Visual Odometry

Christos Kokas

May 2022

Contents

1 Introduction

In robotics and computer vision, visual odometry is the process of determining the position and orientation of a robot by analyzing images from the robot's camera. It has been used in a wide variety of robotic applications, such as on the Mars Exploration Rovers [1].

The accurate localization of the robot poses a great challenge for many applications, even more in GPS-Denied environments (e.g. indoors) where GPS systems, such as RTK GPS, have difficulty producing an accurate estimate of the position of the robot. Visual Odometry has gained popularity because it can produce accurate estimates of the position of the camera even in GPS-Denied environments and at the same time be cheaper than other alternatives (e.g. RTK GPS, LiDARS). Combinations of Visual Odometry and other techniques or products, such as Visual Odometry combined with an IMU (also know as Visual Inertial Odometry) or Visual Odometry combined with Deep Learning, can also yield satisfying results.

2 Definitions/Appendix

Bundle Adjustment is simultaneous refining of the 3D coordinates describing the scene geometry, the parameters of the relative motion, and the optical characteristics of the camera(s) employed to acquire the images, given a set of images depicting a number of 3D points from different viewpoints. It amounts to an optimization problem on the 3D structure and viewing parameters (i.e., camera pose and possibly intrinsic calibration and radial distortion), to obtain a reconstruction which is optimal under certain assumptions regarding the noise pertaining to the observed image features.

Reprojection Error is a geometric error corresponding to the image distance between a projected point and a measured one.

Pose Graph contains nodes connected by edges that represents the pose of the robot.

Bayesian inference is a method of statistical inference in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

Voxel each of an array of elements of volume that constitute a notional three-dimensional space, especially each of an array of discrete elements into which a representation of a three-dimensional object is divided.

Euclidean clustering groups points that are close together. You must set a ”closeness” threshold, such that points within this threshold are considered to be part of the same cluster.

DBoW2 DBoW2 is an open source C++ library for indexing and converting images into a bag-of-word representation. It implements a hierarchical tree for approximating nearest neighbours in the image feature space and creating a visual vocabulary.

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates.

Parallax Parallax is a displacement or difference in the apparent position of an object viewed along two different lines of sight, and is measured by the angle or semi-angle of inclination between those two lines.

Keyframe is a drawing or shot that defines the starting and ending points of any smooth transition.

Structure from motion (SfM) is a photogrammetric range imaging technique for estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals.

3 Literature Review

In this section papers using Visual Odometry will be presented each using a different method of producing an accurate estimate of the camera.

3.1 ORB_SLAM2

3.1.1 Abstract

We present ORB-SLAM2 a complete SLAM system for monocular, stereo and RGB-D cameras, including map reuse, loop closing and relocalization capabilities. The system works in real-time on standard CPUs in a wide variety of environments from small hand-held indoors sequences, to drones flying in industrial environments and cars driving around a city. Our backend based on bundle adjustment with monocular and stereo observations allows for accurate trajectory estimation with metric scale. Our system includes a lightweight localization mode that leverages visual odometry tracks for unmapped regions and matches to map points that allow for zero-drift localization. The evaluation on 29 popular public sequences shows that our method achieves state-of-the-art accuracy, being in most cases the most accurate SLAM solution. We publish the source code, not only for the benefit of the SLAM community, but with the aim of being an out-of-the-box SLAM solution for researchers in other fields [2].

3.1.2 System Overview

ORB-SLAM2 for stereo and RGB-D camera has three main parallel threads.

- Find Feature Matches to the local map with every frame to localize the camera, and apply motion-only Bundle Adjustment (BA) to minimize the reprojection error. (Tracking)
- Perform local BA to manage and optimize the local map. A local map is a simplified representation of the immediate environment around the robot. (Local Mapping)
- Detect large loops and correct the accumulated drift by performing a pose-graph optimization. (Loop Closure)

