Distributed Dual-Resolution 3D Scene Representations

Colton Stearns, Ian Huang, Congyue Deng, Archit Sharma
{coltongs, ianhuang, congyue, architsh}@stanford.edu

Abstract

We address the problem of autonomous 3D mapping via a group of autonomous agents. Given n autonomous agents that continuously perceive the surrounding environment, we propose Discern3D (Distributed Scene Representation in 3D), a protocol for creating a course scene representation for autonomous navigation in addition to a fine scene representation for HD mapping. Discern3D is designed to (1) be highly available for navigation, (2) reduce network traffic between agents, and (3) be fault tolerant to agent failure and network partition. In contrast to classical distributed storage and replicated state machines, Discern3D leverages the structure inherent in 3D data to increase availability and reduce network traffic. We introduce a new consistency model which we call eventual geometric consistency; we argue that this weaker consistency guarantee is sufficient for online 3D scene representation.

1. Introduction

In recent years, high quality 3D scene reconstruction and mapping has been used for a variety of applications, including HD mappings of cities for autonomous vehicles, reconnaissance of disaster-stricken sites, creating digital twins for construction, and collecting 3D data for training neural networks. Inspired by recent advances in computer vision, many researchers are attempting to automate the process of 3D scene reconstruction and mapping with fleets of robots [3, 16]. As two motivating examples, fleets of autonomous vehicles simultaneously navigate and update 3D city maps, and unmanned aerial vehicles (UAVs) are popular tools for scanning and reconstructing remote areas.

A distinct challenge in using fleets of autonomous agents for 3D mapping is the low reliability in agent communication and liveness. Inter-agent communication is often essential for safe planning. However, the real-world navigation of affordable robots through cluttered areas results in frequent network delays, network partitions, and agent failures. A thorough distributed protocol is necessary to ensure both availability and fault-tolerant replication of data across agents.

Given n agents, a replicated state machine with a consensus protocol [4, 11, 18] guarantees fault-tolerance for up to $\lfloor n/2 \rfloor$ agent failures, in addition to ensure that all agents path-plan with the same 3D information. However, guaranteeing strong consistency across all agents jeopardizes availability and could lead to unsafe planning. Furthermore, storing all data on all n agents is often unnecessary. Replicated storage systems [9, 12, 24] can provide the greater flexibility to clone data on and communicate data to appropriate subsets of agents. However, replicated storage systems often provide atomic or linearizable consistency guarantees and allow data in the form of an arbitrary file system or SQL table. Such features are unnecessary for 3D data, adding complexity and overhead. Finally, distributed key-value storage systems [6] can offer highly available replication. While these systems are a reasonable choice for 3D scene replication, their system design is often optimized for a very large number of agents storing many small-sized key-value pairs. In distributed scene representations, there often exist fewer agents that each scan and store dense 3D data.

In this work, we propose Discern3D (Distributed Scene Representation in 3D), a protocol for distributed 3D scene sharing and replication in the face of unreliable network and unpredictable agent liveness. Discern3D disentangles 3D data *sharing* from data *replication*. Specifically, Discern3D uses a highly available low-resolution *coarse scene representation* for sharing data essential for agent navigation. Then, to provide fault-tolerance of the 3D mapping, Discern3D replicates a high-resolution *fine scene representation* on a subset of agents. In addition, Discern3D makes use of a new consistency model which we denote *eventual geometric consistency*. Unlike traditional data types, 3D information is order-invariant with respect to when it is received and may be redundant in the underlying geometric information. Eventual geometric consistency uses these characteristics by weakening constraints on transaction ordering (and some transactions altogether) while still providing eventual convergence of the replicated 3D information.

In summary, we contribute: (i) a novel protocol for distributed 3D scene representation, (ii) a new consistency model for geometric data, and (iii) experimental analysis of our protocol on many example shapes, under different

network conditions, and with different replication requirements.

2. Background

In this section, we provide additional background of multi-agent logic 3D scene representations. We assume the reader is familiar with distributed storage protocols.

