Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints

Yan Lu, *Student Member, IEEE,* Dezhen Song, *Senior Member, IEEE*

***Abstract*—We present a heterogeneous landmark-based visual navigation approach for a monocular mobile robot. We utilize heterogeneous visual features, such as points, line segments, lines, planes, and vanishing points, and their inner geometric constraints managed by a novel multilayer feature graph (MFG). Our method extends the local bundle adjustment based visual simultaneous localization and mapping (SLAM) framework by explicitly exploiting the heterogeneous features and their inner geometric relationships in an unsupervised manner. As the result, our heterogeneous landmark-based visual navigation (HLVN) algorithm takes a video stream as input, initializes and iteratively updates MFG based on extracted key frames; it also refines robot localization and MFG landmarks through the process. We present pseudo code for the algorithm and analyze its complexity. We have evaluated our method and compared it with state-of-the-art point landmark based visual SLAM methods using multiple indoor and outdoor datasets. In particular, on the KITTI dataset our method reduces the translational error by 52.5% under urban sequences where rectilinear structures dominate the scene.**

***Index Terms*—Heterogeneous landmarks, visual navigation, SLAM**

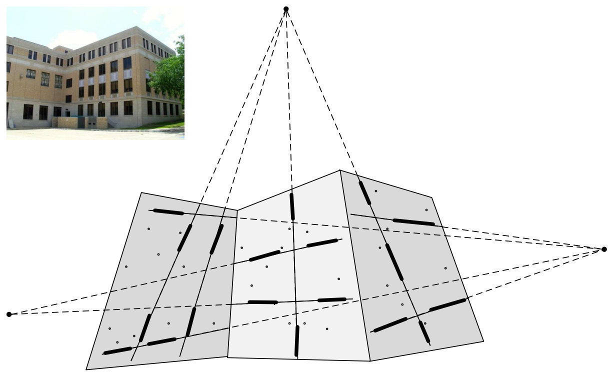
I. Introduction

VISUAL navigation using a single low-cost camera gains more and more attention recently. The fact that cameras in mobile devices like cell phones and tablets are readily available and the increasing needs for navigation assistance in indoor and/or GPS-challenged environments are the main driving forces behind the scene.

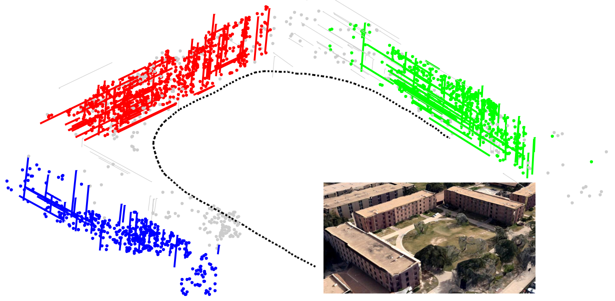
Visual navigation is often conducted under the simultaneous localization and mapping (SLAM) framework. Despite the success reported in literature, state-of-the-art visual SLAM still suffers from drifting and sensitivity to unevenly-distributed features. Besides the limitations of the camera itself (as a bearing-only sensor), another possible reason is that most systems adopt homogeneous low level features (e.g. salient points) as landmarks. Despite the simplicity and elegance in the homogeneous landmarks, the choice cannot fully exploit the information possessed in the higher level landmarks from man-made environments. The top-left thumbnail of Fig. 1(a) shows an example image of a typical urban environment. Observe the abundant lines in parallel directions and the salient building facades. These higher level landmarks can help improve navigation performance if the geometric constraints between them are modeled and utilized properly.

This project is supported in part by the National Science Foundation under IIS-1318638 and NRI-1426752.

Y. Lu and D. Song are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843, USA. Emails: {*ylu, dzsong*}@*cse.tamu.edu*



(a)



(b)

Fig. 1: (a) An illustration of the multilayer feature graph, a data structure organizing heterogeneous landmarks. (b) Sample result of our algorithm, and a Google Earth^{TM}view of the same site from a similar perspective. Coplanar landmarks (points and lines) are coded in the same color, while general landmarks are in gray color. The dotted line is the estimated camera trajectory.

We utilize heterogeneous visual features, including points, line segments, lines, planes, and vanishing points, and their inner geometric constraints to assist robot navigation. This is managed by a novel multilayer feature graph (MFG), our open data structure capturing geometric relationships such as parallelism and coplanarity (see Fig. 1(a)). Our method extends the local bundle adjustment (LBA)-based visual SLAM framework by explicitly exploiting heterogeneous features and their geometric relations in an unsupervised manner. The resulting heterogeneous landmark-based visual navigation (HLVN) algorithm takes a video stream as input, initializes and iteratively updates MFG based on extracted key frames, and refines robot localization and MFG landmarks. We present the algorithm pseudo code and analyze its complexity. We have evaluated the algorithm and compared it with state-of-the-art point landmark-based visual SLAM methods using multiple indoor and outdoor datasets. On the KITTI odometry dataset, our methods reduces the translational error by 52.5% under urban sequences where rectilinear structures dominate the scene.

The major contributions of this work include

- a novel data structure, MFG, that organizes five types of important features and their inner geometric relations for visual navigation,
- a system design that builds up an MFG from a video stream by iteratively verifying, augmenting, and pruning the MFG, and
- an LBA formulation that integrates heterogeneous features and geometric constraints in an unsupervised manner. By "unsupervised" we mean that the LBA incorporates the geometric constraints automatically based on the MFG information, not based on any human inputs.

The rest of the paper is organized as follows. We begin with a review of related work in Section II. We introduce MFG and define the problem in Section III. System design is presented in Section IV, followed by the MFG-based LBA formulation in Section V. The overall HLVN algorithm and complexity analysis are presented in Section VI. We validate our system and algorithm with physical experiments in Section VII and conclude the paper in Section VIII.

II. Related Work

Visual navigation using heterogeneous landmarks mainly relates to two research fields: 3D reconstruction and SLAM.

In computer vision and graphics, 3D reconstruction has been a very popular topic for research as well as commercial applications. Besides regular cameras, sensors used for 3D reconstruction also include laser range finder [1] and more often, aerial cameras [2]. Google Earth and Microsoft Virtual Earth are successful showcases for 3D reconstruction of city models [3]. Following the taxonomy of Seitz et al. [4], 3D reconstruction algorithms are categorized into the following classes: voxel approaches [5], level-set techniques [6], line segment matching [7], polygon mesh methods [8], and algorithms that compute and merge depth maps [9]. Unlike those methods, our work does not pursue a full scale reconstruction. This is because 3D reconstruction usually needs repetitive scene scanning, which is often not the main task for robots.

In robotics research, the most common external sensors for robot navigation include sonar arrays, laser range finders, GPS, cameras and their combinations. SLAM is the typical framework employed in robot navigation [10], [11]. In SLAM, the physical world is represented as a collection of landmarks. For example, point clouds serve as landmarks when a laser range finder is the primary sensor [12]. In particular, our work belongs to the visual SLAM category, where cameras provide the main sensory input [13]–[16].

There are two prevalent methodologies in visual SLAM: the bundle adjustment (BA) approaches (e.g., [13]) rooted in the structure from motion area in computer vision, and the filtering methods (e.g. [17]) originated from the traditional SLAM field of robotics research. Strasdat et al. have analyzed the advantages of each method in [18]. For both methods, various camera configurations/modalities have been studied, including a monocular camera [19], a stereo camera [20]–[22], an omnidirectional camera [23], a camera with range sensors [24], [25], and an RGB-D camera [26], [27].

Besides methodology and sensor configuration, another critical issue in visual SLAM is scene representation. For example, point clouds and sparse feature points [28] are often employed as landmarks in a map. Recently, many researchers have realized that landmark selection is an important factor in visual odometry and SLAM [29]. Lower level landmarks such as corners [30] and SIFT points [31] are relatively easy to use due to their geometric simplicity. However, point features are merely mathematical singularities in color, texture and/or geometric spaces. They are difficult to interpret and use for scene understanding or human-robot interaction. They are also easily influenced by lighting and shadow conditions. To overcome these shortcomings, higher level landmarks have received more and more attention for visual SLAM, such as line segments [32]–[35], straight lines [36], vanishing points [37], and planes [38]–[41].

These works have demonstrated the advantages of higher level landmarks in robustness and accuracy, but they either treat these landmarks as isolated objects, or partially explore the inner relations between them. This treatment simplifies the SLAM problem formulation but cannot fully utilize the power of heterogeneous landmarks. Very recently, Tretyak et al. present an optimization framework for geometric parsing of image by jointly using edges, line segments, lines, and vanishing points [42]. However, this method has not been applied to navigation yet.

We propose the initial concept of MFG for organizing heterogeneous landmarks and build a prototype based on two views in 2012 [43]. The two view-based MFG is then applied to building facades mapping under an EKF framework [44]. Inspired by [18], we further develop an image sequence-based MFG for visual navigation using heterogeneous features under the LBA framework [45]. This paper extends our conference papers [43], [45] by adding algorithm analysis and physical experiments.

III. Problem Formulation

A. Assumptions and Notations

Consider a mobile robot navigating in a previously unknown environment with a monocular camera. We make two assumptions here:

a.1 The robot operates in a largely static man-made environment with rectilinear structures consisting of parallel lines which are not necessarily in orthogonal directions.

a.2 The camera is pre-calibrated with its radial distortion removed.

Let us define the following notations,

$\mathcal{V}$	Input camera video,
I_k	k-th key frame extracted from $\mathcal{V}$, $I_k \in \mathcal{V}, k \in \mathbb{N}$,
$\{C_k\}$	3D camera coordinate system for I_k, a right-handed coordinate system with its origin at the camera

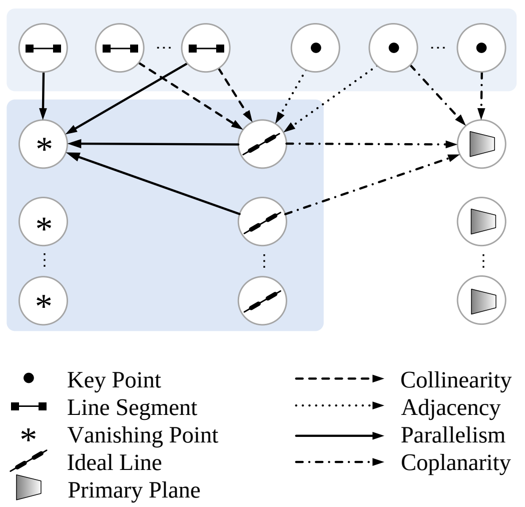


Fig. 2: The whole graph represents a 3D MFG, and the shaded regions jointly represent a 2D MFG. Geometric relationships between nodes are represented by edges of different line types.

	optical center, its Z-axis coinciding with the optical axis and pointing to the forward direction of the camera, and its X-axis and Y-axis parallel to the horizontal and vertical directions of the CCD sensor plane, respectively,
$\{I_k\}$	2D image coordinate system for I_k, with its origin on the image plane and its u-axis and v-axis parallel to the X-axis and Y-axis of $\{C_k\}$, respectively,
$\{W\}$	3D world coordinate system,
K	Camera calibration matrix,
R_k	Camera rotation matrix at I_k with respect to $\{W\}$,
t_k	Camera translation vector at I_k with respect to $\{W\}$,
P_k	Camera projection matrix, $\mathrm{P}_k = \mathrm{K}\,[\mathrm{R}_k \,\vert\, \mathrm{t}_k]$,
$\mathrm{R}_{k_2}^{k_1}$	Relative rotation matrix between I_{k_1} and I_{k_2} defined as $\mathrm{R}_{k_2}^{k_1} = \mathrm{R}_{k_2}\mathrm{R}_{k_1}^{-1}$
$\mathrm{t}_{k_2}^{k_1}$	Relative translation between I_{k_1} and I_{k_2} defined as $\mathrm{t}_{k_2}^{k_1} = \mathrm{t}_{k_2} - \mathrm{R}_{k_2}^{k_1}\mathrm{t}_{k_1}$,
$X_{i:j}$	Collection defined as $X_{i:j} = \{X_k \vert i \le k \le j\}$,
$\mathbf{m}_k$	A 2D MFG (defined later) constructed for I_k,
$\mathcal{M}_k$	3D MFG (defined later) constructed based on $I_{0:k}$,
$\mathbb{E}^n$	n-dimensional Euclidean space,
$\mathbb{P}^n$	n-dimensional projective space, and
$\mathbf{X}$	A homogeneous vector, $\mathbf{X} = [\tilde{\mathbf{X}}^\mathsf{T}, 1]^\mathsf{T}$, where $\tilde{\mathbf{X}}$ denotes the inhomogeneous counterpart of $\mathbf{X}$. $\mathbf{X} \in \mathbb{P}^n \Rightarrow \tilde{\mathbf{X}} \in \mathbb{E}^n$.