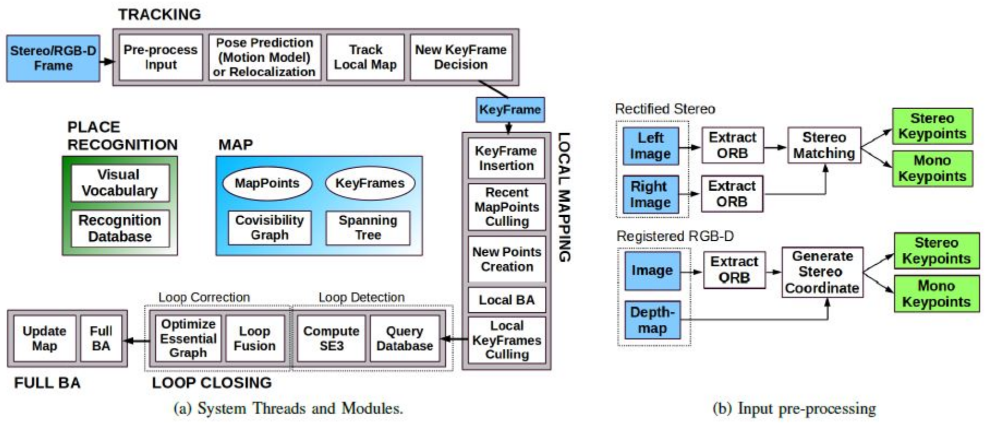

Figure 3.1: ORB-SLAM2 Diagram of Parallel Threads

3.1.3 Key Points

ORB-SLAM2's key points include :

- ORB-SLAM2 treats differently close and far points, first described in the work of Paz et al. [3], which indicates that the points' depth cannot be reliably estimated due to little disparity in the stereo camera. If the points' depth is more than ~ 40 times the stereo baseline it cannot be reliably estimated.

- Stereo keypoints (points that are found both in left and right image) contribute to distance from the object as well as rotation and translation estimation. On the other hand, Monocular keypoints (points that are only found in one of the cameras) are used for rotation and translation estimation only.

- For camera pose optimization the Levenberg–Marquardt method is implemented in g2o [4].

- Full BA optimization is used to achieve the optimal solution. This optimization is very costly and therefore is performed on a separate thread allowing the system to continue creating map and detecting loops. If a loop is detected before the Full BA finishes, it is aborted, the loop is closed, and the Full BA optimization is launched again. To merge the updated subset of keyframes and points optimized by the full BA with the non-updated keyframes and points that where inserted while the optimization was running is done by propagating the correction of updated keyframes (i.e. the transformation from the non-optimized to the optimized pose) to non-updated keyframes through the spanning tree. Non-updated points are transformed according to the correction applied to their reference keyframe.

3.2 Hydra: A Real-time Spatial Perception Engine for 3D Scene Graph Construction and Optimization

3.2.1 Abstract

3D scene graphs have recently emerged as a powerful high-level representation of 3D environments. A 3D scene graph describes the environment as a layered graph where nodes represent spatial concepts at multiple levels of abstraction (from low-level geometry to high-level semantics including objects, places, rooms, buildings, etc.) and edges represent relations between concepts. While 3D scene graphs can serve as an advanced "mental model" for robots, how to build such a rich representation in real-time is still uncharted territory.

This paper describes the first real-time Spatial Perception engINe (SPIN), a suite of algorithms to build a 3D scene graph from sensor data in real-time. Our first contribution is to develop real-time algorithms to incrementally construct the layers of a scene graph as the robot explores the environment; these algorithms build a local Euclidean Signed Distance Function (ESDF) around the current robot location, extract a topological map of places from the ESDF, and then segment the places into rooms using an approach inspired by community-detection techniques. Our second contribution is to investigate loop closure detection and optimization in 3D scene graphs. We show that 3D scene graphs allow defining hierarchical descriptors for loop closure detection; our descriptors capture statistics across layers in the scene graph, ranging from low-level visual appearance, to summary statistics about objects and places. We then propose the first algorithm to optimize a 3D scene graph in response to loop closures; our approach relies on embedded deformation graphs to simultaneously correct all layers of the scene graph. We implement the proposed SPIN into a highly parallelized architecture, named Hydra, that combines fast early and midlevel perception processes (e.g., local mapping) with slower highlevel perception (e.g., global optimization of the scene graph). We evaluate Hydra on simulated and real data and show it is able to reconstruct 3D scene graphs with an accuracy comparable with batch offline methods, while running online [5].

3.2.2 Building 3D scene Graphs

A 3D scene graph is a layered graph where nodes represent spatial concepts at multiple levels of abstraction (from low-level geometry to objects, places, rooms, buildings, etc.) and edges represent relations between concepts.

- Euclidean Signed Distance Function (ESDF) : From a field of booleans and produces a field of scalars such that each value in the output is the distance to the nearest "true" cell in the input. Unfortunately, ESDFs scale poorly in the size of the environment [6].

