Imperial College
London

PROJECT INTERIM REPORT

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

ElasticFusion on the Google Tango Tablet

Author:
Jiahao Lin (CID: 00837321)

Date: January 23, 2017

1 Abstract

In recent years, Google has released an augmented reality Computing platform, Tango[1]. Supporting by with this platform, Google's Tango tablet is an Android device that equip with sensors that enable computer vision functionalities, Including NVIDIA Tegra K1 processor, 4MP color camera, fisheye-lens (motion-tracking) camera, Inertial Measurement Unit (IMU), and Infra-Red projector with RGB-IR camera for integrated depth sensing. Given this cheap commodity mobile hardware and software, a fully self-contained 3D dense surface reconstruction solution can be built by applying state-of-the-art RGB-D SLAM (Simultaneous Localization And Mapping) algorithms on the hardware. This means that high quality 3D room-scale scanning using portable device in real time would become possible.
In the recently published dense RGB-D SLAM algorithms, ElasticFusion[1] published by Dyson Robotics Lab[2], is one that achieve state-of-the-art performance by applying local and global loop closures using a deformation graph, therefore drifts raised by accumulated errors can be recovered and global consistency of the map is maintained. This is particularly useful in 3D surface reconstruction. Also, the current code exist for ElasticFusion made heavily use of GPU programming, in which CUDA is mainly used in the tracking process, which is supported by the NVIDIA Tegra K1 processor on Google tango tablet to enable fast calculation. The fact Elastic Fusion scored high marks on benchmarks for trajectory estimation and surface estimation makes it perfect to be applied on Google Tango Tablet.

2 Introduction (1-3 pages)

2.1 Project Objectives

The objectives of this project is to develop a fully self-contained mobile dense 3D surface reconstruction solution running on Google Tango Tablet. The 3D colored depth map should be rendered on screen in real time from the tablet's viewpoint, and after the 3D scanning stopped the 3D map created should be able saved on device to allow for later use or shown on other devices. The main point of this solution is to produce high quality reconstructed surface map that avoids misalignment, which is often caused by loopy motion of the scanning device during the process. When doing 3D reconstruction, simple fusion of 3D point clouds would produce drifts caused by accumulating errors during the process, therefore by applying ElasticFusion algorithm to the solution, even extremely loopy trajectories of the tablet and long duration of scanning process will not decrease accuracy of the map. On the contrary, revisit areas that has previously been mapped will only help maintain the global consistency of the map by ElasticFusion Algorithm. The part of algorithm that made this possible is the local and global loop closure checking after camera tracking process, which will be discussed in detail later. In our solution, we intend to use Android Native

[1] `https://get.google.com/tango/`
[2] `http://www.imperial.ac.uk/dyson-robotics-lab/`

Development Kit (NDK) to allow the use of Tango C API, which makes it possible to reuse some of the existing code for ElasticFusion. If the development of the solution is successful, an augmented reality application can be created as demo to illustrate the interaction of virtual object with scanned 3D surface map.

3 Background (10- 20 pages)

3.1 Related Work (3 pages)

3.1.1 SLAM basic

The paper: past present and future of SLAM [5] gave a survey about the current development state of modern SLAM, including standard formulation and models, different methods used in SLAM. Below we give a brief summary of basic knowledge about SLAM from this paper.
Simultaneous Localization and Mapping (SLAM) means constructing the model of environment (map) that the robot is in, and estimating the robot state within the environment at the same time. Usually the robot is equipped with some kind of sensor: RGB camera, depth sensor, IMU, or GPS, So that the robot is able to perceive the environment in some way from these sensor. In standard models, the robot state includes its position and orientation, velocity, sensor biases and calibration parameter. Using the robot status and data read from sensors, the environment(map) constructed could be representation in different forms. With the existence of the map, errors raised in estimating robot state could be eliminated and also doing ”loop closure” when robot revisits a place in the map allows drift to be corrected and hence improve accuracy of the estimated status of robot.

