2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Using Vanishing Points to Improve Visual-Inertial Odometry

Federico Camposeco and Marc Pollefeys[1]

Abstract— **This work presents a method for increasing the accuracy of standard visual inertial odometry (VIO) by effectively removing the angular drift that naturally occurs in feature-based VIO. In order to eliminate such drift, we propose to leverage the predominance of parallel lines in man-made environments by using the intersection of their image projections, known as vanishing points (VPs). First, an efficient inertial-based method is presented that accurately and efficiently detects such points. Second, a strategy to deal with these measurements within the framework of an EKF-based VIO system is presented. Furthermore, special care is taken in order to ensure the real-time execution of the estimator in order to comply with time-critical applications running on computationally constrained platforms. Experiments are performed in a mobile device on challenging environments and evaluated against the same VIO system without the use of VPs, demonstrating the superior accuracy when employing the proposed framework.**

I. Introduction

Having accurate and low-latency estimates of a mobile robot's pose is fundamental for the success of autonomous missions. In order to achieve this, several solutions have been proposed in the literature. In particular, recent advances in visual-inertial egomotion estimation, as well as the low cost and high computational power of consumer-grade mobile electronics, make Camera-IMU systems a very attractive candidate to tackle the localization problem [1], [2], [3].

However, standard VIO techniques rely on the use of point-features as visual updates to an EFK framework, while using the measurements from the IMU as an input to the system used to propagate the filter's state. These type of methods have been shown in [4], [5] and [3] to have unobservable degrees of freedom in the global x, y, z position of the Camera-IMU system and its rotation around the direction of gravity (yaw). Under several motion profiles this will lead to a loss in accuracy for yaw [6]. This may lead to accumulation of errors along these unobservable states which will tend to grow linearly with time [7].

The dominant strategy to deal with drift inherent in VIO is to perform *loop closure* whenever the same place is visited twice. This method has some important disadvantages with respect to our approach: it requires the trajectory to actually close, computational resources need to be allocated to back-propagate the refined trajectory whenever a loop closure is detected and it's memory consuming. Although loop closure and VP detection incur in additional computational costs, VP detection provides to the system line segments as a needed step of the detection. These can potentially be used as measurements for the VIO.

By recognizing the presence of predominant vanishing points in man-made scenarios, the estimate of the VIO can be improved by providing a global measurement of the rotation around gravity. In this work, a real-time method for making use of this information is proposed.

Herein we present a novel approach of integrating information provided by the Manhattan World assumption in the form of vanishing points to a VIO system. For this purpose, we propose a full system contributing:

- Efficient and robust line-segment classification using an inertial-aided RANSAC, along with a least square solution for the computation of a refined VP using such classified segments.
- An extension to a VIO system to effectively include the detected VPs using a delayed update for real-time execution. This effectively allows many underpowered systems to calculate the VP on the background and update the estimate of the filter whenever the VP becomes available.

The rest of the paper is organized as follows. Relevant work in the matter is given in Section II, and providing an overview to the VIO system used in Section IV. The method proposed includes VP detection and its integration to a VIO system, described in Sections III and V respectively. Details on the experiments conducted and results obtained are in Section VI. Finally, closing remarks are provided in Section VII.

II. Related Work

A. Visual Inertial Fusion

Aiming to improve computer vision techniques of relative pose estimation, many approaches have been proposed to use relative orientation information from an IMU as a prior to the pose problem, such as [8] and [9].

Because of the complementary nature of visual and inertial systems [10], other approaches focus on *fusing* visual and inertial data in an optimal way. These are usually classified into two categories, *tightly* and *loosely*-coupled visual-inertial fusion. In loosely-coupled systems, such as [11], the measurements from the IMU are used as an auxiliary and independent input to a stereo pose estimation framework. Weiss et. al. in [12] do the opposite and use a vision-only keyframe-based visual SLAM pose estimate to update an EKF that uses IMU readings as input for propagation.

In contrast, tightly-coupled visual inertial fusion jointly estimates the state of the IMU and Camera, taking into

[1]The authors are with the Computer Vision and Geometry Group, Institute for Visual Computing, Computer Science Department, ETH Zürich {fede, marc.pollefeys}@inf.ethz.ch

account the correlations between the internal states of both sensors. For instance, Kelly et al. [1] as well as Lynen et al. [13] propose a tightly coupled method that estimates not only the kinematic state of the system, but the relative pose of the camera and the IMU, as well as the biases of the accelerometer and gyroscope. Also, Mourikis et al. [2] presented a multi-state Kalman filtering method (MSCKF) that keeps a constant complexity and uses a sliding window of past poses.

