drones

MDPI

Article

DAGmap: Multi-Drone SLAM via a DAG-Based Distributed Ledger

Seongjoon Park and Hwangnam Kim *

School of Electrical Engineering, Korea University, Seoul 02841, Korea; psj900918@korea.ac.kr
* Correspondence: hnkim@korea.ac.kr; Tel.: +82-2-3290-4821

Abstract: Simultaneous localization and mapping (SLAM) in unmanned vehicles, such as drones, has great usability potential in versatile applications. When operating SLAM in multi-drone scenarios, collecting and sharing the map data and deriving converged maps are major issues (regarded as the bottleneck of the system). This paper presents a novel approach that utilizes the concepts of distributed ledger technology (DLT) for enabling the online map convergence of multiple drones without a centralized station. As DLT allows each agent to secure a collective database of valid transactions, DLT-powered SLAM can let each drone secure global 3D map data and utilize these data for navigation. However, block-based DLT—a so called blockchain—may not fit well to the multi-drone SLAM due to the restricted data structure, discrete consensus, and high power consumption. Thus, we designed a multi-drone SLAM system that constructs a DAG-based map database and sifts the noisy 3D points based on the DLT philosophy, named DAGmap. Considering the differences between currency transactions and data constructions, we designed a new strategy for data organization, validation, and a consensus framework under the philosophy of DAG-based DLT. We carried out a numerical analysis of the proposed system with an off-the-shelf camera and drones.

Keywords: multi-agent SLAM; distributed ledger technology; DAG-based DLT

Citation: Park, S.; Kim, H. DAGmap: Multi-Drone SLAM via a DAG-Based Distributed Ledger. *Drones* **2022**, *6*, 34. https://doi.org/10.3390/drones6020034

Academic Editors: Jacky C. K. Chow, Mozhdeh Shahbazi and Ajeesh Kurian

Received: 10 December 2021
Accepted: 18 January 2022
Published: 20 January 2022

1. Introduction

Autonomous navigation of single or multiple drones requires both the information of the environments and its own location. For indoor environments or similar environments, without the support of a reference system, such as a geographic information system (GIS) or a global positioning system (GPS), a drone must rely on its own sensing and recognition system. From the intuitive relevance between the map reconstruction and localization, simultaneous localization and mapping (SLAM) technology emerges [1]. SLAM accumulates the spatial information of the given environment by processing the incoming sensor data (vision, LIght Detection, and And Ranging (LIDAR), acoustic, and so on) and its own location changes. On the other hand, SLAM tracks the drone's location changes by matching the incoming sensor data and the collected map features. This combined approach enables the all-in-one system of the drone navigation, which results in the cost-efficiency and simplicity of the SLAM sensors, and availability of the indoor operations.

Since SLAM normally takes long time to obtain a large-scale map, its scalability is a challenging topic [2]. There are two major ways to expand the size of the map—multi-session style and multi-agent style [3,4]. Multi-session SLAM stores previous SLAM data and reuses it for subsequent sessions of the SLAM operation, which focuses on continuous map improvement and fast localization of the robot. On the other hand, multi-agent SLAM involves a simultaneous SLAM of multiple robots, which requires reconstruction of the global map using map merging techniques. Since the SLAM operated in the robot results in the localized map data whose data are structured from the robot's predefined frame, it requires post-processing to merge multiple maps obtained from multiple robots. Although multi-session SLAM shows remarkable advances [5–7], the multi-agent mapping system is

still at the limit of its implementation since it collects features when synchronizing and it performs centralized map merging [8–11]. The reason for the centralization is mainly the limitation of computational and network resources; in particular, it is difficult to share and process large point clouds in energy-constraint multi-drone networks.

To overcome these limitations, we propose a new approach for multi-drone SLAM called DAGmap, which continuously shares 3D map data based on the operation of distributed ledger technology (DLT). Recently, the distributed ledger system took a big step via blockchain [12], which enabled a consensus of transactions through a distributed network. The advantage of DLT is its resilience from a single point of failure since each participant in the network manages a synchronized database of confirmed transactions. If invalid or malicious transactions are issued, multiple honest participants validate the transactions and exclude them from the database. In addition, proof-of-work (PoW) restrains dishonest actions of malicious participants and encourages honest participants to competitively validate the recent transactions [13].

We constructed DAGmap, observing the concept of the DLT mechanism and composed of the multiple agents that organize a graph-based data structure of 3D features; we share them through the network, and synchronize the 3D maps constructed by the drones. Figure 1 shows the overall architecture of our proposed system. To implement the multi-drone SLAM with DLT, we introduced the detail mechanism of the recent DLTs. As DLT follows a sequence of validations, broadcasts, and confirmation processes, DAGmap performs the same procedure for transactions, each of which contains one or more features. Then, as DLT results in the confirmed sequence of the transactions through the consensus, our system results in a reconstructed 3D map through the consensus among the drones, named as the map consensus.

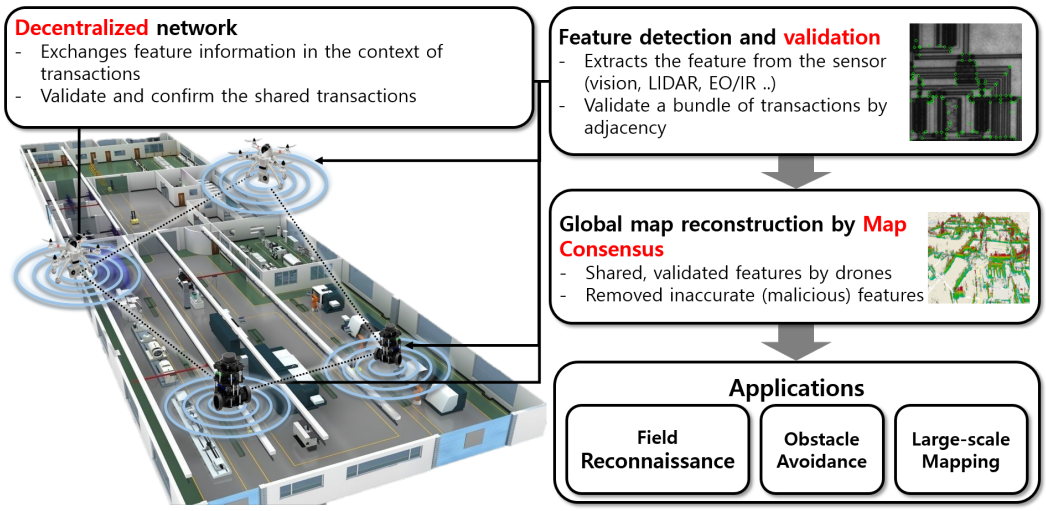

Figure 1. Conceptual overview of DAGmap.

To cope with the differences in the map reconstruction and ledger management, we innovated the existing DLT schemes to a novel architecture to meet the performance requirements of the multi-drone SLAM. In order to continuously build a 3D global map, we adopted directed acyclic graph (DAG)-based DLT technologies [14], which perform validation and confirmation processes for each transaction. In DAGmap, 3D features construct a DAG structure where each point is connected to other points. In addition, we designed our novel DAG construction algorithm for fast validation and confirmation of points while taking advantage of the DLT structure. To handle a large number of the 3D features, we devised a bundle-level transaction construction according to the credibility of the 3D features.

For compatibility, we designed DAGmap as a distinction layer from SLAM that could be implemented separately from the existing SLAM system. We implemented DAGmap in

an empirical environment, showing the possibility of the distributed, online, multi-drone SLAM with COTS devices.

Compared to conventional mapping technologies, our proposed design has several advantages.