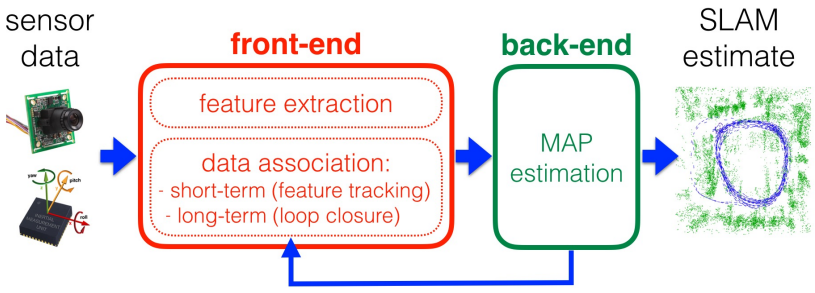

Figure 1: Front-end and back-end in a typical SLAM system[5]

The structure of SLAM system is mainly divided into two parts: front-end and back-end The front-end extract features from sensor data and associate them to the model used to make predictions, then the back-end does estimation of robot state and the map using this model produced by front-end. As you can see in the figure 1, this process forms a continuous process of tracking and update.
Classical formulations of SlAM is to take probabilistic approaches such as models based on Maximum Likelihood Estimation, Maximum-a-posteriori (MAP) estimation,

EKF(Extended Kalman Filters), and particle filters.
Many of the popular SLAM algorithm models it as a maximum-a-posteriori estimation problem by taking probability approach. And usually using a factor graph to show relationships and constraints between variables. In these models, often an observation or measurement model is used to represent the probability of observe measurements given the state of robot, given some random measurement noise that usually model to be under a zero-mean Gaussian distribution. By using Bayes theorem, the MAP estimate, in which the posterior is the probability of robot state given a measurement, can be calculated given some prior belief of the robot state. In the case that there is no prior information known about robot state, the MAP estimate would simply become a maximum-likelihood-estimate. When we are given a set of independent robot states and measurements, these variable would become factors(nodes) in the factor graph, and their probabilistic relationships would become the constraints(edges) in the factor graph.
Factor graph allows us to visualize our problem in a simple way and provides an insight into how the variables are related together by their constraints. 2 from the paper have shown an example of it for a simple SLAM problem.

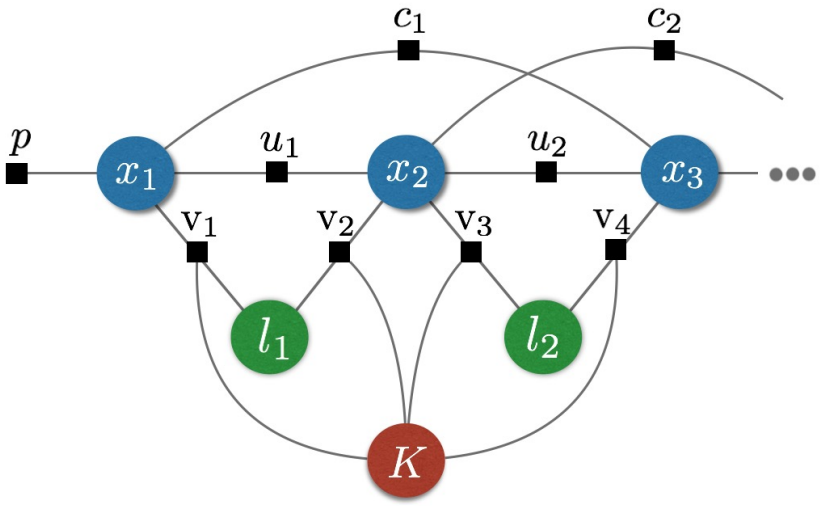

Figure 2: SLAM as a factor graph[5]

As explained in [5]: "Blue circles denote robot poses at consecutive time steps ($x1$, $x2$ ...), green circles denote landmark positions ($l1$, $l2$ ...), red circle denotes the variable associated with the intrinsic calibration parameters (K). Factors are shows are black dots: the label "u"marks factors corresponding to odometry constraints, "v "marks factors corresponding to camera observations, "c"denotes loop closures, and "p"denotes prior factors."

