappearing trajectories are also calculated. Objects typically disappear when the object becomes too small.

Once the Kalman filter has smoothed the bounding box trajectories, we use them in an MCL-based localization within the floor map. This is described in the next section.

### *D. MCL-based Localization*

In this work, we make use of Monte-carlo Localization (MCL) [18]. Typically, MCL uses laser range-finders and compares bearing-distance measurements to a known 2D geometric map of the environment. In this work, we replace the distance measurements with the smoothed bounding box measurements defined as $\mathbf{Z}= \{(\mathrm{c}_k, \alpha_k)\}_{k=1}^{K}$, where $K$ is the number of objects in each sensor reading. For MCL, we also require odometry information. For this, we make use of an inertial measurement unit (IMU).

The Monte Carlo Localization (MCL) [19] is a recursive Bayes filter for estimating the pose of the user. Bayes filters estimate the state of a dynamical system with Markov property assumption for the environment. The probability distribution of the current state only depend on the previous state. Readings from different sensors such as rangefinder, cameras etc. can be used to compare against a pre-built map. Each particle in MCL is a pose hypothesis with a weight. The weights sum to one. The particles get updated as the user moves around. if the system does not have any prior information about the real pose, MCL generally starts with uniform distribution of particle.

MCL has prediction, update, and resampling steps to make particles converge towards the user's location. At the prediction step, particles are propagated according to the last odometry information. The prediction step is followed by an update step. In the update step, each particle is evaluated on how much it correlates with the current observations and its weight is updated proportionally. The last step is the resampling step, where particles are resampled according to the their new weights. Our method requires an annotated 2D map $\mathbf{M}$ which stores Cartesian Coordinates $(\mathrm{x}, \mathrm{y})$ of annotated objects from different predetermined object classes c, state$(\mathrm{x}, \mathrm{y}, \theta)$ of the user $\mathbf{s}$, odometry information $\mathbf{u}$ and sensor update $\mathbf{Z}$ in our method. The 2D annotated map $\mathbf{M}$ is defined as $\mathbf{M}= \{\mathrm{x}_n, \mathrm{y}_n, \mathrm{c}_n\}_{n=1}^{N}$ where $N$ is the number of objects in the map.

The smoothed bounding box measurements $\mathbf{Z}$ contain detected object class name $c$ and bearing angle $\alpha$ to the center of the object.The measurement update is $\mathbf{Z}= \{(\mathrm{c}_k, \alpha_k)\}_{k=1}^{K}$, where $K$ is the number of objects in each sensor reading.

Particles from $P(\mathbf{s_{t-1}}|\mathbf{z_{t-1}}, \mathbf{u_{t-1}})$ are propagated with a motion model at the prediction step. At the update step each particle weight gets updated according to how much the current measurements match to the map by a sensor model $P(\mathbf{z_t}|\mathbf{s_t}, \mathbf{M})$. The update step is followed by a re-sampling step based on particle weights into a posterior $P(\mathbf{s_t}|\mathbf{z_t}, \mathbf{u_t})$.The posterior is calculated as;

$$\begin{aligned} &P(\mathbf{s}_t|\mathbf{Z}_t, \mathbf{u}_t, \mathbf{M}) = \\ &P(\mathbf{Z}_t|\mathbf{s}_t, \mathbf{M})P(\mathbf{s}_t|\mathbf{u}_t, \mathbf{s}_{t-1})P(\mathbf{s}_{t-1}|\mathbf{Z}_{t-1}, \mathbf{u}_{t-1}, \mathbf{M}) \end{aligned} \tag{1}$$

There are two common ways to compute a joint likelihood distribution.The first one is to assume measurements from the same reading $\mathbf{z}_p$ and $\mathbf{z}_q$ to be conditionally independent to each other given the location and apply product of likelihood as

$$\begin{aligned} P(\mathbf{z}_p|\mathbf{z}_q, \mathbf{s}) = P(\mathbf{z}_p|\mathbf{s}) \qquad \forall p \neq q, \\ P(\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k\}|\mathbf{s}, \mathbf{M}) = \prod_{k=1}^{K} P(\{\mathbf{z}\}_k|\mathbf{s}, \mathbf{M}) \end{aligned} \tag{2}$$

