LC: add: “Probabilistic Structure from Motion with Objects (PSfMO)” **LC:** add: “ObjectFusion: An Object Detection and Segmentation framework with RGB-D SLAM and Convolutional Neural Networks G Tian, L Liu, JH Ri, Y Liu, Y Sun - Neurocomputing, 2019” **LC:** add: [PDF] GEN-SLAM: Generative Modeling for Monocular Simultaneous Localization and Mapping P Chakravarty, P Narayanan, T Roussel - arXiv preprint arXiv:1902.02086, 2019

LC: add: [PDF] GEOMetrics: Exploiting Geometric Structure for Graph-Encoded Objects EJ Smith, S Fujimoto, A Romero, D Meger - arXiv preprint arXiv:1901.11461, 2019

LC: Mesh Based Semantic Modelling for Indoor and Outdoor Scenes Julien P. C. Valentin1,3 Sunando Sengupta1,3 Jonathan Warrell1 Ali Shahrokni2 Philip H. S. Torr1

LC: work from Soatto

LC: https://journals.sagepub.com/doi/abs/10.1177/0278364919839761

Semantic 3D Mesh VIO

Antoni Rosinol[1], Siyi Hu[1], Luca Carlone[1]

***Abstract*—We present a suite of algorithms and simulators with open-source implementations suitable for general Robotics development. As robotics is starting to have an increasingly important role with applications such as drone delivery and autonomous cars, it is ever more important to test, refine and provide a proof-of-concept of our algorithmic designs in a fast yet reliable way. With the advent of powerful desktop computers with multiple graphic cards and high-performance processors, we have the possibility to accurately simulate worlds that are always closer to being indistinguishable from the real world. Although we still have not reach the level where one might discard completely a real-life demonstration for the sake of proving the validity of a system, we are getting there at an exponential pace. Soon we might reach the point where we will have difficulties to decide what is real from what is computer-generated. If the reader is not convinced that this will happen, we suggest checking the latest developments in Generative Adversarial Networks TODO: cite .**

Classical implementations of Visual-Inertial Odometry (VIO) algorithms ignore semantic information of the scene, as they rely solely on sparse landmarks. Nevertheless, recent work has shown the advantage of using richer representations of the scene, such as 3D meshes, to extract higher-level information such as structural regularities. In this work, we show that a 3D mesh of the scene can be further utilized to accommodate semantic information, which enhances the mapping side of a classical VIO beyond a sparse and uninformative point-cloud. Towards this end, we use recent work on semidefinite programming and conditional random fields to generate semantic information in real-time on a single-core CPU.

***Index Terms*—Vision-Based Navigation, Semantic Segmentation.**

Supplementary Material

Videos of the experiments: TODO: Add video url

I. Introduction

Real-life applications of robotic platforms that are versatile enough to navigate autonomously necessitate from the following main modules: control, path-planning, state-estimation, and perception.

In order to have high-fidelity simulations to test algorithmic implementations of such modules, we need to accurately simulate the dynamics of the robot, as well as the exteroceptive and propioceptive sensors it has. A simulator that is extensively used by experienced roboticists to this purpose is Gazebo TODO: cite . The latest release of Gazebo provides highly accurate physics engine capable of computing collisions and body dynamics. Coupled with specialized third-party software for particular robotic platforms, Gazebo is a powerful tool that is the backbone of robot simulation in ROS, the Robotic Operating System.

[1]A. Rosinol, S. Hu and L. Carlone are with the Laboratory for Information & Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, USA, {arosinol,siyihu,lcarlone}@mit.edu

This work was partially funded by ARL DCIST CRA W911NF-17-2-0181.

Moreover, Gazebo allows for simulation of sensors such as: TODO: add , and in particular cameras. Nevertheless, as many experienced roboticists might have noticed already, Gazebo lacks high-fidelity image rendering, therefore making the simulator not usable for computer vision algorithms that necessitate realistic images.

Other simulators, such as AirSim TODO: cite , developped by Microsoft, provide both the dynamics and the photo-realistic renderings that we need. Nevertheless, its integration with ROS is in a very early stage, and must practitioners might find its installation and architecture inadequate for fast development (for now). Hopefully, in the next years, an up-to-date integration with ROS will happen and allow many more to use AirSim's simulator. AirSim achieves realistic renderings by making use of game development engines that are highly optimized to develop realistic simulated environments.

