A distributed framework for monocular visual SLAM

Ruwan Egodagamage, Mihran Tuceryan
Department of Computer and Information Science
Indiana University Purdue University Indianapolis
Indianapolis, Indiana 46202, USA
Email: {rjegodag, tuceryan}@iupui.edu

Abstract

In Distributed Simultaneous Localization and Mapping (SLAM), multiple agents generate a global map of the environment while each performing its local SLAM operation. One of the main challenges is to identify overlapping maps, especially when agents do not know their relative starting positions. In this paper we are introducing a distributed framework which uses an appearance based method to identify map overlaps. Our framework generates a global semi-dense map using multiple monocular visual SLAM agents, each localizing itself in this map.

1 Introduction

In Simultaneous Localization and Mapping (SLAM), an agent creates a map of an unknown environment, at the same time localizing itself in it. These two tasks, cannot be solved independent to each other. An accurate estimation of the pose is required, to build a map. At the same time, a good map is required for agent localization.

In certain applications such as indoor tracking, there could be multiple agents moving in a given environment. The agents (cameras for our purposes in this paper) can enter and exit the environment at any time. The environment may be unknown. If there is a map of the environment, the agents can utilize it to localize themselves in it. If an agent moves in a part of the environment that is not mapped, it can start building the map and localize itself in it as part of the SLAM process. Each agent can do this independently, however, when they are operating in a common environment, it makes sense to use their locally built maps to complement each other and complete and/or improve the global map and, therefore, help each other in their respective tasks.

One can use a camera as the only input device to perform Visual SLAM. Ubiquitous cameras (e.g., on mobile devices) make visual SLAM a more popular choice. However, cameras impose an additional challenge since they provide bearing only data. In recent work, both feature based and featureless direct methods are used in visual SLAM. Direct methods like [Engel *et al.*, 2014] generate denser maps. Dense maps can be more attractive in certain applications, such as augmented reality, in which a user is interacting with the environment and virtual objects in the environment. It is desirable that this interaction be realistic and seamless. A dense map of the environment makes this interaction possible.

Additionally, using multiple agents to perform SLAM increases the robustness of SLAM process and makes it more fault tolerant and less vulnerable for catastrophic failures. One of the main challenges in distributed SLAM is to compute map overlaps, especially when agents have no prior knowledge of their relative starting positions. Usually agents also have limited bandwidth to communicate with each other.

In this paper we introduce a distributed framework for monocular visual SLAM agents with no initial knowledge of their relative positions.

2 Related Work

2.1 Visual SLAM

Building a 3D map of the environment from motion has been studied in computer vision under the name of structure from motion [Faugeras and Lustman, 1988; Sturm and Triggs, 1996]. However, this 3D scene structure estimated from motion does not necessarily mean that the result is a coherent global map. In SLAM this is accomplished by "loop closure" in which features or structures already seen before are used to refine the map as well as correct the camera's path and, therefore, the localization.

Davison et al. introduced a visual SLAM method of capturing the path of a freely moving camera (6 Degrees of Freedom — DoF), while generating a sparse map [Davison *et al.*, 2007]. The method was called Monocular visual SLAM (MonoSLAM). MonoSLAM is limited to work in a room-sized environment. A more robust method called Parallel Tracking and Mapping (PTAM) was introduced by Klein et al. in [Klein and Murray, 2007]. It focused on accurate and fast mapping in a similar environment to MonoSLAM.

In contrast to feature based methods, recent Monocular SLAM work, DTAM by Newcombe et al. in [Newcombe *et al.*, 2011] and LSD-SLAM by Engel et al. [Engel *et al.*, 2014], utilize image pixel intensities directly, and generate dense or semi-dense maps of the environment. These maps describe the environment better. And, generally, they are more robust to blur.

2.2 Distributed SLAM

Multiple agents in a distributed SLAM system could be handled using a naive brute-force method, where nodes communicate all sensor observations and map updates with each other in a complete graph topology. However, the communication bandwidth and computational resources available for an agent are typically limited and the distributed network is subject to failures of nodes and links. So it is necessary to come up with an intelligent approach to cope with these challenges.