Related to our approach [7] proposed to use observations of lines with a known direction along with point feature measurements to remove the unobservability around yaw. In their method they directly observe image lines and assign them to a world direction using a Mahalanobis check, for which they need to initially align the system's yaw with one of the building's VPs. In contrast to this method, ours directly observes VPs and tracks them by keeping them as system states. This allows us to use multiple vanishing directions at the same time (see Fig. 2c and the accompanying video) as well as deal with cases in which vanishing directions are not observed for extended periods of time (in the case of high drift, a new VP could be initialized instead). Additionally our approach emphasizes realtime operation by proposing a delayed update as well as a 1-line RANSAC VP detection which cheaply discards outlier lines.

B. Vanishing Point Detection

Several methods have been proposed for vanishing point detection. Recently, an algebraic method Mirzaei et al. [14] was proposed to accurately and robustly estimate a triad of vanishing points from a single image. However, this approach assumes that all three vanishing directions are visible on the image, while this is not necessarily the case in many real-world scenarios.

Bazin et al. [15] propose two methods for VP detection. A 3-line RANSAC [16] approach, which also assumes all three vanishing directions visible, and a 1-line RANSAC that instead assumes a known vertical. In order to detect the vanishing direction we also leverage the direction of gravity similar to [15]. However, we do not assume to be able to obtain the triplet of vanishing points by doing cross product between the vertical (g_C) and one VP (which assumes a perfect estimate of the vertical). Instead we treat the observation of a vanishing point independently and use the direction of gravity as a queue. Because of this, we do not observe the full rotation from the vanishing directions (as in [14] or in [17]), but simply keep track of the VP direction as a state of the filter and use its observation to update our estimates of the orientation of the system and the directions of the vanishing points, similar to [18].

III. Inertial-Aided Vanishing Point Detection

The steps followed to provide the system with the VP estimates are as follows:

(a) Detect line segments, then use the camera intrinsic calibration to normalize the segments' end and start points. This gives two camera-centered rays for each segment.

(b) Calculate the vanishing line (VL) using the current estimate of the direction of the vertical (see Fig. 1). Notice that we only use the current estimate of the vertical to efficiently find the VP, however for the final estimate of the VP the current state estimate is not used (see III-C). A VL is defined as the intersection of parallel plane (in this case, parallel also to the ground plane) and the plane at infinity [19].

(c) Classify the segments into two sets using RANSAC. For the highest voted VP, the first set, $\mathbf{S}_{\parallel}$, contains all the segments that point in the direction of the VP. The second set, $\mathbf{S}_{\perp}$, contains the segments that intersect the VL at a point orthogonal to the VP.

(d) Refine the estimates of the VP given by RANSAC using both sets of segments.

Because of the computational cost of these steps (specially the line detection step), the system is adapted to handle the computation of VPs as a background process in order to maintain an uninterrupted execution of the pose estimator (see Section V-E).

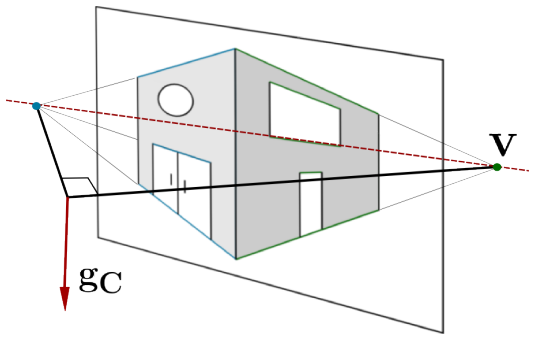


Fig. 1: Schematic of the VP detection method. Segments are classified into $\mathbf{S}_{\parallel}$ and $\mathbf{S}_{\top}$, shown as green and blue segments. Using the direction of gravity, we can trace the VL, shown as the red dotted line.

A. Line detection

We use a simplified version of the Line Segment Detector (LSD) proposed by [20] to first detect line segment in the image. The LSD algorithm takes a gray-scale image as an input and returns a set of detected line segments.

First, the image's gradient is computed. Starting seeds at the pixels with high gradient magnitude, the algorithm then does region growing of gradient-aligned pixels neighboring the seed pixel. Each region, called line-support region, is used to calculate a segment by fitting rectangle. However, the original LSD has subsequent steps to minimize the number of false detections. To decrease the computational burden from the line detection, we decided to remove the refinement and false positive detection and replace it by a simple line segment size threshold.

