2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Global Localization Over 2D Floor Plans with Free-Space Density Based on Depth Information

Renan Maffei[†] Diego Pittol[†] Mathias Mantelli[†] Edson Prestes[†] Mariana Kolberg[†]

Abstract— **Many applications with mobile robots require self-localization in indoor maps. While such maps can be previously generated by SLAM strategies, there are various localization approaches that use 2D floor plans as reference input. In this paper, we present a localization strategy using floor plan as map, which is based on spatial density information computed from dense depth data of RGB-D cameras. We propose an interval-based model, called Interval Free-Space Density, that bounds the uncertainty of observations and minimizes the effects of movable objects in the environment. Our model was applied in a Monte Carlo Localization strategy and compared with traditional observation models. The results of experiments showed the robustness of the proposed method in single-camera and multi-camera experiments in home environments.**

I. Introduction

Global localization is a fundamental problem of mobile robotics that consists in estimating the pose of a robot, that is initially unknown, relative to a map of the environment known a priori [1]. Most approaches model such estimate as a probability distribution, which is updated through Bayesian filtering. Among those approaches, Monte Carlo Localization (MCL) [2] stands out as one of the most popular, due to its ability to model arbitrary distributions, robustness and simplicity. However, the efficiency of the localization process depends on a series of factors, such as the map representation, and the way sensor measurements are modelled.

Different map representations have been used in robotics throughout the years and each of them are more suitable to some types of sensors and environments. For example, grid maps are easily generated from rangefinders sensors such as lasers and sonars; feature-based maps can be generated from camera images, etc. That said, in order to localize a robot in some specific map, this map must have been previously built, for instance, using some SLAM technique. Nevertheless, when we are dealing with localization in indoor environments, often there are maps of the environment already available in the form of floor plans. Floor plans are not perfect representations of the real environments, due to the absence of objects such as furniture that impact the sensors observations. Still, they describe the complete structure of environments, serving as good representations of them.

[†]Institute of Informatics, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil rqmaffei, dpittol, mathias.fassini, prestes, mariana.kolberg@inf.ufrgs.br

This study was financed in part by the Brazilian National Council for Scientific and Technological Development (CNPq) and by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

In recent years, many localization approaches use floor plans as reference input [3]–[6]. Among them is Ribacki et al.'s approach [6], which is based on the concept of Free-Space Density (FSD) [7]. The FSD of a given position in space is a measure of the free-space surrounding such position computed using a circular kernel. For instance, wide regions (e.g. middle of large rooms) have high FSD values, while narrow regions (e.g. small corridors, places near corners of rooms) have low FSD values. The value depends on the selected kernel radius, but usually a radius of couple meters is a safe choice for domestic scenarios [7]. Due to its conciseness, FSD is suitable for efficient queries, and applied in MCL it is able to handle a large number of particles very fast. Additionally, FSD describes a characteristic of a given region which is the same regardless the sensors used. Thus, it is applicable with different sensors, such as laser rangefinders [7] and omnidirectional cameras capturing ceiling images [6].

In this paper we propose a localization strategy based on free-space density for a robot equipped with RGB-D cameras. This type of sensor provides data in the form of dense 3D point clouds, which is much more information than what was dealt with in previous FSD works. However a large part of such information is associated with objects that are not present in the reference map, which needs to be taken into account. The main contributions of this paper are:

- A strategy to compute FSD from dense 3D point clouds that seeks to minimize the effect of objects that are absent in the reference map.
- A modification to FSD, called **Interval FSD**, that determines the uncertainty of the FSD value in interval form and makes the technique more robust.

The method was evaluated in multi-cameras and single-camera scenarios of a robot moving in different domestic environments. Our code is available online[1].

This paper is organized as follows. Section II presents relevant work on the localization problem. Sections III and IV present the proposed approach and experimental results. Section V presents conclusion and future work.

II. Related Work

Robot localization is a widely studied problem and since the seminal work of Dellaert et al. [2], several localization approaches were proposed based on Monte Carlo Localization. It is possible to find approaches for robots with different

[1]https://github.com/phir2-lab/fsd_localization

sensors like laser rangefinders, monocular cameras, or RGB-D cameras.

Boniardi et al. [8] used a LiDAR sensor and proposed a scan-to-map-matching method to online augment a 2D floor plan with 3D information extracted from a pose-graph-based SLAM. However, their approach not intent to solve the global localization problem and need a known coarse start pose. Later, Boniardi et al. [5] extended [8] to handle both static and changing environments in long-term localization.

