A Visual Inertial Odometry Framework for 3D Points, Lines and Planes

Shenbagaraj Kannapiran[†1], Jeroen van Baar[2], and Spring Berman[1]

***Abstract*— Recovering rigid registration between successive camera poses lies at the heart of 3D reconstruction, SLAM and visual odometry. Registration relies on the ability to compute discriminative 2D features in successive camera images for determining feature correspondences, which is very challenging in feature-poor environments, i.e. low-texture and/or low-light environments. In this paper, we aim to address the challenge of recovering rigid registration between successive camera poses in feature-poor environments in a Visual Inertial Odometry (VIO) setting. In addition to inertial sensing, we instrument a small aerial robot with an RGBD camera and propose a framework that unifies the incorporation of 3D geometric entities: points, lines, and planes. The tracked 3D geometric entities provide constraints in an Extended Kalman Filtering framework. We show that by directly exploiting 3D geometric entities, we can achieve improved registration. We demonstrate our approach on different texture-poor environments, with some containing only flat texture-less surfaces providing essentially no 2D features for tracking. In addition, we evaluate how the addition of different 3D geometric entities contributes to improved pose estimation by comparing an estimated pose trajectory to a ground truth pose trajectory obtained from a motion capture system. We consider computationally efficient methods for detecting 3D points, lines and planes, since our goal is to implement our approach on small mobile robots, such as drones.**

I. INTRODUCTION

Odometry and SLAM are critical for robots to navigate in previously unseen environments. Although various solutions are available [1], they rely on discriminative visual features in color camera images in order to determine robust correspondences. Texture-rich environments can provide the necessary visual features and many encountered environments are texture-rich, e.g., outdoor environments. However, other environments may lack texture almost completely, e.g., stairwells and elevator shafts. In addition, even if the environment is texture-rich, low light levels or atmospheric conditions such as fog will negatively impact the ability to compute robust correspondences.

Without good visual features in camera images, robots will have to rely on other sensing modalities to provide 3D measurements for registration and pose estimation. A popular modality is LIDAR, which can provide a 360° point cloud around the robot. However, LIDAR sensors are both expensive and and, more importantly, heavy. In the case where the robot is an Unmanned Aerial Vehicle (UAV), the weight of a LIDAR sensor severely limits its feasibility as a payload. RGBD cameras, on the other hand, are inexpensive and lightweight, and thus a more desirable choice for obtaining 3D measurements. Although current 3D reconstruction and SLAM approaches incorporate RGBD cameras and utilize their depth information [2], these approaches still rely on visual features from the color camera images.

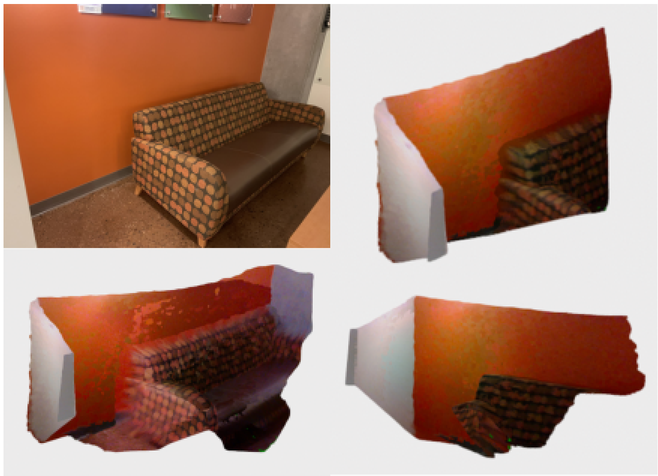

Fig. 1. Registration result of point clouds from 93 frames with our proposed framework using 3D points, 3D lines and 3D planes. The scene contains texture-poor areas, which are difficult for methods that rely on 2D features derived from color images.

Additional sensors, such as inertial measurement units (IMUs), have been introduced to further assist 3D reconstruction and SLAM. A major drawback of IMUs is drift in their measurements due to integration of a slowly changing bias instability and high-frequency noise. Relying mostly on IMUs for registration, dead-reckoning accumulates errors over time and can lead to registration errors that are prohibitively large for accurate reconstruction and mapping. Visual Inertial Odometry (VIO) frameworks [3] have been introduced to reduce or eliminate this drift. In addition to 2D point features, 2D line features have been recently considered for VIO [4]. Assuming that the environment can be (locally) approximated as a Manhattan-world configuration, the approach relies on tracking 2D lines as features in the camera images.

We propose a VIO framework that can handle texture-poor environments and incorporates 3D geometric entities: 3D points, 3D lines and 3D planes from point clouds obtained with an RGBD camera. The 3D points themselves may be prone to noise in the point cloud data, and fitting of 3D

[†] Work partially done during an internship at Mitsubishi Electric Research Laboratories (MERL). This work was supported in part by the Arizona State University Global Security Initiative.