- No centralized hierarchy. Previous approaches [8,15,16] required a synchronization point to collect all parts of the map data, which could lead to network bottlenecks. In DAGmap, all drones are equally involved in the global map construction process, so any drone can have the updated global map online.
- Quick handover in mission. The previous studies had little consideration for the single point failures that have a significant impact on the map reconstruction process [17]. In our system, each drone has the entire feature map, so the participation of a new drone can track the current map construction progress. When a new drone joins, any nearby drones can share its entire map without the network connection to the central station. Then, the new drone can process its DAGmap operations to contribute to the ongoing map reconstruction process.
- Reliability assurance. Previous studies have fought against errors in feature points, especially when the drone was purely rotating [18,19]. Due to the inherent features of DLT, our system can render the noise function obsolete through repeated observations, and the resulting map can clearly see the target space.

The rest of this paper is organized as follows. In Section 2, we introduce the related researches and show the differences and similarities with our system. Then, we describe the overall architecture of our system in Section 3, and address the design detail in Section 4. Finally, Section 7 concludes the paper.

2. Related Work

In this section, we address the motivation and the major approaches to the system design. One major issue of the distributed system involves the consensus among participants, since malicious data can seamlessly harm the entire operation of the system [20,21]. Nakamoto proposed a solution called blockchain, which allows more than half of the honest participants to prevent malicious actions through PoW and the chaining of blocks. This work accelerates the research on the distributed ledger system and its applications [22], aiming at the absence of the single-point authority with the higher security of the asset trading. Recent studies have focused on the fast transaction issuance, higher applicability, and the eco-friendly consensus [23]. Since PoW of the blockchain consumes a large amount of computing power—energy efficiency and transaction issuance time are regarded as major challenges that need to be solved. A DAG-based distributed ledger system is one solution that could be used to resolve the blockchain limitations [24,25].

DAG-based ledgers issue each transaction to the network [14], unlike a blockchain that gathers the transactions and generates a block to make the transactions valid. Transactions are stored in a structure of DAG, where each transaction stores its data in a vertex. Each vertex has one or multiple edges towards other vertices; this is called a vertex referencing other vertices. Before a new vertex gets appended to the graph, the system selects the vertices not referred by the other vertices, and validates a set of the transactions that are directly or indirectly referred by the new vertex. If the set of the transactions are valid, then a new vertex is appended to the graph and the system broadcasts the issuance of the transaction. Otherwise, the new vertex re-selects the vertices, and repeats until it finds the valid set of the transactions. Periodically, the system confirms the transaction that has enough ratio of the references, since the higher ratio implies that the number of references for a transaction determines its validity.

In the case of double-spending transactions, they could be appended to the graph if one of them does not directly or indirectly refer the other. However, after a few advances in the graph, the following transactions "confront" the case that validates both double-spending transactions. By the aforementioned vertex selection rule, the following transactions consequently select the referencing nodes that directly or indirectly refer only one of

the double-spending transactions. Finally, only one of the double-spending transactions reaches the threshold of the reference ratio, and gets confirmed. IOTA—to accelerate the differences of the ratio—adopts a weighted random walk at the vertex selection [25]. DAG-based DLT proposes continuous accumulation and verification of the transactions without PoW, which results in the promising approach for mobile, lightweight devices, such as IoT. In the case of DAGmap, even though all nodes are honest, noisy data can unintendedly be obtained, which could be filtered in the consensus stage. We address the details of our DAGmap graph design in the following sections through an example of a multi-drone 3D mapping scenario.

3. System Overview

Figure 2 graphically represents our DAGmap structure, compared to existing DLT and multi-drone SLAM. In the case of our system, PoW is not efficient on mobile devices, such as drones, and we needed to reduce the map completion time to account for the limited drone lifetime. Therefore, we adopted the DAG-based distributed ledger system, where feature points were issued with a minimalized PoW and were validated individually. The newly detected feature was broadcast through the multi-drone network, and the other drones appended the feature at the DAG. The difference between the existing DAG-based DLT and our system is the shape of the graph, because of the difference between the asset management and the object reconstruction. In the case of the distributed ledger, the entire amount of the electronic currency generated by administrators, and all transactions, are validated from the given asset [26]. In the case of 3D mapping, nodes have no specific references for the map reconstruction, so a newly found 3D object should be represented as a new subgraph, where nodes represent a part of a certain object.

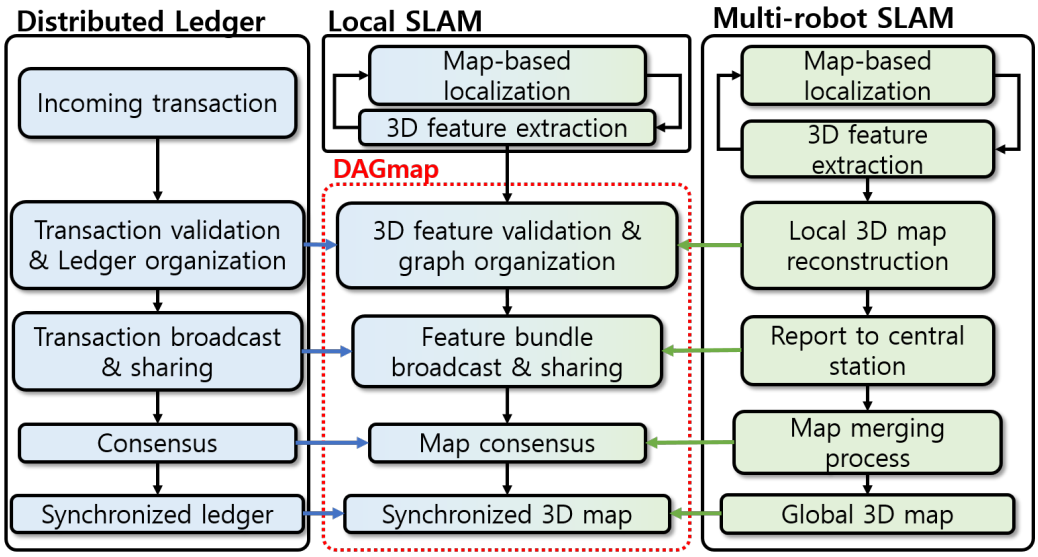

Figure 2. Technical relationship between the components of DLT, SLAM and DAGmap.

The introduction of a 3D graph to the multi-robot SLAM might not be novel, as many previous research works have adopted [15,27,28]. Most research studies used a graph to express the camera position change, and few systems used the graph to organize a 3D map. The graph-based structure in SLAM is designed to suit researcher tastes to optimize the performance. In DAGmap, a specific shape of the graph named DAG was used in the context of DLT, which was not attempted in the previous research.

As shown in Figure 3, we let multiple drone-equipping vision sensors move around the target space while establishing the wireless network. After detecting features on an object, each drone checks if the new feature point is valid; it checks if the feature can reconstruct the 3D object or if it is placed in an acceptable position. If a feature is determined to be valid, then the drone issues the feature as a form of a transaction at the network, and stores

in its database until the transaction gets confirmed by incoming transactions. In the end, the drone gets a database of equivalent maps formed by confirmed transactions that other transactions validate. With blockchain, transactions are verified and confirmed—not in blocks, but individually.

The detailed mechanisms on the validation and the confirmation schemes are introduced, in detail, in the system components section.

To conduct thw system work properly, several detailed issues should be considered.