We abuse "=" to denote real equality and up-to-scale equality for inhomogeneous and homogeneous vectors, respectively.

B. Overview of Multilayer Feature Graph

As illustrated in Fig. 2, we propose multilayer feature graph for organizing heterogeneous features and their inner geometric relations. We first introduce the five types of features in MFG.

1) **Key points** represent point features. We refer to point features detected from images as 2D key points, which only reside in image space. Thus the set of 2D key points detected in I_k is denoted by $\{\mathbf{p}_{i,k} \in \mathbb{P}^2, i = 1, 2, \cdots\}$. We refer to spacial points as 3D key points, and represent them w.r.t. $\{W\}$ by $\{\mathbf{P}_j \in \mathbb{P}^3, j = 1, 2, \cdots\}$. The observation of $\mathbf{P}_j$ in I_k, if existing, is denoted by $\mathbf{p}_{j(k)}$. Therefore, if $\mathbf{p}_{i,k}$ is the observation of $\mathbf{P}_j$ in I_k, then we have $\mathbf{p}_{j(k)} = \mathbf{p}_{i,k}$ by definition. Note that the subscript convention used in naming $\mathbf{p}_{i,k}$ and $\mathbf{p}_{j(k)}$ also applies to other types of features in this paper.
2) **Line segments** represent finite linear objects. We denote a 2D line segment in I_k by $\mathbf{s}_{i,k} = [\mathbf{d}_{i1,k}^\mathsf{T}, \mathbf{d}_{i2,k}^\mathsf{T}]^\mathsf{T}$, where $\mathbf{d}_{i1,k}$ and $\mathbf{d}_{i2,k}$ are the endpoints. We represent a 3D line segment in $\{W\}$ by $\mathbf{S}_i = [\mathbf{D}_{i1}^\mathsf{T}, \mathbf{D}_{i2}^\mathsf{T}]^\mathsf{T}$.
3) **Ideal lines** (defined later in Definition 2) represent infinite lines. A 2D ideal line in I_k is represented by $\mathbf{l}_{i,k} \in \mathbb{P}^2$. We represent a 3D ideal line by $\mathbf{L}_i = [\mathbf{Q}_i^\mathsf{T}, \mathbf{J}_i^\mathsf{T}]^\mathsf{T}$, where $\mathbf{Q}_i \in \mathbb{P}^3$ is a finite 3D point located on $\mathbf{L}_i$, and $\mathbf{J}_i \in \mathbb{P}^3$ is an infinite 3D point defining the direction of $\mathbf{L}_i$. The observation of $\mathbf{L}_i$ in I_k is denoted by $\mathbf{l}_{i(k)}$.
4) **Vanishing points** represent particular directions of parallel 3D lines. We denote a 2D vanishing point in I_k by $\mathbf{v}_{i,k}$, and a 3D vanishing point in $\{W\}$ by $\mathbf{V}_i$. $\mathbf{V}_i \in \mathbb{P}^3$ is an infinite 3D point, and its observation in I_k is denoted by $\mathbf{v}_{i(k)}$.
5) **Primary planes** represent dominant planar surfaces (e.g. building facades) and only exist in 3D space. We denote a primary plane by $\mathbf{\Pi}_i = [\mathrm{n}_i^\mathsf{T}, d_i]^\mathsf{T}$, where $\mathrm{n}_i \in \mathbb{E}^3$ and $d_i \in \mathbb{R}$, such that $\mathbf{X}^\mathsf{T}\mathbf{\Pi}_i = 0$ for any point $\mathbf{X}$ on the plane.

MFG exists in both $\{I_k\}$ and $\{W\}$. In $\{I_k\}$, we name it as a 2D MFG, which consists of 2D key points, 2D line segments, 2D ideal lines, and 2D vanishing points as its nodes. The geometric relationships between 2D features, including adjacency, collinearity, and parallelism, are represented by the edges of 2D MFG. A 2D MFG effectively summarizes the feature information of a frame. Thus we construct a 2D MFG for each key frame I_k and denote it by $\mathbf{m}_k$. In Fig. 2, the shaded regions jointly represent the structure of a 2D MFG. The top shaded region consists of raw features that are directly extracted from images, while the lower shaded region includes features that need to be abstracted from raw features.

In $\{W\}$, we define a 3D MFG, which contains 3D key points, 3D line segments, 3D ideal lines, 3D vanishing points, and primary planes as its nodes. The edges of 3D MFG represent geometric relationships including collinearity, parallelism, and coplanarity. There is only one 3D MFG in $\{W\}$ and we use $\mathcal{M}_k$ to denote the 3D MFG constructed/updated based upon $I_{0:k}$.

C. Problem Definition

Our ultimate goal is to construct a 3D MFG from an input video. To achieve this goal, we utilize an iterative method to solve the following problem.

Definition 1: Given video $\mathcal{V}$, MFG $\mathcal{M}_{k-1}$, and historical camera poses $\{\mathrm{R}_{0:k-1}, \mathrm{t}_{0:k-1}\}$ for $k \ge 1$, select key frame I_k, estimate camera pose $\{\mathrm{R}_k, \mathrm{t}_k\}$, refine $\{\mathrm{R}_{0:k}, \mathrm{t}_{0:k}\}$, and update the nodes and edges of $\mathcal{M}_{k-1}$ to obtain $\mathcal{M}_k$.

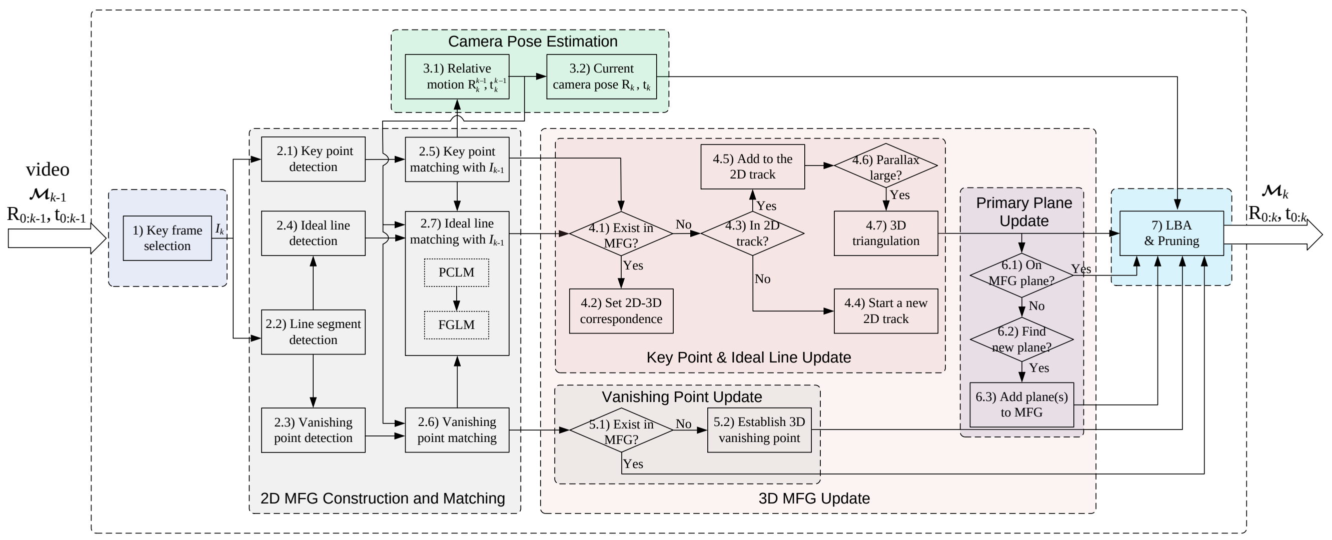

video
$\mathcal{M}_{k\text{-}1}$
$R_{0:k\text{-}1}$, $t_{0:k\text{-}1}$
1) Key frame selection
I_k
Camera Pose Estimation
3.1) Relative motion R_k^{k-1}, t_k^{k-1}
3.2) Current camera pose R_k, t_k
2.1) Key point detection
2.5) Key point matching with $I_{k\text{-}1}$
2.4) Ideal line detection
2.7) Ideal line matching with $I_{k\text{-}1}$
PCLM
FGLM
2.2) Line segment detection
2.3) Vanishing point detection
2.6) Vanishing point matching
2D MFG Construction and Matching
4.1) Exist in MFG?
No
Yes
4.2) Set 2D-3D correspondence
4.3) In 2D track?
4.4) Start a new 2D track
4.5) Add to the 2D track
4.6) Parallax large?
4.7) 3D triangulation
Key Point & Ideal Line Update
Vanishing Point Update
5.1) Exist in MFG?
5.2) Establish 3D vanishing point
3D MFG Update
Primary Plane Update
6.1) On MFG plane?
6.2) Find new plane?
6.3) Add plane(s) to MFG
7) LBA & Pruning
$\mathcal{M}_k$
$R_{0:k}$, $t_{0:k}$

Fig. 3: System Diagram

IV. System Design and Multilayer Feature Graph

Fig. 3 shows our system architecture, where the main blocks are shaded by different colors. The system takes a video as input and proceeds iteratively. During each iteration, the system selects a key frame I_k, extracts a 2D MFG $\mathbf{m}_k$, and finds 2D feature correspondences between $\mathbf{m}_k$ and $\mathbf{m}_{k-1}$, which are used to estimate camera pose and establish 3D features for $\mathcal{M}_k$. The last step of each iteration performs LBA to jointly refine camera poses and 3D MFG features.

To start, we select the first video frame as key frame I_0. We let $\mathcal{M}_0 = \emptyset$ and $\{W\}$ coincide with $\{C_0\}$.