This paper focuses on indoor environments and adopts the 3D scene graph introduced in [7]. The 3D scene graph consists of 5 layers that represent semantics, objects and agents, places, rooms and lastly a building node connecting all rooms. Edges connect nodes within each layer or across layers.

- Mesh and Objects (Layers 1-2) : The construction of the metric-semantic 3D mesh is an extension of the approach in [8]. The authors in [8] use Voxblox to integrate semantically-labeled point clouds into a monolithic Truncated Signed Distance Field (TSDF) and an ESDF of the environment, while also performing Bayesian inference (Defined in Chapter 2) over the semantic label of each voxel. This paper forms a volumetric model of the robot's surroundings within a radius to bound the memory used by the ESDF. Subsequently, they extract he 3D metric-semantic mesh using Voxblox' marching cubes implementation and the places, and they are passed to the Scene Graph Frontend. They segment objects by performing euclidean clustering (EC) of the 3D meshes. The results of the EC are then used to create a bounding box for each putative object.

- Places (Layer 3) : To extract the subgraph of places the authors of this paper use a Generalized Voronoi Diagram (GVD). After the GVD they identify distinctive points and connect them with edges to form the graph of places.

- Rooms (Layer 4) : The segmentation of rooms was created directly from the sparse subgraph of places. Firstly, by inflating objects, small apertures in the environment (i.e., doors) will gradually close, naturally partitioning the voxel-based map into disconnected components (i.e., rooms). Secondly, with a dilation of the map by a distance δ, every place with obstacle distance smaller than δ will disappear from the graph (since it will no longer be in the free space). A visualization of this idea is given in Figure 3.2.

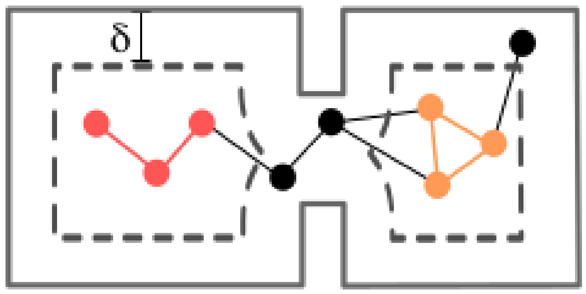

Figure 3.2: Map Dilation for Room Detection

3.2.3 Key Points

This paper overcomes that ESDFs scale poorly depending on the size of the environment by implementing algorithms that reconstruct a local ESDF of the robot's surroundings and incrementally convert the ESDF into a metric-semantic 3D mesh as well as a Generalized Voronoi Diagram, from which a topological graph of places can be quickly extracted. The main contribution of this paper is the development of the first real-time Spatial Perception engINe (SPIN), a suite of algorithms and implementations to build a 3D scene graph from sensor data in real-time.

This paper additionally investigates loop closure detection and optimization in 3D scene graphs. It proposes a hierarchical approach for loop closure detection :

- top-down loop closure detection that uses hierarchical descriptors —capturing statistics across layers in the scene graph— to find putative loop closures
- a bottom-up geometric verification that attempts estimating the loop closure pose by registering putative matches
- optimize a 3D scene graph in response to loop closures, which relies on embedded deformation graphs to simultaneously correct all layers of the scene graph, from the 3D mesh, to places, objects, and rooms.

Lastly the paper proposes a highly paralllelized implementation, named Hydra,that combines fast early and mid-level perception processes (e.g., local mapping) with slower high-level perception (e.g., global optimization of the scene graph).

3.2.4 Loop Closure Detection

Agent nodes describe the robot's trajectory in the 3D scene graph. A keyframe, containing appearance information, is stored for each agent node. Using these agent nodes loop closure detection tries to find a past agent node that matches the last agent node (current robot pose).

For each agent node a hierarchy of descriptors is constructed, that describe statistics of the node's surroundings. The hierarchical descriptors include standard DBoW2 (Described in Chapter 2) appearance descriptor. The appearance descriptor is augmented with an object-based and a place-based descriptor. For loop closure detection, the place-based descriptor is examined and subsequently object-based and then appearance descriptors. If any of the descriptors return a putative result, geometric verification (Described in Paragraph 3.2.5) is performed.

3.2.5 Geometric Verification

After a putative loop closure, bottom-up geometric verification is performed. In particular, whenever there is a match (e.g.,between agent i and agent j) at a given layer, (e.g., between appearance descriptors at the agent layer, or between object descriptors at the object layer), an attempt is made to register frames i and j. For registering visual features standard RANSAC-based geometric verification as in [9] is used. If that fails, TEASER++ [10] is used, discarding loop closures that also fail object registration.