A unique, globally consistent map can be easily generated, if the agents know either their relative locations or map overlaps. For example, in [Nettleton *et al.*, 2006] relative locations of the agents are found by global positioning sensors (GPS). It is also relatively easier to determine map overlaps if all relative initial poses of agents are known. For example, Paull et al. in [Paull *et al.*, 2015] initialize agents with known GPS location information.

The problem becomes more difficult if the relative locations of the agents are unknown. In some contributions, agents continue building local sub maps until the agents meet each other. Howard et al. [Howard *et al.*, 2006] propose a method where each agent could detect the other agents. The agents use these coincidental encounters to find their relative locations. Dieter Fox et al. in [Fox *et al.*, 2006] present a method where each agent is actively seeking the other agents in the environment to find relative locations between them.

We used our experimental framework for distributed SLAM [Gamage and Tuceryan, 2016] for the development of the methods in this paper.

3 Methodology

The distributed computing framework is implemented as computing nodes that communicate with each other over a network. The nodes have different computing tasks depending on what their function is.

3.1 The distributed SLAM framework

We have two types of nodes in our distributed framework, *exploring* nodes and *monitoring* nodes. Nodes are physically located in different computers. They communicate with each other using the underlying computer network. At any given time, there is one monitoring node and multiple exploring nodes. Each node has a unique identifier. The exploring nodes are linked to the monitoring node and exploring nodes may be linked to each other if their maps overlap resulting in a network of nodes as shown in Figure 1.

The exploring nodes perform monocular visual SLAM. Each node receives images from a single camera. Also these nodes receive data and commands from the monitoring node. They exchange data between each other and they send data to the monitoring node. Our framework adapted the monocular visual SLAM work by [Engel *et al.*, 2014] in exploring nodes. The details of the exploring nodes are described in section 3.3.

The monitoring node continuously receives data from exploring nodes. By looking at the maps of exploring nodes, it detects map overlaps and loop closures. Accordingly, it issues commands to relevant exploring nodes. The details of the monitoring nodes are described in section 3.2.

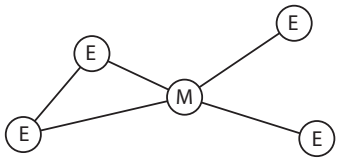

Figure 1: The network of nodes, all exploring nodes (E) are connected to the monitoring node (M). Some exploring nodes are also connected to each other

The Robot Operating System

We use the ROS infrastructure [Quigley *et al.*, 2009] to implement the distributed SLAM framework. A ROS node is responsible of performing computations. ROS also provides a message passing communication framework between nodes. Nodes in our framework are implemented as ROS nodes.

In its communication framework, ROS provides named communication buses called topics. Multiple nodes can publish messages to a topic while multiple subscribed nodes could receive them. Based on the requirement, ROS could either use UDP or TCP for message passing. In our framework data and commands are communicated between nodes as ROS messages.

ROS has a master server to list all available topics of a system. After identifying providers and subscribers of a topic by communicating with master server, ROS nodes can communicate with each other, peer-to-pear via topics.

The Map

In our framework, a map is represented using a set of *keyframes* in a *pose graph*.

The i^{th} *keyframe*, $\mathcal{K}_i$ consists of an *absolute pose* $\boldsymbol{\xi}_{Wi}$, an image I_i, an inverse depth map D_i, an inverse depth variance map V_i, and a list of features $f_i^{(n)}$. Each keyframe gets a globally unique 32 bit identifier as shown in Algorithm 1.

Algorithm 1 Unique identifier for a keyframe

```
1: procedure GETUNIQUEID(keyframe_id, node_id)
2:     id ← SHIFTLEFT(node_id, 20)
3:     id ← id + keyframe_id
4:     return id                    ▷ A globally unique identifier
5: end procedure
```

The pose graph contains edges, ε_{ji}, having similarity transformation $\boldsymbol{\xi}_{ji} \in \mathbb{R}^7$ and corresponding covariance matrix $\boldsymbol{\Sigma}_{ji}$, between i^{th} and j^{th} key frames as shown in Figure 2. The relative pose between the two nodes is encoded with three components of the translation and the four components of the quaternion.