A. *Key Frame Selection*

Given a video, it is necessary to select a set of key frames for motion estimation and 3D reconstruction. The basic principle is to find a good balance between two needs: 1) wide baseline to avoid ill-posed epipolar geometry problems and 2) sufficient overlap of scene between key frames. Based on existing methods (e.g. [19]), we use the following criteria for key frame selection when $k \geq 1$. Given I_{k-1} and $\mathcal{M}_{k-1}$, a video frame I is chosen as key frame I_k if it satisfies:

1) the number of 2D point matches between I_{k-1} and I is not less than a threshold N_2,
2) for $k \geq 2$, the number of 3D key points (from $\mathcal{M}_{k-1}$) observable in I is not less than a threshold N_3,
3) for $k \geq 2$, the rotation angle between I_{k-1} and I is not larger than a threshold τ_R, and
4) there are as many video frames as possible between I_{k-1} and I.

B. *2D MFG Construction and Matching*

From I_k we construct a 2D MFG $\mathbf{m}_k$, and match $\mathbf{m}_k$ with $\mathbf{m}_{k-1}$ for $k \geq 1$ to establish 2D-2D matching for heterogeneous features. We discuss the extraction and matching for each type of features separately.

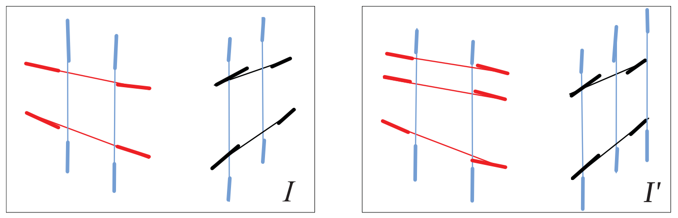


Fig. 4: An example of vanishing point matching. The line segments and ideal lines associated with the same vanishing points are drawn in the same color.

1) ***Key Points****:* We detect 2D key points from I_k using the corner detector proposed in [30], though alternatives such as SIFT are also applicable. We track 2D feature points across frames using the iterative Lucas-Kanade method with pyramids [46]. Thus, putative key point correspondences between I_k and I_{k-1} (for $k \geq 1$) are readily obtained from the tracking result (see Box 2.5 in Fig. 3). To remove false matching, the putative matches are fed into a five-point algorithm-based RANSAC [47] to estimate the essential matrix E. We also compute the relative camera rotation R_k^{k-1} by decomposing E. Note that although the relative motion estimation (in Box 3.1) belongs to the "camera pose estimation" block in Fig. 3, it is indeed conducted as soon as putative key point correspondences are available.

2) ***Line Segments****:* We detect 2D line segments from I_k using LSD [48] (see Box 2.2 in Fig. 3). Line segments provide more information in addition to key points, but line segment matching is hard due to the lack of distinctive descriptors and the instability of endpoint detection. However, we use line segments to find vanishing points.

3) ***Vanishing Points****:* We detect vanishing points from 2D line segments using RANSAC (see Box 2.3 in Fig. 3). In a 2D MFG, each vanishing point has a set of child line segments, which are actually parallel to each other in 3D space.

To find correspondences between two vanishing point sets,

$\{\mathbf{v}_{i,k-1}|i=1,\cdots\}$ and $\{\mathbf{v}_{j,k}|j=1,\cdots\}$, we compute

$$\theta_{ij} = \cos^{-1}(|(\mathrm{K}^{-1}\mathbf{v}_{i,k-1})^{\mathsf{T}}\mathrm{R}_k^{k-1}\mathrm{K}^{-1}\mathbf{v}_{j,k}|), \forall i,j, \quad (1)$$

which represents the angle between the two vanishing point directions in 3D.

Let $\theta^*_{\cdot j} = \min_\iota(\theta_{\iota j})$, and $\theta^*_{i\cdot} = \min_\iota(\theta_{i\iota})$. We claim $\mathbf{v}_{i,k-1} \leftrightarrow \mathbf{v}_{j,k}$ as a correspondence if it holds that

$$\theta_{ij} = \theta^*_{\cdot j} = \theta^*_{i\cdot} \leq \tau_\theta, \quad (2)$$

where τ_θ is a user-specified upper bound. Fig. 4 shows an example of vanishing point matching result.

*4) **Ideal Lines**:* As illustrated in Fig. 5(a), line segments tend to be short and fragmented. However, there are usually many small groups of collinear line segments which come from the same 3D linear structure. If we fuse the information of collinear line segments to form a single line, the accuracy and robustness should be improved. Therefore, we introduce ideal lines.

Definition 2 (Ideal Line): An ideal line is defined as a real or virtual line passing through a set of collinear line segments.

We detect 2D ideal lines from line segments using sequential RANSAC (see Box 2.4 in Fig. 3). To reduce the problem size, we group line segments by vanishing point and detect ideal lines group by group. After a set of collinear line segments $\{\mathbf{s}_i\}$ is found, we compute the ideal line as

$$\mathbf{l}^* = \operatorname*{argmin}_{\mathbf{l}} \sum_i \sum_{j=1}^{2} d_\perp(\mathbf{d}_{ij}, \mathbf{l})^2, \quad (3)$$

where $d_\perp(\cdot,\cdot)$ denotes the perpendicular distance from a point to a line in 2D. This method is effectively the maximum likelihood estimation (MLE) when each line segment endpoint is subject to an isotropic Gaussian noise as illustrated in Fig. 5(a). In MFG, a line segment must have only one ideal line as its parent node, and an ideal line may have multiple line segment nodes as its children.

We consider two ideal lines from different images to be matched if they correspond to the same 3D line. Since vanishing point matching is done, we narrow down the ideal line matching problem by only considering lines connected to the same vanishing point. We present a two-stage approach to ideal line matching here (see Box 2.7 in Fig. 3).

Stage 1: Point Correspondence-based Line Matching

In Stage 1, we adopt a point correspondence-based line matching (PCLM) method proposed by [49]. To apply this method, we first introduce the neighbor region of an ideal line. For an ideal line $\mathbf{l}$, let $\mathbf{d}_A$ and $\mathbf{d}_B$ be the two farthest line segment endpoints associated with $\mathbf{l}$, and $\mathbf{d}'_A$ and $\mathbf{d}'_B$ be the projections of $\mathbf{d}_A$ and $\mathbf{d}_B$ onto $\mathbf{l}$, respectively. Let $\mathbf{b}$ be the perpendicular bisector of line segment $\mathbf{d}'_A\mathbf{d}'_B$, as illustrated in Fig. 5(b). We define the neighbor region of $\mathbf{l}$ to be

$$\lambda(\mathbf{l}) := \left\{\mathbf{x} \in \mathbb{P}^2 : d_\perp(\mathbf{x},\mathbf{b}) \leq \frac{\|\mathbf{d}'_A\mathbf{d}'_B\|}{2}, d_\perp(\mathbf{x},\mathbf{l}) \leq \frac{\sigma_b}{2}\right\}, \quad (4)$$

where $\|\mathbf{d}'_A\mathbf{d}'_B\|$ is the length of line segment $\mathbf{d}'_A\mathbf{d}'_B$, and $\sigma_b = \min\{100, (\text{image width})/12\}$. In Fig. 5(b), $\lambda(\mathbf{l})$ is the rectangular area enclosed by dotted lines. If a key point $\mathbf{p} \in \lambda(\mathbf{l})$, then we say $\mathbf{p}$ is adjacent to $\mathbf{l}$ and they are connected in 2D MFG.

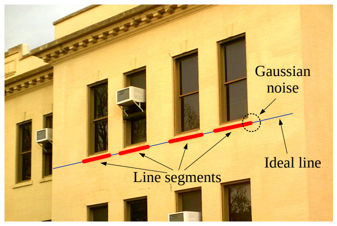
The intuition of the PCLM method is that for an ideal line match $\mathbf{l}_{i,k-1} \leftrightarrow \mathbf{l}_{j,k}$, point correspondences in their neighbor regions must satisfy

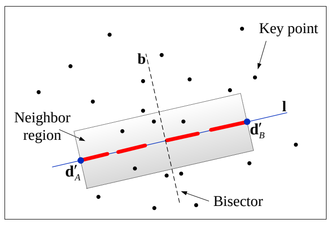
$$\frac{\mathbf{l}_{i,k-1}^{\mathsf{T}}\mathbf{p}_{a,k-1}}{\mathbf{l}_{i,k-1}^{\mathsf{T}}\mathbf{p}_{b,k-1}} = \frac{\mathbf{l}_{j,k}^{\mathsf{T}}\mathbf{p}_{c,k}}{\mathbf{l}_{j,k}^{\mathsf{T}}\mathbf{p}_{d,k}} \quad (5)$$

where $\mathbf{p}_{a,k-1} \leftrightarrow \mathbf{p}_{c,k}$ and $\mathbf{p}_{b,k-1} \leftrightarrow \mathbf{p}_{d,k}$ are two pairs of point correspondences satisfying $\mathbf{p}_{a,k-1}, \mathbf{p}_{b,k-1} \in \lambda(\mathbf{l}_{i,k-1})$, and $\mathbf{p}_{c,k}, \mathbf{p}_{d,k} \in \lambda(\mathbf{l}_{j,k})$.

The PCLM method provides very accurate matching result, but tends to fail when the neighbor regions are textureless. To handle this issue, in Stage 2 we use an F-guided line matching (FGLM) method [50], which utilizes the fundamental matrix F. To be specific, we take the line matches found by PCLM out of the candidate sets, and apply FGLM to the remaining ideal lines. The fundamental matrix is computed as $\mathrm{F} = \mathrm{K}^{-\mathsf{T}}\mathrm{E}\mathrm{K}^{-1}$.

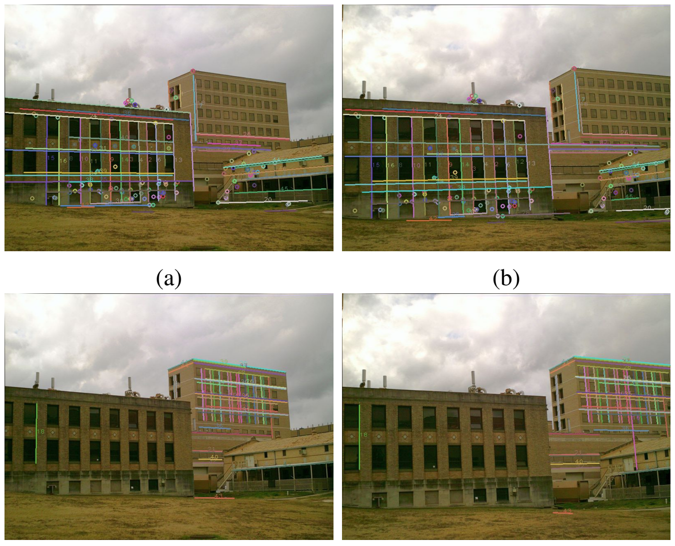
Stage 2: F-Guided Line Matching





(a) (b)

Fig. 5: (a) An example of an ideal line, (b) An example of the adjacency between key points and ideal lines



(a) (b) (c) (d)

Fig. 6: An example of our two-stage approach to ideal line matching. (a) and (b): ideal line matches found in Stage 1 by the PCLM method, each pair of line match is plotted in the same color, and small circles represent point correspondences used by PCLM; (c) and (d): additional matches found by the FGLM method in Stage 2. (Best viewed in color)

Since the FGLM method essentially works with line segments, we treat an ideal line as an augmented line segment. To be specific, we only use the part of ideal line inside the neighbor region (for example, $\mathbf{d}'_A\mathbf{d}'_B$ in Fig. 5(b)) since this is the part that bears most appearance information.