When calculating the MAP estimate of robot state, we can transform the problem into minimizing the negative log of posterior, assuming the noise is zero-mean Gaussian distributed, this becomes a non-linear least squares problem, since we will be minimising the measurement error's $l2$ norm. This method is similar to the BA (Bun-

dle Adjustment) method in computer vision area, which minimise the reprojection error. However factor graph can not only contain visual and geometry factor, but also a variety of data from different sensors. Also the MAP estimate is calculated every time a measurement is arrived, instead of performing calculation after all the data is given.
Different from factor graph optimization, a similar approach is called pose graph optimization, which estimate trajectory of robot using measurements between relative poses as constraints, and optimized these poses to achieve better accuracy trajectory.
Other than optimization-based estimation, some filtering approach using EKF model have also achieve great results, such as MSCKF(Multi-State Constraint Kalman Filter of Mourikis and Roumeliotis)[**?**], EKF is used for performing filtering on the linearised point of estimate, and have shown to be accurate in visual-inertial navigation.
In terms of the representation of the map, one way is to have a group of features or landmarks in 3D, these features are usually extracted from sensor images to be allow easy distinguishable from environment, such as edges and corners. The method used to extracting features from images are well-established within computer vision area, using descriptors like ROB, SIFT, and SURF. The discriminative and repetitive properties of landmarks allow the correspondence of each landmark and sensor measurement(image), and then via triangulating, the robot is able to compute the relative pose between measurements and geometric information about landmarks, therefore achieve localization and mapping.

Other than feature based approach, sometimes raw dense representation of the environment is used. different than extracting landmarks, this approach stores a dense 3D map using high resolution points, the 3D geometry of environment is described by these vast amount of points, so called point clouds. With stereo or depth cameras, lase r depth sensors, 3D information of the points can be easily obtained, therefore this method is popular in RGB-D SLAM. Sometimes, these points doesn't only stores simple 3D information, in Elasticfusion[1], each 3D point is represented by a surfel, which is a disk with certain radius centred by the 3D point and stored information such as normal vector of that point on the surface and color. These suefel disk combine together can encode the geometry of environment better than simple points.
At more a higher level than raw, unstructured points, dense related element that partitioned by space is also used to represent surface and boundaries, such as surface mesh models(connected polygons formed by points) and implicit surface representations like octree and occupancy grid. KinectFusion[4] storing surfaces using volumetric representation by uniformly subdivide a 3D physical space into a 3D grid of fixed size voxels, voxels defining surface has a zeros value of the truncated signed-distance function (TSDF) function that stores distance information relative to the surface. We can see that represent the environment in these types of dense data requires huge amount of storage, however they give very low level geometry information which is suitable for 3D reconstruction and rendering.

Comparing feature-based and dense SLAM methods, for the purpose of this project,

we will use dense SLAM algorithm such as Elasticfusion[1] for 3D reconstruction, since it maintains a dense 3D surfel map that make use of all the information about the environment from sensor thus more robust for localization and mapping.

3.1.2 Kinectfusion

KinectFusion[4] is a dense RGB-D SLAM method that enables high quality 3D reconstruction and interaction in real time. This method make use of mobile low cost RGB-D camera, scanning the environment in real time and acquire 3D point information about the space. During the reconstructing process, holes caused by absence of depth measurement in original 3D point cloud is filled, and model is more refined by repetitive scanning over time.
Firstly the 3D points reading are then stored in a 3D vertex map and their normal vectors are calculated by reprojection of neighbouring vertexes. After that the camera pose is tracked by using Iterative Closest Point (ICP) algorithm to align current 3D points with the previous frame. In ICP algorithm, first the corresponding point of each current 3D point is found by doing projective data association, which reprojects each previous point into image coordinates and use this to look up current vertexes that may be correspondence points. In the second part these correspondences are tested by calculated distance and angle between them and reject any outliers that are beyond certain threshold. Please be noted that the running of ICP in real time is because the use of GPU to parallel processing a current vertex per thread, according to the original paper.