Therefore, roboticists are usually faced with two choices: either go for a photo-realistic approach, like AirSim's one, despite a fragile ROS integration, or use the decades long integration of Gazebo with ROS at the expense of realistic renderings. For a full-stack roboticist this means developing with AirSim when building state-estimation and perception modules, while switching to Gazebo for control and path-planning development for high-fidelity dynamics and fast development with ROS.

In this work, we propose to use Gazebo coupled with a photo-realistic rendering game engine such as Unity to allow for the development of full-stack robotic platforms with control/path-planning and state-estimation/perception algorithms. In particular, we will use the newly released software FlightGoggles, which uses the Unity game engine as the backbone for photo-realistic rendering while providing a ROS-Unity bridge that is simple and well-integrated with ROS.

Figure TODO: ... provides an overview of the different modules used in this work.

TODO: mention CARLA TODO: mention synthetic datasets such as VirtualKitti, SYNTHIA, ApolloScape, NuScenes etc

RECENT advances in VIO are enabling a wide range of applications, ranging from virtual and augmented reality to agile drone navigation [1].

Contributions. In this paper, we propose to *incrementally build a 3D mesh restricted to the receding horizon of the VIO optimization.* In this way, we can map larger areas than a per-frame approach, while memory footprint and computational complexity associated to the mesh remain bounded.

Paper Structure. Section III presents the mathematical formulation of our approach, and discusses the implementation of our VIO front-end and back-end. Section IV reports and discusses the experimental results and comparison against related work. Section V concludes the paper.

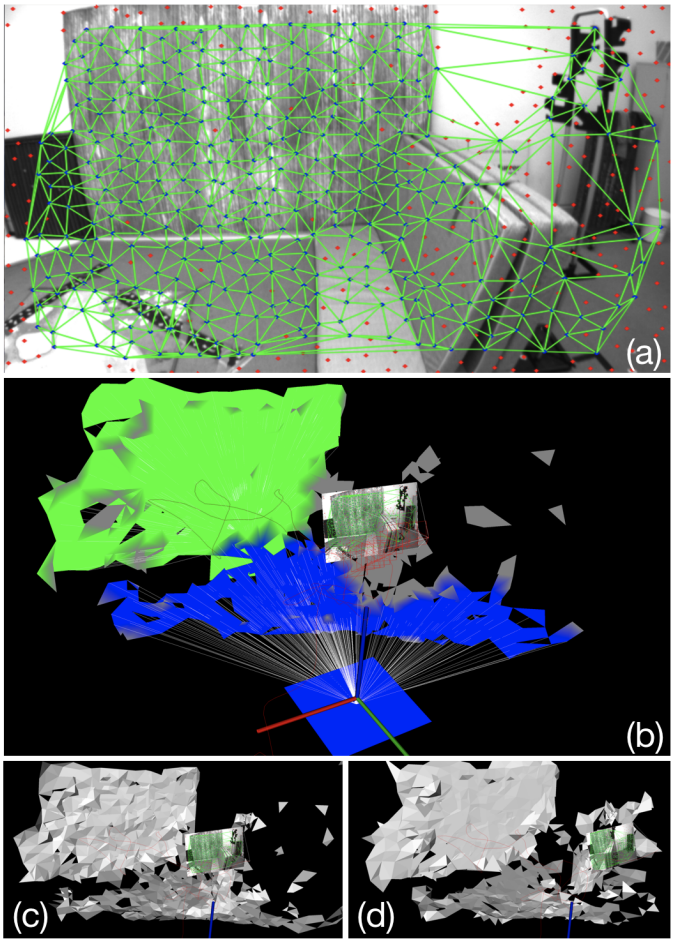

Fig. 1: We propose a VIO pipeline that incrementally builds a 3D mesh of the environment starting from (a) a 2D Delaunay triangulation of keypoints. We also detect and enforce *structural regularities*, c.f. (b) planar walls (green) and floor (blue). The bottom row in the figure compares the mesh constructed (c) without and (d) with structural regularities.