Cooperative multi-agent planning. Distributed protocols for cooperative planning are of great interest in safe robot-fleet navigation [20, 25, 28]. The goal of cooperative robot planning is to efficiently share information to improve agent planning with respect to a global environment. In distributed SLAM, agents transmit sensor readings with for localizing into an common reference frame [10]. In distributed path planning, agents are tasked with simultaneously routing to many target destinations; many works have focused on extend classic single-agent heuristics to the distributed setting [17, 21, 22]. Beyond SLAM and planning, recent works address the problem of forming agent coalitions in a distributed manner, targeting agent collaboration [1, 2, 15, 19]. In contrast to multi-agent planning, we focus on the less-studied *upstream* task of shared 3D scene reconstruction. We assume a known the global reference frame and a downstream path planning module.

HD mapping. High definition (HD) mapping is gaining increasing attention in 3D navigation which provides high quality maps with both global accuracy and local accuracy. Key to the acquisition of HD maps is the coordination between an array of sensors where the construction of an appropriate distributed systems can be of great help. Many works have been studying the distribution of HD maps to individual agents [5, 13, 14, 26, 27]. [14] improves transmission efficiency by leveraging multiple communication schemes. [5, 26] adopts named data networking (NDN) to support agent mobility. [13] introduces a fast content delivery mechanism using multiple data sources and a probabilistic handover strategy. RLSS [27] attempts to find the best agent or nearby tower for querying an existing HD map. On the other hand, works are also done on bringing more intricate geometric data into distributed HD maps. [8] builds a TSDF scene representation by fusing a collection of local sub-maps kept by each agent and [7] further introduces decentralization. However, to the best of knowledge, we are the first to consider significant network unreliability and agent failure under more realistic assumptions.

3. Scene Representation

Discern3D is a protocol for *fault tolerant* and *highly efficient* distributed scene representation across n agents that actively perceive the surrounding 3D environment. Specifically, Discern3D addresses the task of high-fidelity autonomous scene acquisition, which is at the core of many tasks such as urban HD mapping, construction surveillance, topographic surveying, machine-learning training data, and more. In high-fidelity autonomous scene acquisition, agents simultaneously (1) acquire high-fidelity 3D information and (2) use acquired scene information for downstream navigation, i.e. to avoid obstacles and to plan appropriate scanning routes. To improve the efficiency of scene representation, Discern3D makes use of a *dual-resolution* scene representation (both low resolution and high resolution), where the low-resolution scene is used for navigation.

As Discern3D addresses the use case of robot fleets for 3D mapping, the protocol is motivated by many task-based assumptions. First, Discern3D assumes reasonable network connectivity between agents, and no network connectivity with an external or centralized server. Additionally, while we assume reasonable inter-agent connection, we also assume that the network bandwidth will observe throughput limitations and can easily become a bottleneck. As a motivating example, a modern Lidar scanner acquires megabytes of 3D data per second per agent. In addition to throughput limits, we also assume that network partitions and agent failure are very common. We argue this a reasonable assumption given the complexity of real-world environments. Finally, we assume that high-availability of scene data is essential for safe navigation. Given these assumptions, we build Discern3D to value *availability* of the necessary 3D scene information, *fault tolerance* towards network partition and agent failure, *network efficiency* (i.e. minimizing network load).

3.1. Scene Voxel Representation

We represent a 3D scene as a coarse voxel grid, i.e. a grid of evenly spaced cubes marked with xyz coordinates. We note that such a grid can always exist by using global (lat, lon, z) coordinates in the locally planar reference frame and that voxel size can vary depending on application. As time progresses, voxels are populated with fine-grained 3D data in the form of a second voxel grid that lies within the original voxel. Concretely, in addition to it's xyz coordinate in 3D space, a voxel is associated with an append-only file that grows as agents acquire more and more fine-grained 3D information. Thus, each voxel can be viewed as a primitive data "chunk". Moving forward, we refer to these append-only files as *voxel files*.