Winterhalter et al. [3] localize a Google Tango tablet in a 2D floor plan map. The device provided the visual-inertial odometry, and the similarity between the observation and the map is computed using a subset of RGB-D measures randomly selected. Fang et al. [9] focused on the 3D localization problem in visually degraded environments using an RGB-D camera. They proposed a 6DoF odometry estimation method, and 6DoF Localization based on MCL, to localize the robot in a given 3D global map.

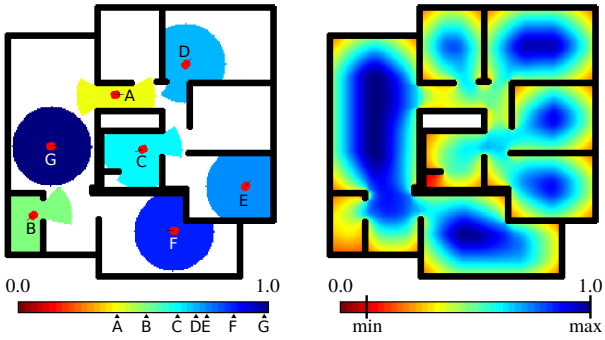
The RGB-D MCL method proposed by Ito et al. [4] uses the WiFi signal strength to estimate a coarse initial distribution, removing ambiguities. Then, they extract planes from the point cloud and project them down onto the 2D floor plan. Fallon et al. [10] proposed to generate a simplified 3D map composed of large planar segments, such as walls and ceilings, in a preliminary mapping step. Then, the likelihood of each particle is evaluated comparing the downsampled RGB-D image with each particle's prediction of the scene, synthesized using a GPU to run in real-time. Biswas et al. [11] introduced the Fast Sampling Plane Filtering algorithm that samples the depth image to produce a set of points corresponding to planes, significantly reducing the amount of data to be processed. Then, these points are matched against the lines in the 2D floor plan.

Boniardi et al. [12] proposed a monocular localization system that employs a CNN to predict the room layout edges from the image. Then, an MCL was used to localize the robot, comparing these edges with those inferred from the 2D floor plan. A 3D metrical point cloud obtained from a monocular visual SLAM was used by Chu et al. [13] to extract information about doors, architectural lines, and the observed free space, incorporating them into the observation model. Mendez et al. [14] extracted semantic information from RGB images by processing them with a CNN. They proposed a sensor model to MCL that rely on likelihood fields computed for walls, doors, and windows. A similar approach, but without semantics, was proposed by Merriuax et al. [15] that extended the likelihood field to 3D based only in the obstacles, and used it in the observation model.

III. Localization in 2D floor plans using free-space density based on depth information

A. Free-space density (FSD)

The proposed method relies on the concept of Free Space Density (FSD) [7], which is a kernel density estimate of the amount of free space around a given position of a grid map.



(a) Examples of FSD values (b) FSD field

Fig. 1: Free-space density computed over a gridmap. (a) The colored regions show the cells inside a circular kernel that are visible from its center, placed at different positions. (b) The FSD values of all cells of the gridmap are illustrated by different colors going from dark red ($\Psi = 0$) to dark blue ($\Psi = 1$).

The FSD (Ψ) of a region centered at cell $\mathbf{m}_0$ is a real value between 0 and 1 that can be computed as follows

$$\Psi(\mathbf{m}_0) = \sum_{\mathbf{m}_i} s(\mathbf{m}_i, \mathbf{m}_0) K(||\mathbf{m}_i - \mathbf{m}_0||) \qquad (1)$$

where $K(.)$ is a circular kernel profile[2], $\mathbf{m}_i$ is a cell limited by the kernel radius and

$$s(\mathbf{m}_i, \mathbf{m}_0) = \begin{cases} 1 & \text{, if } \mathbf{m}_i \text{ is a } \textit{free-space} \text{ cell} \\ & \text{and } \textit{visible} \text{ from } \mathbf{m}_0 \\ 0 & \text{, otherwise} \end{cases} \qquad (2)$$

The value of FSD is independent of orientation, which means that every position in a given 2D grid map is associated to only one FSD value. Fig. 1a shows examples of FSD values computed in different positions of a map, considering a circular kernel of $1.5m$ radius. In wide areas such as position G, the FSD value is near the maximum (1.0) because the whole circular kernel is applied over free space cells; on the other hand, in narrow regions such as A and B, the FSD value is much smaller. Fig. 1b shows the resulting FSD field computed over the grid map.

The cost of computing FSD in a given position of a grid map is proportional to the kernel radius (k_r), because, at the most, it requires checking the visibility of all cells inside the kernel. This can be done performing a raycast from the kernel center to all border cells[3], stopping at obstacles.