For a point $\mathbf{x}$ on the augmented line segment of $\mathbf{l}_{i,k-1}$, we find its correspondence $\mathbf{x}' = \mathbf{l}_{j,k} \times (\mathrm{F}\mathbf{x})$, assuming $\mathbf{l}_{i,k-1}$ and $\mathbf{l}_{j,k}$ correspond to each other. The basic idea of FGLM is to compute the matching score of a pair of line segments as the average of the individual correlation scores for the points (pixels) of the lines. Although FGLM does not produce very accurate results, but it is complementary to the PCLM method. As an example, Fig. 6(a) and 6(b) show the matching result of the PCLM method, and Fig. 6(c) and 6(d) show the result of the FGLM method. It is obvious that the two-stage matching approach is able to find more ideal line correspondences.

C. Camera Pose Estimation

With 2D feature correspondences obtained, estimating the 6 degrees of freedom (DoF) camera pose R_k and t_k for I_k is a key step for constructing and updating 3D MFG. Existing methods (e.g. [13]) usually solve this problem using 3-point algorithm based on the 3D-2D correspondences $\{\mathbf{P}_i \leftrightarrow \mathbf{p}_{i(k)}\}$ between known 3D points and their observations in I_k. This method omits those 2D-2D correspondences between I_{k-1} and I_k whose 3D positions are unknown yet. This omission will lead to large estimation error when observed 3D points are few. Various approaches exist to handle this issue, e.g. using Kalman filtering [51] or three-view constraints [52]. A good fit for our system is a method proposed by Tardif et al. [23] that decouples the estimation of R_k from t_k in two steps. We adopt this method with the modification as follows.

Step 1: Compute essential matrix E as described in Section IV-B1. Decompose E to recover the relative camera rotation R_k^{k-1} and translation t_k^{k-1}, with $\|\mathrm{t}_k^{k-1}\|$ unknown.

Step 2: Compute the translation distance $\|\mathrm{t}_k^{k-1}\|$ using 3D-2D correspondences through a RANSAC process where only one correspondence is needed for a minimal solution. This completes the 6 DoF estimation.

In the Step 2 of [23], Tardif et al. estimate the full 3 DoFs of t_k^{k-1} using two 3D-2D correspondences for a minimal solution. This difference can be justified by the different cameras used - an omnidirectional camera in [23] with 360° horizontal field of view (HFOV) vs. a regular camera we use with $40°-80°$ HFOV. Narrower HFOV results in fewer observable 3D landmarks in view and thus fewer 3D-2D correspondences, especially in a turning situation. Therefore, we choose to reduce the problem dimension in Step 2 to fit our needs.

It is worth noting that when $k = 1$, we do not need Step 2, but set $\|\mathrm{t}_k^{k-1}\| = 1$. This fixes the scale of the following estimations.

D. 3D MFG Update

We initialize $\mathcal{M}_k$ by letting $\mathcal{M}_k = \mathcal{M}_{k-1}$ and then perform 3D MFG update for $\mathcal{M}_k$ using 2D information just obtained. This is a process of associating 2D features in $\mathbf{m}_k$ with 3D landmarks in $\mathcal{M}_k$ and introducing new 3D landmarks into $\mathcal{M}_k$. We present details for each type of landmarks as follows.

*1) **Key Point Update:*** Key point update involves associating image observations to existing 3D key points, and establishing new 3D key points using new 2D key point correspondences (see Boxes 4.1-4.6 in Fig. 3).

A 2D point correspondence must have sufficient parallax to be used for computing a 3D point. Here we define the parallax of a 2D key point correspondence $\mathbf{p}_{i,k_1} \leftrightarrow \mathbf{p}_{j,k_2}$ as

$$\rho(\mathbf{p}_{i,k_1}, \mathbf{p}_{j,k_2}) := \langle \mathrm{K}^{-1}\mathrm{H}_\mathrm{r}\mathbf{p}_{i,k_1}, \mathrm{K}^{-1}\mathbf{p}_{j,k_2}\rangle \tag{6}$$

$$\text{with } \mathrm{H}_\mathrm{r} = \mathrm{K}\mathrm{R}_{k_2}^{k_1}\mathrm{K}^{-1} \tag{7}$$

where H_r represents a rotational homography [53], $\langle\cdot,\cdot\rangle$ indicates the angle between two vectors, and $\mathrm{R}_{k_2}^{k_1}$ has been computed in Section IV-C.

For a 2D key point correspondence $\mathbf{p}_{i,k-1} \leftrightarrow \mathbf{p}_{j,k}$,

- if it is a re-observation of key point $\mathbf{P}_\iota$, we make the association by letting $\mathbf{p}_{\iota(k)} = \mathbf{p}_{j,k}$.
- if it is a newly discovered point, compute its parallax $\rho(\mathbf{p}_{i,k-1}, \mathbf{p}_{j,k})$ using (6). If $\rho(\mathbf{p}_{i,k-1}, \mathbf{p}_{j,k}) > \tau_\rho$ where τ_ρ is a parallax threshold, we triangulate it and add the 3D point to $\mathcal{M}_k$ as a new key point. Otherwise, we set up a new 2D key point track $\mathcal{Q}_q = \{\mathbf{p}_{i,k-1}, \mathbf{p}_{j,k}\}$ to keep track of it for potential triangulation in the future. A 2D key point track is a collection of 2D key points corresponding to a 3D point whose position is not computed yet due to insufficient parallax.
- if it is an observation of an existing 2D key point track $\mathcal{Q}_q$, we append it to the track $\mathcal{Q}_q = \mathcal{Q}_q \cup \{\mathbf{p}_{j,k}\}$, and check whether $\mathcal{Q}_q$ can be converted to a 3D key point. To do this, we compute the parallax between $\mathbf{p}_{j,k}$ and each of the rest points in $\mathcal{Q}_q$. If anyone is larger than τ, we compute a 3D point from all points in $\mathcal{Q}_q$ and add it to $\mathcal{M}_k$; $\mathcal{Q}_q$ is then deleted.

*2) **Vanishing Point Update:*** Vanishing point update is straightforward (see Boxes 5.1-5.2 in Fig. 3). Given a 2D vanishing point $\mathbf{v}_{i,k}$, if it is a re-observation of existing $\mathbf{V}_j$, let $\mathbf{v}_{j(k)} = \mathbf{v}_{i,k}$. Otherwise, establish a new vanishing point node $\mathbf{V}_j = [\mathbf{v}_{i,k}^\mathsf{T}\mathrm{R}_k, 0]^\mathsf{T}$. It is trivial but important to update the edges between ideal lines and vanishing points whenever a new ideal line or vanishing point node is added.

*3) **Ideal Line Update:*** Before presenting the ideal line update algorithm, we need to define the parallax for ideal lines. Generally speaking, parallax has not been clearly defined for lines. Here we propose a heuristic parallax measurement for ideal lines by leveraging their line segment endpoints. For a 2D ideal line correspondence $\mathbf{l}_{i,k_1} \leftrightarrow \mathbf{l}_{j,k_2}$, define

$$\varrho(\mathbf{l}_{i,k_1}, \mathbf{l}_{j,k_2}) := \frac{1}{n}\sum_{\iota=1}^{n}\rho(\mathbf{d}_{\iota,k_1}, \mathbf{d}_{\iota,k_2}^{+}) \tag{8}$$

where $\{\mathbf{d}_{\iota,k_1} | \iota = 1, \cdots, n\}$ denotes the endpoints of line segments that support $\mathbf{l}_{i,k_1}$, and $\mathbf{d}_{\iota,k_2}^{+}$ is the perpendicular foot of $\mathbf{d}'_{\iota,k_2} := \mathrm{H}_\mathrm{r}\mathbf{d}_{\iota,k_1}$ on $\mathbf{l}_{j,k_2}$ in I_{k_2}, as illustrated in Fig. 7.

The rationale is that we want to reward line correspondences which have larger distance in their perpendicular direction. If

$\mathbf{l}'_{\iota,k_2} := \mathrm{H}_\mathrm{r}^{-\mathsf{T}}\mathbf{l}_{\iota,k_1}$ overlap with $\mathbf{l}_{j,k_2}$, their parallax should be zero.

With the parallax defined, the ideal line update is performed in a similar fashion to the key point case (i.e. Boxes 4.1-4.6 in Fig. 3), and thus skipped here.

Remark 1: 3D Line segments are also updated in this process. Since a line segment always has an ideal line parent, when a 2D ideal line is converted to 3D, its associated line segments are also converted to 3D. Their 3D positions are computed based on the 3D ideal line parameters.

*4) **Primary Plane Update:*** Detecting primary planes is of great importance for robot navigation. Here we detect primary planes by finding coplanar 3D key points and ideal lines using RANSAC. To be specific, let $\mathcal{C}$ be a collection of 3D key points and ideal lines which are not yet associated with any primary plane. We briefly describe two key steps of RANSAC below.

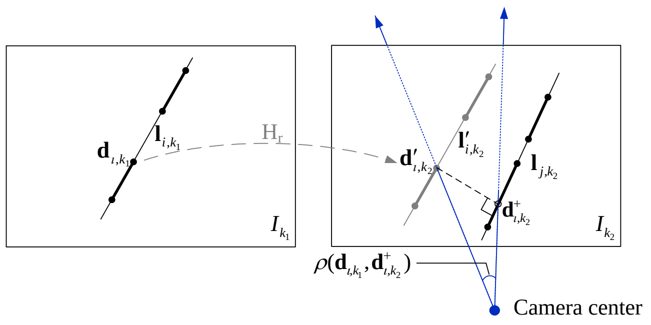
1) Compute a plane candidate $\mathbf{\Gamma}$ from a minimal solution set, which could include either 3 key points, or 2 parallel ideal lines, or 1 key point plus 1 ideal line.
2) $\forall c \in \mathcal{C}$, compute a consensus score $f(c, \mathbf{\Gamma})$ as follows.

$$f(c,\mathbf{\Gamma}) = \begin{cases} \delta_\perp(c,\mathbf{\Gamma}) & \text{if } c \text{ is a key point} \\ \frac{1}{n}\sum_{i=1}^{n}\delta_\perp(D_i,\mathbf{\Gamma}) & \text{if } c \text{ is an ideal line} \end{cases} \tag{9}$$

where $\delta_\perp(\cdot,\cdot)$ denotes the perpendicular distance from a point to a plane in 3D, and $\{D_i | i = 1, \cdots, n\}$ is the set of 3D endpoints associated with ideal line c. Therefore, if c is an ideal line, $f(c, \mathbf{\Gamma})$ is the average of the distances from its associated line segment endpoints to $\mathbf{\Gamma}$.

If the size of the largest consensus set is greater than a threshold N_{cp}, we add the corresponding plane candidate to $\mathcal{M}_k$ as a primary plane, and establish edges between it and the key points and ideal lines in the consensus set. To control the problem size, we do not include all 3D key points or ideal lines in $\mathcal{C}$. Instead, we only take into account those recently established landmarks. Here we enforce $|\mathcal{C}| \leq$450.

Moreover, when new 3D key points or ideal lines are established, we check if they belong to existing primary planes using the metric defined by (9) and add edges accordingly. An ideal line may have two parent primary planes if it is a boundary line.