B. 1-Line RANSAC for segment classification

Given that in a Camera-IMU system the direction of the gravity in the camera frame is known (provided the extrinsic calibration between camera and IMU is known), we use this fact to our advantage in order to find VPs in an efficient manner. In order to classify the segments into $\mathbf{S}_{\parallel}$ and $\mathbf{S}_{\perp}$, we employ a RANSAC scheme. From the full set of normalized segments, we randomly produce VP hypotheses and check for the size of the combined sets. The ouput of this is a rough estimate of the most-voted VP and, more importantly, the two sets of inliers.

Usually in order to produce a VP hypothesis we need to intersect at least two segments. However, if we have knowledge of its corresponding vanishing line (VL), we know that the VP must lie on the VL [19]. Thus, intersecting any segment with this VL will yield a VP hypothesis, as shown in Fig. 1. We employ a geodesic metric in order to define a segment as inlier,

$$d_{\mathbf{u}_k,\mathbf{v}} = \left| \arctan\left(\frac{|\mathbf{u}_k \times \mathbf{v}|}{\mathbf{u}_k \cdot \mathbf{v}} \right) \right| \tag{1}$$

where $\mathbf{v}$ is the hypothesis of the vanishing point, $d_{\mathbf{u}_k,\mathbf{v}}$ is the distance in radians and $\mathbf{u}_k$ is the normal of the k^{th} element of the segment set. This gives the advantage that the metric can be used for segments that intersect the same VP and those which intersect the VL orthogonal to the current VL hypothesis by setting the residual to $\min\left(d_{\mathbf{u}_k,\mathbf{v}}, |\pi/2 - d_{\mathbf{u}_k,\mathbf{v}}|\right)$. If this is lower than an angle threshold, then the segment is counted as an inlier, adding it to its corresponding set.

C. Least Squares Solution

Once we have found the most supported hypothesis, we get two supporting sets, parallel lines to the VP ($\mathbf{S}_{\parallel}$) and orthogonal lines to the VP ($\mathbf{S}_{\perp}$). From these two sets we now intend to compute a refined direction for the VP. In order to do so we propose a simple least squares scheme that minimizes the cost function

$$E(\mathbf{v}) = \sum_{i \in \mathbf{S}_{\parallel}} \left(\mathbf{v} \cdot \mathbf{u_i}\right)^2 + \sum_{j \in \mathbf{S}_{\perp}} \left((\mathbf{v} \times \mathbf{a}_C) \cdot \mathbf{u_j}\right)^2, \tag{2}$$

subject to $\|\mathbf{v}\| = 1$, where $\mathbf{v}$ is the vanishing point being refined, and $\mathbf{a}_C = \mathbf{C}\left({}^{C}_{I}q\right) \cdot \frac{\mathbf{a}_m}{\|\mathbf{a}_m\|}$, where $\mathbf{a}_m$ is the measured acceleration at the time of the acquired frame[1]. Notice that we no longer use the geodesic sphere distance and instead have opted for a simple dot product. This is valid since we no longer need to admit both parallel and orthogonal normal vectors using the same expression. We now proceed to minimize (2), for that we rewrite

$$\begin{aligned} E(\mathbf{v}) &= \mathbf{v}^\top \left(\sum\nolimits_i \mathbf{u_i}\mathbf{u_i}^\top \right) \mathbf{v} - \lambda \mathbf{v}^\top \mathbf{v} + \\ &\quad (\mathbf{v} \times \mathbf{a}_C)^\top \left(\sum\nolimits_j \mathbf{u_j}\mathbf{u_j}^\top \right) (\mathbf{v} \times \mathbf{a}_C) \qquad (3) \\ \frac{\partial E}{\partial \mathbf{v}} &= 0 = \left(\sum\nolimits_i \mathbf{u_i}\mathbf{u_i}^\top \right) \mathbf{v} + \\ &\quad \lfloor \mathbf{a}_C \rfloor_\times^\top \left(\sum\nolimits_j \mathbf{u_j}\mathbf{u_j}^\top \right) \lfloor \mathbf{g}_C \rfloor_\times \mathbf{v} - \lambda \mathbf{v} \end{aligned}$$

$$\lambda \mathbf{v} = \left[\sum\nolimits_i \mathbf{u_i}\mathbf{u_i}^\top - \lfloor \mathbf{g}_C \rfloor_\times^\top \left(\sum\nolimits_j \mathbf{u_j}\mathbf{u_j}^\top \right) \lfloor \mathbf{g}_C \rfloor_\times \right] \mathbf{v} \tag{4}$$

where $\lfloor \mathbf{a} \rfloor_\times$ is the skew-symmetric matrix of the vector $\mathbf{a}$. The resulting expression (4) is an eigenvalue problem of the form $A\mathbf{x} = \lambda \mathbf{x}$, obtained by applying the Lagrange multiplier to (3) to enforce $\|\mathbf{v}\| = 1$. The problem can thus be solved using standard numerical methods. In our implementation, the terms are weighted using the length of the segment, assigning a larger importance to segments that are usually detected more reliably and with more accuracy by LSD. This experimentally showed an increase in the accuracy of the detection.