However, the fact that FSD is just a single value favors pre-caching, differently from approaches with more complex observation models, in which pre-caching can be high costly in terms of memory requirements or only small parts of the process can be pre-computed. This is particularly suitable for particle filters, because the process of particles' weighting (generally the costly step of the filter) becomes a single-valued table look-up.

[2]In this work we simply use a Uniform kernel profile, but other profiles, such as Gaussian, can be used to give more weight to cells near the center of the kernel [6], [7].

[3]Considering that there are $2\pi k_r = O(k_r)$ border cells, and a single raycast operation is $O(k_r)$, the cost of the FSD computation is $O(k_r^2)$.

The map of the environment is obtained from a 2D floor plan such as in [3], [6]. For each dataset tested, a 2D grid map is built defining as *obstacles* the cells associated to walls (with the exception of internal doorways); as *unknown*, the cells outside the map; and as *free-space*, the remaining cells. The FSD field map is computed over all free-space cells prior the start of the localization process.

B. Computing FSD from camera depth information

Since FSD describes a basic spatial characteristic of environment regions, the FSD map (i.e., the one with reference values used to evaluate the robot observations) is always computed the same way, independent from the type of robot sensors. This is the case for robots equipped with 2D laser rangefinders [7], with omnidirectional cameras [6], or, for instance, with RGB-D cameras, as proposed in this paper. On the other hand, computing the FSD based on the actual robot observations changes according to the sensor used.

Computing FSD from camera depth data takes five steps:

1) Horizontal downsampling of the depth data;
2) Projection of selected measurements over a 2D plane;
3) Filtering max ranges in each projected direction;
4) Updating local grid map with filtered measurements;
5) Computing FSD on local grid map.

Note that this algorithm can be applied for any type of 3D point cloud, since it does not use color information, but only positional information.

The first step, i.e. downsampling the depth data information, is not mandatory, but gives a great contribution to reduce the processing cost. Since the method deals with a 2D floor plan, the downsampling is made by selecting horizontal slices of the depth data, to cover more area with less points. Fig. 2c shows an example of a downsampled point cloud, generated from the camera image in Fig. 2a, in comparison with the full point cloud shown in Fig. 2b.

In the second step, the selected measurements are projected from 3D to 2D. Then, during the third step, we filter the measurements by keeping only the one with the maximum projected range, among all that happen in the same orientation[4]. The goal is to ignore dynamic obstacles, by keeping only the measurements associated with static obstacles, like walls, that ideally are the maximum ranges measured indoor. Of course, this is not true when the robot senses through windows or in front of mirrored surfaces, but in general it is shown to be a good strategy. We also keep only measurements of points above $0.5m$ and below $2.5m$ to remove detected measurements of the floor and the ceiling.

The fourth step is updating a local grid map with the filtered ranges. Like [7], we update the grid map using the HIMM method [16]. Also, we reset all cells with a distance at least $2 \times K_r$ from the robot position to avoid inconsistencies from past observations. Fig. 2d shows an example of map update, where the updated cells are highlighted in pink. Finally, the fifth step is computing the FSD, as defined in section III-A, by applying a circular kernel of radius K_r over the grid and counting all visible free-space cells inside it, as shown in Fig. 2e.

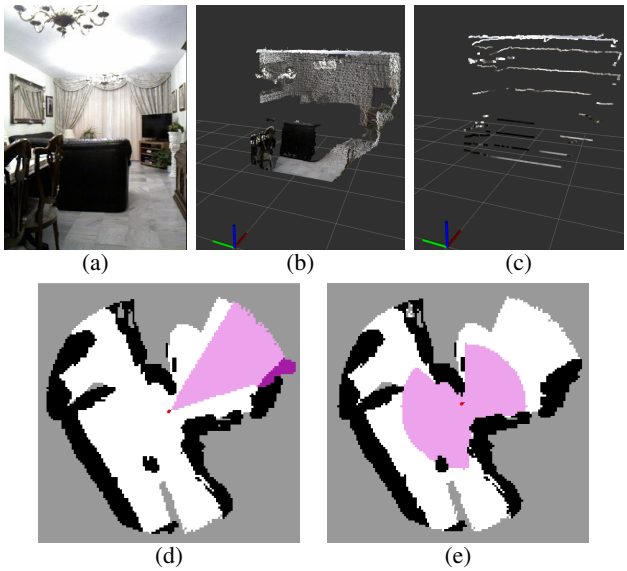


Fig. 2: FSD obtained from camera depth information. (a) Camera image. (b) Full point cloud obtained from depth data. (c) Downsampled point cloud. (d) Update of local grid map with the maximum projected measured ranges. (e) Visible free-space cells used for FSD computation.