II. Related Work

A. TODO

B. Semantic Segmentation

III. Approach

TODO: Apprach We consider a stereo visual-inertial system and adopt a *keyframe*-based approach [2]. This section describes our VIO front-end and back-end. Our front-end proceeds by building a 2D Delaunay triangulation over the 2D keypoints at each keyframe. Then, the VIO back-end estimates the 3D position of each 2D keypoint, which we use to project the 2D triangulation into a 3D mesh. While we incrementally build the 3D mesh, we restrict the mesh to the time-horizon of the VIO optimization, which we formulate in a fixed-lag smoothing framework [3], [4]. The 3D mesh is further used to extract structural regularities in the scene that are then encoded as constraints in the optimization problem.

A. Front-end

Our front-end has the same components as a keyframe-based indirect visual-inertial odometry pipeline [2], [5], but it also incorporates a module to generate a 3D mesh, and a module to detect structural regularities from the 3D mesh. We refer the reader to [6, Sec. 4.2.1] for details on the standard modules used, and we focus here instead on the 3D mesh generation and regularity detection.

1) 3D Mesh Generation: Building a consistent 3D Mesh of the environment using a sparse point cloud from VIO is difficult because (i) the 3D positions of the landmarks are noisy, and some are outliers; (ii) the density of the point cloud is highly irregular; (iii) the point cloud is constantly morphing: points are being removed (marginalized) and added, while the landmarks' positions are being updated at each optimization step. Therefore, we avoid performing a 3D tetrahedralisation directly from the sparse 3D landmarks, which would require expensive algorithms, such as space carving [7].

2) 3D Mesh Propagation: While some algorithms update the mesh for a single frame [8], [9], we attempt to maintain a mesh over the receding horizon of the fixed-lag smoothing optimization problem (section III-B), which contains multiple frames. The motivation is three-fold: (i) A mesh spanning multiple frames covers a larger area of the scene, which provides more information than just the immediate field of view of the camera. (ii) We want to capture the structural regularities affecting any landmark in the optimization problem. (iii) Anchoring the 3D mesh to the time-horizon of the optimization problem also bounds the memory usage, as well as the computational complexity of updating the mesh.

a) Temporal propagation: deals with the problem of updating the 3D mesh when new keypoints appear and/or old ones disappear in the new frame.

b) Spatial propagation: deals with the problem of updating the global 3D mesh when a new local 3D mesh is available, and when old landmarks are marginalized from the optimization's time-horizon. We solve the first problem by merging the new local 3D mesh to the previous (global) mesh, by ensuring no duplicated 3D faces are present.

3) Regularity Detection:

4) Data Association:

B. Back-end

1) State Space: If we denote the set of all keyframes up to time t by $\mathcal{K}_t$, the state of the system $\mathbf{x}_i$ at keyframe $i \in \mathcal{K}_t$, is described by the IMU orientation $\mathbf{R}_i$, position $\mathbf{p}_i$, velocity $\mathbf{v}_i$ and biases $\mathbf{b}_i$:

$$\mathbf{x}_i \doteq [\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i], \tag{1}$$

where the pose $(\mathbf{R}_i, \mathbf{p}_i) \in \mathrm{SE}(3)$, $\mathbf{v}_i \in \mathbb{R}^3$, and $\mathbf{b}_i = [\mathbf{b}_i^g \quad \mathbf{b}_i^a] \in \mathbb{R}^6$, and $\mathbf{b}_i^g, \mathbf{b}_i^a \in \mathbb{R}^3$ are the gyroscope and accelerometer biases, respectively.

We will only estimate the 3D positions $\boldsymbol{\rho}_l$ for a subset Λ_t of all landmarks $\mathcal{L}_t$ visible up to time t: $\{\boldsymbol{\rho}_l\}_{l \in \Lambda_t}$, where $\Lambda_t \subseteq \mathcal{L}_t$. We will avoid optimizing over the rest of the landmarks $\mathcal{L}_t \setminus \Lambda_t$ by using a structureless approach, as defined in [10, Sec. VII], which circumvents the need to add the landmarks' positions as variables in the state. This allows to trade accuracy for speed; as the optimizations complexity increases with the number of variables to be estimated.