- As existing DLT experienced, multi-robot SLAM should overcome the large overhead of the data processing, due to the large number of 3D features. In DAGmap, we found the breakthrough of the system acceleration method in the shape of the data structure and validation mechanism, similar to the approaches of the DLT domain. We devised a novel DAG construction scheme, considering the requirements of the multi-drone SLAM, which can be referred to as our main contribution.
- The confirmation process should be entirely revised due to the differences of the inherent characteristics between the 3D features and currency transfers. The main "adversary" of the map reconstruction involves the false-positive features, due to the sensor noise or the artifact of the feature detection method, which accumulatively degrades the resulting map's quality. Thus, to achieve the clean view of the map, DAGmap should establish a novel confirmation procedure, different from the distributed ledger scheme for canceling the noisy features.
- Since DAGmap operates as a higher layer of the SLAM module—the generation and removal of the 3D map points should be reflected at the DAGmap agent. The SLAM module—through point filtering and the loop closure [29]—frequently removes the existing 3D points, which is one of the main differences with the ledger system. DAGmap tracks the removal of the 3D feature, reorganizes the graph, and shares the point clouds with enough validation.

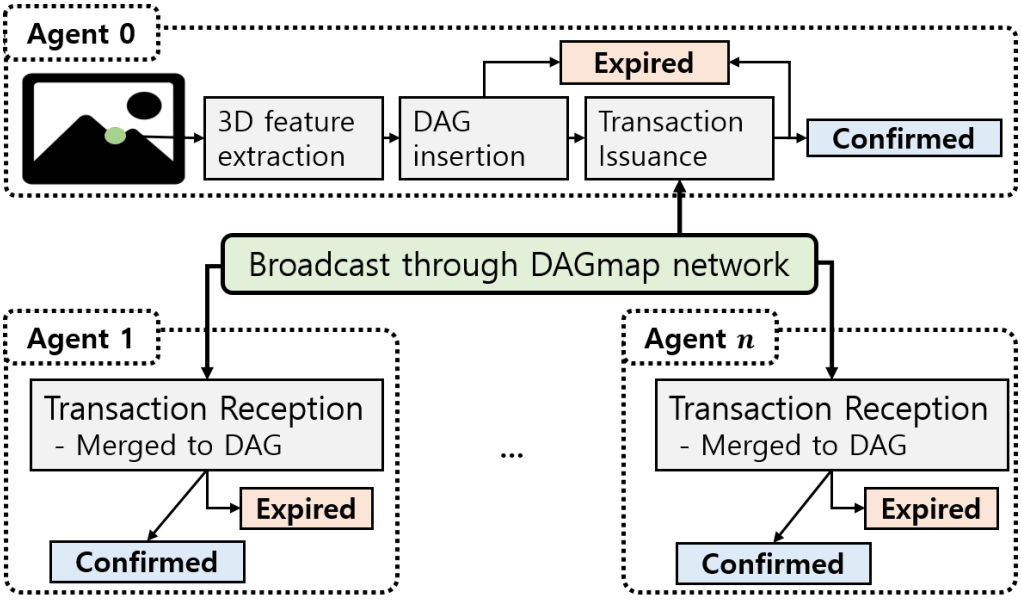

Figure 3. Feature transaction issuance, validation, and confirmation procedure.

4. System Design

The following subsections discuss the more detailed designs of DAGmap. For readability, we define the terminologies used in this paper in Table 1.

Table 1. DAGmap terminologies.

Term	Math Expression	Meaning
Agent	G	Participant of the DAGmap network
Node	n	makecellA vertex of adjacency DAG, containing a 3D map point information
Total nodes	U	A set of nodes that an agent has
Terminal	e	A node that any node does not refer
Terminals	E	A set of terminals, managed by an agent
Parents	p_n	A set of nodes directly referred by the node n
Children	cd_n	A set of nodes directly referring the node n
Ascendants	A_n	A set of nodes directly or indirectly referred by the node n
Descendants	D_n	A set of nodes directly or indirectly referring the node n
Number of elements	$n(s)$	The number of the elements in a set s

4.1. Graph Organization

Most SLAM systems manage the 3D features as point cloud data structures due to their large numbers. This approach accelerates the operation delay of the individual SLAM, but also makes a large overhead, to merge multiple maps, which results in the centralized structure of a multi-robot SLAM. DAGmap organizes a DAG-based data structure to hierarchically manage the features and reduce the operational cost of the map merging process. However, the difference of the mechanism between the ledger validation and map construction should be considered. In the case of DLT, currency transactions occur sequentially, following the initially-given global reference. Therefore, the ledger network should validate all of the transactions in the global consistency check of the account. On the other hand, validation of a feature is not definitive; the system cannot perfectly determine the validity of the point with the other points. However, in the 3D map, the features are densely extracted if a distinguishable object exists. It means that a 3D feature can prove its validity with adjacent features that present a part of the physical object in a map. By using this observation, DAGmap estimates the validity of the features by constructing adjacency DAG for map points.

At first, each agent generates a set of grid-pattern reference nodes, shown as the green circles in Figure 4. Then, for a new 3D feature obtained by SLAM module, DAGmap measures the distance between its position and the terminals, and select N_p number of nearest nodes $n_1, n_2, \ldots, n_{N_p}$, where N_p refers to the maximum number of a node's parents. Finally, a new node n' that satisfies $p_{n'} = \{n_1, n_2, \ldots, n_{N_p}\}$ and $p_{n_i} \in n'$ for $i = \{1, 2, \ldots, n_{N_p}\}$.

This node insertion procedure is similar with conventional DAG-based DLT technologies [25], except for the adjacency-based parent selection instead of random walk [30]. However, there are some differences in terms of the incoming pattern of the nodes. Normally, SLAM updates a bundle of the 3D features collectively, while DLT transactions arrive frequently and individually. Furthermore, newly added features of SLAM usually focus on the specific view angle of the camera. This difference of data generation patterns and the spatial locality of data highly affect the shape of the adjacency DAG. A node (parent, p_n) in cryptocurrency DLT has a chance to be referred to by a proper number of the following nodes (children, cd_n) due to the validation overhead and network delay [31]. On the other hand, each DAGmap agent attaches a number of the features locally, so the DAG results in the long and shallow one, which is not expected for efficient functional management. Thus, we manually handle the list of terminals E, to stall the referred nodes until $n(c_e) < N_c$, where N_c refers to the maximum number of children.

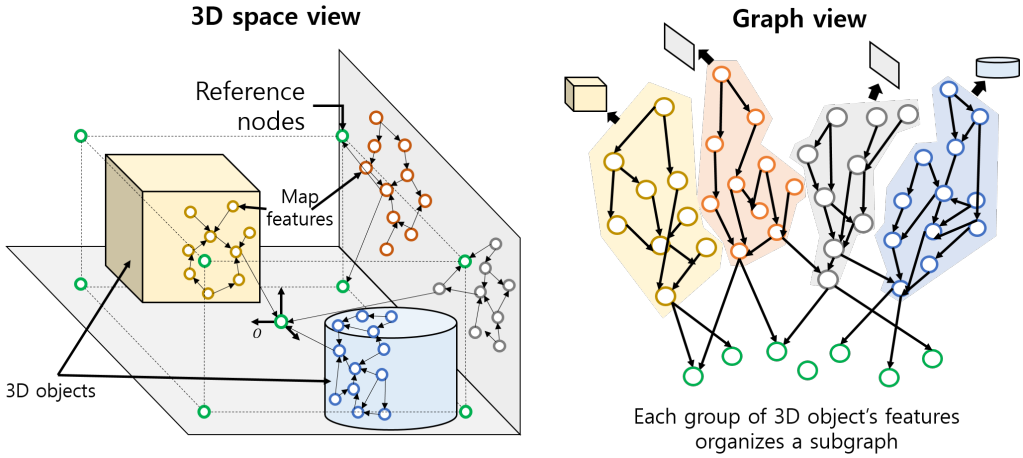

Figure 4. Adjacency DAG composed of 3D features.

