Sensor Localization by Few Distance Measurements via the Intersection of Implicit Manifolds

Michael M. Bilevich[1] Steven M. LaValle[2] Dan Halperin[1]

***Abstract*— We present a general approach for determining the unknown (or uncertain) position and orientation of a sensor mounted on a robot in a known environment, using only a few distance measurements (between 2 to 6 typically), which is advantageous, among others, in sensor cost, and storage and information-communication resources. In-between the measurements, the robot can perform predetermined local motions in its workspace, which are useful for narrowing down the candidate poses of the sensor. We demonstrate our approach for planar workspaces, and show that, under mild transversality assumptions, already two measurements are sufficient to reduce the set of possible poses to a set of curves (one-dimensional objects) in the three-dimensional configuration space of the sensor $\mathbb{R}^2 \times \mathbb{S}^1$, and three or more measurements reduce the set of possible poses to a finite collection of points. However, analytically computing these potential poses for non-trivial intermediate motions between measurements raises substantial hardships and thus we resort to numerical approximation. We reduce the localization problem to a carefully tailored procedure of intersecting two or more implicitly defined two-manifolds, which we carry out to any desired accuracy, proving guarantees on the quality of the approximation. We demonstrate the real-time effectiveness of our method even at high accuracy on various scenarios and different allowable intermediate motions. We also present experiments with a physical robot. Our open-source software and supplementary materials are available at https://bitbucket.org/taucgl/vb-fdml-public. This is an abridged version of a paper originally presented at ICRA 2023.**

I. Introduction

Localization can be carried out in various ways using a variety of sensors, which may be intrinsic (attached to the robot) or extrinsic (attached to the environment).

A common type of sensor for localization is distance-based, e.g., LiDAR. Many distance measurements do not only allow for localization inside a known environment but are also able to map an unknown environment. This is the task of the intensively investigated Simultaneous Localization And Mapping (SLAM, see [1] for a survey), usually by equipping the robot with either a LiDAR sensor [2] or with a camera [3]. But equipping each robot with a LiDAR sensor or a camera might be expensive both in the price of each sensor and the hardware required to process the information, as well as in the network throughput of sending, receiving, and processing large point clouds.

As we focus in this paper on distance measurement, we will refer to *sensor localization* for short, with the understanding that the sensor is geometrically a point that is rigidly attached to a robot. While our approach is general, we will analyze it in detail for a sensor attached to a mobile robot translating and rotating in the plane. Hence the configuration space of the sensor is three-dimensional and we represent each configuration as the triple (x, y, θ). We will use the robot motions as part of our solution technique.

Note that instead of moving the robot between measurements, one can alternatively fix the pose of the robot and measure simultaneously from a few distinct sensors mounted on the robot—each sensor's pose can be thought of as a translation and rotation relative to the robot's origin.

A. Related Work

1) Localization techniques: There are many approaches to localization. To localize in an already known environment, one can use particle filters and probabilistic methods [4], [5], [6]. We have already mentioned SLAM, which can be achieved for example by matching LiDAR samples [2] or by using a camera (known as visual SLAM, [3]). Recent works aim to speed up SLAM using GPUs [7], [8], [9]. See [1] for a more complete survey. One can also localize a robot by using RFID tags and sensors [10], or with RSSI signals [11].

2) Methods in meshing and implicit surface intersection: In this work we deal with the problem of meshing an implicit surface, which is finding an explicit representation for the surface defined as the zeros of some function $f : \mathbb{R}^3 \to \mathbb{R}$, i.e., the set $\{x : f(x) = 0\}$. This can be done in several ways, including the marching-cubes algorithm [12], [13] and dual contouring [14], which evaluate the function on some grid, or Delaunay refinement [15], which refines and filters a Delaunay triangulation of a three-dimensional point set. See [16] for a comprehensive survey.

We further deal with the intersection of implicit surfaces, which can also be carried out using a variety of techniques (see, e.g., [17], [18], [19]). In this paper, we will present methods that have provable guarantees on the error and convergence rate, when dealing specifically with robot/sensor localization.