In contrast to more general distributed storage systems, Discern3D only handles the storage of a voxel grid across all agents. Because it specializes in this single datatype, it is able to leverage the following property:

Property. *The sequence of data within a voxel file is order-invariant. Also, adding data to a sequence results in having weakly more 3D information. Therefore, updating a voxel file will NEVER have negative consequences.*

This suggests that commonly-desired guarantees such as linearizability and atomicity are of little importance for our 3D scene representation. Furthermore, it suggests that our voxel-files do not need traditional consistency guarantees; instead, we are satisfied with an order-invariant consistency that only ensures underlying *geometric equivalence*.

3.2. Voxel Partitioning and Replication

We represent our scene as a grid of high-resolution voxel-files that are replicated on autonomous agents. To achieve fault tolerance to agent failure without significant network overhead, we replicate voxel-files across a subset of K agents.

Instead of using methods such as consistent hashing [6] to statically assign each voxel to a subset of K agents for replication, we propose that the assignment be done dynamically given the locations and trajectories of each agent. At every timestep, each agent maintains a dynamic preference list mapping from each voxel ID to a subset of all agents. Setting agent preferences dynamically allows us to lower the network load, as we will analyze in Section 4.5.

In order to ensure eventual consensus on which K agents replicate a voxel, the dynamic preference-list must converge to a globally-accepted total ordering. to reach total ordering in Discern3D, we use timestamps of initial data acquisition in addition to a gossiping protocol. We will further discuss this is Sec. 4.3.3.

3.3. Low Resolution Representation

While the high-resolution storage strategy in Sec. 3.2 is an efficient and fault-tolerant storage scheme, it overlooks the fact that all agents need *high availability* of scene information for downstream navigation. Noting that modern 3D vision methods inherently operate on lower-resolution or abstracted scene representations, we use a low-resolution scene representation to address this need. Concretely, Discern3D uses the abstracted low-resolution voxel-grid as a light-weight scene representation to be replicated across all agents. While this low-resolution representation can be any arbitrary 3D representation, we choose to use the original coarse-resolution voxel grid in our implementation.

4. Agent Coordination

In this section, we cover the symmetric protocol deployed on each agent in a fleet of robots.

4.1. Agent Behavior and State

At any given timestamp, every agent scans a part of the 3D environment. Each scan is decomposed into data chunks belonging to a set of low-resolution voxels, and each low-resolution voxel is assigned a separate preference list of the K agents that should store the data. Depending on whether the agent is or is not one of the K storing agents, it either transmits data to the other $K-1$ agents and keeps a copy for itself or transmits to the K appropriate agents and deletes the copy.

In addition to transmitting high-resolution data, the agent must have a refreshed version of the low-resolution (coarse) representation of the entire scene; this is crucial for downstream tasks like navigation.

In order to carry out the above actions, each agent stores the following in memory:

1. R: the preference list, where $R[C, i]$, for some coarse voxel id C and agent i would give the timestamp when agent i first attained data from voxel C. $R[C, i]$ is initialized to positive infinity at start time.
2. D: a database of high-resolution data per voxel.
3. F: a temporary storage space of voxel data that is attained by an agent, but is not yet clear if the agent is the end-destination for the data.
4. L: a low resolution voxel representation of the high-resolution data.

4.2. Agent RPC's

Discern3D is a symmetric protocol consisting of the following remote procedure calls:

1. **`GET`**: retrieves the coarse representation stored at an agent at a queried coarse voxel ID.
2. **`GET_ALL`**: retrieves the coarse representation stored at an agent across all coarse voxels ID's.
3. **`GET_PREFERENCE_LIST`**: retrieves the preference list of a remote agent at its reference timestamp.
4. **`UPDATE`**: sends a remote agent a set of high-resolution scans.

4.3. Agent Implementation

An agent concurrently scans, stores, shares and updates information. To achieve this, we implement the following procedures as concurrent threads running at fixed frequencies.