$\mathbf{d}_{\iota,k_1}$
$\mathbf{l}_{\iota,k_1}$
H_r
I_{k_1}
$\mathbf{d}'_{\iota,k_2}$
$\mathbf{l}'_{\iota,k_2}$
$\mathbf{l}_{j,k_2}$
$\mathbf{d}^+_{\iota,k_2}$
I_{k_2}
$\rho(\mathbf{d}_{\iota,k_1}, \mathbf{d}^+_{\iota,k_2})$
Camera center

Fig. 7: Illustration of parallax computation for 2D ideal lines. H_r is a rotational homography mapping defined in (7). Bold lines are supporting line segments of the underlying (thin) ideal line. $\rho(\mathbf{d}_{\iota,k_1}, \mathbf{d}^+_{\iota,k_2})$ is the parallax between points $\mathbf{d}_{\iota,k_1}$ and $\mathbf{d}^+_{\iota,k_2}$.

V. LBA and Pruning

A. LBA with Geometric Constraints

After the 3D MFG is updated, we further refine the estimated camera poses and 3D landmarks jointly using LBA (see Box 7 in Fig. 3). Inspired by [23], we use w latest key frames to bundle adjust m latest camera poses and MFG nodes established since I_{k-m+1}, with $w \geq m$ usually. To account for the various feature types and geometric constraints in MFG, we define cost functions accordingly.

1) Key Points: Denote the reprojection of key point $\mathbf{P}_i$ in I_k by $\hat{\mathbf{p}}_{i(k)} := \mathrm{P}_k\mathbf{P}_i$. Recall that the observation of $\mathbf{P}_i$ in I_k is $\mathbf{p}_{i(k)}$. Usually $\hat{\mathbf{p}}_{i(k)} \neq \mathbf{p}_{i(k)}$ due to image noise. Here we assume $\mathbf{p}_{i(k)}$ is subject to a zero-mean Gaussian noise $\mathcal{N}(\mathbf{0}, \Lambda_\mathbf{p})$.

Define the cost function for $\mathbf{P}_i$ in I_k to be

$$\mathcal{C}_p(\mathbf{P}_i, k) = (\tilde{\hat{\mathbf{p}}}_{i(k)} - \tilde{\mathbf{p}}_{i(k)})^\mathsf{T}\Lambda_\mathbf{p}^{-1}(\tilde{\hat{\mathbf{p}}}_{i(k)} - \tilde{\mathbf{p}}_{i(k)}). \tag{11}$$

2) Ideal Lines & Collinearity: Denote the reprojection of ideal line $\mathbf{L}_i$ in I_k by $\hat{\mathbf{l}}_{i(k)} := (\mathrm{P}_k\mathbf{Q}_i) \times (\mathrm{P}_k\mathbf{J}_i)$. Since the observation of $\mathbf{L}_i$ in I_k, i.e. $\mathbf{l}_{i(k)}$, is estimated from its supporting line segments $\{\mathbf{s}_{\iota,k} | \iota = 1, \cdots\}$, we directly treat these line segments as its observations for cost function definition. The measurement noise of 2D line segments can be modeled in various ways. Here we adopt a simple but well-accepted modeling [54], which assumes each line segment endpoint is subject to a zero-mean Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma_d^2\mathbf{I}_2)$, where σ_d is a scalar, and $\mathbf{I}_2$ is a 2×2 identity matrix.

Define the cost function for $\mathbf{L}_i$ in I_k as

$$\mathcal{C}_l(\mathbf{L}_i, k) = \sum_\iota\sum_{j=1}^{2}\left(\frac{d_\perp(\tilde{\mathbf{d}}_{\iota j,k}, \hat{\mathbf{l}}_{i(k)})}{\sigma_d}\right)^2. \tag{12}$$

This cost function effectively captures the *collinearity* constraint between ideal lines and line segments.

3) Vanishing Points & Parallelism: Let the reprojection of vanishing point $\mathbf{V}_i$ in I_k be $\hat{\mathbf{v}}_{i(k)} := \mathrm{P}_k\mathbf{V}_i$. The observation of $\mathbf{V}_i$ in I_k is $\mathbf{v}_{i(k)}$ which is the intersection of 2D line segments from the same parallel group. Since $\mathbf{v}_{i(k)}$ is estimated from line segments, its estimation covariance $\Lambda_{\mathbf{v}_{i(k)}}$ is easily derived as well [54]. Define the cost function for $\mathbf{V}_i$ in I_k by

$$\mathcal{C}_v(\mathbf{V}_i, k) = (\hat{\mathbf{v}}_{i(k)} - \mathbf{v}_{i(k)})^\mathsf{T}\Lambda_{\mathbf{v}_{i(k)}}^{-1}(\hat{\mathbf{v}}_{i(k)} - \mathbf{v}_{i(k)}). \tag{13}$$

In particular, for all ideal lines $\{\mathbf{L}_j\}$ connected to $\mathbf{V}_i$ in MFG, we enforce $\mathbf{L}_j = [\mathbf{Q}_j^\mathsf{T}, \mathbf{V}_i^\mathsf{T}]^\mathsf{T}$ such that these lines are strictly parallel. Recall that $\mathbf{Q}_j$ is a finite point on $\mathbf{L}_j$. This parameterization and cost function (13) together account for the *parallelism* relationship in MFG.

4) Primary Planes & Coplanarity: Primary plane $\mathbf{\Pi}_i$ has neither reprojection nor direct observation in image space. Therefore, we define its cost function by leveraging 3D key points and ideal lines, respectively. For key point $\mathbf{P}_j$ and primary plane $\mathbf{\Pi}_i$, define

$$\mathcal{C}_\pi(\mathbf{P}_j, \mathbf{\Pi}_i) = \begin{cases} [\delta_\perp(\mathbf{P}_j, \mathbf{\Pi}_i)]^2 & \text{if } \mathbf{P}_j \in \mathbf{\Pi}_i \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

$$\mathcal{C}_{\text{LBA}}(\mathcal{M}_k) = \sum_{\kappa=k-w+1}^{k} \left[\eta_p \sum_{\mathbf{P}\in\mathcal{S}_p^k} \mathcal{K}_{\delta_p}\big(\mathcal{C}_p(\mathbf{P},\kappa)\big) + \eta_l \sum_{\mathbf{L}\in\mathcal{S}_l^k} \mathcal{K}_{\delta_l}\big(\mathcal{C}_l(\mathbf{L},\kappa)\big) + \eta_v \sum_{\mathbf{V}\in\mathcal{S}_v^k} \mathcal{K}_{\delta_v}\big(\mathcal{C}_v(\mathbf{V},\kappa)\big) \right] \tag{10}$$
$$+ \eta_\pi \sum_{\mathbf{\Pi}\in\mathcal{S}_\pi^k} \left[\sum_{\mathbf{P}\in\mathcal{S}_p^k} \mathcal{K}_{\delta_\pi}\big(\mathcal{C}_\pi(\mathbf{P},\mathbf{\Pi})\big) + \sum_{\mathbf{L}\in\mathcal{S}_l^k} \mathcal{K}_{\delta_\pi}\big(\mathcal{C}_\pi(\mathbf{L},\mathbf{\Pi})\big) \right]$$

where $\mathbf{P}_j \in \mathbf{\Pi}_i$ indicates that $\mathbf{P}_j$ is connected with $\mathbf{\Pi}_i$ in MFG.

For ideal line $\mathbf{L}_j$ and primary plane $\mathbf{\Pi}_i$, define

$$\mathcal{C}_\pi(\mathbf{L}_j, \mathbf{\Pi}_i) = \begin{cases} \frac{1}{n}\sum_{\iota=1}^{n}[\delta_\perp(\mathbf{D}_\iota, \mathbf{\Pi}_i)]^2 & \text{if } \mathbf{L}_j \in \mathbf{\Pi}_i \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\{\mathbf{D}_\iota | \iota = 1, \cdots, n\}$ denote the endpoints of all the line segments that support $\mathbf{L}_j$.

Eqs. (14) and (15) represent the *coplanarity* constraint in MFG.

5) Overall Metric: Denote the last m camera poses by $\mathcal{S}_c^k = \{\mathrm{R}_i, \mathrm{t}_i | i = k-m+1, \cdots, k\}$, and the last m key frames by $\mathcal{I}^k = \{I_i | i = k-m+1, \cdots, k\}$. The key points to be refined in LBA are those that are observed in at least one frame of $\mathcal{I}^k$, and we denote them by $\mathcal{S}_p^k$. Similarly we define $\mathcal{S}_l^k$ and $\mathcal{S}_v^k$ for ideal lines and vanishing points, respectively. The primary planes to be refined are those that have edges connected to key points from $\mathcal{S}_p^k$ or ideal lines from $\mathcal{S}_l^k$, and we denote them by $\mathcal{S}_\pi^k$.

The cost function of LBA is defined as a weighted sum of the costs of MFG features/constraints in (10), where η_p, η_l, η_v and η_π are the weights for key points, ideal lines, vanishing points and primary planes, respectively, and $\mathcal{K}_\delta(\cdot)$ is a robust kernel function with parameter δ. Currently the weights are chosen empirically, and the kernel function is the Huber loss defined as

$$\mathcal{K}_\delta(e^2) = \begin{cases} e^2 & \text{for } |e| < \delta, \\ 2\delta e - \delta^2 & \text{otherwise.} \end{cases} \tag{16}$$

The LBA problem at time k is

$$\min_{\mathcal{S}_c^k, \mathcal{S}_p^k, \mathcal{S}_l^k, \mathcal{S}_v^k, \mathcal{S}_\pi^k} \mathcal{C}_{\text{LBA}}(\mathcal{M}_k). \tag{17}$$

This problem is solved using the Levenberg-Marquardt (LM) algorithm [53], and the solution is used to refine camera poses $\mathcal{S}_c^k$ and $\mathcal{M}_k$ nodes including key points $\mathcal{S}_p^k$, ideal lines $\mathcal{S}_l^k$, vanishing points $\mathcal{S}_v^k$ and primary planes $\mathcal{S}_\pi^k$.

B. MFG Pruning

False data association inevitably results in erroneous estimation in the 3D MFG. We thus constantly prune the MFG after performing LBA. We start with key point pruning.

Recall that $\mathbf{p}_{i(k)}$ represents the observation of $\mathbf{P}_i$ in I_k. Here we define a set $\mathcal{S}_{ob}(\mathbf{P}_i) = \{\mathbf{p}_{i(\kappa)}, \kappa \geq 0\}$ to contain all the detected observations for $\mathbf{P}_i$. The key point pruning procedure is summarized in Algorithm 1. The basic idea is that a key point outlier resulted from false matching must have observations inconsistent with its reprojections. After removing inconsistent observations, if a key point has very few surviving observations, then we consider it as mis-estimated.