3.2 Monitoring nodes

The monitoring node contains a feature store to collect all the SURF [Bay *et al.*, 2008] features (and SIFT [Lowe, 2004]

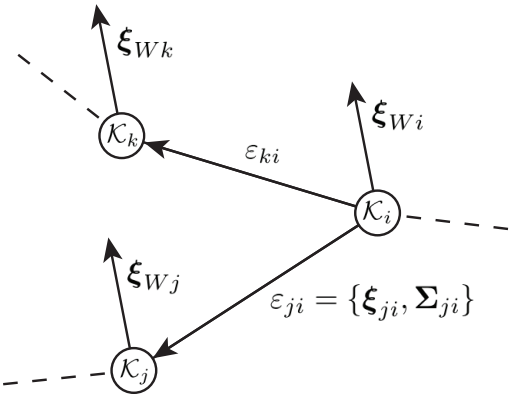

Figure 2: Keyframes and pose graph representing the map.

descriptors) received from the exploring nodes. It also maintains a fusion graph of exploring nodes to keep track of map overlaps.

Map overlap detection
All features found in incoming the key frames are compared against features in the feature store. If there are more than N matching features with a key frame of a different node, it is concluded that those two nodes have overlapping maps. The threshold N is determined empirically.

Exploring nodes are represented as vertices in the fusion graph. All identified map overlaps are represented as edges in the fusion graph. The weight corresponds to the number of matching features found across multiple key frames.

Loop closure detection
Similar to map overlap detection, if there are matching features between two key frames of the same node and there is no edge between those key frames in the map, then it is considered a loop closure. Loop closures that connect temporally far apart key frames, corresponds to the event where agent revisit a location after a while. These large loop closures help to reduce accumulated pose drift, and improve the accuracy of the entire map. Similarly, small loop closures, which connect key frames that are temporally closer, improve sub regions of the map.

Rigid transformation between key frames
In both map overlap detection and loop closure detection, the relative rigid transformation between key frames are computed using a least-square method [Sorkine-Hornung and Rabinovich, 2017]. RANSAC algorithm [Fischler and Bolles, 1981] is used to get rid of outliers.

3.3 Exploring nodes

The functional modules and the architecture of an exploring node are shown in Figure 3 which consist of the input stream, tracking, mapping, and optimization modules. All these modules run in separate threads.

The *input stream* module accepts incoming frames from the camera as well as commands and data from other nodes.

The *tracking* module accepts a new frame from the *input stream* module, and tracks it against the current key frame. If the current key frame no longer can track the frame, i.e. when the frame has significantly deviated from the key frame, a new key frame is created.

The old key frame is added to the map by the *mapping* module. The new key frame is also sent to the monitoring node and to all other exploring nodes subscribed (linked) to the current exploring node.

The *optimization* module continuously optimizes the pose graph in the background. After each optimization the pose graph is sent to the monitoring node and to the subscribed exploring nodes.

Map merging process
The Figure 4 shows the sequence of operations that take place in the map merging process.

Consider the case in which the exploring node A receives a merge command from the monitoring node. The input stream module puts the command into a command buffer. The tracking module contains multiple *state* submodules, including one for map merging. The map merging state submodule, processes the command from the buffer and changes its state, to *waiting for a map* from the other exploring node B. Furthermore, it sends its own map to B. Once the map of B is received, all new key frames and the new pose graph is added into the A's map, and the state is changed to the default state, where node A is waiting for new merge commands.

Figure 5 shows a resultant map after map merging process.

Multiple instances of the same ROS node
In our distributed framework, we implemented exploring node as a ROS node. Given there are multiple instances of exploring node in the network, we have to configure these instances, so that we can uniquely identify each exploring node's topics. For this we use ROS topic remapping. For example, say we have two exploring node instances, namely A and B, then exploring node's $/dvslam/graph$ topic could be remapped to $/A/dvslam/graph$ and $/B/dvslam/graph$. In this way, node A could listen to topic $/B/dvslam/graph$ and node B could listen to topic $/A/dvslam/graph$.

The following simplified scripts in Listing 1 and Listing 2 show how an exploring node was configured for remapping.