Figure 4 graphically represents the resulting graph of the DAGmap from the given map data. Note that the nodes are not broadcast to the DAGmap network after insertion since they are not validated at this point. As aforementioned, DAGmap should shape the traffic of the transaction, and the transaction should contain the nodes that represent valid features.

4.2. Transaction Issuance

It should be noted that the context of the currency transfer is time-sequential, so the transactions that violate account balances, such as double-spending, can be detected by aggregating former transactions. However, 3D features have no global constraints on the map, so anonymous references between the nodes are not efficient for organizing map data. As DAG-based DLT performs only partial validation of the unconfirmed transactions; we attempted the partial validation of the features to accelerate the map reconstruction process. The insertion process of DAGmap's adjacency DAG (Section 4.1) seamlessly checks the validity through the adjacency among the nodes, while forming a cluster of the 3D points. Following this, DAGmap issues a bundle of features in a transaction that includes a set of valid features, sifted by the weighting method [25].

When DAGmap creates a node, it imposes an integer variable named validity to each node, which is referred to v_i for node i. Reference node validity is set to 1. When a new node n is successfully attached to the adjacency DAG, DAGmap sets its validity to

$$v_n = \sum_{i \in p_n} v_i. \tag{1}$$

If $v_n > V_{th}$, where V_{th} refers to the validation overhead, DAGmap broadcasts a transaction that contains the information of the node n and its ascendants A_n. Figure 5 graphically represents the transaction issuance procedure, and Algorithm 1 is an integrated pseudo code of the node insertion and transaction issuance process. In the algorithm, the method `GetAdjacentNodes`(E, N_p) returns, at most, N_p number of the nearest nodes from the terminal set E, and `UpdateVal`(a) performs Equation (1) for v_a.

From the observations in Section 4.1, it is clear that a subgraph A_n represents a part of a distinguishable 3D object due to the adjacency DAG mechanism. The condition $v_n > V_{th}$ indicates that the 3D features of A_n have been observed with enough density, which means that the features are valid, and could be uploaded to the global 3D map. For noisy features, they are less likely to be referred by the following nodes and, thus, less likely to be broadcast to the network. Through this mechanism, DAGmap agents can broadcast a fully validated bundle of the 3D features only, which results in the clear view of the global 3D map. However, if there is noise or other physical objects that are adjacent, the shape of the detected object is incorrect and needs to be filtered. Although this kind of error can

also be filtered at the local mapping phase, DAGmap determines the uncertainty of the 3D feature by collecting multiple agents' results, which cancels the artifact error of the devices. We address this topic in the following section in detail.

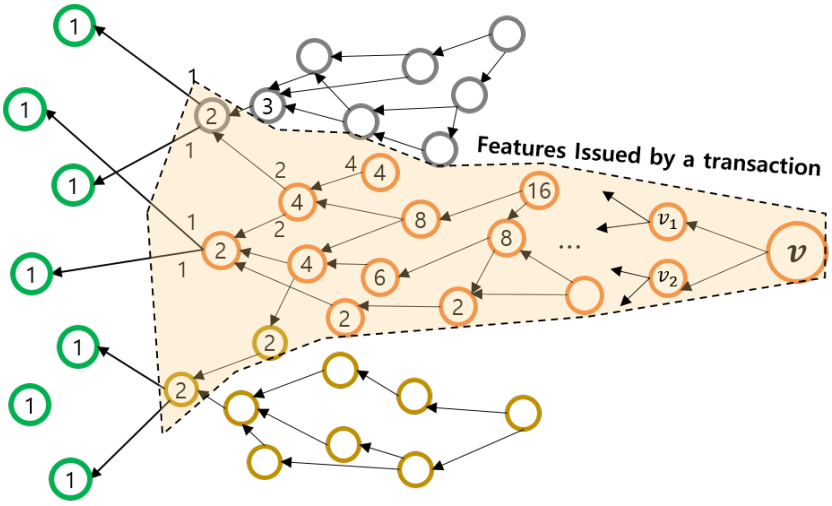

Figure 5. Validation and feature bundle transaction.

Algorithm 1 Node insertion and validation procedure

Input: A new node a
$P \leftarrow$ `GetAdjacentNodes`(E, N_p)
$p_a = P$
for $pt \in P$ **do**
  Add a to cd_p
  if $n(cd_p) == N_c$ **then**
    Remove p from E
  end if
end for
// Broadcast if valid //
`UpdateVal`(a)
if $v_a > V_{th}$ **then**
  $B \leftarrow \{a\} \cup$ `GetAscendants`(a)
  `Broadcast`(B)
end if
Add a to E

4.3. Map Consensus

After enough numbers of the following transactions attach, DLT can confirm the former transactions based on the contextual continuity of the transaction streams. However, the 3D map does not have such a manner, since even an honest robot can unintentionally publish erroneous features. Furthermore, since there is no definite schedule for the transactions that represents the already-observed objects, determining whether a 3D feature is fully confirmed or not is not certain. Therefore, we adopted a feature-wise map consensus scheme, inspired by [32].

In our previous study stated in [32], we designed a DAG-based DLT system named PowerGraph for validating the transactions occurred in the smart grid. Both PowerGraph and DAGmap utilize a similar shape of a database as DAG, but the entire system architecture of each system is separately constructed. When PowerGraph appends a new datum (transaction) to the DAG, it randomly selects the parents, while DAGmap selects the nearest ones in terms of the 3D feature positions. In addition, PowerGraph confirms the transactions according to the reliability of the validation processes, while DAGmap

confirms the features by comparing with the other features obtained from the neighbor drones, which will be explained in Section 4.3 in detail. The characteristics of the data dealt with in the PowerGraph and DAGmap are essentially different, so their detailed designs are distinguished to work properly.

Figure 6 graphically represents the error compensation of a DAGmap consensus scheme. Assuming the feature coordination error forms a normal distribution, the repetitive observation of multiple drones results in overlapped feature sets, which are shared as a transaction through the DAGmap network. In the confirmation process, we introduce the hit/miss ratio for refining the map.

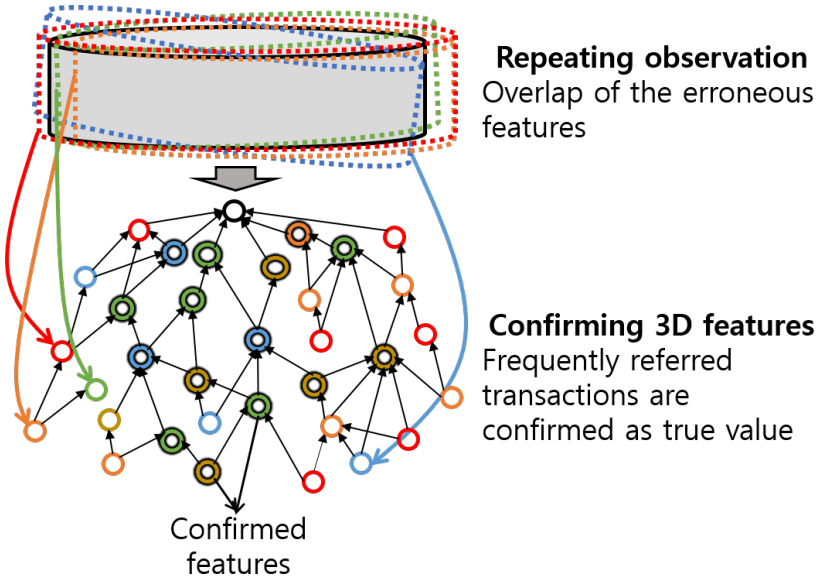

Figure 6. Finding true feature by confirmation process.