B. Contribution

Our contributions are as follows:

[1]Blavatnik School of Computer Science, Tel-Aviv University, Israel. Work by M.B. and D.H. has been supported in part by the Israel Science Foundation (grant nos. 1736/19 and 2261/23), by NSF/US-Israel-BSF (grant no. 2019754), by the Israel Ministry of Science and Technology (grant no. 103129), by the Blavatnik Computer Science Research Fund, and by the Yandex Machine Learning Initiative for Machine Learning at Tel Aviv University.

[2]Center for Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland. Work by SL has been supported by a European Research Council Advanced Grant (ERC AdG, ILLUSIVE, 101020977) and the Academy of Finland (PERCEPT 322637).

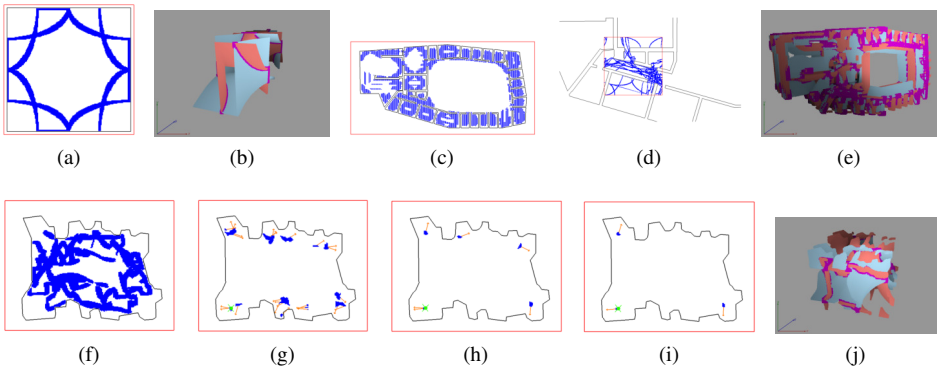


Fig. 1. Various screenshots from our software of the implicit manifolds described in Section III, in 3D space and projected onto 2D space. Figures 1a and 1b correspond to the scene `square-room`, Figures 1c,1d and 1e correspond to `floor-plan` and Figures 1f,1g,1h,1i and 1j correspond to `lab`, as described in Section IV.

- A general, efficient and robustly implemented method for sensor localization from few distance measurements.
- Theoretical guarantees on the quality of the approximation, with, for a parameter n chosen by the user, a linear bound $O(1/n)$ on the error, which is the distance between the best pose estimate and the true pose of the robot.
- An open-source code suite which utilizes GPU and CPU multi-core parallelism for computing the few-distance-measurement localization in real time.

We believe that our method is a deterministic novel outlook on the robot localization problem.

II. Problem Statement

The sensor is placed in the interior of a planar workspace $W \subseteq \mathbb{R}^2$. A distance measurement is a mapping

$$h : W \times \mathbb{S}^1 \to \mathbb{R}_+, \tag{1}$$

such that $h(x, y, \theta)$ is the length of the shortest segment connecting the position (x, y) with the boundary of W along a ray emanating from (x, y) in direction θ.

We are now ready to state the problem that we study.

The problem: Given a workspace W, a set $g_1, \ldots, g_k$ of rigid-body transformations, and a set $d_1, \ldots, d_k$ of positive real values, find all the poses (x, y, θ) such that $(x, y) \in W$ and $d_i = h(g_i(x, y, \theta))$ for all $i \in [1, k]$.

We aim to find the configuration (x, y, θ), which is the original pose of the robot. Before each one of the k measurements, the robot moves to another pose or stays put. The pose of the sensor when making the ith distance measurement is $g_i(x, y, \theta)$, where, as just stated, (x, y, θ) is the original pose of the robot, which we aim to find.

In this paper, we will deal with workspaces that have a polygonal boundary, namely polygons or polygons with holes. However, we believe that the techniques that we present here could be used for more involved workspaces for which we can evaluate the distance function $h(X)$, where X is the pose of the sensor. One such example for evaluating the distance function is computing $h(X)$ using a neural network representation, similar in spirit to [20],[21].

III. Voxel-Based Approximation: The Method and Its Guarantees

We now reformulate our problem so that it reduces to the intersection of implicit manifolds. We will compute these intersections numerically, by first bounding the manifolds inside sets of voxels.