4.3.1 High-Resolution (Fine) Representation Update

The agent constantly scans the environment. Let C_{ID} be the coarse voxel ID, P_{ID} be the packet ID, X be the high-resolution voxel data, and t be a timestamp of the time of scan. Whenever the agent acquires (C_{ID}, P_{ID}, X, t), it enqueues it onto a packet queue Q. The same is done when the agent receives (C_{ID}, P_{ID}, X, t) from any other agent through calls of the `UPDATE` RPC; however, in this case t is the time of message receipt and not of the original scan. As such, the elements on the data queue grows both from

the scans acquired by the native agent as well as from messages sent by other agents. When Q is not empty, the fine representation update procedure is triggered. This procedure:

1. Dequeues a data tuple from Q.
2. Compares the timestamp t of the dequeued data tuple with the timestamp in the current preference-list value at $R[C_{ID}, i]$ (where i is the agent's own index). If t indicates is a lower value than that recorded in R, then (C_{ID}, t, i) is added to G, the gossip Queue (used in section 4.3.3).
3. Transfers the fine representation X to temporary storage F within the the agent's memory for C_{ID}.
4. It keeps X in F until conditions allow for 1) packet replication to other agents (by calling the `UPDATE` RPC's) or 2) transferal of X from the temporary storage to the final storage in D.

The decision to move the data off of temporary storage F is made based on R. If an agent, through looking at the timestamps in R, decides with certainty that it is not the first K (when sorting agents by ascending timestamps for that voxel id), it will attempt to send the data to the first K agents. Otherwise, it keeps the data in temporary storage F until the timestamps in R indicates that it has to be one of the top K. Regardless of its own placement in R, the agent will keep the data in temporary storage F until it receives acknowledgement of transmission from all K top agents (perhaps including itself).

4.3.2 Low-Resolution (Coarse) Reprentation Update

A coarse representation is attained at a fixed frequency by calling the `GET_ALL` RPC. This simply aggregates the coarse voxels stored at each agent with its own, by taking a boolean OR on its own L and the L of another agent. Even though our current implementation does not use it, we additionally note that a user could create more intricate logic for the coarse representation updates using the `GET` RPC (e.g. only get data along a planned trajectory).

4.3.3 Preference-List Gossiping Update

Ensuring that agents have an updated and synchronized version of the preference list R is at the core of Discern3D's efficiency. For instance, for newly attained voxel data, R informs agents of whether to store it or pass it to another agent.

We implement gossiping between agents to reach eventual consistency on the preference list R. A thread iteratively calls the `GET_PREFERENCE_LIST` RPC on other agents, and merges their preference lists into R. The merging of R is done by the timestamp of $R[C_{ID}, i]$ and $R'[C_{ID}, i]$, where R' is the preference list of another agent. Effectively, $R[C_{ID}, i] := \min(R[C_{ID}, i], R'[C_{ID}, i])$. Since R stores timestamps (for which we can establish total ordering), gossiping and merging preference lists provides eventual consistency of R.

In addition to updating R via the `GET_PREFERENCE_LIST` call, the gossip procedure actively updates R with values dequeued from G, a queue that records all newly attained timesteps of data.

4.4. Consistency Model

Discern3D guarantees a type of eventual consistency which we denote *eventual geometric consistency*. Specifically, Discern3D guarantees eventual geometric consistency for all high-resolution data replicated across the subset of K agents in addition to the fully-replicated coarse scene representation. In contrast to other (stronger) forms of consistency such as eventual consistency, causal consistency, strong consistency, and others [23], *eventual geometric consistency* guarantees that, as $t \to \infty$, a sequence of append-operations will be equivalent up to a permutation in the order.

Discern3D makes use of a total ordering on timestamps of the preference list to ensure eventual geometric consistency. That is, by dynamically setting preference-lists based on loosely synchronized timestamps, the ordering of all preference lists is guaranteed to converge. Due to the global convergence of which K agents receive data, we can guarantee that all data will eventually reach each agent (even if the order of transmission is unknown).