C. Interval FSD

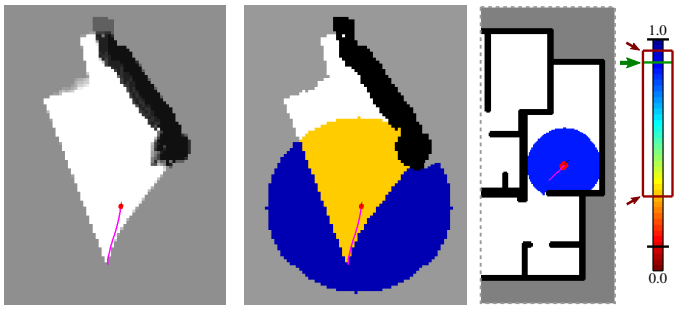
An issue about comparing free-space densities from a map known a priori and a local map that is being built during runtime, is that, depending on the sequence of observations made by the robot, the local map may look very different while not fully completed, thus the computed FSD value may vary. A fully completed local map is a grid with no reachable unknown cells[5] inside the kernel. In previous works [7], the FSD is considered *undefined* in situations like that, and the *weighting+resampling* steps of the localization process are momentarily suspended, causing an increase of uncertainty.

Nonetheless, we can estimate an interval of possible FSD values considering the current local map. In the worst case, this interval encompasses all values from 0 to 1, and the FSD will be *undefined* just like in [7]. Fortunately, this interval is generally smaller. Given the example in Fig. 3, the minimum possible FSD value is the current FSD considering the known free-space, which usually is larger than 0. The maximum possible FSD value must be computed with the maximum possible visible free-space cells inside the kernel, which is the sum of current visible free-space cells and the current visible unknown cells, as shown in Fig. 3b. Commonly, after the initial moments when the robot enters a room, the amount of unknown cells surrounding the robot pose tends to decrease fast, thus most of the time the interval of possible FSD values is small.

According to interval arithmetics definitions [17], we define the Interval FSD, $[\Psi(\mathbf{m}_0)]$, of a region centered at

[4]For this, we discretize the orientations in steps of 1°.

[5]That is, no unknown cells neighboring free-space cells inside the kernel.



(a) (b) (c)

Fig. 3: Interval FSD. (a) Local map with known free-space (white), known obstacles (black), unknown cells (gray). (b) Minimum visible free-space inside kernel (yellow) and maximum possible free-space (blue). (c) True FSD value computed in the reference map (pointed by green arrow) is inside the interval of FSD values.

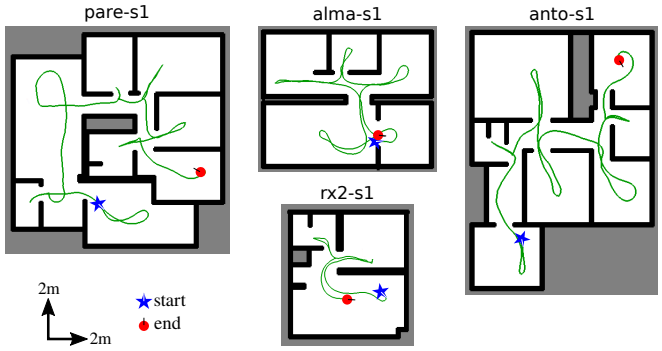


Fig. 4: Maps and trajectories of the four tested scenarios from the *Robot@home dataset* [18]. **pare-s1**: area $10.2\times10.3m^2$, path length $43.2m$; **alma-s1**: area $8.2 \times 6.6m^2$, path length $39.9m$; **anto-s1**: area $8.7 \times 12.4m^2$, path length $43.7m$; **rx2-s1**: area $5.7 \times 6.1m^2$, path length $15.7m$.

cell $\mathbf{m}_0$, as follows

$$[\Psi(\mathbf{m}_0)] = \left[\underline{\Psi(\mathbf{m}_0)}, \overline{\Psi(\mathbf{m}_0)}\right] = \{\Psi \in \mathbb{R} \,|\, \underline{\Psi(\mathbf{m}_0)} \leq \Psi \leq \overline{\Psi(\mathbf{m}_0)}\} \quad (3)$$

where the infimum $\underline{\Psi(\mathbf{m}_0)}$ of the interval corresponds to the definition of the FSD in Eq. 1, that is $\underline{\Psi(\mathbf{m}_0)} = \Psi(\mathbf{m}_0)$; and the supremum $\overline{\Psi(\mathbf{m}_0)}$ of the interval is

$$\overline{\Psi(\mathbf{m}_0)} = \sum_{\mathbf{m}_i} s_{unk}(\mathbf{m}_i, \mathbf{m}_0) K(||\mathbf{m}_i - \mathbf{m}_0||), \quad (4)$$

with

$$s_{unk}(\mathbf{m}_i, \mathbf{m}_0) = \begin{cases} 1 & \text{, if } \mathbf{m}_i \text{ is a } \textit{free-space} \text{ or } \textit{unknown} \\ & \text{cell and } \textit{visible} \text{ from } \mathbf{m}_0 \\ 0 & \text{, otherwise} \end{cases} \quad (5)$$