[1] Shenbagaraj Kannapiran and Spring Berman are with the School for Engineering of Matter, Transport and Energy, Arizona State University, Tempe, AZ 85287, USA shenbagaraj@asu.edu; spring.berman@asu.edu

[2] Jeroen van Baar is with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA jeroen@merl.com

lines and planes with robustness to outliers—due to noise—helps to mitigate sensitivity to such noise. A major advantage of our proposed approach, due to its reliance on depth measurements rather than visual features, is its potential to operate effectively in environments with fog, low-light conditions, or man-made structures devoid of visual textures. Moreover, our framework is agnostic to the particular RGBD camera that is used to obtain the point clouds. Our focus is on developing an approach which can be implemented on UAV robots that can access relatively small spaces. This limits the UAV size, and hence the size and weight of its payload. Our contributions can be summarized as follows:

- Algorithms for detection and tracking of 3D points, 3D lines and 3D planes in 3D point clouds, including methods to determine world space estimates from tracked 3D entities in local camera frames; and
- A unified VIO framework that incorporates the tracked 3D points, 3D lines and 3D planes as geometric constraints in an Extended Kalman Filtering (EKF) framework.

We next discuss related work, followed by a technical description of our approach in Section III, and discussion of experiments and results in Section IV. Section V concludes the paper and provides an outlook on future work.

II. RELATED WORK

In this section, we give an overview of previous work related to 3D reconstruction, SLAM and visual odometry. Given the wealth of research in these fields, e.g., see [1], [5], [6], a comprehensive overview is beyond the scope of this paper. Feature matching and determining correspondences lie at the heart of all these approaches. Several approaches use monocular RGB(D) cameras to determine features and correspondences. ORB-SLAM [7], [8] computes so-called ORB features for matching, providing sparse correspondence information. To ensure greater robustness to mismatches in 2D feature points, SLAM with points and planes [2], [9] rely on both SIFT [10] (or alternatively SURF [11]) features and plane correspondences. CPA-SLAM [12] incorporates a plane-based approach using expectation-maximization (EM) pose estimation and graph-based optimization for registration. More recent techniques use data-driven approaches, e.g., [13], but still rely on correspondences from visual data. An interesting data-driven registration approach samples correspondences [14] in a deep learning framework. Although featureless approaches are being investigated, e.g., [15], these methods still rely on textures for template matching.

A. Methods for Low-Texture Environments

Classical methods rely on texture-rich environments for computing visual features which are discriminative for matching. Low-texture environments, however, cannot provide such features. Several approaches have been introduced for low-texture environments. Pop-up SLAM [16] detects wall and ground planes in single images and formulates SLAM based on Pop-up 3D planes, while also incorporating sparse features when available. Newcombe et al. [17] perform fast Iterative Closest Point (ICP) directly on depth scans acquired with an RGBD sensor. Although this method does not rely on features, scans between successive frames have to overlap significantly to avoid incorrect ICP solutions. The method proposed in [18] uses range flow and sparse geometry features from depth maps directly to handle low-texture environments. The method in [19] leverages point and line features in dynamic Manhattan worlds by performing dynamic object tracking, visual odometry, local mapping, and loop closing. End-to-end deep learning approaches aim to learn feature correspondences in depth maps [20]. A major drawback of deep learning approaches is their reduced accuracy in novel environments.

B. Inertial Odometry

Incorporating additional sensors besides cameras can further improve results. The authors in [21] propose to combine odometry or inertial sensor measurements with ORB-SLAM to improve performance in low-texture environments. MonoSLAM [22] introduced an Extended Kalman Filtering (EKF) approach to SLAM, including the use of a gyro sensor. Visual Inertial Odometry (VIO) [3], [23] also uses an EKF framework to prevent drift by integrating constraints, i.e. 2D features, that are generated from visual information in successive frames. Constraints from 2D lines [4] can be added by assuming that the scene is locally Manhattan-world. Similarly, there are other tightly coupled VIO techniques, such as [24], [25], [26], [27], [28], that exploit points and lines to reduce IMU drift and obtain optimized camera pose estimates.

To handle low-texture environments, we build on the VIO frameworks described in [3], [4], [23] and propose to directly incorporate ***3D geometric entities***, i.e., 3D points, 3D lines and 3D planes, computed from the point cloud provided by an RGBD camera, into a single unified VIO framework. To the best of our knowledge, this is the first such proposed VIO framework.

III. VIO WITH 3D GEOMETRIC ENTITIES

In our VIO framework, the visual information, which is incorporated as geometric constraints in an EKF, relies on tracking a set of 2D correspondences, i.e., points and lines. To compensate for the potential lack of texture, or visual information necessary to determine 2D correspondences, we directly utilize 3D information from the underlying geometry instead. We first give a brief overview of the notation and VIO framework using EKF and then describe our proposed approach in more detail.