A. Problem reformulation as an intersection of implicit manifolds

Given some measurement d, we wish to find its preimage [22] $h^{-1}(d)$, namely the set of poses (position and orientation) of a sensor placed inside W from which we can measure a distance d to a wall of W. We will denote this preimage by M_d. Formally:

Definition 3.1: Given some distance measurement d, its *preimage* is defined as

$$M_d = \left\{(x, y, \theta) : h(x, y, \theta) = d, (x, y) \in W, \theta \in \mathbb{S}^1\right\}. \tag{2}$$

Theorem 3.1: For any polygonal workspace and for any $d > 0$, the preimage M_d is the finite (possibly empty) union of manifold patches, namely

$$M_d = \bigcup_j M_d^j \,, \tag{3}$$

where each M_d^j is either a 2-manifold (without a boundary), a curve or a point.

Observation 3.1: The preimage M_d is the zero set of the following piecewise-continuous function

$$f_d(x, y, \theta) = h(x, y, \theta) - d \,, \tag{4}$$

and thus the preimage has an implicit representation.

We will use the straightforward mapping $\phi : \mathbb{S}^1 \to [-\pi, \pi)$ and embed our configuration space in $\mathbb{R}^3$, to simplify further discussions and analysis. We are now ready to cast our problem in terms of *implicit manifold intersection*:

Given a workspace W, a set $g_1, \ldots, g_k$ of rigid-body transformations $g_i : \mathbb{R}^3 \to \mathbb{R}^3$, and a set $d_1, \ldots, d_k$ of non-negative real values, find the intersection:

$$\bigcap_{i=1}^{k} M_{d_i} \ , \tag{5}$$

where M_{d_i} is the implicit manifold derived, as above, for measurement d_i after applying a transformation g_i with respect to the original pose of the robot:

$$M_{d_i} = (f_{d_i} \circ g_i)^{-1}(\{0\}) \ . \tag{6}$$

See Figure 1 for screenshots of produced with our implementation, which present 2D projections and 3D embeddings of the implicit manifolds and their intersections in various scenarios.

B. Overview of the method

We assume that we are given a set of measurements d_i with corresponding rigid-body transformations $g_i : \mathbb{R}^3 \to \mathbb{R}^3$. We are also given a parameter n, which determines the resolution and accuracy of our computation: we split a bounding cube of the configuration space into $n \times n \times n$ voxels. Our method comprises the following steps:

1) For each measurement d_i, we bound the two-dimensional preimage M_{d_i} inside a set of (three-dimensional) voxels, which we will refer to as a *voxel cloud*, i.e., a union of pairwise disjoint voxels, with an edge length of $\frac{1}{n}$. For each vertex in the voxel grid, we evaluate the distance function, and keep only voxels for which there is at least one vertex having a distance $\geq d_i$, and at least one vertex having a distance $\leq d_i$. We call this method *marching voxels*, as a simplified version of the marching-cubes algorithm [13].
2) We intersect two or more voxel clouds by keeping only the voxels that appear in all of the clouds.
3) If there are more than two preimages, then the resulting intersection should be a union of dot-like clusters of voxels (where each is purportedly representing a single point of intersection of manifolds), from which we can extract estimations for possible poses of the robot. Using some heuristic function (denoted $\mathcal{L}$) for giving worse score for outliers and improbable estimates, we can filter and clean our returned pose estimates.

We prove that this method is complete, in the sense that the true pose of the robot is contained in the voxel-cloud approximation. Additionally, we prove that the L_2 distance between the true pose of the robot and the closest estimation returned by the algorithm is bounded by $O\left(\frac{1}{n}\right)$. We also show that in general position, for distance errors bounded by at most $\varepsilon > 0$, we get that the closest estimation is bounded by $O\left(\frac{1}{n} + \varepsilon\right)$, i.e., we are robust to small bounded measurement errors.

We note that most operations presented are local; hence, parallelization (e.g., using GPUs) can be applied, as we demonstrate below in the experiments section.