Algorithm 2 and Figure 7 represent the map consensus procedure of DAGmap. In the algorithm, U refers to the overall set of the DAGmap agent's nodes, while `GetCoreNode` and `GetRemains` decode the incoming transaction into the root node n and its ascendants A_n (Section 4.2). If a new transaction arrives, the DAGmap agent searches for the nearest node with n in its graph. If a node with almost the same position is found, it is possible that the other agent observes the similar 3D features with this agent. Then, the agent increases the hit count of the found node n' by 1, and it starts to merge two subgraphs rooted by n and n', named bundle merging. Otherwise, it is possible that the transaction represents the other object, so the agent appends the whole subgraph to its adjacency DAG. The bundle merging task searches the highly approximated nodes with each ascendant a_n, and operates the same procedure with the case of n and n'. During this procedure, if a nearest node a'_n cannot be approximated to a_n, the agent newly adds a_n and increases the miss count of them by 1. Figure 7 shows the case of bundle merging, where 5 nodes hit, 4 nodes miss, and finally 13 ascendants appear at the resulting graph, except the reference nodes. Note that when the transaction broadcaster and the receiver are reversed, the same result appears due to the same mechanism of the hit/miss procedure. Through this consensus algorithm, every agent comes to have the same shape of the global map with the same properties of each feature, and evaluates the accuracy of the features by hit/miss investigation.

Algorithm 2 Map consensus algorithm

Input: A new transaction R
$n \leftarrow$ `GetCoreNode`(R)
$A_n \leftarrow$ `GetRemains`(R)
$n' \leftarrow$ `GetAdjacentNodes`$(U, 1)$
$A_{n'} \leftarrow$ `GetAscendants`(n')
if $A'_n == \phi$ **then**
  Add n,A_n to U
else
  if `GetDistance`(n,n′) $< \delta$ **then**
    $h_{n'} \leftarrow h_{n'} + 1$
  else
    Add n to U
    $m_n \leftarrow m_n + 1$
    $m_{n'} \leftarrow m_{n'} + 1$
  end if
  for $a_n \in A_n$ **do**
    $a'_n \leftarrow$ `GetAdjacentNodes`$(A_{n'}, 1)$
    if `GetDistance`$(a_n, a'_n) < \delta$ **then**
      $h_{a'_n} \leftarrow h_{a'_n} + 1$
      `ChangeParent`(a_n, a'_n)
    else
      Add a_n to U
      $m_{a_n} \leftarrow m_{a_n} + 1$
      $m_{a'_n} \leftarrow m_{a'_n} + 1$
    end if
  end for
end if

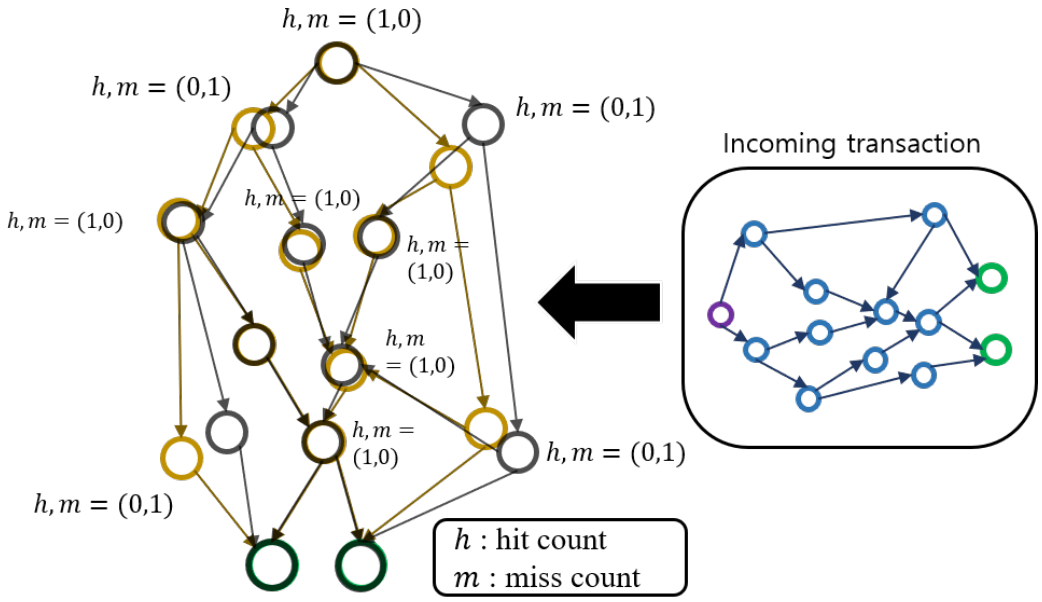

Figure 7. Map consensus algorithm.

5. System Analysis

Since DAGmap operates above the SLAM modules in a drone system, performance becomes important to guarantee the online operation. This section analyzes the system overheads of the DAGmap mathematically and proposes the optimization solution.

5.1. DAGmap System Analysis

As shown in Figure 3, newly added 3D features periodically appear at every time cycle. We let T refer to the average time cycle that the SLAM module outputs the 3D features, and N_f refer to the average number of the new features for each cycle. If we let N_{tx} refer to the average number of transactions, which is highly varied by the environment, the objective performance of N_G agents can be expressed as

$$S = N_f t_{insertion} + N_{tx} t_{broadcast} + N_G N_{tx} t_{merge} < T \tag{2}$$

where S refers to the estimated system overhead, and $t_{insertion}, t_{broadcast}, t_{merge}$ refer to the system overhead of feature insertion, transaction broadcast, and map consensus scheme, respectively.

At first, $t_{insertion}$ is mainly determined by $n(E)$, the number of the terminals. With constraints on N_c, the dimension of the graph can be approximated, so $t_{insertion}$ can be expressed as

$$t_{insertion} = \frac{n(U)}{\log_{N_c} n(U)} l_{dist} \tag{3}$$

where l_{dist} refers to the delay of the distance calculation per a pair of position vectors. $t_{broadcast}$ is mainly determined by the function call `getAscendants`() and the network delay, which can be expressed as

$$t_{broadcast} = 2^{\log_{N_c} n(U)} l_{search} \tag{4}$$

where l_{search} refers to the graph search delay. In the case of t_{merge}, root node especially runs $n(U)$ number of calculations, so it can be calculated to

$$t_{merge} = (n(U) + 2^2 \log_{N_c} n(U)) l_{dist}. \tag{5}$$

From Equations (3)–(5), we derive the complexity of the DAGmap procedure. For readability, we let $N_c = 2$. By Equation (3), the insertion process has $O(N/\log_2 N)$, while assuming l_{dist} as constant. By Equations (4) and (5), transaction validation and confirmation process have $O(N)$, respectively. Since N_f and N_{tx} refer to the average rate of the SLAM outputs, these factors can be regarded as constant, as N_G. Therefore, DAGmap results in the complexity of $O(N)$, which is acceptable for the onboard system. If N_c sets to be the higher value, the complexity reduces to $O(2^{\log_{N_c} N})$, but it greatly increases the validation overhead and map completion time due to the large width of the graph. In our DAGmap implementation, we control N_c while monitoring $n(U)$ and estimating the system overhead by the above equations.

5.2. Network Delay Analysis

DAGmap agents transmit the transactions through the distributed network. To sustain the synchronization of the global map, each agent should transfer the transactions to the other agents, and receive the transactions from them. Regarding network topology and size, overflown transactions on the network can delay the completion of the global map reconstruction, resulting in the performance degradation of the DAGmap. Owing to the continuity of the DAGmap feature streaming, the network traffic can be shaped to guarantee the online synchronization of the DAG. Since the network status can be varied due to the mobility of drones, DAGmap agent should estimate the transmission delay of the transaction and periodically adjust its traffic, by modifying N_{tx}.