A. Notation and Prerequisites

We use the following notation and definitions. We identify several coordinate spaces: the local coordinate frame I for the inertial sensor; the local coordinate frame C for the RGBD camera; and the world coordinate frame W. A (rigid) transform $\mathbf{M}$ from a space A to another space B is denoted by ${}^{B}_{A}\mathbf{M}$. Rotational matrices are denoted by $\mathbf{R}$, and $\mathbf{R}_q$ is obtained from a quaternion vector $\bar{q}$ by the operation $\mathbf{C}(\bar{q})$.

The notation $\lfloor \mathbf{V} \times \rfloor$ denotes a 3×3 skew-symmetric matrix from some 3-vector $\mathbf{V}$. An estimated quantity is decorated with a $\hat{}$ symbol, and an error quantity with $\tilde{}$, defining $\tilde{x} = x - \hat{x}$ for some quantity x (e.g., a state vector).

The goal of VIO is to estimate IMU states in an EKF estimation framework, and "ground" the estimates according to visual features. The IMU state at time k is defined as $\mathbf{X}_{\text{IMU}_k} = \left[{}^{I}_{W}\bar{q}^T \; b_g^T \; {}^{W}\mathbf{v}_I^T \; b_a^T \; {}^{W}p_I^T\right]^T$, whose components are the rotation between the world and IMU frames, the gyro bias, the IMU velocity in the world frame, the accelerometer bias, and the IMU position in the world frame, respectively. For the purpose of grounding, the EKF state estimate at time k, $\hat{\mathbf{X}}_k$, augments the IMU state with N camera poses: $\hat{\mathbf{X}}_k = \left[\hat{\mathbf{X}}_{\text{IMU}_k}^T \; {}^{C_1}_{W}\hat{\bar{q}}^T \; {}^{W}\hat{p}_{C_1}^T \cdots {}^{C_N}_{W}\hat{\bar{q}}^T \; {}^{W}\hat{p}_{C_N}^T\right]^T$ (note the $\hat{}$).

In the EKF framework, state estimates are updated at each time step as follows: $\hat{\mathbf{X}}_{k+1} = \hat{\mathbf{X}}_k + \Delta\mathbf{X} = \hat{\mathbf{X}}_k + \mathbf{K}_k r_k$, where $\mathbf{K}_k$ denotes the matrix of Kalman gains, and r_k is a residual of the form

$$r_k = \mathbf{H}\tilde{\mathbf{X}}_k + \text{noise}. \tag{1}$$

Here, $\mathbf{H}$ is the measurement Jacobian matrix, and the noise is zero-mean Gaussian white noise, uncorrelated with the state vector $\mathbf{X}$. The IMU error-state is: $\tilde{\mathbf{X}}_{\text{IMU}} = \left[\delta\boldsymbol{\theta}_I^T \; \tilde{b}_g^T \; {}^{W}\tilde{v}_I^T \; \tilde{b}_a^T \; {}^{W}\tilde{p}_I^T\right]^T$, with the corresponding EKF error-state: $\tilde{\mathbf{X}} = \left[\tilde{\mathbf{X}}_{\text{IMU}} \; \delta\boldsymbol{\theta}_{C_1}^T \; {}^{W}\tilde{p}_{C_1}^T \cdots \delta\boldsymbol{\theta}_{C_N}^T \; {}^{W}\tilde{p}_{C_N}^T\right]^T$. For some quantity x, the error $\tilde{x}$ here means $\tilde{x} = x - \hat{x}$. The vectors $\delta\boldsymbol{\theta}_{C_i}^T$ are errors for the quaternions representing camera poses i. The EKF error-state is an $(15 + 6N)$-dimensional vector. For more details on the EKF framework, we refer the reader to [23].

The measurement model relies on the notion that static features are observed from multiple poses, which impose constraints on the IMU estimates: the pose estimates should be consistent with the observations of the (tracked) geometric entities. Given Eq. (1), the objective is to determine the measurement Jacobian matrix $\mathbf{H}$. We will now describe how to incorporate observations of 3D points, 3D lines and 3D planes into a single framework for VIO.

B. Points in 3D

Given a pair of successive RGBD images, labeled *source* and *target*, we compute Fast Point Feature Histogram (FPFH) features [29] for the 3D points in the point cloud. The choice of FPFH is based on the required computational efficiency, given the necessity for real-time computation if deployed on an actual UAV. We then use the FPFH features to determine correspondences between 3D points in the source and target.