Algorithm 1: Key Point Pruning

```
Input : $\mathcal{V}$, $\mathcal{M}_k$, $\mathrm{R}_{0:k}, \mathrm{t}_{0:k}$
Output: $\mathcal{M}_k$
for $\mathbf{P}_i \in \mathcal{S}_p^k$ do
    for $\mathbf{p}_{i(\kappa)} \in \mathcal{S}_{ob}(\mathbf{P}_i)$ do
        Compute $e_{i\kappa} = \mathcal{C}_p(\mathbf{P}_i, \kappa)$ using (11);
        if $e_{i\kappa} > \varepsilon$ then
            Remove $\mathbf{p}_{i(\kappa)}$ from $\mathcal{S}_{ob}(\mathbf{P}_i)$;
    if $|\mathcal{S}_{ob}(\mathbf{P}_i)| < N_{ob}$ then
        Remove $\mathbf{P}_i$ and the associated edges from $\mathcal{M}_k$;
return $\mathcal{M}_k$
```

The ideal line pruning procedure is similar to Algorithm 1 except that $e_{i\kappa} = \mathcal{C}_l(\mathbf{L}_i, \kappa)$. Ideal line mis-estimation results not only from false line matching between images but also from wrong association with vanishing points. In the latter case, the true direction of a 3D line is not parallel to the assigned vanishing point. Since we enforce the estimated line direction to be parallel to the vanishing point (see Section V-A3), the line reprojections must have discrepancies with observations. This allows the pruning algorithm to detect this kind of mis-estimation.

At current stage, we do not prune vanishing points and primary planes because 1) they are rarely mis-estimated, and 2) the Huber loss functions in (10) allow the LBA to be robust to such mis-estimations.

VI. Algorithms

The HLVN algorithm is summarized in Algorithm 2 to facilitate our analysis. Let the image resolution of I_k be r pixels. Then detecting 2D key points and line segments can be done in $O(r)$ time [30], [48]. Suppose on average we detect n_p 2D key points, n_s 2D line segments, n_l 2D ideal lines, and n_v 2D vanishing points in each image. Obviously, $r > n_p$, $r > n_s \geq n_l > n_v$. Usually, n_v is very small and can be considered as constant. Primary plane update takes $O(1)$ time because $|\mathcal{C}|$ is bounded by constant (see Section IV-D4). MFG pruning takes $O(n_T(|\mathcal{S}_p^k| + |\mathcal{S}_l^k|))$ time, where n_T denotes the average number of observations for a 3D key point (or ideal line). In general visual navigation, we can bound n_T by a large constant. Thus, MFG pruning has a time complexity $O(w(n_p + n_s))$ because $|\mathcal{S}_p^k| < wn_p$ and $|\mathcal{S}_l^k| < wn_s$.

Algorithm 2: HLVN Algorithm

```
Input  : $\mathcal{V}$, $\mathcal{M}_{k-1}$, $\mathrm{R}_{0:k-1}, \mathrm{t}_{0:k-1}$
Output: $\mathcal{M}_k$, $\mathrm{R}_{0:k}, \mathrm{t}_{0:k}$
Select key frame $I_k$;                                   $O(r+n_p^2)$
Detect 2D key points from $I_k$;                          $O(r)$
Detect 2D line segments from $I_k$;                       $O(r)$
Detect 2D ideal lines for $I_k$;                          $O(n_s^2)$
Detect vanishing points for $I_k$;                        $O(n_s^2)$
Match key points between $I_{k-1}$ and $I_k$;             $O(n_p^2)$
Compute epipolar geometry;                                $O(n_p)$
Match vanishing points between $I_{k-1}$ and $I_k$;       $O(1)$
Match ideal lines between $I_{k-1}$ and $I_k$;            $O(n_s^2)$
Estimate camera pose $\mathrm{R}_k, \mathrm{t}_k$;        $O(n_p^2)$
$\mathcal{M}_k = \mathcal{M}_{k-1}$;                      $O(1)$
Update 3D key points;                                     $O(n_p)$
Update 3D ideal lines;                                    $O(n_s)$
Update 3D vanishing points;                               $O(1)$
Update 3D primary planes;                                 $O(1)$
Perform LBA on $\mathcal{M}_k$;                           $O(w^3(n_p+n_s)^3/\epsilon^2)$
Prune $\mathcal{M}_k$;                                    $O(w(n_p+n_s))$
return $\mathcal{M}_k$, $\mathrm{R}_{0:k}, \mathrm{t}_{0:k}$
```

The most computationally-expensive step is LBA, which refines a parameter vector of dimension

$$d_p = 3|\mathcal{S}_p^k| + 6|\mathcal{S}_l^k| + 3|\mathcal{S}_v^k| + 4|\mathcal{S}_\pi^k| + 7|\mathcal{S}_c^k|.$$

This is because we use a 3-vector for a 3D key point, a 6-vector for a 3D ideal line, a 3-vector for a vanishing point, a 4-vector for a primary plane and a 7-vector (unit quaternion for rotation) for a camera pose. Since $|\mathcal{S}_v^k| < wn_v$, $|\mathcal{S}_\pi^k| < |\mathcal{S}_p^k|$ and $|\mathcal{S}_c^k| < w$, we have $d_p = O(w(n_p + n_s))$. Similarly, the observation vector's dimension is $d_o = O(w(n_p+n_s))$. In each iteration of LM, the computational complexity is $O(w^3(n_p + n_s)^3)$ for a dense matrix solver. According to [55], the total iterations needed by LM is upper-bounded by $O(1/\epsilon^2)$, with a stopping criterion $\|\nabla \mathcal{C}_{\text{LBA}}\| \le \epsilon$.

Theorem 1: The computational complexity of the HLVN algorithm is $O(r + w^3(n_p + n_s)^3/\epsilon^2)$.

VII. Experiments

We have implemented our algorithm using C++. We first validate the proposed two stage line matching (TSLM) approach on real image data. Then we evaluate the visual odometry performance of HLVN under both indoor and outdoor scenarios and compare it with state-of-the-art algorithms.

A. Line Matching Test

Ideal lines play a pivotal role in MFG. A good matching algorithm for ideal lines should find as many matches as possible while maintaining high accuracy. Here we validate our TSLM approach by comparing it with PCLM, the state-of-the-art in line matching.

As illustrated in Fig. 8, we have collected 20 pairs of images covering a variety of man-made scenes (available on line [56]). Due to the wide baselines in the image data, SIFT matches are utilized as point correspondences for both line matching methods. Table I shows the number of total matches (TM) and the true positive rate (TPR) obtained by TSLM and PCLM, respectively. On average, TSLM is able to find 62.5% more line matches than PCLM while achieving a TPR of 92.3%. The significant increase in line match number can greatly benefit the MFG construction process. The slight decrease in TPR is not a big problem because false matches are handled by other procedures such as MFG pruning.

Fig. 8: Sample images used for line matching test.

TABLE I: Ideal Line Matching Results

No.	PCLM		TSLM	
	#TM	TPR	#TM	TPR
1	48	97.9%	93	97.8%
2	84	96.4%	125	88.8 %
3	54	92.6%	73	94.5%
4	51	86.2%	62	87.1%
5	83	90.4%	125	89.6%
6	58	96.5%	87	94.3%
7	62	93.5%	82	90.2%
8	74	89.2%	110	90.9 %
9	50	94.0%	84	96.4%
10	62	98.3%	86	97.7%
11	29	96.5%	51	94.1%
12	55	98.1%	76	97.4%
13	42	95.2%	87	96.6%
14	79	97.4%	104	95.2%
15	22	86.3%	80	76.3%
16	35	97.1%	74	87.8%
17	29	100%	56	91.1%
18	13	84.6%	28	82.1%
19	25	100%	68	97.1%
20	6	100%	19	89.5%
Avg.	48	93.7%	78	92.3%

B. Visual Odometry Test

We now focus on the visual odometry performance of our HLVN algorithm. For comparison, we have chosen the following two state-of-the-art algorithms in feature-based monocular visual odometry/SLAM.

- PTAM: Parallel Tracking and Mapping [28], one of the most successful BA based visual SLAM algorithms, and
- 1-Point-EKF: 1-Point RANSAC-based EKF-SLAM [14], a representative sequential filtering based visual odometry method.

Both algorithms above provide open-source code, allowing more fair comparisons. Although system parameter settings largely depend on real applications, we list the relevant parameter values used in our experiments in Table II.

TABLE II: PARAMETER SETTINGS

Parameter	Value	Description
N_2	50	2D point match number
N_3	7	3D key point number
τ_R	15°	rotation angle
τ_θ	10°	vanishing point angle
τ_ρ	0.9°	parallax threshold
N_{cp}	100	coplanar feature number
w	10	LBA window size
m	8	LBA pose number
η_p	1	key point weight
η_l	1	ideal line weight
η_v	15	vanishing point weight
η_π	100	primary plane weight
δ_p	1	Huber kernel size
δ_l	3	Huber kernel size
δ_v	1	Huber kernel size
δ_π	1	Huber kernel size
ε	4	reprojection error
N_{ob}	2	observation number

TABLE III: ABSOLUTE TRAJECTORY ERRORS

(a) BICOCCA

Method	RMSE (m)	SD (m)	Max (m)	RMSE/trajectory
1-Point-EKF	2.23	1.10	5.20	2.90%
PTAM	0.93	0.35	1.97	1.21%
HLVN	0.88	0.28	1.86	1.14%

(b) HRBB4

Method	RMSE (m)	SD (m)	Max (m)	RMSE/trajectory
1-Point-EKF	1.99	1.17	6.94	2.84%
PTAM	1.61	0.87	4.65	2.30%
HLVN	0.85	0.41	2.45	1.21%

(c) MALAGA6

Method	RMSE (m)	SD (m)	Max (m)	RMSE/trajectory
1-Point-EKF	77.16	44.68	175.98	6.43%
PTAM	—	—	—	—
HLVN	14.10	6.23	45.03	1.18%

*1) **Evaluation Metric:*** To evaluate the localization accuracy, we adopt the widely used absolute trajectory error (ATE) [14] as explained below.

Since the ground truth and the estimation of camera poses are usually represented in different coordinate systems, we need to align them before computing ATE. Let $\mathbf{g}_k^{W'}$ be the ground truth of camera position at time k in a coordinate system $\{W'\}$ and $\mathbf{r}_k^W$ the estimated one in $\{W\}$. We need to find a similarity transformation that maps $\mathbf{r}_k^W$ to $\{W'\}$:

$$\mathbf{r}_k^{W'} := s\mathbf{R}_W^{W'}\mathbf{r}_k^W + \mathbf{t}_W^{W'}, \tag{18}$$

where the transformation is defined by rotation matrix $\mathbf{R}_W^{W'}$, translation vector $\mathbf{t}_W^{W'}$ and scaling factor s. The similarity transformation is obtained by minimizing $\sum_k \|\mathbf{r}_k^{W'} - \mathbf{g}_k^{W'}\|^2$. The ATE ε_k at time k is then defined as: $\varepsilon_k = \|\mathbf{r}_k^{W'} - \mathbf{g}_k^{W'}\|$. We compute the root-mean-square-error (RMSE) of ATE's over all the time indexes, as well as the standard deviation (SD), maximum (Max), and the ratio between RMSE and the trajectory length for comparison purpose.

*2) **Datasets:*** We evaluate the aforementioned three methods on two indoor and one outdoor sequence, as described below.