Listing 1: Setting up environment variables

```
export ROS_IP=10.1.2.3
export ROS_MASTER_URI=http://10.1.2.2:11311
export EXP_NODE=2
roslaunch exploring_node.launch
```

Listing 2: ROS node launch script

```
<launch>
  <node pkg="exploring_node"
  type="live_slam"
  name="node_$(env EXP_NODE)"
  args="image:=/image_raw
  camera_info:=/camera_info"
  output="screen">
    <remap from="/dvslam/graph"
     to="/$(env EXP_NODE)/dvslam/graph"/>
```

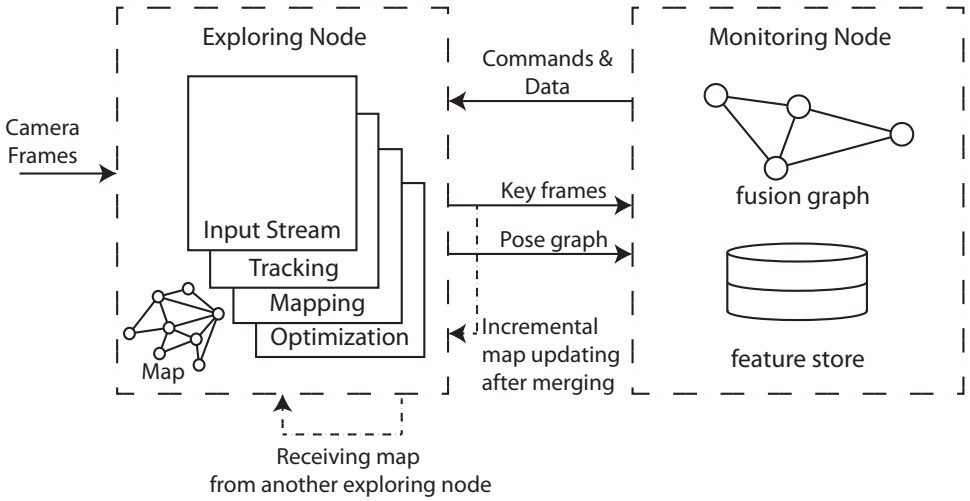

Figure 3: The distributed SLAM framework. Returning dashed arrows represent communication between two exploring nodes.

```
    <remap from="/dvslam/keyframes"
     to="/$(env EXP_NODE)/dvslam/keyframes"/>
  </node>
</launch>
```

3.4 Communication between nodes

Nodes use ROS topics and messages to issue commands and to exchange data. Four topics named, *keyframes*, *graph*, *command* and *map* are used in the framework. These topics are fully-qualified using the namespace */dvslam/*.

ROS topics used in the framework are shown in Table 1. The publisher node writes data and subscriber nodes receive the data via the topic.

Network statistics

A distributed SLAM could easily reach the bandwidth limitations of a network, especially, when dense maps are transferred between nodes. In our distributed framework we could generate statistics by using ROS **Topic Statistics**.

The measurement of traffic volume in bytes of every connection between nodes helps us to identify bandwidth utilization. Furthermore, measurements like number of dropped messages between nodes is a good indication of reaching the bandwidth limit. Other statistics such as mean & standard deviation of the age of messages, and period of messages of all nodes can also be used to investigate the communication between nodes.

4 Experimental Results

4.1 The experimental setup

Our experimental setup is designed to precisely define the ground truth against which the estimated camera positions along the motion path can be measured. To test our framework we mounted a global shutter camera, Point Grey Firefly MV, on a Computer Numeric Control (CNC) machine as shown in Figure 6. We then prepared a $1m \times 1.5m$ table surface containing wooden objects. We then moved the camera above the wooden objects, roughly 4 minutes each time, along known paths and captured both the image stream and the ground truth from the CNC controller. The CNC machine has 0.2mm accuracy in all 3 axes. The camera operated in 640×480 pixel resolution. To capture ground truth from the tiny CNC controller, we wrote an open source ROS node in [Egodagamage, 2017].

4.2 Running experiments

We defined two different 3D paths with 10% overlap, $Path_A$ and $Path_B$ with two different known starting points A and B respectively. Next, we collected four datasets, with two different camera orientations for each path. We created two experiments. In the first experiment, we selected two datasets having parallel camera optical axes. In the second experiment we used remaining two datasets, where the optical axes is 40° to each other. Each dataset is used to deploy an exploring node.

In each experiment, two exploring nodes and one monitoring node are used. These nodes are deployed in three different computers. Once the framework detected map overlaps, we recorded the relative transformation between the maps of exploring nodes. Each experiment is repeated 100 times. The average transformation for each experiment is then compared against the ground truth to compute errors in rotation and translation.