We denote the position of the 3D point j observed in frame i as $z_i^{(j)} = {}^{C_i}p_j = {}^{C_i}_{W}\mathbf{R}_{\bar{q}}({}^{W}p_j - {}^{W}p_{C_i})$. Since ${}^{W}p_j$ is unknown, it is estimated by tracking 3D point correspondences. We denote this estimate as ${}^{W}\hat{p}_j$ and can obtain measurement estimate $\hat{z}_i^{(j)} = {}^{C_i}\hat{p}_j = {}^{C_i}_{W}\mathbf{R}_{\bar{q}}({}^{W}\hat{p}_j - {}^{W}p_{C_i})$. We can define the measurement residual as $r_i^{(j)} = z_i^{(j)} - \hat{z}_i^{(j)}$.

In order to incorporate 3D points in the VIO measurement model, we update the Jacobian of a measurement $z_i^{(j)}$ with respect to the state, cf. Eq. (22) in [23], as follows:

$$\mathbf{H}_{\mathbf{X}_i}^{(j)} = \left[\mathbf{0}_{3\times15} \;\; \mathbf{0}_{3\times6} \;\; \cdots \;\; \lfloor {}^{C_i}\hat{p}_j \times \rfloor \;\; -{}^{C_i}_{W}\mathbf{R}_{\bar{q}} \;\; \cdots\right]. \tag{2}$$

The skew-symmetric matrix in Eq. (2) is obtained using Lie theory [30]. Given a matrix $\mathbf{R} \in SO(3)$ and a function $\mathbf{y} = f(\mathbf{R}, x) = \mathbf{R} \cdot x$, we compute:

$$\begin{aligned} \frac{\partial \mathbf{y}}{\partial \mathbf{R}} &= \frac{\partial}{\partial \omega}|_{\omega=0}(\exp(\omega) \cdot \mathbf{R}) \cdot \mathbf{x} \\ &= -\lfloor \mathbf{y} \rfloor_{\times}, \end{aligned}$$

where ω represents the parameters of $\mathfrak{so}(3)$.

The Jacobian of $z_i^{(j)}$ with respect to feature position, cf. Eq. (23) in [23], is

$$\mathbf{H}_i^{(j)} = {}^{C_i}_{W}\mathbf{R}_{\bar{q}}. \tag{3}$$

The residuals for all M_{pt} observations of 3D point j are then stacked.

Given tracked correspondences for 3D point j in local coordinate frame C, the estimate ${}^{W}\hat{p}_j$ can be easily determined by taking the average over ${}^{W}\hat{p}_j^i$, $i \in \{1, ..., M_{pt}\}$, where ${}^{W}\hat{p}_j^i = ({}^{C_i}_{W}\mathbf{R}_{\bar{q}})^{-1}\,{}^{C_i}p_j$. We employ RANSAC to eliminate potentially incorrect correspondences.

C. Lines in 3D

In addition to 3D points, we aim to incorporate 3D lines. In the presence of noisy observations, fitting a line may mitigate some of the noise. Furthermore, some scenes may not provide ***many*** discriminative point features, e.g., if the geometry consists of mostly planar wall regions. Our goal is to incorporate 3D lines which are robust with respect to several observations. Various 3D line finding techniques have been introduced, e.g., [31], [32], [33]. To avoid an expensive search, we identify only those lines for which the points lie on a boundary between different depth values, as determined from the depth map. Although we could use the FPFH features computed for the 3D point correspondences, instead for each 3D point we analyze the normals in a small local neighborhood, and define a histogram-based feature similar to SIFT features [10]. We group together all points with similar features and fit a 3D line ${}^{C_i}\mathbf{L}_j$ to them. The line fitting procedure is based on Eigen analysis.

The authors in [4] parameterize lines with start and end points. In the case of 3D lines, we would need to track these 3D points and update them if they become occluded or clipped by the camera field of view. To avoid having to determine such corresponding anchor points for a line between observations, we parameterize a line by its direction d only. A line j is then transformed by:

$$\begin{aligned} z_i^{(j)} = {}^{C_i}\mathbf{L}_j &= f({}^{C_i}_{W}\mathbf{R}_{\bar{q}}, {}^{W}\mathbf{L}_j) \\ &= {}^{C_i}_{W}\mathbf{R}_{\bar{q}} \cdot {}^{W}d_j. \end{aligned}$$

The Jacobians of $z_i^{(j)}$ with respect to the state and feature position are then:

$$\mathbf{H}_{\mathbf{X}_i}^{(j)} = \left[\mathbf{0}_{3\times15} \;\; \mathbf{0}_{3\times6} \cdots - \lfloor {}^{C_i}\hat{\mathbf{d}}_j \times \rfloor \;\; \mathbf{0}_{3\times3} \cdots\right], \tag{4}$$

$$\mathbf{H}_i^{(j)} = {}^{C_i}_{W}\mathbf{R}_{\bar{q}}. \tag{5}$$

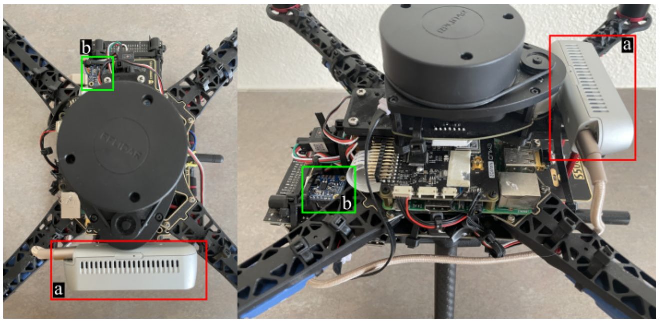

Fig. 2. Two views of the instrumented UAV illustrating the locations of the (**a**) RGBD camera (in red box), and (**b**) IMU (in green box). Additional on-board compute devices are for future use.