- Bicocca sequence: a subsequence of over 4,500 images from the Bicocca-2009-02-25b session of the Rawseeds datasets [57]. The images are recorded in a library using a camera with 79° HFOV at a resolution of 320×240. The trajectory has an approximate length of 77 m, with ground truth provided.
- HRBB4 sequence: a sequence of 12,000 images collected in an office corridor environment by ourselves (available on line [56]). Fig. 9(b) shows the sample images, which are recorded using a camera (with 65° HFOV) mounted on a PackBot (see Fig. 9(a)). In our experiment, we reduce the image resolution from 1920×1080 to 640×360 for faster computation. The robot trajectory has an approximate length of 70 m as illustrated in Fig. 9(c). The ground truth of camera poses is obtained by using artificial landmarks. As shown in Fig. 9(b), a set of markers are posted along the lower parts of walls. The 3D positions of the 4 outmost corners of each marker are manually measured, and their projections in images are manually selected. Based on these 3D-2D point correspondences, the camera pose of each frame is recovered using a PnP (perspective-n-point) solver (e.g. [58]). According to our test, this method achieves an accuracy of ±1 cm in camera position, owing to the following facts.
  - The marker locations are designed such that the camera clearly sees at least 3 or 4 markers most of the time.
  - The marker corners' 3D positions are carefully measured using a BOSCH GLR225 laser distance measurer with a range up to 70 m and measurement accuracy of ±1.5 mm.
  - High resolution images (i.e. 1920×1080) are used to suppress image noise when finding the projections of marker corners.

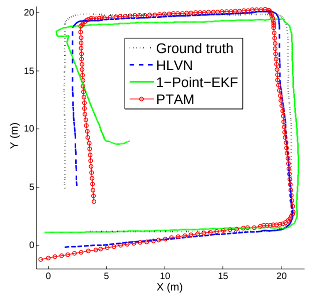
  Due to their minimal sizes in images, these markers do not bring much influence to the approaches in our test.
- Malaga6 sequence: a sequence of over 4,600 urban images from the 6th section of the Málaga Stereo and Laser Urban Data Set [59]. Although the original dataset provides stereo images, we only use the left channel at a resolution of 800×600. The trajectory length is over 1,200 m with GPS data available.

*3) **ATE:*** We have fine-tuned parameters for PTAM and 1-Point-EKF with best efforts. The results below represent their best performance in our test.

Table III(a) shows the ATE's on Bicocca for each method. Both PTAM and HLVN outperform 1-Point-EKF, which manifests the superiority of BA approaches over filtering. On the other hand, the difference between PTAM and HLVN is almost negligible. This is because Bicocca is recorded in an environment with rich texture (on average, 478 corner points detected per image), and the camera HFOV is relatively wide. This favors key point-based approaches like PTAM, whereas not allowing HLVN to demonstrate much advantage.

(a) PackBot

(b) Sample images



(c) Trajectories

Fig. 9: HRBB4 sequence. (a) Camera and robot, (b) Sample images, (c) Trajectory estimates aligned with the ground truth using similarity transforms as described in (18).

TABLE IV: ABSOLUTE TRAJECTORY ERRORS (M) OF HLVN VARIANTS

Variant	Bicocca	HRBB4	Malaga6
PT	1.04	2.05	39.68
PT+LN	0.98	1.42	25.24
PT+VP	1.01	1.56	22.08
PT+PL	0.96	1.62	25.71
PT+LN+VP	0.95	1.33	17.85
PT+LN+PL	0.91	1.17	19.37
PT+VP+PL	0.92	1.26	19.52
HLVN	0.88	0.85	14.10

The ATE's in the Malaga6 column are larger than other columns because Malaga6 is an outdoor sequence with a much longer trajectory (1,200 m).

In contrast HRBB4 is a much more challenging dataset. Despite its larger image resolution than Bicocca, HRBB4 only detects 355 corner points per image on average due to the textureless scene. The robot also makes sharp turns (i.e., small translation along with large rotation) in HRBB4, which easily leads to large scale drift. The narrower HFOV in HRBB4 further increases the difficulty. Nonetheless, Table III(b) shows that HLVN outperforms both competing approaches with an RMSE of 0.85 m, 1.21% of the overall trajectory length. Specifically, the RMSE of HLVN is 47.2% less than that of PTAM. As shown in Fig. 9(c), HLVN suffers less scale drift at sharp turns than PTAM and 1-Point-EKF by leveraging more types of features.

Designed for small-scale indoor use, PTAM is not quite applicable to Malaga6. In fact, we are not even able to have PTAM completely process Malaga6 because of tracking failure. In Table III(c), HLVN outperforms 1-Point-EKF again by exploiting heterogeneous features and LBA. This demonstrates the benefit of HLVN for visual navigation in urban environments.

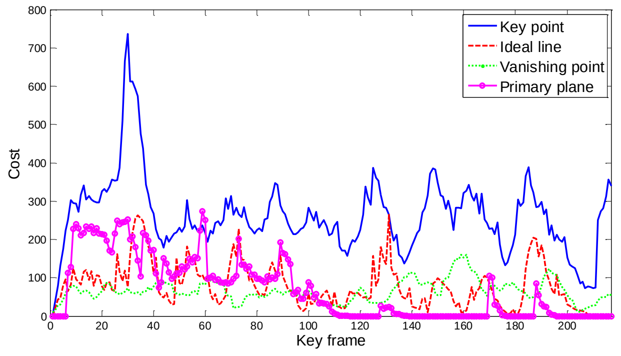
4) ***Feature Contributions****:* Fig. 10 illustrates the values of each component in (10) over key frames of the Bicocca and HRBB4 sequences, respectively. We see how much each type of feature contributes to the LBA problem dynamically. As a result of the rich texture in Bicocca, the contribution of key points dominates all other costs throughout the sequence. On the other hand, the contributions of key points and ideal lines are mostly comparable to each other in HRBB4. This phenomenon shows that HLVN is adaptive in the sense that the contribution of each type of feature varies as the scene structure changes.

For a further insight of feature contributions, we investigate the performance of our system under different combinations of feature types using the following variants of HLVN:

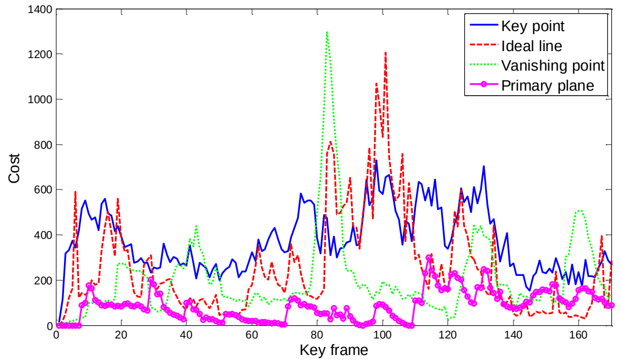
- PT: using only key points,
- PT+LN: using key points and ideal lines,
- PT+VP: using key points and vanishing points,
- PT+PL: using key points and primary planes,
- PT+LN+VP: using key points, ideal lines and vanishing points,
- PT+LN+PL: using key points, ideal lines and primary planes,
- PT+VP+PL: using key points, vanishing points and primary planes.

Table IV shows the ATE's resulted from these HLVN variants on the three datasets. We find that the inclusion of more feature types helps reduce the ATE's, though the improvement may vary in different scenarios. For example, the error reduction brought by more feature types is less significant on Bicocca than on HRBB4 or Malaga6; this conforms to the observation in Fig. 10(a) that key points dominate the overall cost of LBA throughout Bicocca. Unfortunately it is hard to judge the relative importance between individual feature types in general since it is essentially a scene-dependent problem. Nonetheless, the result implies that exploiting more features types and geometric constraints, whenever available, effectively reduces the overall estimation error. This justifies our choice of fusing heterogeneous landmarks for visual navigation in man-made environments, despite higher computational demands.

5) ***Time Consumption****:* The LBA of HLVN is implemented using g^{2}o (general graph optimization) [60], which allows to leverage sparse optimization solvers. Our current implementation of HLVN is single-threaded and not yet optimized. The computation time on a desktop with an Intel Core i7-3770 CPU is reported in Table V. Although relatively slow, HLVN can be accelerated for real time use in at least 3 ways, i.e., optimizing the implementation, using graphics processing units or more powerful CPUs, and parallelizing the algorithm. We expect that the algorithm can run in real time in the near future.



(a) Bicocca



(b) HRBB4

Fig. 10: Contributions to LBA costs by different features.

TABLE V: Run Time of HLVN

Sequence	Duration	Run time	n_p	n_s	#Key frame
Bicocca	153 s	290 s	478	192	218
HRBB4	400 s	210 s	355	250	170
Malaga6	231 s	600 s	518	413	267

As defined in Section VI, n_p and n_s are the average numbers of 2D key points and line segments detected from each image, respectively. Duration means the length of the video duration, and run time means computation time.

C. Test on KITTI Odometry Dataset

The KITTI odometry dataset [61] contains 22 image sequences, 11 of which (i.e. sequences 00-10) are provided with ground truth and thus used for our test. For general autonomous driving testing, this dataset covers various scenarios including urban, countryside and highway roads. In our experiment, however, only sequences 00 and 07 are of particular interest because they have rectilinear buildings dominating the scene, which conforms to our assumption **a.1**. Due to the lack of feature heterogeneity on other sequences, our method is not expected to outperform other approaches. For comparison, we choose the following point-based methods

- VISO2-M: the monocular visual odometry algorithm associated with the dataset [62], and
- SCG: a state-of-the-art large-scale monocular system proposed by Song, Chandraker and Guest [63], referred to as SCG here. SCG is a top-ranked monocular algorithm on the KITTI odometry benchmark.

TABLE VI: Comparison on KITTI Dataset

Seq	VISO2-M		SCG		Ours	
	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)	Rot (deg/m)	Trans (%)
00	0.0369	12.62	0.0142	7.14	0.0151	4.39
02	0.0194	3.71	0.0097	4.34	0.0122	5.60
03	0.0288	9.05	0.0093	2.90	0.0122	3.71
04	0.0163	7.58	0.0064	2.45	0.0088	2.74
05	0.0575	12.74	0.0107	8.13	0.0188	4.93
06	0.0275	3.71	0.0108	7.56	0.0181	4.09
07	0.1235	25.77	0.0234	9.92	0.0199	4.71
08	0.0369	16.88	0.0122	7.29	0.0171	6.69
09	0.0227	3.94	0.0096	5.14	0.0231	5.27
10	0.0596	29.36	0.0121	4.99	0.0119	4.43

Highlighted rows indicate urban sequences with rectilinear buildings dominating the scene, which allow our method to stand out by design.

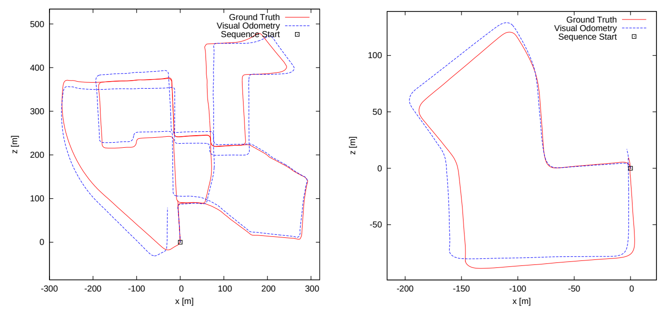
The evaluation metric provided by the dataset (see [61] for detail) requires estimated trajectories to be in real-world scale. Therefore, we have augmented our system with a ground plane detection component to remove scale ambiguity by assuming a fixed camera height. Similar to [62], [63], in each iteration our algorithm finds point correspondences within a pre-defined image region between key frames, reconstructs 3D points by triangulation, and detects ground plane from these points using RANSAC.