4.5. Network Load with a Dynamic Preference-List

Assume we have N agents, and a requirement that each scan needs to be duplicated on K agents. Assume that there are V coarse voxels and for a sufficient quality of the scan, our system needs M scans of each of the voxels. For comparison, we consider a static preference list, where instead of using a dynamic list, we pre-allocate K agents per voxel.

In expectation, the following number of data packets will be sent over the network:

$$VMK\left(\frac{N-1}{N}\right)$$

In contrast, with our dynamic assignment based on which agents have scanned the region, the total number of data packets sent over the network is:

$$V(\min(M,K)(\max(M,K)-1))$$

As such, the number of data packets removed from the total network traffic, in expectation, is:

$$V\left(\frac{N\min(M,K)-MK}{N}\right)$$

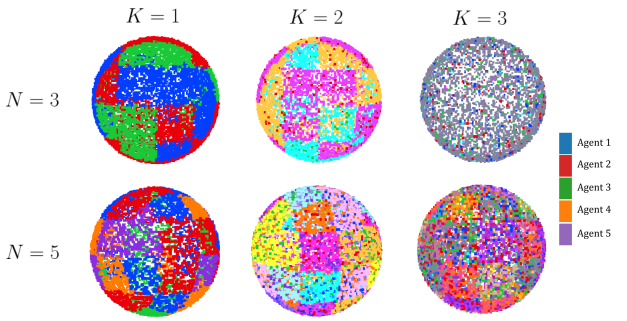

Figure 1. Twenty seconds of scanning a sphere using the Discern3D protocol with differing numbers of agents, N, and varying replication requirements, K. We appropriately observe agent colors mixing as we require a higher number of replicas. We additionally observe clear edges where the 3x3 coarse voxel-grid partitions the sphere.

This shows great utility in our dynamic preference list when the total number of coarse voxels V is large, and when the number of agents N is far greater than M and K. Concretely, dynamic preference lists can save significant network traffic over static lists when there is a need to scan a large area with a large number of agents (i.e. the scan quality may be high s.t. M is small and the reliability of agents is decently high s.t. K is small).

5. Experiments

5.1. System Behavior in Reliable Conditions

Although Discern3D is designed for conditions with network partitions and agent failure, we first verify its efficient behavior in the "ideal case", i.e. without network partitions or agent failure. As a toy example of the protocol, we scan a sphere with varying numbers of agents and varying replication requirements for the high-resolution representation. For visual clarity, we set an extremely coarse 3x3 grid as the low-resolution representation. We showcase Discern3D's high-resolution representation after 20 seconds in Fig. 1. As depicted, the trivial case of $K = 1$ replication yields a simple square-like partition of the data across each agent. As the data is replicated across more agents (greater K), we observe mixed colors in addition to heterogeneous specs where the data is not yet fully replicated.

In addition to the sphere, we test Discern3D on a variety of 3D objects. Fig. 4 illustrates the scanning of an airplane and car over time. As depicted, continuing to scan the object results in convergence of the coarse representation and appropriate replication of fine representation

5.2. System behavior under network partitions

Network partitions is especially important for our use-case, as agents navigating the real world environments may

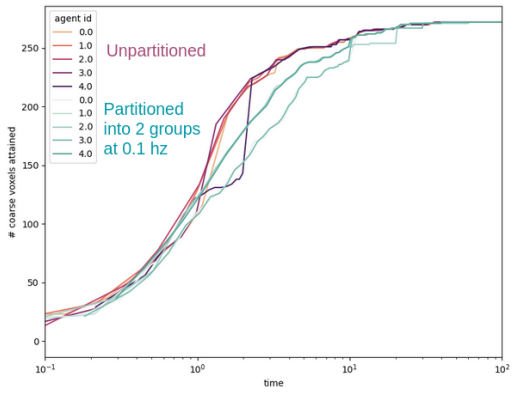

Figure 2. **Coarse (low-resolution) data acquisition rate affected by network partitions.** We show the number of coarse voxels attained by all agents for two different conditions: 5 agents in the absence of network partitions (purple), and 5 agents in the presence of network partitions refreshed at 0.1 Hz (cyan). This shows that the availability of coarse representation is more robust to network partitions, unlike the high-resolution representation shown in Figure 3. Time in log-scale to magnify the difference between the two conditions.