3.2.6 3D Scene Graph Optimization

the Scene Graph Backend (i) optimizes the graph using a deformation graph approach and (ii) postprocesses the results to remove redundant subgraphs corresponding to the robot visiting the same location multiple times.

- Scene Graph Frontend : The frontend builds an initial estimate of the 3D scene graph that is uncorrected for drift. More precisely, the frontend takes as input the latest mesh, places subgraph, objects, and pose graph of the agent. Then, the frontend populates inter-layer edges from each object or agent node to the nearest place node in the active window using nanoflann [11].

- Scene Graph Backend : When a loop closure is detected, the backend optimizes an embedded deformation graph built from the frontend scene graph and then reconstructs the other nodes in the scene graph via interpolation [12].

3.2.7 The Hydra Architecture

The authors the this paper implement their spatial perception engine into a highly parallelized architecture, named Hydra. Hydra involves a combination of processes that run at sensor rate (e.g., feature tracking for visual-inertial odometry), at sub-second rate (e.g., mesh and place reconstruction), and at slower rates (e.g., the scene graph optimization, whose complexity depends on the map size). Therefore these processes have to be organized such that slow-but-infrequent computation (e.g., scene graph optimization) does not get in the way of faster processes. The visualization of Hydra is presented in Figure **??**

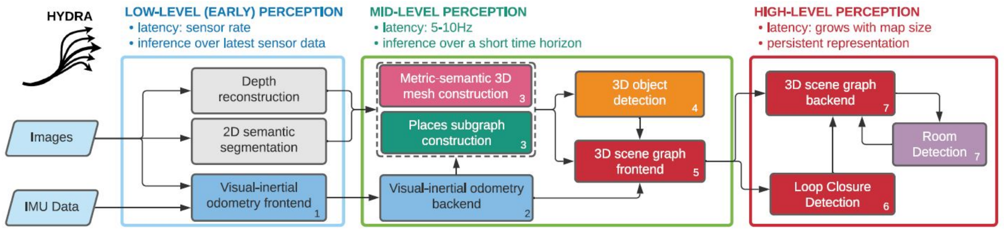
HYDRA
LOW-LEVEL (EARLY) PERCEPTION
• latency: sensor rate
• inference over latest sensor data
MID-LEVEL PERCEPTION
• latency: 5-10Hz
• inference over a short time horizon
HIGH-LEVEL PERCEPTION
• latency: grows with map size
• persistent representation
Images
IMU Data
Depth reconstruction
2D semantic segmentation
Visual-inertial odometry frontend 1
Metric-semantic 3D mesh construction 3
Places subgraph construction 3
Visual-inertial odometry backend 2
3D object detection 4
3D scene graph frontend 5
3D scene graph backend 7
Room Detection 7
Loop Closure Detection 6

Figure 3.3: Hydra's Functional Blocks

3.3 VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator

3.3.1 Abstract

One camera and one low-cost inertial measurement unit (IMU) form a monocular visual-inertial system (VINS), which is the minimum sensor suite (in size, weight, and power) for the metric six degrees-of-freedom (DOF) state estimation. In this paper, we present VINS-Mono: a robust and versatile monocular visual-inertial state estimator. Our approach starts with a robust procedure for estimator initialization. A tightly coupled, nonlinear optimization-based method is used to obtain highly accurate visual-inertial odometry by fusing preintegrated IMU measurements and feature observations. A loop detection module, in combination with our tightly coupled formulation, enables relocalization with minimum computation. We additionally perform 4- DOF pose graph optimization to enforce the global consistency. Furthermore, the proposed system can reuse a map by saving and loading it in an efficient way. The current and previous maps can be merged together by the global pose graph optimization. We validate the performance of our system on public datasets and real-world experiments and compare against other state-of-the-art algorithms. We also perform an onboard closedloop autonomous flight on the microaerial-vehicle platform and port the algorithm to an iOS-based demonstration. We highlight that the proposed work is a reliable, complete, and versatile system that is applicable for different applications that require high accuracy in localization. We open source our implementations for both PCs (`https://github.com/HKUST-Aerial-Robotics/VINS-Mono`) and iOS mobile devices (`https://github.com/HKUST-Aerial-Robotics/VINS-Mobile`) [13].