D. Monte Carlo Localization using Interval FSD

A popular solution for the global localization problem, in which the initial pose of a robot inside a known environment is unknown, is the Monte Carlo Localization (MCL) [2]. In MCL, the posterior distribution of the robot pose is estimated with a set of weighted particles that are updated in a process of sampling, importance weighting and resampling. The steps of sampling and weighting depend on the robot motion model and the measurement model, respectively. In this work, we use a traditional odometry-based motion model [1], while the observation model is based on Interval FSD values.

At instant t, the weight of the i-th particle, $\boldsymbol{p}_t^{[i]}$, is defined as

$$w(\boldsymbol{p}_t^{[i]}) = \begin{cases} 1 & \text{, if } \Psi(\boldsymbol{m}_t^{[i]}) \in [\Psi(\boldsymbol{m}_t^r)] \\ f_\Psi(\Psi(\boldsymbol{m}_t^{[i]}), \overline{\Psi(\boldsymbol{m}_t^r)}) & \text{, if } \Psi(\boldsymbol{m}_t^{[i]}) > \overline{\Psi(\boldsymbol{m}_t^r)} \\ f_\Psi(\Psi(\boldsymbol{m}_t^{[i]}), \underline{\Psi(\boldsymbol{m}_t^r)}) & \text{, if } \Psi(\boldsymbol{m}_t^{[i]}) < \underline{\Psi(\boldsymbol{m}_t^r)} \end{cases} \quad (6)$$

where $\Psi(\boldsymbol{m}_t^r)$ is the Interval FSD computed at the local map given the current observations, $\Psi(\boldsymbol{m}_t^{[i]})$ is the FSD at the particle position, and f_Ψ is the following function,

$$f_\Psi(\Psi_a, \Psi_b) = 1.0 - \frac{\min(|\Psi_a - \Psi_b|, \Delta_\Psi)}{\Delta_\Psi}, \quad (7)$$

with Δ_Ψ being the difference between the maximum and minimum FSD values in the reference map. In short, particles with FSD value inside the boundaries of the computed Interval FSD receive the maximum weight; the remaining particles are weighted in function of the distance to the nearest interval boundary – the smallest the difference, the higher the particle's weight.

IV. Experiments

The experimental validation was made using the *Robot@home dataset* [18] produced by researchers from the Univ. of Malaga. They used a mobile robot equipped with five sensors in parallel to the ground: a 2D laser range finder near the ground and four RGB-D cameras with orientations (yaw) of $-45°$, $0°$, $45°$, $90°$. Four domestic scenarios[6], described in Fig. 4, were selected to evaluate our work. The ground truth of the robot pose and the odometry were not directly available, and thus were respectively generated using SLAM and scan matching techniques[7]. For each scenario, we performed two types of tests: multi-camera (using the four RGB-D cameras) and single-camera (using only the RGB-D camera facing forward).

We compared our proposal (***Interval FSD***) to other five approaches of particles weighting: ***Cloud Likelihood*** - scan matching between K random measurements[8] from the point cloud, analyzing beam endpoints using a likelihood map, as used in [3]; ***Cloud Raycast*** - similar to the previous method, but performing raycasting to obtain more precise results; ***Laser*** - scan matching using the 2D laser readings; ***Pure Motion*** (no observation model) - only removing particles that go over obstacles or outside the map; and ***Absolute FSD*** - using in all situations the FSD considering only the known free-space. All experiments were performed in a notebook with 16GB and an i7 processor, using ROS

[6]The *Robot@home dataset* contains a fifth scenario, but collected with a very low framerate that prevents its use in our application.

[7]The floor plan for each environment was manually generated over a gridmap created by a laser-based SLAM, along with depth information from the 3D point cloud. The package developed for this is available at `https://github.com/phir2-lab/robotathome_at_ros`

[8]We selected K=100 random measurements in the experiments.