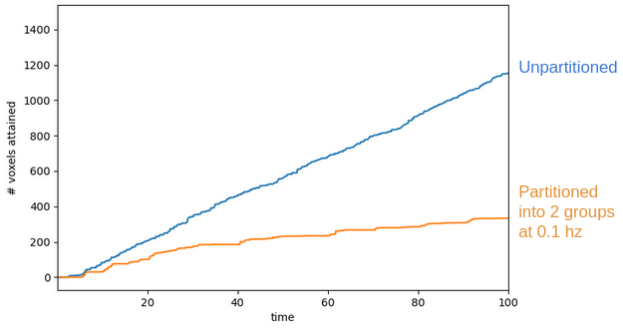

Figure 3. **High-resolution data acquisition rate with (orange) and without (blue) network partitions.** Unlike the coarse representation (Figure 2), the fine representation here is less robust to network partitions.

naturally suffer from partitions given how pairwise signals between agents may be completely blocked by the object being scanned as well as the surrounding environment.

To investigate the behavior of our system under different network partitions, we assume that the network can have up to P partitions at any point in time. Each partition is allowed to communicate with one another (e.g. for sending and receiving point packets or allocation rosters), but no information can be transmitted between groups. We assume that the sampled partition at each step is independent of that of the previous step. During our experiments, the frequency at which the network is re-partitioned is a hyperparameter that indicates the length of partitions. In practice, this frequency can vary drastically. The lower the frequency, the longer network partitions may last, which can hinder gos-

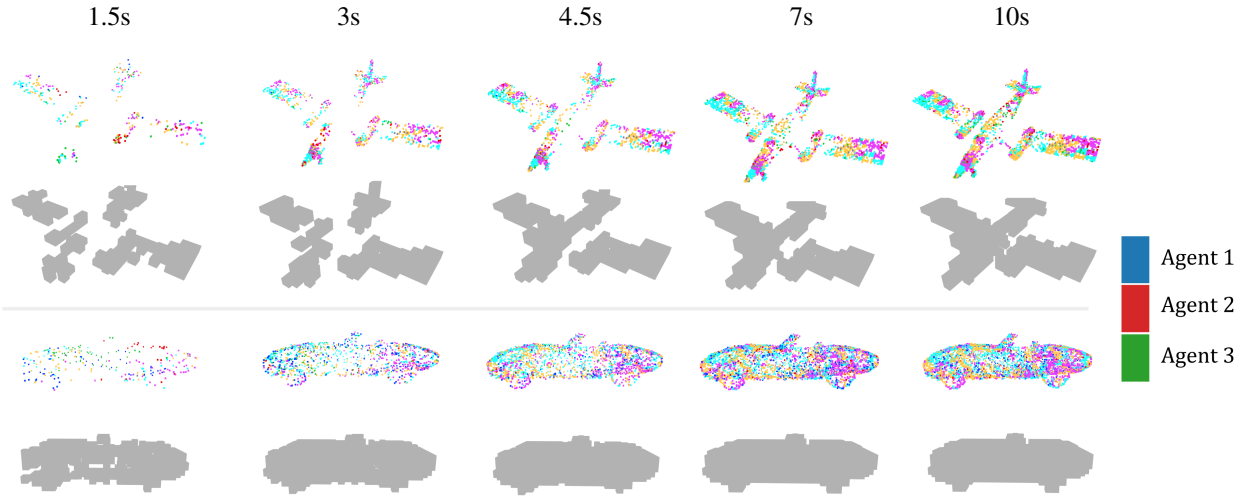

Figure 4. Two examples of the fine representation (upper row) and coarse representation (bottom row) produced by the Discern3D protocol. Points in the fine representation are colored by the agent(s) that store the data. We visualize the coarse representation of a single agent. Both representations monotonically grow as more data is scanned.

siping, as well as data replication to the top K agents.