The residuals for all M_{ln} observations of 3D lines j are then stacked.

As for 3D points, the 3D line estimate ${}^{W}\hat{d}_j$ can be determined by averaging the observations in the world coordinate frame for the 3D line j. Tracking lines between different observations is performed by comparing line directions in the world coordinate frame, and designating those with directions within some threshold as corresponding lines. However, this may potentially designate lines with similar directions, but different 3D positions, as corresponding. To prevent this, for a given line $\mathbf{L}_j$ we determine the 3D points which *support* the 3D line, denoted as $\mathbf{P}_{\mathbf{L}_j} = \{p_1, \cdots, p_m\}_{\mathbf{L}_j}$. First, lines with $|\mathbf{P}_{\mathbf{L}_j}| < \tau$, where τ is chosen as the threshold to determine the number of inliers are discarded, since we found empirically that they cannot be robustly matched. Second, for two lines $\mathbf{L}_j$, $\mathbf{L}_k$ we can use the associated $\mathbf{P}_{\mathbf{L}_j}$ and $\mathbf{P}_{\mathbf{L}_k}$ to calculate Euclidean distances. If the distance is within some threshold, lines $\mathbf{L}_j$, $\mathbf{L}_k$ are flagged as corresponding, and non-corresponding otherwise.

D. Planes in 3D

Finally, we also consider 3D plane entities for the VIO. Planes can further mitigate uncertainty due to noise for points and lines. The treatment of 3D planes in this framework follows that of lines. The main difference between lines and planes is that we parameterize the planes by their normal direction.

The Jacobians of $z_i^{(j)}$ wtih respect to the state and feature position are now:

$$\mathbf{H}_{\mathbf{X}_i}^{(j)} = \left[\mathbf{0}_{3\times 15} \;\; \mathbf{0}_{3\times 6} \cdots - \left\lfloor {}^{C_i}\hat{\mathbf{n}}_j \times \right\rfloor \;\; \mathbf{0}_{3\times 3} \cdots \right], \quad (6)$$

$$\mathbf{H}_i^{(j)} = {}^{C_i}_{W}\mathbf{R}_{\bar{q}}, \quad (7)$$

where n denotes the plane normal. The residuals for all M_{pl} observations of 3D planes j are then stacked.

Tracking 3D plane correspondences across observations occurs in the same way as for 3D lines, including the calculation of supporting 3D points for the plane. Planes with the number of supporting points lower than a threshold are discarded for the same robustness reason as lines (see above).

E. Dimensionality Reduction

For a world coordinate frame feature ${}^{W}f$, i.e., a 3D point, line or plane, we have that $r^{(j)} \simeq \mathbf{H}_{\mathbf{X}}^{(j)}\tilde{\mathbf{X}} + \mathbf{H}^{(j)\,W}\tilde{f} + \text{noise}$. In order to formulate the residual in the form of Eq. (1), we perform the same nullspace projection as in [3], resulting in the (stacked) residuals $r_o^{(j)}$ and Jacobian $\mathbf{H}_o^{(j)}$ for a feature j. Prior to the nullspace projection, the Jacobian is a $(3M^{(j)} \times 3)$-dimensional matrix, and after projection, its dimensions are $((3M^{(j)} - 3) \times 3)$. Here $M^{(j)}$ denotes the number of observations for an entity j: a point, line or plane.

If we assume that $\forall j$, $M^{(j)} = M = M_{pt} = M_{ln} = M_{pl}$, then given N_{pt} 3D points, N_{ln} 3D lines and N_{pl} 3D planes, the final stacked Jacobian matrix, by combining all entities, is $((N_{pt} + N_{ln} + N_{pl})(3M - 3) \times 3)$-dimensional. It is evident that dimensionality reduction via QR decomposition is essential for reducing the computational complexity. We follow the same approach as in [3] for reducing the dimensionality.

F. EKF Updates

We now have the necessary information to compute the desired Kalman gains to perform an EKF update, i.e., determine the updated state and covariance estimates. In addition to augmenting the EKF state, the arrival of a new observation also augments the covariance matrix necessary for the EKF updates. In the next section, we perform an evaluation of our proposed framework.