IV. Experiments and Results

A. Implementation Details

The code is written as a C++ library and has Python bindings. We used OpenCL [23] to compute the marching voxels and the manifold intersection in parallel. The code was run on a macOS machine with Intel Core i5 CPU and Intel HD Graphics 6000 GPU, on a macOS machine with Apple M1 Pro and on a Windows machine with Intel Core i9-10850K CPU and NVIDIA RTX 3090 GPU. Running times for all systems are presented below. The parallelization of the marching voxels using a GPU was inspired by [24].

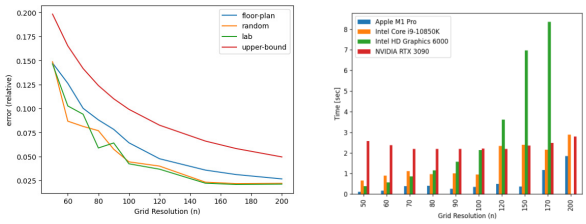
We demonstrate the performance of our algorithm on the following test scenes: `square-room`: A square room with dimensions of $2[m] \times 2[m]$; `non-convex-5gon`: A non-convex room with 5 vertices; `lab`: A polygon based on a LiDAR scan of our lab, bounded inside a $6.5[m] \times 5[m]$ rectangle `floor-plan`: Plan of the floor where our lab is located, bounded inside a $40[m] \times 20[m]$ rectangle; `random`: Randomly generated non-convex polygons, bounded inside a $2[m] \times 2[m]$ rectangle.

B. Error and Success Rate in Simulations

To test the error and success rate of our method, we performed the following simulation. Sample a random point p in some workspace $W \subseteq \mathbb{R}^2$ and a random orientation $\theta \in \mathbb{S}^1$. We call this pose the "ground truth". We then measure the distance to the walls from the point p with orientations $\theta + (i-1) \cdot \pi/4$ for $i = 1, \ldots, 6$ and random uniform offsets in x and y coordinates, to get measurement $d_1, \ldots, d_6$, and run our method to get a set S of pose estimations. We return the point $s_{\mathcal{L}}$ in S with the lowest value of $\mathcal{L}$, where $\mathcal{L}$ is the heuristic mentioned in Section III-B. We also return the point $s_{\mathcal{W}}$ in S whose Euclidean distance in $\mathbb{R}^3$ to the ground truth pose is minimal. If the set S is empty or for the returned pose the Euclidean distance in $\mathbb{R}^3$ to the ground truth is farther than twice the theoretical upper bound, we consider this as a failure. Otherwise, we measure the distance between the ground truth pose and the best estimation to get the approximation error.

We performed this experiment on the workspaces `lab`, `floor-plan` and `random`, for grid resolutions of $n \in \{50, 75, 100, 125, 150, 175, 200\}$. For each workspace and for each n, results were averaged over 100 trials. We measure the success rate for the estimation achieving the lowest value of $\mathcal{L}$, which we call the $\mathcal{L}$-success rate, and the success rate and error for the estimation with lowest Euclidean distance in $\mathbb{R}^3$ from the ground truth, which we call the $\mathcal{W}$-success rate and the $\mathcal{W}$-error rate respectively. The $\mathcal{L}$-success rate is the performance of our proposed method in practice, while the $\mathcal{W}$-success rate tests the completeness of the returned set of predictions. Error rate is the Euclidean distance in $\mathbb{R}^3$ to the ground truth, when the workspace W and $\mathbb{S}^1$ are both normalized to the unit cube $[0,1] \times [0,1] \times [0,1] \subseteq \mathbb{R}^3$ (so that the units are comparable).

Table I shows the average success rate for each workspace for both predictions, as well as the average number of pose



(a) (b)

Fig. 2. On the left: $\mathcal{W}$-Error rate as function of n, for different workspaces. The theoretical upper bound on the error $\mathrm{UB}(n)$ is also drawn for a reference. On the right: running time (in seconds) for different values of n for the `lab` workspace.