Figures 2 and 3 show the behavior of the system with 5 agents, $K = 2$, with the network partition refresh rate at 0.1 Hz. Figure 2 shows that across the 5 different agents in two different situations, the rate at which coarse representations is being acquired of the world is approximately same. However, this is not the case for high-resolution representations, as can be seen in Figure 3. This is expected, and justifies why agents should maintain data of the world represented at two different resolutions. The data which needs to be highly available is the coarse representation, as it may be important for mission critical time-sensitive tasks like navigation. The design of our protocol for syncing coarse representations across agents allows for some robustness to network partitions.

Figure 5 shows the behavior of the system under different time duration of each network partition, controlled by the network partition refresh rate. For a system of 5 agents, with $K = 2$, a doubling in the network partition duration from 0.1 Hz to 0.05 Hz decreases the rate of high-resolution data collection by less than that factor. The 5x increase in network partition duration (from 0.1 Hz to 0.02 Hz) similarly does not lead to a 5x decrease in data collection rate. This shows that because of the gossiping protocol for the preference list allows the rate of data collection to decrease sub-linearly to the length of the network partitions, which is a desirable property to have. In the presence of intensely long network partitions (0.01 Hz), however, the agents may plateau in the amount of data collected for the first 100 seconds before the next network partition refresh happens (which, likely allows any agent to fall into a group with a different set of agents, allowing the gossiping and data replication to continue). A version of this behavior can also be seen for the 0.02 Hz and 0.05 Hz lines in Figure 5.

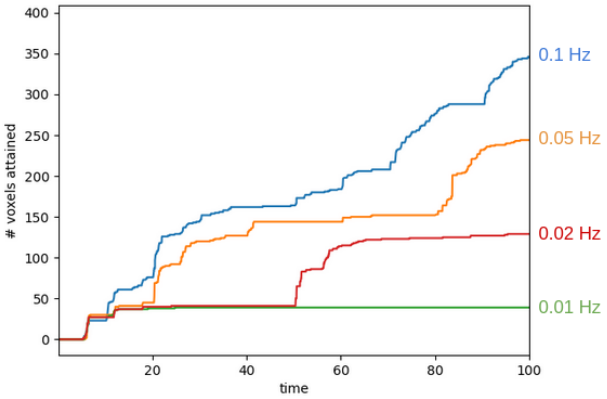

Figure 5. **Effect of network re-partition frequency on high-resolution data acquisition rate.** Lower network re-partition frequencies adversely affect the rate at which fine-grained voxels are accumulated in the shared high-resolution representation.

6. Limitations

Discern3D is a first step toward robust distributed scene representation in the face of network uncertainty and agent failure. In future works, we hope to build a more scalable protocol for syncing preference lists (gossiping has many drawbacks). In addition, we are interested in extending the low-resolution scene protocol to go beyond simple queries of surrounding agents.

References

[1] Manoj Agarwal, Nitin Agrawal, Shikhar Sharma, Lovekesh Vig, and Naveen Kumar. Parallel multi-objective multi-robot coalition formation. *Expert Syst. Appl.*, 42:7797–7811, 2015. 2

[2] Manoj Agarwal, Lovekesh Vig, and Naveen Kumar. Multi-objective robot coalition formation for non-additive environments. In *ICIRA*, 2011. 2

[3] Siddharth Agarwal, Ankit Vora, Gaurav Pandey, Wayne Williams, Helen Kourous, and James R. McBride. Ford multi-av seasonal dataset. *CoRR*, abs/2003.07969, 2020. 1

[4] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association. 1

[5] Rodolfo WL Coutinho, Azzedine Boukerche, and Antonio AF Loureiro. Design guidelines for information-centric connected and autonomous vehicles. *IEEE Communications Magazine*, 56(10):85–91, 2018. 2