IV. EXPERIMENTS

We first provide a short description of the experimental hardware platform, and then describe the results of experiments we conducted to demonstrate the effectiveness of incorporating 3D points, lines and planes.

A. Experimental Hardware Platform

We instrumented a UAV with an IMU and RGBD camera. The IMU is an Adafruit 9-DOF Absolute Orientation IMU Fusion Breakout - BNO055, providing IMU readings at 100 Hz. The RGBD camera is an Intel RealSense D435 that acquires data at 10 Hz[1]. The experimental hardware is shown in Figure 2. We employed IMU-camera calibration as proposed by [34], [35]. The IMU-camera calibration also provides estimation for the camera's intrinsic parameters. We used all the depth frames produced by the RGBD camera, around 150 frames for each experiment. The compute time of registration between adjacent frames was around 5 s using non-optimized Python code; we expect that C++ code with modest optimization could run at an interactive rate. Due to restrictions, the results in this paper were obtained by manually moving the experimental UAV platform.

B. Results

Quantitative Results. We want to evaluate how incorporating different 3D geometric entities improves the accuracy of registration for a given texture-poor scene. The scene is shown in Figure 3(a). In order to be able to perform

[1] The IMU and camera are integrated in the Intel RealSense D435i, but we did not have access to that model.

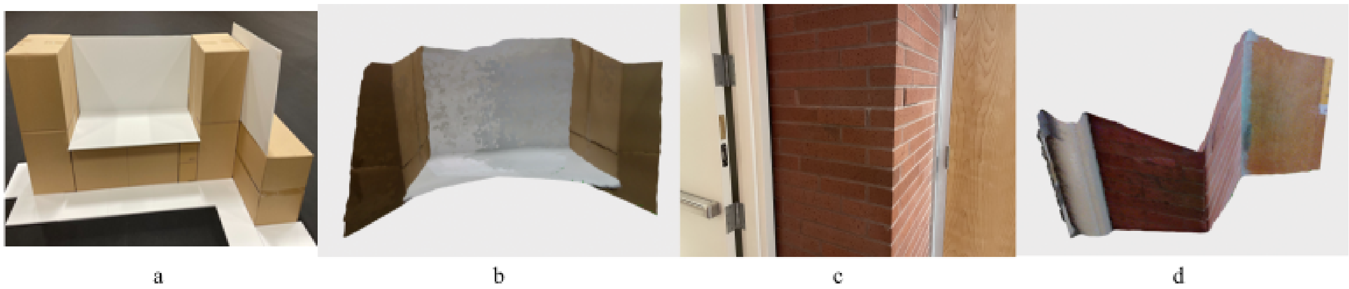

Fig. 3. Point cloud registration results for two environments. (**a**) Box Scene, (**b**) registered point clouds for Box Scene from 125 frames, (**c**) Hallway Scene, (**d**) registered point clouds for Hallway Scene from 59 frames.

TABLE I

COMPARISON OF RMSE, THE AVERAGE DEVIATION FROM THE GROUND TRUTH TRAJECTORY, FOR DIFFERENT COMBINATIONS OF 3D ENTITIES.

Points	Lines	Planes	Pos. RMSE (m)	Rot. RMSE (rad)
-	-	-	0.02465	0.00728
✓	-	-	0.01547	0.00379
-	✓	-	0.01553	0.00380
-	-	✓	0.01554	0.00378
✓	✓	-	0.00235	0.00387
✓	-	✓	0.00235	0.00388
-	✓	✓	0.00548	0.00386
✓	✓	✓	0.00233	0.00388

a quantitative comparison, we used an OptiTrack motion capture system [36] to track six infrared reflective markers mounted on the UAV, using 34 cameras operating at 120 Hz. The motion capture system produces a pose trajectory, i.e. 3D locations and orientations, of the UAV and we refer to this as *ground truth*. The results are summarized in Table I. Columns four and five report position and orientation RMS errors, respectively, between the ground truth and the position and orientation estimated by our framework. Each row indicates which 3D entities are incorporated, except for the first row where we rely on the IMU only, or dead-reckoning. The errors are largest when none of the 3D entities are used. The errors are nearly identical when only one of the entities is incorporated. When incorporating 3D points and lines, and 3D points and planes, the errors improve. However, omitting 3D points but incorporating 3D lines and planes results in an increased RMS error. A possible explanation for this is that the lines and planes do not sufficiently constrain all degrees of freedom in our framework, resulting in drift. Finally, the inclusion of all three entities gives the lowest RMS error. We point out that although these are RMS errors, as more input data are processed, the accumulation of errors causes the estimated trajectory of the UAV to drift more. As is evident from Table I, apart from the case when none of the entities are incorporated, the rotational RMS errors do not vary significantly among the different combinations. One explanation for this is the fact that the gyroscope provides more accurate measurements compared to the accelerometer.