From the variables introduced in Section 5.1, DAGmap agents should determine the parameter that holds

$$\frac{N_f' N_G}{T} < C_{Th} \tag{6}$$

where N_f' is the number of transactions to be broadcast, T is the link usage count of the transaction issuances, and C_{Th} refers to the capacity of the DAGmap network, which

can be allocated by the global drone network system. C_{Th} is determined by the network architecture that the DAGmap network adopts, which is not deeply addressed in here due to the scope of the paper. DAGmap adjusts N'_f in the transaction issuance stage (Section 4.2), by excluding the furthest descendants in the transaction packet. The nearest truncated descendants inherit the root node's validity, so they are allowed to be broadcast at the next loop.

6. Implementation

We implemented the DAGmap software by Python, and organized a test-bed as Figure 8. We adopted `RTABmap_ros` [33] to utilize the ROS architecture, and Tara stereo camera, which supports a USB 3.0 port. In `RTABmap_ROS`, we set all parameters to default, and the map update frequency was 1 Hz, which means $T = 1$. To prevent the multi-session operation of `RTABmap_ROS`, we did not save the resulting map database file `database.db` during the experiment.

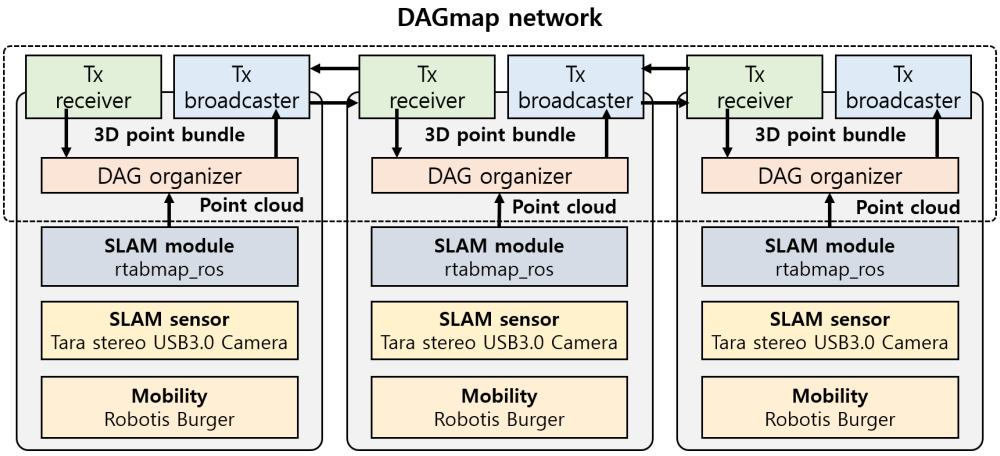

Figure 8. Implementation scheme of DAGmap.

We organized the ROBOTIS burger for mobility. We selected the mobility to an unmanned ground vehicle (UGV) to effectively verify the contribution of DAGmap. The role of DAGmap involves the efficient and distributed merging of 3D map data, so we constructed the test scenario while minimizing the artifact of the SLAM module and the hardware devices. To obtain stabilized 3D features from the SLAM module, we prepared the mobility enabling micro movement of the camera. Moreover, to observe the merging of the short-range 3D features, we progressed the experiment with UGV for safety.

Due to the differences in the overall mechanism from existing DLT, we implemented DAGmap from the scratch. We constructed an adjacency DAG structure and comprehensively implemented feature addition, broadcast, and a map consensus process in an application. Furthermore, we realized DAGmap software as an augmented layer of ROS-based SLAM node, and our implementation result can be linked with other ROS-portable SLAM modules. This portability is the core objective that DAGmap oriented, so that the resulting demonstration shows the feasibility of DAGmap for any desires of collective data refinements.

As shown in Figure 9, we operated three DAGmap agents to search for a space around a working desk. We previously surveyed the EuRoC [34] and Kitti [35] dataset for evaluation, but we could not find the dataset for multi-robot SLAM, so we concluded that the existing datasets would not fully show the validity of DAGmap. Thus, by referring to the previous empirical studies [36], we conducted the experiment, with three mobile nodes with three cameras. We intentionally made the searching area overlap to invoke the map merging process, addressed in Section 4.3. Figure 10 shows the resulting 3D maps of the DAGmap demonstration. In the figure, the top left subfigure shows the overall features obtained by the three SLAM modules, without DAGmap operation. The top right subfigure

shows the graph view that we snapshot in the middle of the demonstration. The green, red, and blue circles indicate the features that agents 1, 2 and 3 broadcast, respectively. The purple ones indicate the features that one of the agents had merged, and the gray circles indicate the features that the agent did not broadcast. As shown in the subfigure, the edges among the nodes start with a set of the reference nodes and form clusters representing discrete objects. Small size clusters are not broadcast because there are only a few nodes with low v value, which can be broadcast by the following nodes, if the features represent the actual objects.

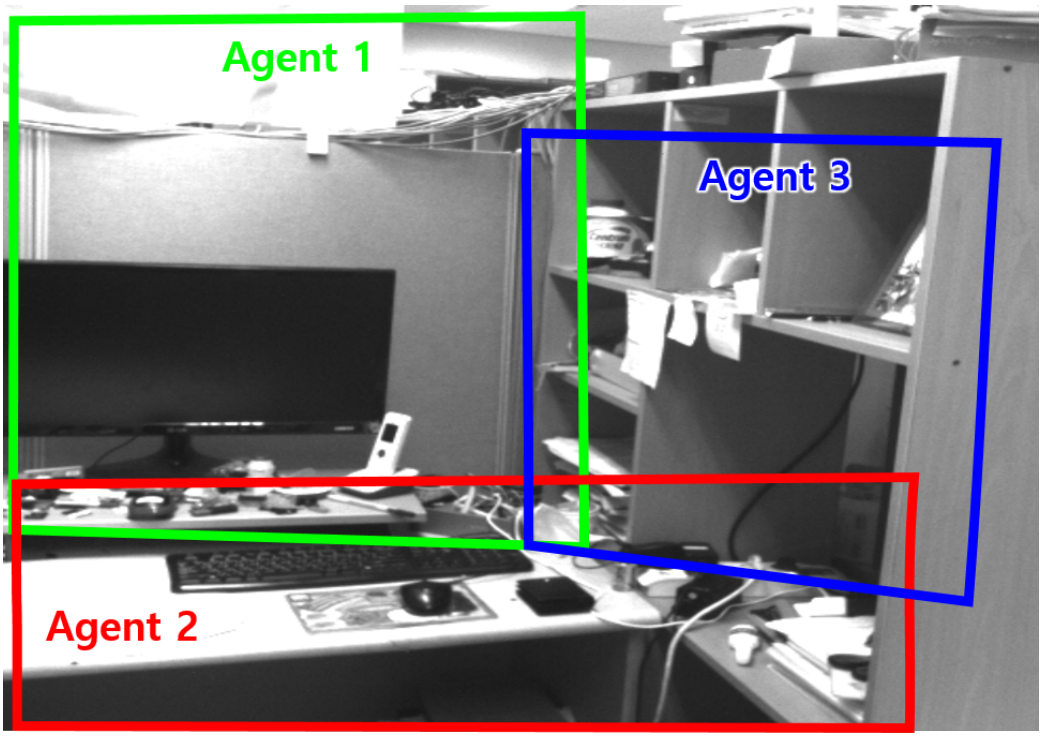

Figure 9. Searching areas of DAGmap agents.