3.3.2 Issues

- Rigorous Initialization : Difficult to fuse the monocular visual structure with inertial measurements.
- VINSs are highly non-linear : challenges in terms of estimator initialization. (In most cases, the system should be launched from a known stationary posi- tion and moved slowly and carefully at the beginning, which limits its usage in practice.)
- Long-Term Drift : unavoidable for visual-inertial odometry (VIO). In order to eliminate the drift, loop detection, relocalization, and global optimization has to be developed.

3.3.3 VINS-Mono Features

- robust initialization procedure that is able to bootstrap the system from unknown initial states
- tightly coupled, optimization-based monocular VIO with camera–IMU extrinsic calibration and IMU bias correction

- online relocalization and four degrees-of-freedom (DOF) global pose graph optimization
- pose graph reuse that can save, load, and merge multiple local pose graphs

The IMUs usually acquire data at a much higher rate than the camera. Different methods have been proposed to handle the high-rate IMU measurements. The most straightforward approach is to use the IMU for state propagation in EKF-based approaches [14], [15].

In a graph optimization formulation, an efficient technique called IMU preintegration is developed in order to avoid the repeated IMU reintegration. This technique was first introduced in [16], which parameterize the rotation error using Euler angles. Shen et al. [17] derived the covariance propagation using continuous-time error-state dynamics. The preintegration theory was further improved in [18] and [19] by adding posterior IMU bias correction.

The structure of the monocular visual-inertial state estimator is shown in Figure 3.4.

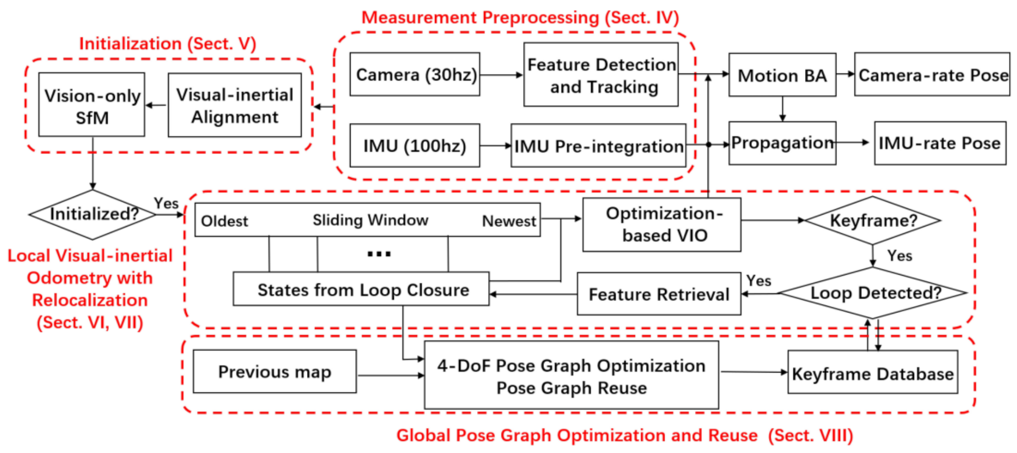

Figure 3.4: Block diagram illustrating the full pipeline of monocular VINS

Existing features are tracked by the KLT sparse optical flow algorithm [20]. Corner features are detected [21] to maintain a minimum number (100– 300) of features in each image. Outlier rejection is performed using RANSAC with a fundamental matrix model [22].

Keyframe selection is based on two criteria :

- the average parallax apart from the previous keyframe. If the average parallax of tracked features is between the current frame and the latest keyframe is beyond a certain threshold, the frame is treated as a new keyframe.
- Tracking quality : If the number of tracked features goes below a certain threshold, The frame is treated as a new keyframe.

The initialization procedure starts with a vision-only SfM to estimate a graph of up-to-scale camera poses and feature positions. First, feature correspondences between the latest frame and all previous frames are checked. The relative rotation and up-to-scale translation between these two frames is recovered using the five-point algorithm [23]. Subsequently, the scale is set arbitrarily and all features observed in these two frames are triangulated. Based on these triangulated features, a perspective-n-point (PnP) method [24] is performed to estimate poses of all other frames in the window. Finally, a global full bundle adjustment [25] is applied to minimize the total reprojection error of all feature observations.

In order to bound the computational complexity of the optimization-based VIO, marginalization is incorporated. The marginalization is carried out using the Schur complement [26].

Place Recognition using DBoW2 [27]. 500 more corners are detected and described by the BRIEF descriptor [28]. All BRIEF descriptors are kept for feature retrieving, but the raw image is discarded to reduce the memory consumption.