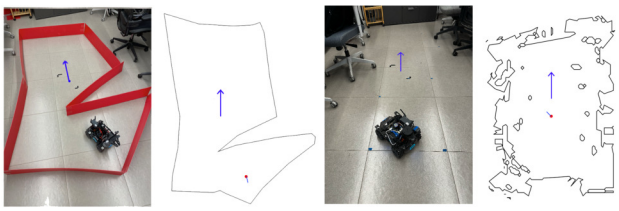


Fig. 3. Two examples of localization results in real life. Left: localization in a polygonal room bounded by cardboard (red) walls. Right: localization in the laboratory itself. In both examples the left column is a picture of the robot in the respective room, and the right column is a sketch of the corresponding best pose estimate returned by the algorithm. A blue arrow is drawn as a reference guide. The origin is marked with black tape on the floor.

estimates generated by the algorithm. The number of pose estimates is a combination of symmetry in the polygons and error incurred by our post-processing heuristic. As we can see, the consistent 100% $\mathcal{W}$-success rate suggests that our algorithm indeed satisfies completeness in the sense that the returned set of estimations also contains the ground truth estimation, justifying our post-processing heuristic. Note that the workspace `floor-plan` is highly symmetrical since many rooms are identical to each other, hence the lower success rate is expected for this scene.

Figure 2a shows the $\mathcal{W}$-error rate for each workspace as a function of n. For reference, we also present an upper bound, denoted by $\mathrm{UB}(n)$ based on the proved upper bound, which is $O(\frac{1}{n})$. As we can see, $\mathrm{UB}(n)$ is indeed an upper bound for the $\mathcal{W}$-error rate, hence the error rates seem to be $O\left(\frac{1}{n}\right)$. Figure 2b compares the running time for the `lab` workspace, for different values of n on the two different hardware systems. Running time for the other workspaces is similar.

C. Physical Robot Experiments

1) Error in localization: As a proof of concept, we also evaluated our method on a physical robot. We used a DJI RoboMaster EP Core, equipped with a distance sensor.

We conducted the following test: In the `lab` workspace (which is also a LiDAR scan of our lab), we chose a random point and asked the robot to advance to that point. We took $k = 3$ measurements with rotation of $\pi/4$ about the robot's center and taking measurements from the front and back of the robot, yielding overall 6 distance measurements. We compare the estimate returned by our algorithm which minimized the Euclidean distance to the ground truth pose. Averaging over 50 experiments, we achieve an average error of $0.0649[m]$ in the workspace (i.e., the Euclidean distance between the position $(x, y$ coordinates) of the estimate and the ground truth), and an average error of 0.0676 radians in orientation. We note that the LiDAR that we used to create the map has an error of up to $0.05[m]$.

We also created simpler polygonal rooms out of cardboard, placed the robot at random points and carefully measured its distance with respect to the origin. After averaging over 20 experiments, we achieve an average error of $0.026[m]$ in the workspace and 0.107 radians in orientation.

For both experiments we chose a resolution value of $n = 200$, which yields precision of at most $0.041[m]$ for the x, y coordinates and 0.0314 radians, as those are the dimensions of the voxels we take.

See Figure 3 for examples of robot placement in these experiments and the pose estimates returned by the algorithm.

2) Comparison against SLAM: As a comparison, we carried out the following experiment: We placed the robot at pre-determined landmarks in the room for which we measured their location manually. We then queried the pose from our algorithm and the pose estimate resulting from the SLAM computation of Google Cartographer [25]. Finally, we measured the Euclidean distance between each pose estimate from the ground truth (the true location of the landmark), and the Euclidean distance between both estimates as well. Averaging on 10 experiments, the Google Cartographer had an error of 0.02377[m] and our method had an error of 0.06[m]. The average Euclidean distance between both pose estimates is 0.05744[m].

We note that both methods incorporate the robot's odometry, which is error prone. However, Google Cartographer also used the robot's IMU.

TABLE I

SUCCESS RATES FOR $k = 6$, FOR DIFFERENT WORKSPACES.

Workspace	$\mathcal{L}$-Success Rate	$\mathcal{W}$-Success Rate	# Estimates
`random`	90.14%	100%	31.92
`lab`	95.28%	100%	10.73
`floor-plan`	80.43%	100%	88.77

V. CONCLUSIONS AND FURTHER WORK

In this work we have presented a numerical method for estimating the unknown pose of a robot inside a known environment using only a few (2 to 6) distance measurements. We show that our method returns conservative results, and can be applied in real-life situations.

In future work, we would like to improve the accuracy and success rate in practice, deal with stochastic errors and errors in odometry, and extend our method to higher dimensions. A related problem that may be solved using similar techniques is finding the optimal motions that the robot needs to perform to reduce the volume of the possible poses.