The bottom left subfigure shows the DAGmap results, representing only the features in the transactions, regardless of the hit/miss ratio. As shown in the subfigure, DAGmap merged the three maps, but included all of the noisy features. The bottom right subfigure separately shows the features with hit rates greater than 0.5 in the observed color, and features with hit ratios less than 0.5 in red. Comparing the top left one, most of the noisy features were dropped at the validation process. In particular, a large number of noises above the desk were removed due to the complexity of the environment. If the dropped ones were valid, then it would appear afterwards with a higher hit/miss ratio. Through this implementation, we confirmed that our DAGmap constructed the global map while filtering out the noisy features.

Finally, we verified the stability of the DAGmap network by measuring the network traffic generated by the DAGmap network. Note that the role of DAGmap is to construct a global map database while sharing and validating the 3D features obtained from the SLAM modules. Therefore, since the quality of global maps is highly dependent on the quality of the SLAM module, it may be inappropriate to compare it to existing SLAM modules in terms of common metrics, such as position error. Although the DAGmap improves the accuracy of the global map through removing the noises in validation and confirmation processes, the core source of quality originates from SLAM logic. Since the main contribution of the DAGmap involves enabling the distributed multi-robot SLAM that operates map merging without the central server, we evaluated our system in the viewpoint of the stability of the inter-robot feature sharing. To analyze the traffic of the network, we collected the transaction issuance time of each agent and the number of features in the transactions. Figure 11 shows the

graphical representation of the feature distribution traffic of the DAGmap network, compared to the case without the DAGmap network. Due to the large amount of differences of N_f, we represent the logarithmic scale of each case for visibility. Note that one of the contributions of the DAGmap is traffic shaping through the adjacency DAG organization and the feature validation. As shown in the figure, the transactions of features are continuously shared through the DAGmap network, whereas the comparison case collectively broadcasts the 3D features after the map update. This evaluation confirms that DAGmap prevents traffic bursts due to the collective feature generation of SLAM and enables fair robotic networking, where other multi-robot applications require a certain amount of network resources.

Note that the total amount of the traffic might be similar to the existing multi-robot SLAM studies.

Since an agent itself lacks certainty about the accuracy of the 3D features it obtained, it is appropriate to share all of the features through the network. However, DAGmap filters and organizes the features in the adjacency DAG and shares the features by a unit of bundles to reduce the overhead of the other agents. Our map consensus scheme, adopting DAG-based DLT, stabilizes the traffic, while accelerating the map merging of the agents.

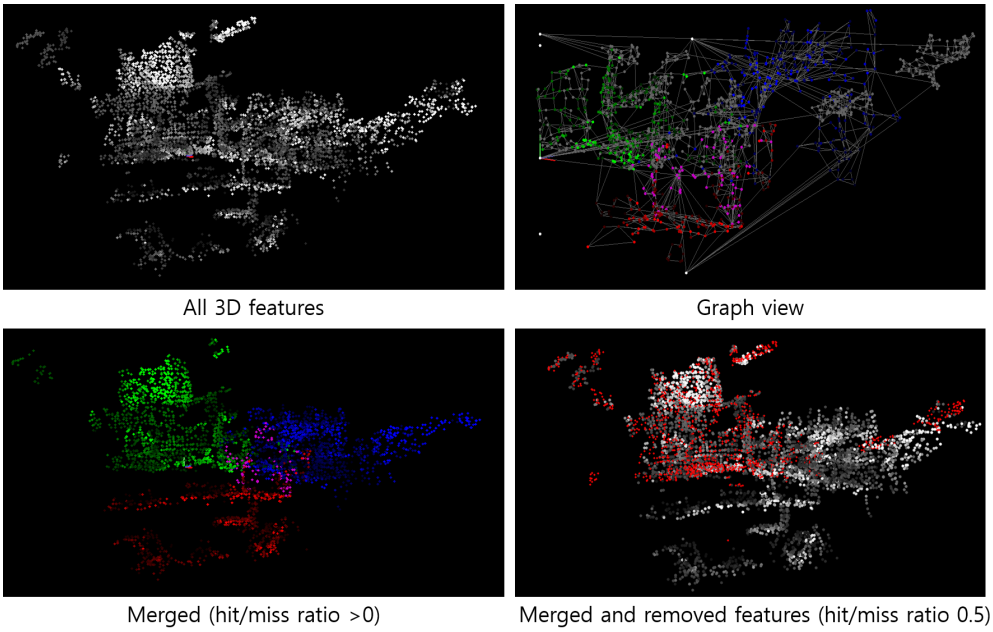

Figure 10. Experiment results of DAGmap.

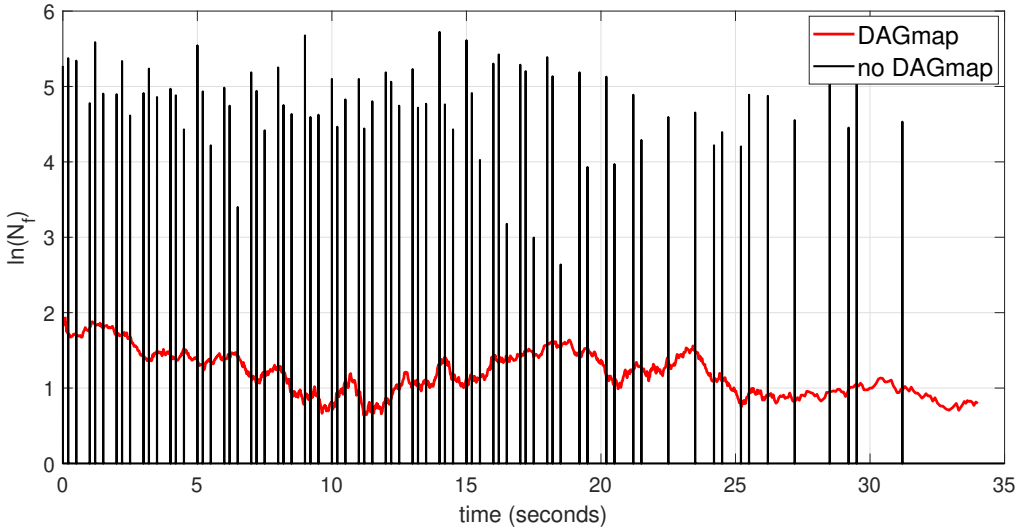

Figure 11. N_f traffic of DAGmap network with respect to time.

7. Conclusions

In this paper, we proposed a design DAGmap for a multi-robot SLAM network, utilizing the concept of the DLT. To improve the computational efficiency and overhead reduction of the 3D map integration, the proposed DAGmap adopted a state-of-the-art distributed ledger methodology and broadcast feature points as forms of transactions. We did not simply replace the data field of the DLT transaction, but we revised the overall mechanism of the validation and confirmation process while reflecting the characteristics of SLAM. We designed the shape of the DAG for our architecture, which is different from a conventional DAG-based DLT, regarding the characteristics of the feature database. Moreover, by analyzing the system overhead and network traffic with respect to the DAGmap parameters, we derived the adaptive control method for online system stabilization. Finally, we implemented and demonstrated DAGmap in an empirical environment, and verified that the system mechanism filters the noisy features and shapes the network traffic of the feature broadcast.

In a following study, we will focus on the empirical analysis of the DAGmap network, which contains the context of the network traffic with respect to the various factors, such as a featureless environment or heterogeneous cameras (RGB-D, stereo, and monocular), and we will derive the parameter optimization methods and structural improvements of the DAGmap.