Benefiting from the inertial measurement of the gravity, the roll and pitch angles are fully observable in the VINS. To take full advantage of valid information and correct drift efficiently, we fix the drift-free roll and pitch, and only perform pose graph optimization in 4-DOF.

Meaning that because the gravity is known the 2 axes roll and pitch are known from the gravity measurement.

Keyframes are added into the pose graph after the VIO process. Every keyframe serves as a **vertex** in the pose graph, and it connects with other vertexes by two types of edges.

- **Sequential Edge** : A keyframe establishes several sequential edges to its previous keyframes. A sequential edge represents the relative transformation between two keyframes, which is taken directly from VIO.

- **Loop-Closure Edge** : If the keyframe has a loop connec- tion, it connects the loop-closure frame by a loop-closure edge in the pose graph.

Although the tightly coupled relocalization already helps with eliminating wrong loop closures, another **Huber norm** [29] is added to further reduce the impact of any possible wrong loops. The pose graph optimization and relocalization run asynchronously in two separate threads.

Every keyframe is a vertex in the pose graph.

3.4 A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors

3.4.1 Abstract

4 Ideas

- Use 2 zed cameras one on head one on belly pointing the same way (distance known from each other), after feature matching on both compare the save features to have an even more accurate position of the pointcloud.

- Create a Voronoi diagram of the features in each room and recognise places when robot finds features that match the Voronoi diagram (Voronoi Diagram : on a 3D surface with points the Voronoi Diagram computes the least distance of all points creates a diagram of shapes that represent the least distance of each point).

- Euclidean Clustering on sets of features to create layers of objects.

- Relative probability of distance from feature matching according to 40 times baseline paper. according to the distance of the feature take into account how accurate it is, like close far points.

- IMU state propagation (IMU higher rate than camera), IMU preintegration (to avoid the repeated IMU reintegration) (Paragraph 3.3.3).

- IMU Noise and Bias equations are shown in [13] page 4.

- Choose sufficient number of features through research. (papers etc.).
- extrinsic parameters (camera to imu) are shown in [13] page 5.
- check the gravity. It always has value of 9.81 but can be on any axis due to camera movement and it should maove the location of the object at most times. It is addressed in [13] page 5.
- create equations for the translation and rotation of the robot. Some equations are presented in [13] page 6.
- account for different types of cameras [13] page 7.
- sliding widow.

List of Figures

References

[1] Mark Maimone, Yang Cheng, and Larry Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics, Special Issue on Space Robotics*, 24:2007, 2007.

[2] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[3] Lina M. Paz, Pedro PiniÉs, Juan D. TardÓs, and JosÉ Neira. Large-scale 6-dof slam with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957, 2008.

[4] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. G2o: A general framework for graph optimization. pages 3607 – 3613, 06 2011.

[5] Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception engine for 3d scene graph construction and optimization, 2022.

[6] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. Voxblox: Incremental 3d euclidean signed distance fields for on-board MAV planning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, sep 2017.

[7] Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans, 2020.

[8] Antoni Rosinol, Andrew Violette, Marcus Abate, Nathan Hughes, Yun Chang, Jingnan Shi, Arjun Gupta, and Luca Carlone. Kimera: From slam to spatial perception with 3d dynamic scene graphs. *The International Journal of Robotics Research*, 40(12-14):1510–1546, 2021.

[9] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping, 2019.

[10] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. 2020.

[11] Jose Luis Blanco and Pranjal Kumar Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. `https://github.com/jlblancoc/nanoflann`, 2014.

[12] Robert Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics*, 26, 07 2007.

[13] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[14] Stephan Weiss, Markus W. Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *2012 IEEE International Conference on Robotics and Automation*, pages 957–964, 2012.

[15] Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007.

[16] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012.

[17] Shaojie Shen, Nathan Michael, and Vijay Kumar. Tightly-coupled monocular visual-inertial fusion for autonomous flight of rotorcraft mavs. *Proceedings - IEEE International Conference on Robotics and Automation*, 2015:5303–5310, 06 2015.

[18] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, feb 2017.

[19] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. 07 2015.

[20] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (ijcai). volume 81, 04 1981.

[21] Jianbo Shi and Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[22] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[23] D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004.

[24] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.

[25] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Workshop on Vision Algorithms*, 1999.

[26] Gabe Sibley, Larry H. Matthies, and Gaurav S. Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27:587–608, 2010.

[27] Dorian Galvez-López and Juan D. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[28] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. volume 6314, pages 778–792, 09 2010.

[29] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.