In both experiments, resultant rotation errors in translation

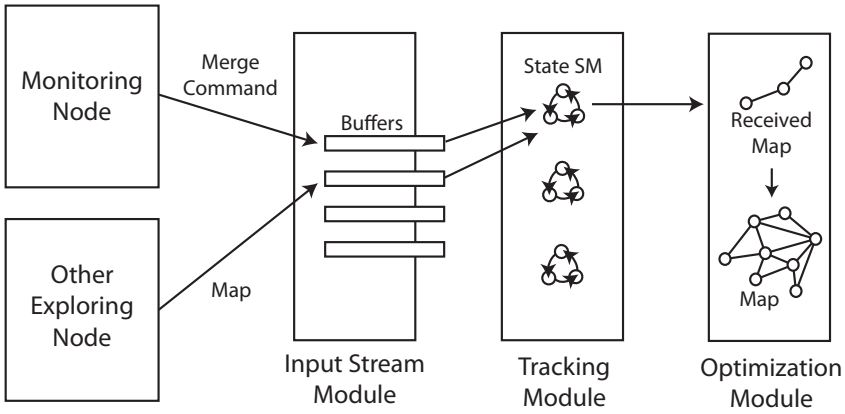

Figure 4: Important components of the map merging process

Topic	Publisher	Subscriber	Description
keyframes	Exploring node	Monitoring node Exploring node	Monitoring nodes always receive key frames from all exploring nodes. After merging maps, exploring nodes exchange key frames between each other.
graph	Exploring node	Monitoring node Exploring node	Monitoring nodes always receive pose graph from all exploring nodes. After merging maps, exploring nodes exchange pose graphs between each other.
command	Monitoring node	Exploring node	All exploring nodes process commands addressed to them and discard others. Commands like map merging, loop closure is received via this topic.
map	Exploring node	Exploring node	During the merge process, relevant exploring nodes subscribe each others' map topic and unsubscribe once done.

Table 1: ROS topics used to communicate between nodes

Experiment	Rotation Error	Translation Error
Experiment 1	3.799°	1.959cm
Experiment 2	5.330°	2.754cm

Table 2: Summary of the experimental results.

and rotation were less than 3cm and 6° respectively as shown in Table 2.

5 Conclusion & Future Work

In this paper, we have introduced a distributed framework for visual SLAM, with no knowledge of relative starting positions of exploring nodes. We have tested our framework using two experiments. Each experiment was repeated 100 times to compute the relative transformation between maps of two exploring nodes. We found that, the error from the ground truth was less than $3cm$ in calculating the translation between maps and less than 6° in calculating the rotation.

As future work, we will be working on supporting more than two exploring nodes. We will also be working on methods to recover random node failures. Moreover, we will be conducting more experiments on network bandwidth utilization.

References

[Bay *et al.*, 2008] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[Davison *et al.*, 2007] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, June 2007.

[Egodagamage, 2017] Ruwan J Egodagamage. An open source ros node for tinyg motion controller. `https://github.com/japzi/rostinyg`, 2017. [Online; accessed in February 2017].

[Engel *et al.*, 2014] Jakob Engel, Thomas Schps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *Computer Vision ECCV 2014*, volume 8690 of *Lecture*