The set Λ_t corresponds to the landmarks which we consider to satisfy a structural regularity. In particular, we are interested in co-planarity regularities, which we introduce in section III-B5. Since we need the explicit landmark variables to formulate constraints on them, we avoid using a structureless approach for these landmarks.

Finally, the co-planarity constraints between the landmarks Λ_t require the modelling of the planes Π_t in the scene. Therefore, the variables to be estimated comprise the state of the system $\{\mathbf{x}_i\}_{i\in\mathcal{K}_t}$, the landmarks which we consider to satisfy structural regularities $\{\boldsymbol{\rho}_l\}_{l\in\Lambda_t}$, and the planes $\{\boldsymbol{\pi}_\pi\}_{\pi\in\Pi_t}$. The variables to be estimated at time t are:

$$\mathcal{X}_t \doteq \{\mathbf{x}_i, \boldsymbol{\rho}_l, \boldsymbol{\pi}_\pi\}_{i\in\mathcal{K}_t, l\in\Lambda_t, \pi\in\Pi_t} \tag{2}$$

Since we are taking a fixed-lag smoothing approach for the optimization, we limit the estimation problem to the sets of variables that depend on the keyframes in a time-horizon of size Δ_t. To avoid cluttering the notation, we skip the dependence of the sets $\mathcal{K}_t$, Λ_t and Π_t on the parameter Δ_t.

By reducing the number of state variables to a given window of time Δ_t, we will make the optimization problem tractable and solvable in real-time.

2) Measurements: The input for our system consists on measurements from the camera and the IMU. We define the image measurements at keyframe i as $\mathcal{C}_i$. The camera can observe multiple landmarks l, hence $\mathcal{C}_i$ contains multiple image measurements $\mathbf{z}_i^l$, where we distinguish the landmarks that we will use for further structural regularities $\mathbf{z}_i^{l_c}$ (where the index c in l_c stands for constrained landmark), and the landmarks that will remain as structureless $\mathbf{z}_i^{l_s}$ (where the s in the index of l_s stands for structureless). We represent the set of IMU measurements acquired between two consecutive keyframes i and j as $\mathcal{I}_{ij}$. Therefore, we define the set of measurements collected up to time t by $\mathcal{Z}_t$:

$$\mathcal{Z}_t \doteq \{\mathcal{C}_i, \mathcal{I}_{ij}\}_{(i,j)\in\mathcal{K}_t}. \tag{3}$$

3) Factor Graph Formulation: We want to estimate the posterior probability $p(\mathcal{X}_t|\mathcal{Z}_t)$ of our state $\mathcal{X}_t$, as defined in eq. (2), using the set of measurements $\mathcal{Z}_t$, defined in eq. (3). Using standard independence assumptions between measurements and states, we arrive to the formulation in eq. (4), where we grouped the different terms in factors ϕ:

$$\begin{aligned} p(\mathcal{X}_t|\mathcal{Z}_t) &\overset{(a)}{\propto} p(\mathcal{Z}_t|\mathcal{X}_t)p(\mathcal{X}_t) \\ &= \phi_0(\mathbf{x}_0) \prod_{l_c\in\Lambda_t} \prod_{\pi\in\Pi_t} \phi_{\mathcal{R}}(\boldsymbol{\rho}_{l_c}, \boldsymbol{\pi}_\pi)^{\delta(l_c,\pi)} && \text{(4a)} \\ &\quad \prod_{(i,j)\in\mathcal{K}_t} \phi_{\text{IMU}}(\mathbf{x}_i, \mathbf{x}_j) && \text{(4b)} \\ &\quad \prod_{i\in\mathcal{K}_t} \prod_{l_c\in\mathcal{C}_i^c} \phi_{l_c}(\mathbf{x}_i, \boldsymbol{\rho}_{l_c}) \prod_{i\in\mathcal{K}_t} \prod_{l_s\in\mathcal{C}_i^s} \phi_{l_s}(\mathbf{x}_i) && \text{(4c)} \end{aligned}$$

where we apply the Bayes rule in (a), and ignore the normalization factor over the measurements since it will not influence the result (section III-B4).