Qualitative Results. Figure 3 shows results of registration for two different environments. The first environment, the *Box Scene* shown in Fig. 3a, was constructed by arranging several cardboard boxes and foam boards to create a challenging, feature-poor scene. We discussed quantitative results for this environment above. The registration result shown in Fig. 3b is for 125 registered point clouds. The second environment is a *Hallway Scene*, depicted in Fig. 3c. The result in Fig. 3d is from 59 registered point clouds. Although the hallway scene is texture-rich, we only rely on 3D entities from the point cloud data. Both environments show qualitatively good registration results.

Figure 4 shows an example of detected 3D points, lines and planes for the Box Scene (Fig. 3a). In general, the number of detected lines and planes will be low. On the other hand, there may be an abundance of 3D points, which would create an imbalance and cause our framework to mainly be corrected by the 3D points. To avoid this imbalance, we limit the number of points for registration by dividing the point cloud of the observed scene into cells and only using a small number of points per cell. The cell size and number of points are hyperparameters that are determined by the user.

To verify whether the lack of texture in the Box Scene indeed hindered 2D visual feature tracking, we employed ORB-SLAM2 [8] to recover the poses of the UAV and to register the point clouds. We found, however, that ORB-SLAM2 was not able to compute the necessary discriminative features for matching and failed entirely.

In general, the accuracy of our method hinges on the quality of the point clouds and the ability to extract 3D features from them.

V. CONCLUSION

We have presented a unified framework to incorporate 3D points, lines and planes for Visual Inertial Odometry. Our approach is suitable for environments which lack texture for computing 2D correspondences, which most prior methods rely on. Our experiments show that by incorporating all three primitives, we can increase the accuracy of registration between subsequent frames. The depth or point clouds acquired with a RGBD camera suffer from noise, and tracking 3D points may be sensitive to such noise. By additionally estimating 3D lines and 3D planes, this noise can be mitigated. To avoid having to estimate so-called anchor points for 3D lines and planes, we only consider the directions of these entities. We thus require 3D lines and planes in different directions to constrain all three dimensions if we cannot robustly track 3D points. However,

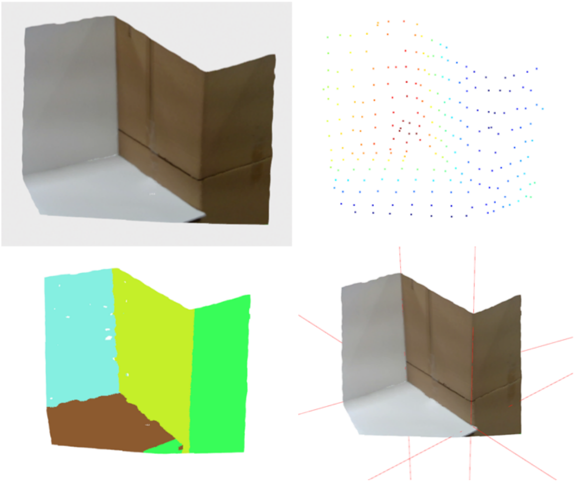

Fig. 4. (Clockwise, from top-left:) Box Scene, detected points, detected lines and detected planes. To avoid an imbalance due to a relatively large number of points, we limit the number of points for registration by dividing the point cloud into cells and only using a small number of points per cell. The numbers of lines and planes tend to be low, and thus are not restricted.

in some parts of an environment there may not be enough information to estimate 3D lines and planes in different directions. In such cases, we can always fall back on *dead reckoning* with the IMU until the necessary conditions are met again, at the expense of some drift. For future work, we would like to implement our framework on an autonomous UAV and further demonstrate its usefulness in texture-poor environments.

References

[1] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An overview to visual odometry and visual SLAM: Applications to mobile robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015.

[2] Y. Taguchi, Y. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *IEEE Intl. Conf. on Robotics and Autom.*, 2013, pp. 5182–5189.

[3] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *IEEE Intl. Conf. on Robotics and Autom.*, 2007, pp. 3565–3572.

[4] D. Zou, Y. Wu, L. Pei, H. Ling, and W. Yu, "StructVIO: Visual-inertial odometry with structural regularity of man-made environments," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 999–1013, 2019.

[5] B. Huang, J. Zhao, and J. Liu, "A survey of simultaneous localization and mapping," *arXiv preprint arXiv:1909.05214*, 2019.

[6] S. Yuan, H. Wang, and L. Xie, "Survey on localization systems and algorithms for unmanned systems," *Unmanned Systems*, vol. 9, no. 02, pp. 129–163, 2021.

[7] R. Mur-Artal, J. Montiel, and J. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, pp. 1147 – 1163, 10 2015.