Author Contributions: Conceptualization, S.P.; methodology, S.P.; software, S.P.; validation, S.P. and H.K.; formal analysis, S.P.; resources, H.K.; data curation, H.K.; writing—original draft preparation, S.P.; writing—review and editing, H.K.; visualization, S.P.; supervision, H.K.; project administration, H.K.; funding acquisition, H.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2021-0-01835) supervised by the Institute of Information & Communications Technology Planning & Evaluation (IITP), and the Human Resources Program in Energy Technology of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy (MOTIE) of the Republic of Korea (no. 20204010600220).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. Furthermore, the funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

SLAM	simultaneous localization and mapping
DLT	distributed ledger technology
DAG	directed acyclic graph

References

1. Taketomi, T.; Uchiyama, H.; Ikeda, S. Visual SLAM algorithms: A survey from 2010 to 2016. *IPSJ Trans. Comput. Vis. Appl.* **2017**, *9*, 16. [CrossRef]
2. Aulinas, J.; Petillot, Y.R.; Salvi, J.; Lladó, X. The slam problem: A survey. *CCIA* **2008**, *184*, 363–371.
3. Krinkin, K.; Filatov, A.; Filatov, A. Modern multi-agent slam approaches survey. In Proceedings of the XXth Conference of Open Innovations Association FRUCT, Saint Petersburg, Russia, 3–7 April 2017; Volume 776, pp. 617–623.
4. Zou, D.; Tan, P.; Yu, W. Collaborative visual SLAM for multiple agents: A brief survey. *Virtual Real. Intell. Hardw.* **2019**, *1*, 461–482. [CrossRef]

5. Labbe, M.; Michaud, F. Online global loop closure detection for large-scale multi-session graph-based SLAM. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 2661–2666.
6. Burguera Burguera, A.; Bonin-Font, F. A Trajectory-Based Approach to Multi-Session Underwater Visual SLAM Using Global Image Signatures. *J. Mar. Sci. Eng.* **2019**, *7*, 278. [CrossRef]
7. Campos, C.; Elvira, R.; Rodríguez, J.J.G.; Montiel, J.M.; Tardós, J.D. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial and Multi-Map SLAM. *arXiv* **2020**, arXiv:2007.11898.
8. Zhang, P.; Wang, H.; Ding, B.; Shang, S. Cloud-based framework for scalable and real-time multi-robot slam. In Proceedings of the 2018 IEEE International Conference on Web Services (ICWS), San Francisco, CA, USA, 2–7 July 2018; pp. 147–154.
9. Badalkhani, S.; Havangi, R.; Farshad, M. Multi-Robot SLAM in Dynamic Environments with Parallel Maps. *Int. J. Humanoid Robot.* **2021**, *18*, 2150011. [CrossRef]
10. Schmuck, P.; Ziegler, T.; Karrer, M.; Perraudin, J.; Chli, M. COVINS: Visual-Inertial SLAM for Centralized Collaboration. In Proceedings of the 2021 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct), Bari, Italy, 4–8 October 2021; pp. 171–176.
11. Saha, O.; Dasgupta, P. A comprehensive survey of recent trends in cloud robotics architectures and applications. *Robotics* **2018**, *7*, 47. [CrossRef]
12. Mainelli, M.; Smith, M. Sharing ledgers for sharing economies: An exploration of mutual distributed ledgers (aka blockchain technology). *J. Financ. Perspect.* **2015**, *3*, 3.
13. Lee, J.; Lee, B.; Jung, J.; Shim, H.; Kim, H. DQ: Two approaches to measure the degree of decentralization of blockchain. *ICT Express* **2021**, *7*, 278–282. [CrossRef]
14. Benčić, F.M.; Žarko, I.P. Distributed ledger technology: Blockchain compared to directed acyclic graph. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–5 July 2018; pp. 1569–1570.
15. Karrer, M.; Schmuck, P.; Chli, M. CVI-SLAM—Collaborative visual-inertial SLAM. *IEEE Robot. Autom. Lett.* **2018**, *3*, 2762–2769. [CrossRef]
16. Atanasov, N.; Le Ny, J.; Daniilidis, K.; Pappas, G.J. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4775–4782.
17. Kegeleirs, M.; Grisetti, G.; Birattari, M. Swarm SLAM: Challenges and Perspectives. *Front. Robot. AI* **2021**, *8*, 23. [CrossRef] [PubMed]
18. Zhou, Y.; Yan, F.; Zhou, Z. Handling pure camera rotation in semi-dense monocular SLAM. *Vis. Comput.* **2019**, *35*, 123–132. [CrossRef]
19. Chng, C.K.; Parra, A.; Chin, T.J.; Latif, Y. Monocular rotational odometry with incremental rotation averaging and loop closure. In Proceedings of the 2020 Digital Image Computing: Techniques and Applications (DICTA), Melbourne, Australia, 29 November–2 December 2020; pp. 1–8.
20. Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System*; Decentralized Business Review: Seoul, Korea, 2008; p. 21260.
21. Yang, R.; Yu, F.R.; Si, P.; Yang, Z.; Zhang, Y. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1508–1532. [CrossRef]
22. Ølnes, S.; Ubacht, J.; Janssen, M. Blockchain in government: Benefits and implications of distributed ledger technology for information sharing. *Gov. Inf. Q.* **2017**, *34*, 355–364.
23. Zheng, Z.; Xie, S.; Dai, H.N.; Chen, X.; Wang, H. Blockchain challenges and opportunities: A survey. *Int. J. Web Grid Serv.* **2018**, *14*, 352–375. [CrossRef]
24. Cullen, A.; Ferraro, P.; King, C.; Shorten, R. Distributed ledger technology for IoT: Parasite chain attacks. *arXiv* **2019**, arXiv:1904.00996.
25. Popov, S. The Tangle. White Paper. 2018. Available online: http://www.descryptions.com/Iota.pdf (accessed on 9 December 2021).
26. Rauchs, M.; Glidden, A.; Gordon, B.; Pieters, G.C.; Recanatini, M.; Rostand, F.; Vagneur, K.; Zhang, B.Z. Distributed Ledger Technology Systems: A Conceptual Framework. 2018. Available online: https://ssrn.com/abstract=3230013 (accessed on 9 December 2021).
27. McCormac, J.; Clark, R.; Bloesch, M.; Davison, A.; Leutenegger, S. Fusion++: Volumetric object-level slam. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 32–41.
28. Whelan, T.; Leutenegger, S.; Salas-Moreno, R.; Glocker, B.; Davison, A. ElasticFusion: Dense SLAM without a pose graph. *Int. J. Robot. Res.* **2016**, *35*, 1697–1716. [CrossRef]
29. Liu, Q.; Duan, F.; Sang, Y.; Zhao, J. A survey of loop-clos ure detection method of visual SLAM in complex environments. *Robot* **2019**, *4*, 112–123.
30. Kusmierz, B.; Sanders, W.; Penzkofer, A.; Capossele, A.; Gal, A. Properties of the Tangle for uniform random and random walk tip selection. In Proceedings of the 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 14–17 July 2019; pp. 228–236.
31. Park, S.; Oh, S.; Kim, H. Performance Analysis of DAG-Based Cryptocurrency. In Proceedings of the 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 22–24 May 2019; pp. 1–6.
32. Park, S.; Kim, H. DAG-Based distributed ledger for Low-Latency smart grid network. *Energies* **2019**, *12*, 3570. [CrossRef]

33. Labbé, M.; Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *J. Field Robot.* **2019**, *36*, 416–446. [CrossRef]
34. Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. The EuRoC micro aerial vehicle datasets. *Int. J. Robot. Res.* **2016**, *35*, 1157–1163. [CrossRef]
35. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
36. Rizk, Y.; Awad, M.; Tunstel, E.W. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–31. [CrossRef]