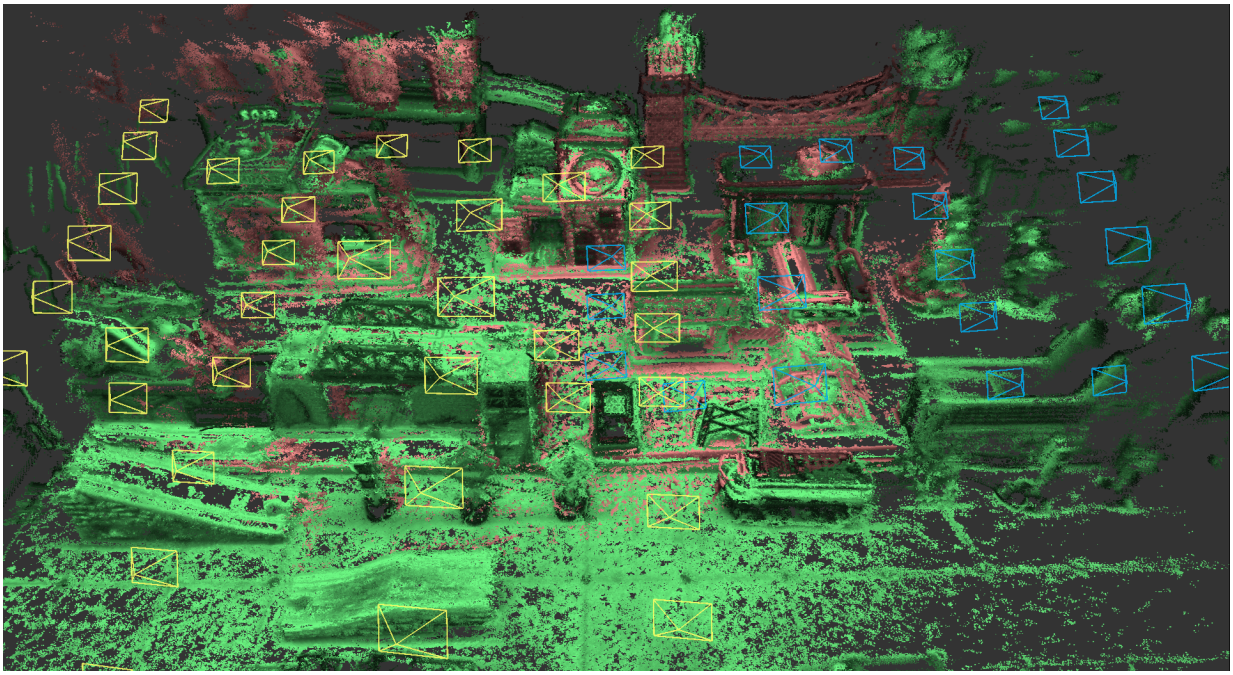

Figure 5: Resultant map of an exploration node after the map merging process. The exploring node's map and keyframes are shown in green and yellow respectively. Received map and keyframes are shown in pink and blue respectively. Constraints of the pose graph is not shown for clarity.

Figure 6: Experimental setup showing a camera mounted on a CNC machine allowing us to capture ground truth information.

Notes in Computer Science, pages 834–849. Springer International Publishing, 2014.

[Faugeras and Lustman, 1988] Olivier D Faugeras and Francis Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(03):485–508, 1988.

[Fischler and Bolles, 1981] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[Fox *et al.*, 2006] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, July 2006.

[Gamage and Tuceryan, 2016] R. Gamage and M. Tuceryan. An experimental distributed framework for distributed simultaneous localization and mapping. In *2016 IEEE International Conference on Electro Information Technology (EIT)*, pages 0665–0667, May 2016.

[Howard *et al.*, 2006] Andrew Howard, LynneE. Parker, and GauravS. Sukhatme. The sdr experience: Experiments with a large-scale heterogeneous mobile robot team. In Jr. Ang, MarceloH. and Oussama Khatib, editors, *Experimental Robotics IX*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 121–130. Springer Berlin Heidelberg, 2006.

[Klein and Murray, 2007] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 225–234, Nov 2007.

[Lowe, 2004] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[Nettleton *et al.*, 2006] Eric Nettleton, Sebastian Thrun, Hugh Durrant-Whyte, and Salah Sukkarieh. Decentralised

slam with low-bandwidth communication for teams of vehicles. In *Field and Service Robotics*, pages 179–188. Springer, 2006.

[Newcombe *et al.*, 2011] Richard A. Newcombe, S.J. Lovegrove, and A.J. Davison. Dtam: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327, Nov 2011.

[Paull *et al.*, 2015] L. Paull, Guoquan Huang, M. Seto, and J.J. Leonard. Communication-constrained multi-auv cooperative slam. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 509–516, May 2015.

[Quigley *et al.*, 2009] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[Sorkine-Hornung and Rabinovich, 2017] Olga Sorkine-Hornung and Michael Rabinovich. Least-squares rigid motion using svd. 2017. Available at `https://igl.ethz.ch/projects/ARAP/svd_rot.pdf`.

[Sturm and Triggs, 1996] Peter Sturm and Bill Triggs. A factorization based algorithm for multi-image projective structure and motion. In *European conference on computer vision*, pages 709–720. Springer, 1996.