[8] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] L. Zhang, D. Chen, and W. Liu, "Point-plane SLAM based on line-based plane segmentation approach," in *IEEE Intl. Conf. on Robotics and Biomim.*, 2016, pp. 1287–1292.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *IEEE Intl. Conf. on Comp. Vision*, vol. 2, 1999, pp. 1150–1157 vol.2.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

[12] L. Ma, C. Kerl, J. Stückler, and D. Cremers, "CPA-SLAM: consistent plane-model alignment for direct RGB-D SLAM," in *IEEE Intl. Conf. on Robotics and Autom.*, 2016, pp. 1285–1291.

[13] C. Duan, S. Junginger, J. Huang, K. Jin, and K. Thurow, "Deep Learning for Visual SLAM in Transportation Robotics: A review," *Transp. Safety and Env.*, vol. 1, no. 3, pp. 177–184, 01 2020.

[14] L. Ding and C. Feng, "DeepMapping: Unsupervised map estimation from multiple point clouds," in *IEEE/CVF Conf. on Comp. Vision Patt. Recog.*, June 2019.

[15] M. Milford and A. George, *Featureless Visual Processing for SLAM in Changing Outdoor Environments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 569–583.

[16] S. Yang, Y. Song, M. Kaess, and S. Scherer, "Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments," in *IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2016, pp. 1222–1229.

[17] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *IEEE Intl. Symp. on Mixed Augm. Real.*, 2011, pp. 127–136.

[18] S. Zhao and Z. Fang, "Direct depth SLAM: Sparse geometric feature enhanced direct depth SLAM system for low-texture environments." *Sensors (Basel)*, vol. 18, no. 10, 2018.

[19] J. Liu, Z. Meng, and Z. You, "A robust visual SLAM system in dynamic man-made environments," *Science China Technological Sciences*, vol. 63, no. 9, pp. 1628–1636, 2020.

[20] G. Georgakis, S. Karanam, Z. Wu, J. Ernst, and J. Kosecka, "End-to-end learning of keypoint detector and descriptor for pose invariant 3D matching," in *IEEE/CVF Conf. on Comp. Vision and Patt. Recog.*, Los Alamitos, CA, USA, June 2018, pp. 1965–1973.

[21] B. A. C. Caldato, R. A. Filho, and J. E. C. Castanho, "ORB-ODOM: Stereo and odometer sensor fusion for simultaneous localization and mapping," in *Latin Amer. Robotics Symp.*, 2017, pp. 1–5.

[22] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE Trans. on Patt. Anal. Machine Intell.*, vol. 29, no. 6, pp. 1052–1067, June 2007.

[23] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," *Technical Report*, 2006.

[24] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C.-C. Chu, and H. Hu, "Trifo-VIO: Robust and efficient stereo visual inertial odometry using points and lines," in *IEEE/RSJ Intl. Conf. on Intell. Robots and Sys.*, 2018, pp. 3686–3693.

[25] K. Jung, Y. Kim, H. Lim, and H. Myung, "ALVIO: Adaptive line and point feature-based visual inertial odometry for robust localization in indoor environments," *arXiv preprint arXiv:2012.15008*, 2020.

[26] X. Luo, Z. Tan, and Y. Ding, "Accurate line reconstruction for point and line-based stereo visual odometry," *IEEE Access*, vol. 7, pp. 185 108–185 120, 2019.

[27] H. Wen, J. Tian, and D. Li, "PLS-VIO: Stereo vision-inertial odometry based on point and line features," in *Intl. Conf. on High Performance Big Data and Intelligent Systems (HPBD&IS)*. IEEE, 2020, pp. 1–7.

[28] X. Li, Y. Li, E. P. Örnek, J. Lin, and F. Tombari, "Co-planar parametrization for stereo-SLAM and visual-inertial odometry," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6972–6979, 2020.

[29] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," Ph.D. dissertation, Computer Science department, Tech. Universitaet Muenchen, Germany, October 2009.

[30] B. Hall and B. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, ser. Graduate Texts in Mathematics. Springer, 2003.

[31] L. Xiaohu, L. Yahui, and L. Kai, "Fast 3D line segment detection from unorganized point cloud," *arXiv preprint arXiv:1901.02532*, 2019.

[32] H. Ni, X. Lin, X. Ning, and J. Zhang, "Edge detection and feature line tracing in 3D-point clouds by analyzing geometric properties of neighborhoods," *Remote Sensing*, vol. 8, no. 9, 2016.

[33] C. Dalitz, T. Schramke, and M. Jeltsch, "Iterative Hough Transform for Line Detection in 3D Point Clouds," *Image Processing On Line*, vol. 7, pp. 184–196, 2017.

[34] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes," *IEEE Intl. Conf. on Robotics and Autom.*, pp. 4304–4311, 2016.

[35] R. S. J. Maye, P. Furgale, "Self-supervised calibration for robotic systems," *IEEE Intell. Vehicles Symp.*, pp. 473–480, 2013.

[36] OptiTrack, "Robotics applications," 2021, last accessed: July 2021, https://optitrack.com/applications/robotics.