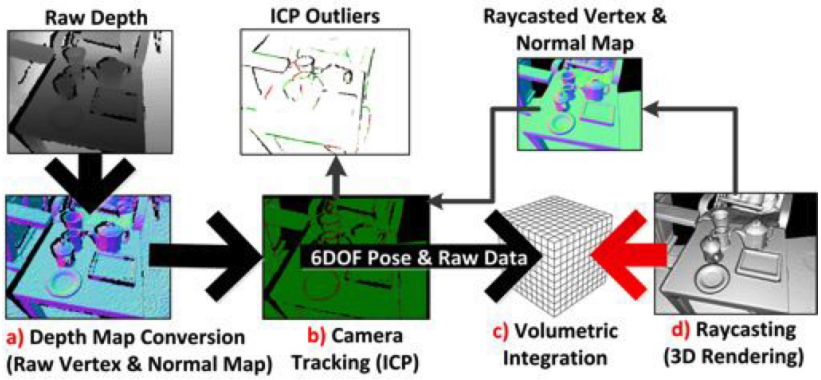

Figure 3: Overview of tracking and reconstruction pipeline from raw depth map to rendered view of 3D scenem[4]

The structure of KinecFusion is given in Fig 7 from original paper. The result of ICP is a transformation matrix from previous camera pose to the current pose, which is later used for converting the coordinates of current vertexes into the global coordinate frame. These coordinate data is then fused into the current model that uses a volumetric representation, as mentioned before. The 3D space is uniform divided into fixed size voxel grid. The geometry of surface is implicitly stores in these voxels by having zero values of Truncated Signed Distance Functions (TSDFs) at the

surface, positive value in the front of surface, and negative at the back of it. Also, to improve efficiency and minimise storage usage only a truncated area around the surface is actually stored, thus the name ”truncated”.
When integrating the current vertexes into voxel grid, each voxel is perspective projected back onto the image to calculate the difference between it and the measurement depth of this coordinate. The distance difference is then normalized to maximum TSDF distance and updated using weighted average with previous value. Similar to the implementation of tracking stage, the integration and update process also make use of GPU parallel processing, each thread is assigned a slice of of grid along the $Z - axis$, and sweep through each voxel by its (x, y) position for processing.
Finally, in order to render the 3D reconstructed surface raycasting is performed. This is done by having each GPU thread to follow a ray from the camera center along one pixel in the image into the voxel grid in camera coordinate space, the ray is extended until a zero-crossing voxel is met, this voxel represents the surface point observed by this pixel and the position is extracted by doing a trilinear interpolation on this voxel point. And the normal at this position is calculated by find the surface gradent of TSDF values around the position. With these information for each pixel, a live 3D reconstruction view from camera view point can be rendered using camera's pose and intrinsic information.

3.1.3 Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization

Other than using depth sensor with RGB camera, another kind of sensor equipped on the Google Tango table is Inertial Measurement Unit (IMU). Okvis[2] is a sparse feature-based SLAM approach that tightly coupled IMU measurement with visual measurement using non linear optimization.
In visual SLAM, a nonlinear optimization is formulated to find the camera poses and landmark positions by minimizing the reprojection error of landmarks observed in camera frames. Figure 2 shows the respective graph representation : it displays measurements as edges with square boxes and estimated quantities as round nodes. As soon as inertial measurements are introduced, they not only create temporal constraints between successive poses, but also between successive speed and IMU bias estimates of both accelerometers and gyroscopes by which the robot state vector is augmented. In this section, we present our approach of incorporating inertial measurements into batch visual SLAM.
Keypoint Matching and Keyframe Selection Our processing pipeline employs a customized multiscale SSE-optimized Harris corner detector combined with BRISK descriptor extraction [12]. The detector enforces uniform keypoint distribution in the image by gradually suppressing corners with weaker score as they are detected at a small distance to a stronger corner. Descriptors are extracted oriented along the gravity direction (projected into the image) which is observable thanks to tight IMU fusion. Initially, keypoints are stereo-triangulated and inserted into a local map. We perform brute-force matching against all of the map landmarks; outlier rejection is simply performed by applying a chi-square test in image coordinates by using the (uncertain) pose predictions obtained by IMUintegration. There is no costly RANSAC step involved another advantage of tight IMU involvement. For the sub-