Equation (4a) corresponds to the prior information we have over the state $p(\mathcal{X}_t)$. In this term, we encode regularity constraints between landmarks $\boldsymbol{\rho}_{l_c}$ and planes π, which we denote by $\phi_{\mathcal{R}}$. We also introduce the data association term $\delta(l_c, \pi)$, which returns a value of 1 if the landmark l_c is associated to the plane π, 0 otherwise. We explain in section III-A4 how the data association is done. The factor ϕ_0 represents a prior on the first pose of the optimization's time-horizon.

In eq. (4b), we have the factor corresponding to the IMU measurements, which depends only on the consecutive keyframes $(i, j) \in \mathcal{K}_t$.

Finally, eq. (4c) encodes the factors corresponding to the camera measurements. Since we want to distinguish between landmarks that are involved in structural regularities (l_c) and landmarks that are not (l_s), we split the product over C_i; where we write $l_s \in \mathcal{C}_i^s$ or $l_c \in \mathcal{C}_i^c$ when a landmark l_s or l_c, respectively, is seen at keyframe i by the camera. Note that $\mathcal{C}_i = \mathcal{C}_i^c \cup \mathcal{C}_i^s$ and $\mathcal{C}_i^c \cap \mathcal{C}_i^s = \emptyset$.

In fig. 2, we use the expresiveness of factor graphs [11] to show the actual dependencies between the variables in eq. (4)[1].

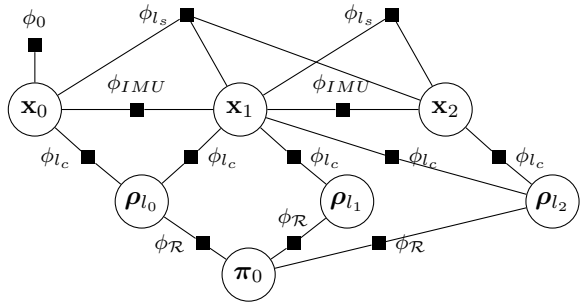

Fig. 2: VIO factor graph combining Structureless (ϕ_{l_s}), Projection (ϕ_{l_c}) and Regularity ($\phi_{\mathcal{R}}$) factors (SPR). The factor $\phi_{\mathcal{R}}$ encodes relative constraints between a landmark l_i and a plane π_0.

4) MAP Estimation: Since we are only interested in the most likely state $\mathcal{X}_t$ given the measurements $\mathcal{Z}_t$, we calculate the *maximum a posteriori* (MAP) estimator $\mathcal{X}_t^{\text{MAP}}$. Maximizing $\mathcal{X}_t^{\text{MAP}}$ is nevertheless not as convenient as minimizing the negative logarithm of the posterior probability, which, using eq. (4), simplifies to eq. (5) for zero-mean Gaussian noise:

$$\begin{aligned} \mathcal{X}_t^{\text{MAP}} = \arg\min_{\mathcal{X}_t} &\|\mathbf{r}_0\|^2_{\Sigma_0} + \sum_{l_c\in\Lambda_t} \sum_{\pi\in\Pi_t} \delta(l_c,\pi) \|\mathbf{r}_{\mathcal{R}}\|^2_{\Sigma_{\mathcal{R}}} \\ &+ \sum_{(i,j)\in\mathcal{K}_t} \left\|\mathbf{r}_{\mathcal{I}_{ij}}\right\|^2_{\Sigma_{ij}} + \sum_{i\in\mathcal{K}_t} \left\{ \sum_{l_c\in\mathcal{C}_i} \left\|\mathbf{r}_{\mathcal{C}_i^{l_c}}\right\|^2_{\Sigma_{\mathcal{C}}} + \sum_{l_s\in\mathcal{C}_i} \left\|\mathbf{r}_{\mathcal{C}_i^{l_s}}\right\|^2_{\Sigma_{\mathcal{C}}} \right\} \end{aligned} \tag{5}$$

where $\mathbf{r}$ represents the residual errors, and $\boldsymbol{\Sigma}$ the covariance matrices. We refer the reader to [10, Sec. VI, VII] for the actual formulation of the preintegrated IMU factors ϕ_{IMU} and structureless factors ϕ_{l_s}, as well as the underlying residual functions $\mathbf{r}_{\text{IMU}}$, $\mathbf{r}_{\mathcal{C}_i^{l_s}}$. For the projection factors ϕ_{l_c}, we use a standard monocular and stereo reprojection error formulation as in [4].