REFERENCES

[1] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A survey on active simultaneous localization and mapping: State of the art and new frontiers," 2023.

[2] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, "A comparative analysis of LiDAR SLAM-based indoor navigation for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6907–6921, 2022.

[3] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, "Visual and visual-inertial SLAM: State of the art, classification, and experimental benchmarking," *Journal of Sensors*, vol. 2021, 2021.

[4] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, pp. 1322–1328 vol.2.

[5] M. Speekenbrink, "A tutorial on particle filters," *Journal of Mathematical Psychology*, vol. 73, pp. 140–152, 2016.

[6] X. Wang, T. Li, S. Sun, and J. M. Corchado, "A survey of recent advances in particle filters and remaining challenges for multitarget tracking," *Sensors*, vol. 17, no. 12, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/12/2707

[7] T. Ma, N. Bai, W. Shi, X. Wu, L. Wang, T. Wu, and C. Zhao, "Research on the application of visual SLAM in embedded GPU," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

[8] M. Abouzahir, R. Latif, M. Ramzi, and M. Sbihi, "OpenCL and OpenGL implementation of simultaneous localization and mapping algorithm using high-end GPU," in *ITM Web of Conferences*, vol. 46. EDP Sciences, 2022, p. 04001.

[9] R. Mittal, V. Pathak, and A. Mithal, "A novel approach to optimize SLAM using GP-GPU," in *Proceedings of International Conference on Data Science and Applications*. Springer, 2021, pp. 273–280.

[10] A. Motroni, A. Buffi, and P. Nepa, "A survey on indoor vehicle localization through RFID technology," *IEEE Access*, vol. 9, pp. 17 921–17 942, 2021.

[11] S. R. Jondhale, R. Maheswar, and J. Lloret, "Survey of existing RSSI-Based L&T systems," in *Received Signal Strength Based Target Localization and Tracking Using Wireless Sensor Networks*. Springer, 2022, pp. 49–64.

[12] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. CRC Press, 2010. [Online]. Available: https://books.google.co.il/books?id=AuXqBgAAQBAJ

[13] E. Chernyaev, "Marching cubes 33: Construction of topologically correct isosurfaces," Tech. Rep., 1995.

[14] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 339–346.

[15] J.-D. Boissonnat and S. Oudot, "Provably good sampling and meshing of surfaces," *Graphical Models*, vol. 67, no. 5, pp. 405–451, 2005, solid Modeling and Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070305000056

[16] J.-D. Boissonnat, D. Cohen-Steiner, B. Mourrain, G. Rote, and G. Vegter, *Meshing of Surfaces*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 181–229. [Online]. Available: https://doi.org/10.1007/978-3-540-33259-6_5

[17] S. Tanaka, Y. Fukuda, and H. Yamamoto, "Stochastic algorithm for detecting intersection of implicit surfaces," *Computers & Graphics*, vol. 24, no. 4, pp. 523–528, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849300000558

[18] O. Caprani, L. Hvidegaard, M. Mortensen, and T. Schneider, "Robust and efficient ray intersection of implicit surfaces," *Reliable Computing*, vol. 6, no. 1, pp. 9–21, 2000.

[19] X. Li, H. Jiang, S. Chen, and X. Wang, "An efficient surface–surface intersection algorithm based on geometry characteristics," *Computers & Graphics*, vol. 28, no. 4, pp. 527–537, 2004.

[20] T. Davies, D. Nowrouzezahrai, and A. Jacobson, "On the effectiveness of weight-encoded neural implicit 3D shapes," *arXiv preprint arXiv:2009.09808*, 2020.

[21] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," 2019.

[22] S. M. LaValle *et al.*, "Sensing and filtering: A fresh perspective based on preimages and information spaces," *Foundations and Trends® in Robotics*, vol. 1, no. 4, pp. 253–372, 2012.

[23] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in Science & Engineering*, vol. 12, no. 3, pp. 66–73, 2010.

[24] J. Lei and K. Jia, "Analytic marching: An analytic meshing solution from deep implicit surface networks," 2020. [Online]. Available: https://arxiv.org/abs/2002.06597

[25] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.