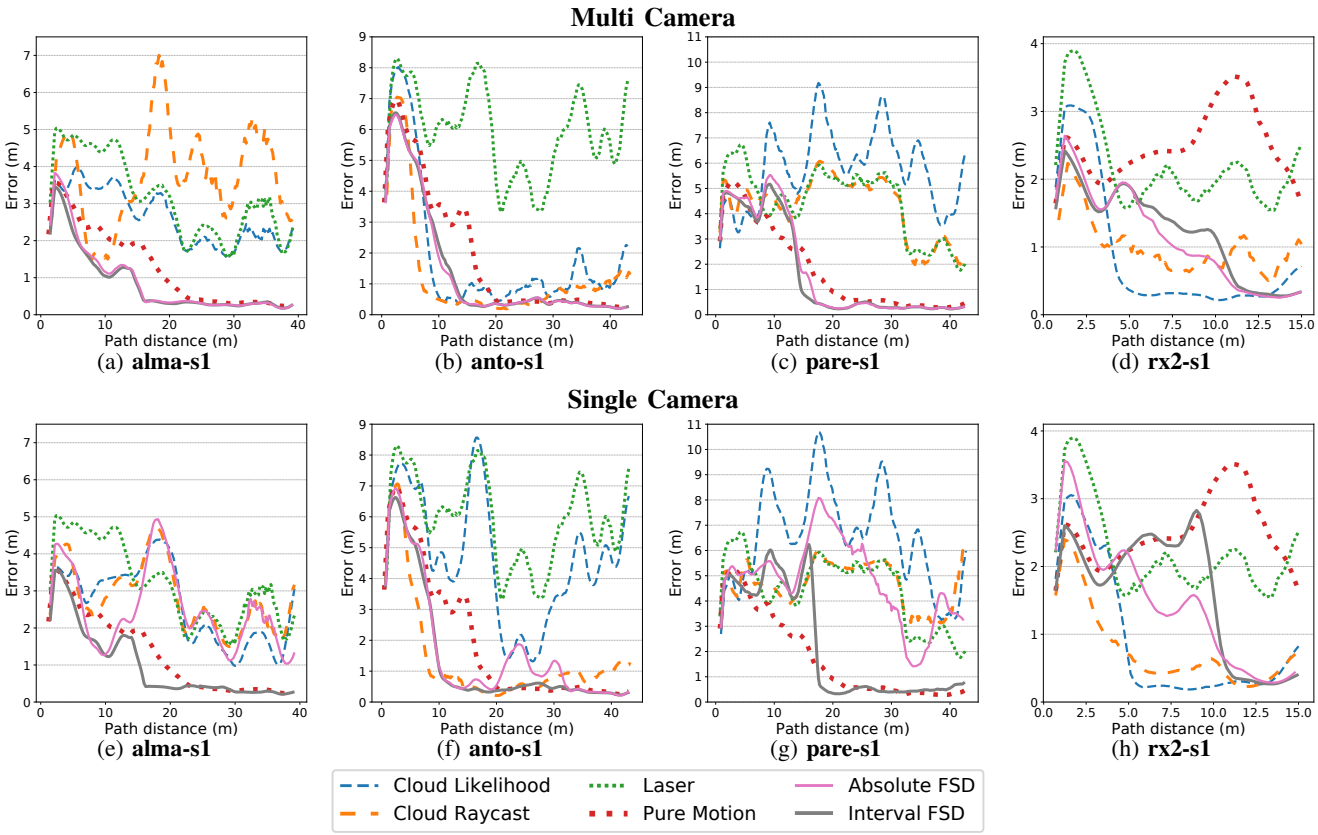


Fig. 5: Mean Particle Error in all experiments.

kinetic (the datasets were converted to the ROS format). Each configuration was run 10 times using 20000 particles.

Fig. 5 and Table I present a summary of the results. Fig. 5 shows the evolution of the **weighted mean error** of the particles position, while Table I analyze other metrics: the **succeed distance**, i.e., the total distance from the start that the robot traveled until the convergence[9] of the particle filter; the **mean position error after convergence**; the **final mean error**; and the **mean time per iteration** of the particle filter.

In general, the Interval FSD was the only method that converged in all datasets. The Absolute FSD converged in all multi-camera experiments, but suffered in the single-camera scenarios. This is expected because multiple cameras covers a large part of the surroundings of the robot, quickly leaving little unknown space in the local map centered at the robot position. In such cases, the Absolute and the Interval FSD behave very similarly. On the other hand, with a single camera the FSD intervals can be quite large when the robot enters a new room, which explains the poor results of the Absolute FSD in datasets such as *pare-s1*.