The second approach elevates each measurement to the power of a "smoothing" coefficient $\lambda$ and computes the joint distributions as weighted geometric means of the individual likelihoods as

$$P(\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k|\mathbf{s}) = \left( \prod_{i=j}^{K} P(\mathbf{z}_j|\mathbf{s})^{\lambda_j} \right)^{1/\sum_{j=1}^{K} \lambda_j} \tag{3}$$

Please refer to [20] for derivation and details. The first approach might yield to peaky, and overconfident results. Especially wrong measurements might have a big impact on the result. The second approach smooths the joint likelihood, and might be able to deal with wrong measurements. In [21], different object classes were treated as different sensors, and they were assigned different smoothing powers to smooth measurement errors bu using Eq 3. Even though assigning fixed smoothing powers were efficient and robust in many cases, it was also proven to be very environment dependent and hard to generalize. Besides, the motivation for applying Kalman Filter is to stabilize the measurements and work with realiable data. Therefore, instead of relying on certain object classes more than others, we rely on all tracked objects. In this work we also assumed independence across objects,and used log-likelihood by taking the natural logarithm of the likelihood function. The annotated objects in map $\mathbf{M}$ and observations in $\mathbf{Z}$ do not have unique IDs. The measurements were matched according to maximum likelihood as

$$P(\mathbf{z}^k|\mathbf{s}, \mathbf{M}) = \quad \mathbf{max}(P(\mathbf{z}^k|\mathbf{s}, \mathbf{M}^n) \forall \mathbf{M}^n \in \mathbf{M} | \mathbf{M}_c^n = \mathbf{z}_c^k) \tag{4}$$

## V. Experimental Evaluation

In order to check the performance of the proposed method and evaluate the contribution of smoothing the bounding box information using feature points, we conducted a real experiment in a classroom environment, as shown in Fig. 5. In order to construct a floorplan, we used ARUCO [22] markers and implemented an ARUCO SLAM inside a Robot Operating System (ROS) environment. This step can be done by hand, as well, by knowing the dimensions of the room.

Fig. 5. Experimental Environment

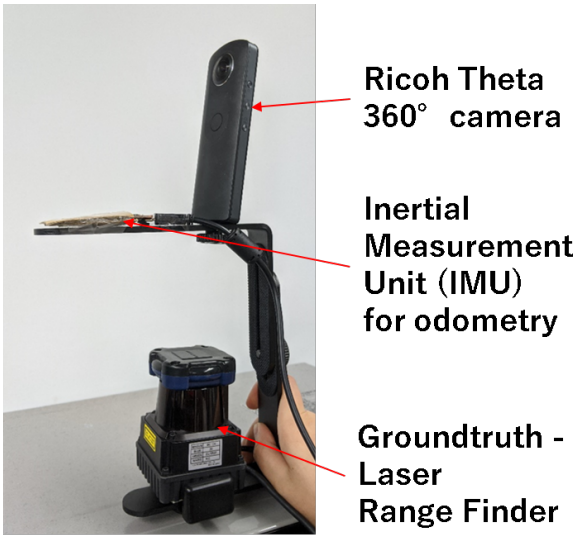


Fig. 6. Experimental Setup

Our experimental setup, shown in Fig. 6, consisted of a Ricoh Theta V 360 degree spherical camera and a Sparkfun IMU. In order to record groundtruth, we made use of a laser range finder combined with the IMU, implementing its own MCL localization. All three were connected to a laptop to run the required ROS nodes, and to a battery. The setup was carried by a person who walked around the room. The entire sequence amounted to around 3 minutes.