As highlighted in Table VI, on sequences 00 and 07 our method achieves clearly smaller translational errors than SCG, and rotational errors of the same level. To be specific, our method reduces the respective translational errors on sequences 00 and 07 by at least 52.5% as claimed in Abstract. As expected, our method outperforms the counterparts on these two sequences by exploiting heterogeneous landmarks and their geometric relationships. Fig. 11 illustrates the estimated trajectories for sequences 00 and 07 by our method. Meanwhile, on other sequences our method yields comparable translational errors with SCG, despite slightly increased rotational errors. In fact, our translational errors on sequences 05 and 06 are also substantially lower than SCG thanks to the sporadic presence of rectilinear structures in the imagery. Sequence 01 is not included in Table VI because its fast speed (up to 90 km/h) fails the ground plane detection for all three methods.

To summarize, our method significantly outperforms the counterpart in urban scenarios (e.g. sequences 00 and 07) where rectilinear buildings dominate the scene. Importantly, this is exactly the scenario where visual SLAM is of great importance due to the fact that GPS signals are often blocked or reflected by tall buildings.

VIII. Conclusions and Future Work

We presented a method utilizing heterogeneous visual features and their inner geometric constraints to assist robot navigation in man-made environments. This was managed by a multilayer feature graph. Our method extended the LBA framework by explicitly exploiting heterogeneous features and their geometric relationships in an unsupervised manner. The algorithm took a video stream as input, initialized and iteratively updated MFG based on extracted key frames, and



(a) Sequence 00 (b) Sequence 07

Fig. 11: Estimated trajectories by our method for sequences 00 and 07 in the KITTI dataset.

refined robot localization and MFG landmarks. We presented the algorithm pseudo code and analyzed its computation complexity. Physical experiments showed that our algorithm outperformed state-of-the-art approaches on datasets where rectilinear structures dominate the scene.

In the future, we will use MFG to facilitate loop closing detection, and consider incorporating appearance information of planes in MFG to enhance its robustness. Using additional sensors such as inertial measurement units to assist MFG construction is another possible direction. We will also consider exploring more geometric constraints from environments such as orthogonality.

Acknowledgment

We would like to acknowledge the insightful discussions from Y. Xu, A. Perera, and S. Oh in Kitware. We also thank H. Li and J. Liu for their early implementation on two-view based MFG, and thank W. Li, M. Hielsberg, J. Lee, C. Chou, H. Cheng, M. Jiang, X. Wang, M. Treat, and R. Liu for their input and contributions to the Networked Robots Lab at Texas A&M University.

References

[1] C. Frueh, S. Jain, and A. Zakhor, "Data processing algorithms for generating textured 3D building facade meshes from laser scans and camera images," *International Journal of Computer Vision*, vol. 61, no. 2, pp. 159–184, 2005.

[2] L. Zebedin, J. Bauer, K. Karner, and H. Bischof, "Fusion of feature-and area-based information for urban buildings modeling from aerial imagery," in *Computer Vision–ECCV 2008*, pp. 873–886, Springer, 2008.

[3] N. Cornelis, B. Leibe, K. Cornelis, and L. Van Gool, "3d urban scene modeling integrating recognition and reconstruction," *International Journal of Computer Vision*, vol. 78, no. 2-3, pp. 121–141, 2008.

[4] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 519–528, IEEE, 2006.

[5] G. Vogiatzis, P. H. Torr, and R. Cipolla, "Multi-view stereo via volumetric graph-cuts," in *Computer Vision and Pattern Recognition, 2005. IEEE Computer Society Conference on*, vol. 2, pp. 391–398, IEEE, 2005.

[6] H. Jin, S. Soatto, and A. J. Yezzi, "Multi-view stereo reconstruction of dense shape and complex appearance," *International Journal of Computer Vision*, vol. 63, no. 3, pp. 175–189, 2005.

[7] H. Bay, V. Ferraris, and L. Van Gool, "Wide-baseline stereo matching with line segments," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 329–336, IEEE, 2005.

[8] T. Yu, N. Xu, and N. Ahuja, "Shape and view independent reflectance map from multiple views," *International journal of computer vision*, vol. 73, no. 2, pp. 123–138, 2007.

[9] D. Gallup, J.-M. Frahm, and M. Pollefeys, "Piecewise planar and non-planar stereo for urban scene reconstruction," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 1418–1425, IEEE, 2010.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[11] J. A. Castellanos, R. Martinez-Cantin, J. D. Tardós, and J. Neira, "Robocentric map joining: Improving the consistency of EKF-SLAM," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 21–29, 2007.

[12] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1, pp. 206–211, IEEE, 2003.

[13] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Generic and real-time structure from motion using local bundle adjustment," *Image and Vision Computing*, vol. 27, pp. 1178–1193, 2009.

[14] J. Civera, O. G. Grasa, A. J. Davison, and J. Montiel, "1-point RANSAC for extended Kalman filtering: Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, no. 5, pp. 609–631, 2010.

[15] A. Majdik, D. Gálvez-López, G. Lazea, and J. A. Castellanos, "Adaptive appearance based loop-closing in heterogeneous environments," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1256–1263, IEEE, 2011.

[16] N. Carlevaris-Bianco and R. M. Eustice, "Learning visual feature descriptors for dynamic lighting conditions," in *Intelligent Robots and Systems (IROS), 2014 IEEE/RSJ International Conference on*.

[17] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1403 –1410 vol.2, oct. 2003.

[18] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2657–2664, IEEE, 2010.

[19] E. Royer, M. Lhuillier, M. Dhome, and J.-M. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.

[20] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-DOF SLAM with stereo-in-hand," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 946–957, 2008.

[21] K. Konolige and M. Agrawal, "Frameslam: From bundle adjustment to real-time visual mapping," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 1066–1077, 2008.

[22] G. Sibley, C. Mei, I. Reid, and P. Newman, "Vast-scale outdoor navigation using adaptive relative bundle adjustment," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 958–980, 2010.

[23] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 2531–2538, IEEE, 2008.

[24] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research*, 2009.

[25] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics, Special Issue on Calibration for Field Robotics*, 2014. In Press.

[26] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2012.

[27] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-D mapping with an RGB-D camera," *Transactions on Robotics*, 2014. In Press.

[28] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pp. 225–234, IEEE, 2007.

[29] S. Frintrop and P. Jensfelt, "Attentional landmarks and active gaze control for visual SLAM," in *IEEE Transactions on Robotics, special Issue on Visual SLAM*, vol. 24, Oct. 2008.

[30] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pp. 593–600, IEEE, 1994.

[31] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, Nov. 2004.

[32] T. Lemaire and S. Lacroix, “Monocular-vision based SLAM using line segments,” in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2791 –2796, april 2007.

[33] E. Eade and T. Drummond, “Edge landmarks in monocular SLAM,” *Image and Vision Computing*, vol. 27, no. 5, pp. 588 – 596, 2009.

[34] J. Sola, T. Vidal-Calleja, and M. Devy, “Undelayed initialization of line segments in monocular SLAM,” in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pp. 1553–1558, 2009.

[35] J. Zhang and D. Song, “Error aware monocular visual odometry using vertical line pairs for small robots in urban areas,” in *Special Track on Physically Grounded AI (PGAI), AAAI Conference on Artificial Intelligence (AAAI)*, (Atlanta, Georgia, USA), July 2010.

[36] P. Smith, I. Reid, and A. Davison, “Real-time monocular SLAM with straight lines,” in *Proc. British Machine Vision Conference*, (Edinburgh), 2006.

[37] G. Zhang, D. H. Kang, and I. H. Suh, “Loop closure through vanishing points in a line-based monocular SLAM,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 4565–4570, IEEE, 2012.

[38] A. Gee, D. Chekhlov, A. Calway, and W. Mayol-Cuevas, “Discovering higher level structure in visual SLAM,” *Robotics, IEEE Transactions on*, vol. 24, pp. 980 –990, oct. 2008.

[39] J. Martinez-Carranza and A. Calway, “Unifying planar and point mapping in monocular SLAM,” in *Proceedings of the British Machine Vision Conference*, pp. 43.1–43.11, BMVA Press, 2010.

[40] A. Flint, C. Mei, I. Reid, and D. Murray, “Growing semantically meaningful models for visual SLAM,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 467–474, IEEE, 2010.

[41] R. F. Salas-Moreno, B. Glocker, P. H. Kelly, and A. J. Davison, “Dense planar slam,” in *Mixed and Augmented Reality (ISMAR), 2014. IEEE and ACM International Symposium on*.

[42] E. Tretyak, O. Barinova, P. Kohli, and V. Lempitsky, “Geometric image parsing in man-made environments,” *International Journal of Computer Vision*, vol. 97, no. 3, pp. 305–321, 2012.

[43] H. Li, D. Song, Y. Lu, and J. Liu, “A two-view based multilayer feature graph for robot navigation,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3580–3587, IEEE, May 2012.

[44] Y. Lu, D. Song, Y. Xu, A. G. A. Perera, and S. Oh, “Automatic building exterior mapping using multilayer feature graphs,” in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, pp. 162–167, 2013.

[45] Y. Lu, D. Song, and J. Yi, “High level landmark-based visual navigation using unsupervised geometric constraints in local bundle adjustment,” in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, p. to appear, IEEE, 2014.

[46] J.-Y. Bouguet, “Pyramidal implementation of the Affine Lucas Kanade feature tracker: Description of the algorithm,” *Intel Corporation*, vol. 5, 2001.

[47] D. Nistér, “An efficient solution to the five-point relative pose problem,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[48] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 722 –732, april 2010.

[49] B. Fan, F. Wu, and Z. Hu, “Line matching leveraged by point correspondences,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 390 –397, june 2010.

[50] C. Schmid and A. Zisserman, “Automatic line matching across views,” in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, (San Juan, Puerto Rico), p. 666, Published by the IEEE Computer Society, June 1997.

[51] R. Eustice, O. Pizarro, and H. Singh, “Visually augmented navigation in an unstructured environment using a delayed state history,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 1, pp. 25–32, IEEE.

[52] V. Indelman, R. Roberts, C. Beall, and F. Dellaert, “Incremental light bundle adjustment.,” in *BMVC*, pp. 1–11, 2012.

[53] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge Univ Pr, 2003.

[54] Y. Xu, S. Oh, and A. Hoogs, “A minimum error vanishing point detection approach for uncalibrated monocular images of man-made environments,” in *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*, pp. 1376–1383, 2013.

[55] K. Ueda and N. Yamashita, “On a global complexity bound of the Levenberg-Marquardt method,” *Journal of optimization theory and applications*, vol. 147, no. 3, pp. 443–453, 2010.

[56] Y. Lu and D. Song, “Multilayer Feature Graph for Visual Navigation.” http://telerobot.cs.tamu.edu/MFG, 2014.

[57] A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos, “Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets,” in *In proceedings of IROS’06 Workshop on Benchmarks in Robotics Research*, 2006.

[58] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate O(n) solution to the PnP problem,” *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009.

[59] J.-L. Blanco-Claraco, F.-Á. Moreno-Dueñas, and J. González-Jiménez, “The Málaga urban dataset: High-rate stereo and LiDAR in a realistic urban scenario,” *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 207–214, 2014.

[60] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g 2 o: A general framework for graph optimization,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3607–3613, IEEE, 2011.

[61] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the KITTI vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[62] A. Geiger, J. Ziegler, and C. Stiller, “Stereoscan: Dense 3d reconstruction in real-time,” in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 963–968, IEEE, 2011.

[63] S. Song, M. Chandraker, and C. C. Guest, “Parallel, real-time monocular visual odometry,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 4698–4705, IEEE, 2013.