The approaches based on scan matching behaved well in the *rx2-s1* dataset, the simplest one. In fact, in such dataset both approaches - Cloud Likelihood and Cloud Raycast - converged much faster than FSD. However, they diverged on the other three datasets. The main problem of such approaches is that by choosing random measurements from the point cloud there is a high risk of making bad selections (i.e. measurements of furniture, movable objects), and bad weighting processes make the filter eliminate too much viable hypotheses over time. As shown in the experiments, when the real environment can be quite different from the reference map (like domestic environments) it is interesting to use conservative approaches, such as the Interval FSD.

Regarding the methods that do not use RGB-D information, the scan-matching of laser readings failed in all datasets, because the observations made by the laser captured all kinds of furniture (e.g. beds, couches, chairs), causing erroneous weighting most of the time. On the other hand, pure motion (i.e. only eliminating particles that go over walls) generally works if odometry is good and the trajectory traversed by the robot is not ambiguous. This is not the case in *rx2-s1*, in which pure motion could not converge, but it is in the other three datasets. The problem of such naive approach is that the convergence generally takes longer to occur.

Finally, Table I also shows the mean time per iteration obtained with the tested methods using the same amount of particles (N=20000). Both types of FSD are very fast because the weighting of a particle is basically a query of a single value in memory. It is important to note that no optimization, such as KLD, was implemented. Still, optimizations can be added to the MCL algorithm to reduce the number of particles, but in any case, the computational cost difference of the particles weighting will be relatively maintained.

[9]We consider convergence when the error becomes smaller than $1m$, and does not exceed this value until the end of the trajectory.

Method		Dataset												Time per iteration (s)
		alma-s1			anto-s1			pare-s1			rx2-s1			
		SucD	PosEAC	FErr	SucD	PosEAC	FErr	SucD	PosEAC	FErr	SucD	PosEAC	FErr	
Laser	μ	-	-	3.36	-	-	7.87	-	-	3.71	-	-	2.79	0.818
	σ	-	-	*1.41*	-	-	*2.86*	-	-	*2.81*	-	-	*1.10*	*0.304*
Pure Motion	μ	14.66	0.39	**0.40**	16.23	0.41	0.45	21.15	0.44	**0.55**	-	-	1.33	**0.062**
	σ	*0.76*	*0.01*	*0.02*	*1.09*	*0.03*	*0.02*	*3.11*	*0.03*	*0.02*	-	-	*0.04*	*0.008*
Multi-camera														
Cloud Likelihood	μ	-	-	3.01	-	-	1.34	-	-	7.45	3.43	0.36	0.79	0.605
	σ	-	-	*1.41*	-	-	*2.82*	-	-	*2.58*	*0.33*	*0.01*	*0.10*	*0.079*
Cloud Raycast	μ	-	-	3.32	-	-	1.61	-	-	4.27	**3.34**	0.49	0.99	1.686
	σ	-	-	*1.55*	-	-	*1.17*	-	-	*1.69*	*1.09*	*0.03*	*0.28*	*0.321*
Absolute FSD	μ	**10.77**	0.31	**0.50**	**11.02**	**0.35**	**0.28**	16.73	**0.32**	**0.40**	7.98	0.45	0.40	**0.064**
	σ	*1.73*	*0.02*	*0.03*	*0.66*	*0.01*	*0.02*	*1.11*	*0.03*	*0.05*	*1.41*	*0.06*	*0.01*	*0.008*
Interval FSD	μ	11.01	**0.29**	0.51	11.45	**0.35**	0.29	**14.34**	0.35	0.42	10.2	**0.35**	**0.39**	0.065
	σ	*0.15*	*0.01*	*0.02*	*0.37*	*0.01*	*0.05*	*0.90*	*0.02*	*0.05*	*0.11*	*0.02*	*0.01*	*0.008*
Single-camera														
Cloud Likelihood	μ	-	-	3.94	-	-	6.48	-	-	8.24	-	-	1.01	0.464
	σ	-	-	*0.78*	-	-	*1.09*	-	-	*3.29*	-	-	*0.05*	*0.097*
Cloud Raycast	μ	-	-	3.89	-	-	1.47	-	-	5.73	**4.86**	0.47	0.79	1.521
	σ	-	-	*3.36*	-	-	*0.88*	-	-	*1.66*	*2.94*	*0.07*	*0.09*	*0.296*
Absolute FSD	μ	-	-	2.07	29.91	0.33	**0.32**	-	-	6.27	9.67	0.44	0.59	**0.088**
	σ	-	-	*2.23*	*0.45*	*0.03*	*0.02*	-	-	*1.32*	*0.13*	*0.02*	*0.01*	*0.008*
Interval FSD	μ	**11.16**	**0.34**	**0.38**	**9.51**	**0.40**	0.33	**18.59**	**0.41**	**0.91**	10.04	**0.36**	**0.47**	**0.088**
	σ	*0.13*	*0.01*	*0.03*	*0.26*	*0.01*	*0.01*	*0.24*	*0.05*	*0.37*	*0.11*	*0.01*	*0.01*	*0.008*

TABLE I: Experiment Results. **SucD**: succeed distance (m); **PosEAC**: Position error after convergence (m); **FErr**: Final error (m). SucD and PosEAC only exist when FErr $< 1m$. Values in red: methods that did not converge. Bold: best results for each metric and dataset.