Frame-by-frame semantic localization was conducted using the proposed method. In order to evaluate the effect of smoothing the bounding box information, evaluation was also conducted without the feature point-based Kalman filter, in accordance to our previous work in [13]. Fig. 7 shows the estimated trajectory in both approaches overlaid on the groundtruth, along with the semantic floor plan map of the room.

The evaluation errors are tabulated in Table I. It can be noticed that the mean position and orientation errors reduced,

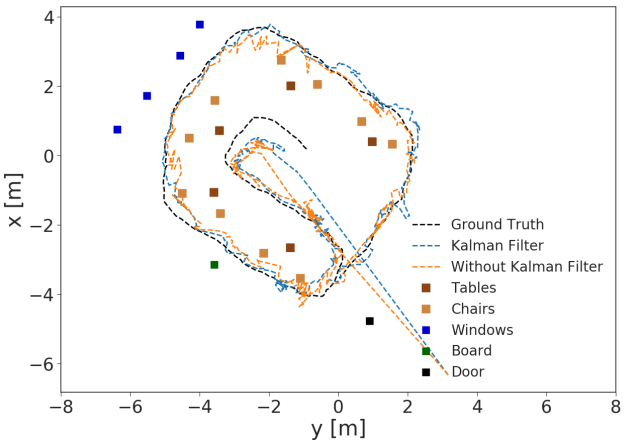


Fig. 7. Floor plan map of the experimental environment showing the trajectory

TABLE I

MEANS ERRORS WITH AND WITHOUT THE USE OF FEATURE POINT INFORMATION

| Error | Without feature points | With feature points |
|---|---|---|
| **Mean(m)** | 0.47 | 0.40 |
| **Std(m)** | 0.19 | 0.16 |
| **Mean(rad)** | 0.20 | 0.09 |
| **Std(rad)** | 0.19 | 0.06 |

and so did their standard deviations.

## VI. DISCUSSIONS AND CONCLUSION

In this research, semantic localization was defined as the localization of an agent within a 2D map consisting of object information, such as an indoor floor plan. It was justified to be of importance for robots and other intelligent agents which need to navigate in and interact with objects in an indoor environment.

This was achieved with the help of a 360 degree camera and was based on two kinds of information: 1. Object detection and 2. Feature point information. Descriptive object detection formed the core of the method and provided landmarks necessary for localization. Meanwhile, non-descriptive feature point information served as a local anchor for jittery and unstable object bounding boxes in order to increase the accuracy of the final output. In addition, the use of an IMU provided odometry information crucial for the MCL-based localization approach that was followed. This was experimentally verified in a real environment. The only preparation the system required was to be provided a floor plan of the target environment, which is easy to obtain in most cases.

While the accuracy of the system is enough for a robot to navigate and approach objects, it remains lacking for interaction with an object. In order to provide this, it is necessary to accurately track local robot motion close to an object and understand its 3D structure, which will be considered in future work.

Including extra iterative steps created extra computation burden. During feature point selection AKAZE features showed better performance for tracking than ORB features. On the other hand, ORB features were faster. Thus, the selection was a trade of accuracy and speed.

The errors of tinyYOLO2 and the tracking algorithm were similar. Despite the improvements made to its first version to detect smaller objects, tinyYOLO2 still struggles with detection of objects are small in user's perspective. Therefore, bigger objects had better performance than the smaller ones. Among the object classes, the longest tracking trajectory belonged to the board class. It was followed by the trajectory of the door. The worst trajectory performance belonged to chairs. During localization only objects which were tracked above a certain threshold of frame numbers were used. When an object disappears, the trajectory was saved for a certain number of frames, and continued when the object is detected again. If the object does not appear then it is deleted. This process was challenging for smaller objects, since their detection is not very reliable and they exist in a big number in the environment. Without any identification, tracking was possible for only some of the members from each class. Motion blur was a problem for the tracking system. It caused losing the tracking when the user was taking corners. This situation resulted in increased errors.