[6] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. *ACM SIGOPS operating systems review*, 41(6):205–220, 2007. 1, 3

[7] Rodolphe Dubois, Alexandre Eudes, Julien Moras, and Vincent Frémont. Dense decentralized multi-robot slam based on locally consistent tsdf submaps. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4862–4869. IEEE, 2020. 2

[8] Thibaud Duhautbout, Julien Moras, and Julien Marzat. Distributed 3d tsdf manifold mapping for multi-robot systems. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2019. 2

[9] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, page 29–43, New York, NY, USA, 2003. Association for Computing Machinery. 1

[10] Andrés Jiménez, Vicente García Díaz, Ruben Gonzalez Crespo, and Sandro Castro. Decentralized on-line simultaneous localization and mapping for multi-agent systems. *Sensors*, 18:1–16, 08 2018. 2

[11] Leslie Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. 2019. 1

[12] Barbara Liskov, Sanjay Ghemawat, Robert Gruber, Paul Johnson, Liuba Shrira, and Michael Williams. Replication in the harp file system. *ACM SIGOPS Operating Systems Review*, 25(5):226–238, 1991. 1

[13] Jialun Lu, Wang Yang, and Fan Wu. High definition map distribution in named data networking based vanets. In *2020 3rd International Conference on Hot Information-Centric Networking (HotICN)*, pages 129–134. IEEE, 2020. 2

[14] Guiyang Luo, Quan Yuan, Haibo Zhou, Nan Cheng, Zhihan Liu, Fangchun Yang, and Xuemin Sherman Shen. Cooperative vehicular content distribution in edge computing assisted 5g-vanet. *China Communications*, 15(7):1–17, 2018. 2

[15] Petra Mazdin and Bernhard Rinner. Distributed and communication-aware coalition formation and task assignment in multi-robot systems. *IEEE Access*, 9:35088–35100, 2021. 2

[16] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: A review. *Applied Geomatics*, 6, 03 2014. 1

[17] Raz Nissim and Ronen I. Brafman. Multi-agent a* for parallel and distributed systems. In *AAMAS*, 2012. 2

[18] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (Usenix ATC 14)*, pages 305–319, 2014. 1

[19] Amit Rauniyar and Pranab K. Muhuri. Multi-robot coalition formation and task allocation using immigrant based adaptive genetic algorithms. 2020. 2

[20] Yara Rizk, Mariette Awad, and Edward W. Tunstel. Cooperative heterogeneous multi-robot systems. *ACM Computing Surveys (CSUR)*, 52:1 – 31, 2019. 2

[21] Michal Stolba, Daniel Fiser, and Antonín Komenda. Admissible landmark heuristic for multi-agent planning. In *ICAPS*, 2015. 2

[22] Michal Stolba and Antonín Komenda. The madla planner: Multi-agent planning by combination of distributed and local heuristic search. *Artif. Intell.*, 252:175–210, 2017. 2

[23] Doug Terry. Replicated data consistency explained through baseball. *Commun. ACM*, 56(12):82–89, dec 2013. 4

[24] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In *SIGMOD 2017*, 2017. 1

[25] Janardan Kumar Verma and Virender Ranga. Multi-robot coordination analysis, taxonomy, challenges and future scope. *Journal of Intelligent & Robotic Systems*, 102, 2021. 2

[26] Jingjing Wang, Chunxiao Jiang, Zhu Han, Yong Ren, and Lajos Hanzo. Internet of vehicles: Sensing-aided transportation information collection and diffusion. *IEEE Transactions on Vehicular Technology*, 67(5):3813–3825, 2018. 2

[27] Fan Wu, Wang Yang, Jialun Lu, Feng Lyu, Ju Ren, and Yaoxue Zhang. Rlss: A reinforcement learning scheme for hd map data source selection in vehicular ndn. *IEEE Internet of Things Journal*, 2021. 2

[28] Zhi Yan, Nicolas Jouandeau, and Arab Ali Chérif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10, 2013. 2