we distinguish two kinds of frames: we introduce a temporal window of the S most recent frames including the current frame; and we use a number of N keyframes that may have been taken far in the past. For keyframe selection, we use a simple heuristic: if the ratio between the image area spanned by matched points versus the area spanned by all detected points falls below 50 to 60
tightly integrate visual measurements with readings from an Inertial Measurement Unit (IMU) in SLAM. An IMU error term is integrated with the landmark reprojection error in a fully probabilistic manner, resulting to a joint non-linear cost function to be optimized. Employing the powerful concept of keyframes we partially marginalize old states to maintain a bounded-sized optimization window, ensuring real-time operation. Comparing against both vision-only and loosely-coupled visual-inertial algorithms, our experiments confirm the benefits of tight fusion in terms of accuracy and robustness.

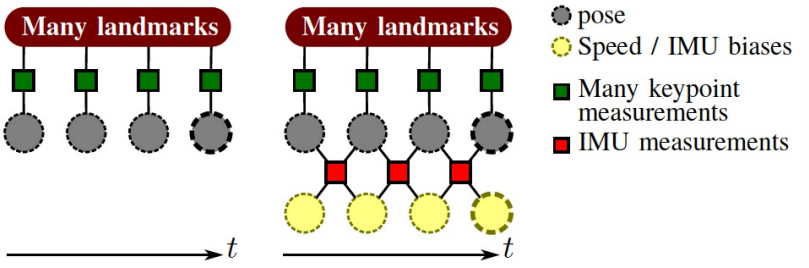

Figure 4: okvis[2]

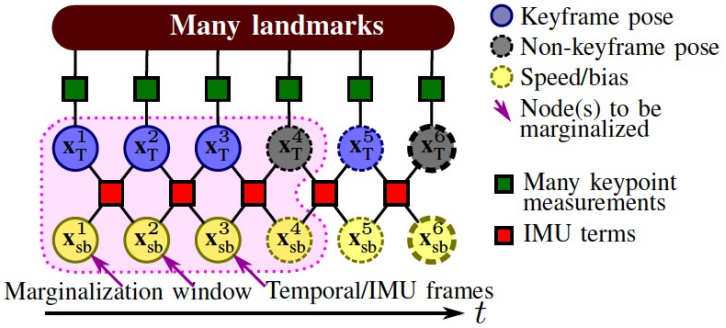

Figure 5: okvis[2]

Maintaining a relatively sparse graph of keyframes and their associated landmarks subject to nonlinear optimization, has since been very popular tightly-coupled approaches jointly estimate all sensor states. In order to be tractable and as an alternative to filtering, Dong-Si and Mourikis [2] propose a fixed-lag smoother, where a window of successive robot poses and related states is maintained, marginalizing out states (following [19]) that go out of scope. A similar approach, but without inertial terms and in the context of planetary landing is used in [16]. With the aim of robust and accurate visual-inertial SLAM, we advocate tightly-coupled fusion for maximal

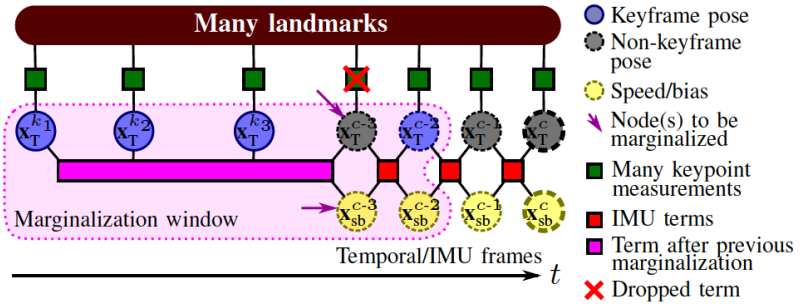

Figure 6: okvis[2]