Even though all objects which were tracked above a frame number threshold were accepted to be completely reliable, there were big differences across their trajectory lengths. Even though, setting fixed smoothing factors according to the object class is not very generalizable, they can be set according to the trajectory length adaptively. However, scaling trajectory length to the fixed smoothing factors could cause the same complications as fixed ones.

## REFERENCES


[1] D. López-de Ipiña, T. Lorido, and U. López, "Indoor navigation and product recognition for blind people assisted shopping," in *International Workshop on Ambient Assisted Living*. Springer, 2011, pp. 33–40.

[2] R. Miyagusuku, A. Yamashita, and H. Asama, "Improving gaussian processes based mapping of wireless signals using path loss models," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4610–4615.

[3] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7263–7271.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 779–788.

[5] W. Winterhalter, F. Fleckenstein, B. Steder, L. Spinello, and W. Burgard, "Accurate indoor localization for rgb-d smartphones and tablets given 2d floor plans," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 3138–3143.

[6] H. Chu, D. Ki Kim, and T. Chen, "You are here: Mimicking the human thinking process in reading floor-plans," in *IEEE International Conference on Computer Vision*, 2015, pp. 2210–2218.

[7] J. M. Coughlan and A. L. Yuille, "The manhattan world assumption: Regularities in scene statistics which enable bayesian inference," in *Proceedings of the Neural Information Processing Systems Conference*, vol. 2, 2000, p. 3.

[8] T. Goto, S. Pathak, Y. Ji, H. Fujii, A. Yamashita, and H. Asama, "Line-based global localization of a spherical camera in manhattan worlds," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2296–2303.

[9] M. Himstedt and E. Maehle, "Semantic monte-carlo localization in changing environments using rgb-d cameras," in *2017 European Conference on Mobile Robots (ECMR)*, Sep. 2017, pp. 1–8.

[10] O. Mendez Maldonado, S. Hadfield, N. Pugeault, and R. Bowden, "Sedar–semantic detection and ranging: Humans can localize without lidar, can robots?" in *IEEE International Conference on Robotics and Automation*, 2018.

[11] M. Jüngel and M. Risler, "Self-localization using odometry and horizontal bearings to landmarks," in *RoboCup 2007: Robot Soccer World Cup XI*, U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 393–400.

[12] A. W. Stroupe and T. Balch, "Collaborative probabilistic constraint-based landmark localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Sep. 2002, pp. 447–453 vol.1.

[13] I. Uygur, R. Miyagusuku, S. Pathak, A. Moro, A. Yamashita, and H. Asama, "A framework for bearing-only sparse semantic self-localization for visually impaired people," in *2019 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2019, pp. 319–324.

[14] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," in *Proceedings of the British Machine Vision Conference*, September 2013.

[15] M. Himstedt and E. Maehle, "Semantic monte-carlo localization in changing environments using rgb-d cameras," in *2017 European Conference on Mobile Robots (ECMR)*, Sep. 2017, pp. 1–8.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[17] S. Pathak, A. Moro, H. Fujii, A. Yamashita, and H. Asama, "Distortion-resistant spherical visual odometry for uav-based bridge inspection," in *Proceedings of SPIE, Vol. 11049 (Proceedings of the 2019 Joint Conference of the International Workshop on Advanced Image Technology (IWAIT2019) and the International Forum on Medical Imaging in Asia (IFMIA2019))*, January 2019, pp. 110 491O–1–110 491O–6.

[18] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term visual localization using semantically segmented images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 6484–6490.

[19] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 1999, pp. 1322–1328.

[20] R. Miyagusuku, A. Yamashita, and H. Asama, "Data information fusion from multiple access points for wifi-based self-localization," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 269–276, April 2019.

[21] I. Uygur, R. Miyagusuku, S. Pathak, A. Moro, A. Yamashita, and H. Asama, "Robust and efficient indoor localization using sparse semantic information from a spherical camera," *Sensors*, vol. 20(15), no. 4128, 2020.

[22] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, p. 2280–2292, 06 2014.