5) Regularity Constraints: For the regularity factors $\phi_{\mathcal{R}}$, we use a co-planarity constraint between a landmark $\boldsymbol{\rho}_{l_c} \in \mathbb{R}^3$ and a plane $\pi = \{\boldsymbol{n}, d\}$, where $\boldsymbol{n}$ is the normal of the

[1]We will use the notation proposed in [12] to represent the factor graph.

plane, which lives in the $\mathrm{S}^2 \doteq \{\mathbf{n} = (n_x, n_y, n_z)^T \big| \|\mathbf{n}\| = 1\}$ manifold, and $d \in \mathbb{R}$ the distance to the origin: $\mathbf{r}_\mathcal{R} = \boldsymbol{n} \cdot \boldsymbol{\rho}_{l_c} - d$. Representing a plane by its normal and distance to the origin is an over-parametrization that will lead to an information matrix that is singular. This is not amenable for Gauss-Newton optimization, since it leads to singularities in the normal equations [13]. To avoid the over-parametrization problem, we optimize in the tangent space $T_{\boldsymbol{n}}S^2 \sim \mathbb{R}^2$ of S^2 and define a suitable retraction $\mathcal{R}_{\boldsymbol{n}}(\boldsymbol{v}) : T_{\boldsymbol{n}}S^2 \in \mathbb{R}^2 \rightarrow \mathrm{S}^2$ to map changes in the tangent space to changes to the normals in S^2 [10]. In other words, we rewrite the residuals as:

$$\mathbf{r}_\mathcal{R}(\boldsymbol{v}, d) = \mathcal{R}_{\boldsymbol{n}}(\boldsymbol{v})^\mathsf{T} \cdot \boldsymbol{\rho} - d \tag{6}$$

and optimize with respect to the minimal parametrization $\boldsymbol{v}$. This is similar to the proposal of Kaess [13], but we work on the manifold S^2, while Kaess adopts a quaternion parametrization. Note that, a single co-planarity constraint, as defined in eq. (6), is not sufficient to constrain a plane variable, and a minimum of three are needed instead. Nevertheless, degenerate configurations exist, e.g. three landmarks on a line would not fully constrain a plane. Therefore, we ensure that a plane candidate has a minimum number of constraints before adding it to the optimization problem. **TODO:** Do we need to mention robust cost functions at all? I think so, reviews were picky about outliers!

IV. Experimental Results

V. Conclusion

References

[1] T. Sayre-McCord, W. Guerra, A. Antonini, J. Arneberg, A. Brown, G. Cavalheiro, Y. Fang, A. Gorodetsky, D. McCoy, S. Quilter, F. Riether, E. Tal, Y. Terzioglu, L. Carlone, and S. Karaman, "Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018.

[2] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial SLAM using nonlinear optimization," *Int. J. Robot. Research*, 2015.

[3] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *arXiv preprint arXiv:1708.03852*, 2017.

[4] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 3886–3893, extended arxiv preprint: 1610.03344 (pdf).

[5] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. IEEE, 2015.

[6] A. Rosinol, "Densifying Sparse VIO: a Mesh-based approach using Structural Regularities." Master's thesis, ETH Zurich, 2018-09-14.

[7] Q. Pan, G. Reitmayr, and T. Drummond, "Proforma: Probabilistic feature-based on-line rapid model acquisition." in *BMVC*, vol. 2. Citeseer, 2009, p. 6.

[8] W. N. Greene and N. Roy, "Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *Int. Conf. Comput. Vis. (ICCV)*, 2017.

[9] L. Teixeira and M. Chli, "Real-Time Mesh-based Scene Estimation for Aerial Inspection," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems(IROS)*, 2016.

[10] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. Robot.*, vol. 33, no. 1, pp. 1–21, 2017.

[11] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, Feb. 2001.

[12] L. Dietz, "Directed factor graph notation for generative models," *Max Planck Institute for Informatics, Tech. Rep*, 2010.

[13] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 4605–4611.