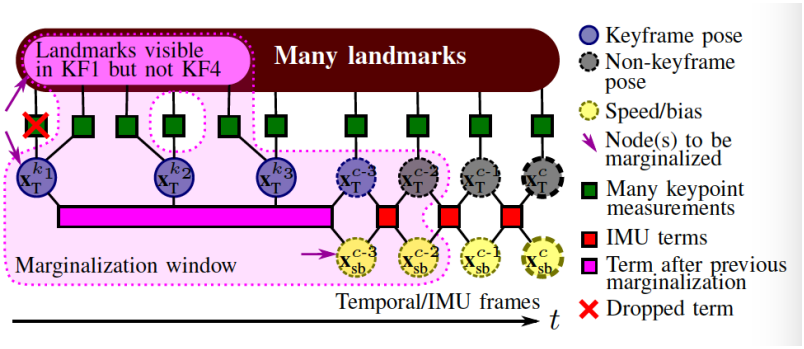

Figure 7: okvis[2]

exploitation of sensing cues and nonlinear estimation wherever possible rather than filtering in order to reduce suboptimality due to linearization. Our method is inspired by [17], where it was proposed to use IMU error terms in batch-optimized SLAM (albeit only during initialization). Our approach is Our approach is closely related to the fixed-lag smoother proposed in [2], as it combines inertial terms and reprojection error in a single cost function, and old states get marginalized in order to bound the complexity. In relation to these works, we see a threefold contribution: 1) We employ the keyframe paradigm for drift-free estimation also when slow or no motion at all is present: rather than using an optimization window of time-successive poses, we keep keyframes that may be spaced arbitrarily far in time, keeping visual constraintswhile still respecting an IMU term. Our formulation of relative uncertainty of keyframes allows for building a pose graph without expressing global pose uncertainty, taking inspiration from RSLAM [13]. 2) We provide a fully probabilistic derivation of IMU error terms, including the respective information matrix, relating successive image frames without explicitly introducing states at IMU-rate. 3) At the system level, we developed both the hardware and the algorithms for accurate real-time SLAM, including robust keypoint matching and outlier rejection using inertial cues. In the remainder of this article, we introduce the inertial error term in batch visual SLAM in II-B, followed by an overview our real-time stereo image processing and keyframe selection in II-C, and the marginalization formalism in II-D. Finally, we show results obtained with our stereo-vision and IMU sensor indoor and outdoor in III.

Streaming SIMD Extensions (SSE) optimised Harris corner detector brisk descriptor

BRISK: Binary Robust Invariant Scalable Keypoints
keyframe-based visual-inertial SLAM [2]

3.1.4 Elasticfusion

Elastic Fusion [1]

random ferns for keyframe encoding [3]

past present and future of SLAM [5]

4 Project Plan (1-2 pages)

You should explain what needs to be done in order to complete the project and roughly what you expect the timetable to be. Dont forget to include the project write-up (the final report), as this is a major part of the exercise. Its important to identify key milestones and also fall-back positions, in case you run out of time. You should also identify what extensions could be added if time permits. The plan should be complete and should include those parts that you have already addressed (make it clear how far you have progressed at the time of writing). This material will not appear in the final report.

May 19 - June 19 :Final Report June 26 : Preliminary Source Code Archive July 4 :Final Project Archive

5 Evaluation plan (1-2 pages)

–
Project evaluation is very important, so it's important to think now about how you plan to measure success. For example, what functionality do you need to demonstrate? What experiments to you need to undertake and what outcome(s) would constitute success? What benchmarks should you use? How has your project extended the state of the art? How do you measure qualitative aspects, such as ease of use? These are the sort of questions that your project evaluation should address; this section should outline your plan.

ElasticFusion [1]
keyframe-based visual-inertial SLAM [2]
random ferns for keyframe encoding [3]
kinectfusion [4]
past present and future of SLAM [5]

References

[1] Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. Elasticfusion: Dense slam without a pose graph. *Proc. Robotics: Science and Systems, Rome, Italy*, 2015. pages 2, 5, 6, 10, 11

[2] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015. pages 7, 8, 9, 10, 11

[3] Ben Glocker, Jamie Shotton, Antonio Criminisi, and Shahram Izadi. Real-time rgb-d camera relocalization via randomized ferns for keyframe encoding. *IEEE transactions on visualization and computer graphics*, 21(5):571–583, 2015. pages 10, 11

[4] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011. pages 5, 6, 11

[5] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. pages 3, 4, 10, 11