V. Conclusion

In this work, we present a localization strategy based on Free-Space Density (FSD) computed from RGB-D images. A robust way to compute spatial density from depth cloud data is proposed, minimizing the effects of movable objects in the environment. We also define an interval-based model that bounds the uncertainty of FSD in incomplete maps. Tests in domestic environments in single and multi-camera scenarios demonstrated the robustness of the proposed method. Additionaly, MCL with FSD does not require parameter tuning like other models[10]: the weights are given by direct comparisons of FSD.

It is important to note that the proposed approach handles well environments that contain unpredicted objects (which may lead to failures of methods based on scan matching), but it still needs to observe part of the structure of the environment (e.g. walls), otherwise there is no way of matching the FSD computed from observations and the reference FSD.

In the future, we plan to study other forms of computing spatial density from different sensors and applications. There is also the possibility of combining multiple kernels to compute FSD, as proposed in [7]. With the Interval FSD, this combination can be analyzed from the interval arithmetics point of view.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proc. of ICRA*. IEEE, 1999, pp. 1322–1328.

[3] 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 *Proc. of IROS*, 2015, pp. 3138–3143.

[4] S. Ito, F. Endres, M. Kuderer, G. D. Tipaldi, C. Stachniss, and W. Burgard, "W-rgb-d: Floor-plan-based indoor global localization using a depth camera and wifi," in *ICRA*. IEEE, 2014, pp. 417–422.

[5] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "A pose graph-based localization system for long-term navigation in cad floor plans," *Robotics and Autonomous Systems*, vol. 112, pp. 84–97, 2019.

[6] A. Ribacki, V. A. M. Jorge, M. Mantelli, R. Maffei, and E. Prestes, "Vision-based global localization using ceiling space density," in *Proc. of ICRA*, 2018, pp. 3502–3507.

[7] R. Maffei, V. A. M. Jorge, V. F. Rey, M. Kolberg, and E. Prestes, "Fast monte carlo localization using spatial density information," in *Proc. of ICRA*. IEEE, May 2015, pp. 6352–6358.

[8] F. Boniardi, T. Caselitz, R. Kümmerle, and W. Burgard, "Robust lidar-based localization in architectural floor plans," in *Proc. of IROS*, 2017, pp. 3318–3324.

[9] Z. Fang and S. Scherer, "Real-time onboard 6dof localization of an indoor mav in degraded visual environments using a rgb-d camera," in *Proc. of ICRA*. IEEE, 2015, pp. 5253–5259.

[10] M. F. Fallon, H. Johannsson, and J. J. Leonard, "Efficient scene simulation for robust monte carlo localization using an rgb-d camera," in *Proc. of ICRA*, 2012, pp. 1663–1670.

[11] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *Proc. of ICRA*. IEEE, 2012, pp. 1697–1702.

[12] F. Boniardi, A. Valada, R. Mohan, T. Caselitz, and W. Burgard, "Robot localization in floor plans using a room layout edge extraction network," in *Proc. of IROS*. IEEE, 2019, pp. 5291–5297.

[13] H. Chu, D. Ki Kim, and T. Chen, "You are here: Mimicking the human thinking process in reading floor-plans," in *Proc. of ICCV*. IEEE, 2015, pp. 2210–2218.

[14] O. Mendez, S. Hadfield, N. Pugeault, and R. Bowden, "Sedar - semantic detection and ranging: Humans can localise without lidar, can robots?" in *Proc. of ICRA*, 2018, pp. 6053–6060.

[15] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "Robust robot localization in a complex oil and gas industrial environment," *J. of Field Robotics*, vol. 35, no. 2, pp. 213–230, 2018.

[16] J. Borenstein and Y. Koren, "Histogramic in-motion mapping for mobile robot obstacle avoidance," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 535–539, Aug 1991.

[17] L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Applied Interval Analysis*. London: Springer, 2001.

[18] J. R. Ruiz-Sarmiento, C. Galindo, and J. González-Jiménez, "Robot@home, a robotic dataset for semantic mapping of home environments," *International Journal of Robotics Research*, 2017.

[10] Except for the kernel radius, which is standard in all scenarios