IV. Visual Inertial Filter

As the underlying feature-based visual-inertial system, we have chosen the one proposed in [2]. The advantage of this VIO system is twofold: First, it has been shown to provide state-of-the art results in VIO while maintaining a computational complexity that is linear with the observed corner features (which in this case are KLT [21] tracks). Second, the method keeps a *sliding window* of the camera poses, which will be of use when performing the delayed update using VPs (see Section V-C). This method, however, is not a full visual-inertial SLAM method, as it does not keep track of the features it uses for its estimate in the state, an thus, makes no attempt at doing any loop closure.

In the following, we provide details regarding the structure of the filter and point feature update strategy employed by the chosen system [2].

A. Structure of the state vector

The core EKF state includes the pose, velocity and IMU biases of the system:

$$\mathbf{X}_{IMU} = \begin{bmatrix} {}^{I}_{G}q^\top & \mathbf{b}_g^\top & {}^{G}\mathbf{v}_I^\top & \mathbf{b}_a^\top & {}^{G}\mathbf{p}_I^\top \end{bmatrix}^\top \tag{5}$$

where ${}^{I}_{G}q$ is the orientation unit quaternion describing the rotation from frame $\{G\}$ to frame $\{I\}$, ${}^{G}\mathbf{v}_I$ and ${}^{G}\mathbf{p}_I$ are the velocity and positions of the IMU in $\{G\}$, and $\mathbf{b}_a$ and $\mathbf{b}_g$ are the biases of the accelerometer and gyroscope expressed as 3-vectors. The error-state of the system can be written as:

$$\widetilde{\mathbf{X}}_{IMU} = \begin{bmatrix} \delta\boldsymbol{\theta}_I^\top & \widetilde{\mathbf{b}}_g^\top & {}^{G}\widetilde{\mathbf{v}}_I^\top & \widetilde{\mathbf{b}}_a^\top & {}^{G}\widetilde{\mathbf{p}}_I^\top \end{bmatrix}^\top. \tag{6}$$

Here the standard additive error is used for all terms except the orientation, which is handled as an *error quaternion* δq that arises from $q = \delta q \otimes \hat{q}$, where $\otimes$ is the quaternion multiplication and $\delta q \simeq [\frac{1}{2}\delta\boldsymbol{\theta}^\top \quad 1]$. Finally, if we assume a

[1]This neglects the effects of acceleration on the system. This proved to be a valid assumption for low acceleration applications, such as handheld applications. Furthermore, we assume that $\mathbf{C}\left({}^{C}_{I}q\right)$ is a known calibration constant.

sliding window of N past IMU poses, the EKF error-state vector becomes:

$$\widetilde{\mathbf{X}} = \left[\widetilde{\mathbf{X}}_{IMU}^\top \;\; \delta\boldsymbol{\theta}_{I_1}^\top \;\; {}^G\widetilde{\mathbf{p}}_{I_1}^\top \cdots \delta\boldsymbol{\theta}_{I_N}^\top \;\; {}^G\widetilde{\mathbf{p}}_{I_N}^\top\right]^\top. \quad (7)$$

At regular frame intervals, the pose of the system is *cloned* and the cloned pose is kept in the state and propagated along with the current pose and biases whenever a new IMU measurement is available. [2] and [22] for details regarding the The reader is referred to propagation step.

B. Point feature update

As mentioned, the system used here [2] relies on point features for its update. These are computed from KLT tracks computed at each frame. The way in which the point feautures are handled as updates to the EKF was proposed in [2] and is summarized here for completeness.

Whenever a tracked point feature, f_j, is no longer detected or when the clone of the pose that originated it is going to be discarded from the state, an update is triggered for such feature. The set of M_j camera poses (see (7)) that observed f_j are used to triangulate the feature into the global frame ${}^G\hat{\mathbf{p}}_{f_j}$, and thus, the point in the global frame is correlated with the state of the system and cannot be directly used by the standard EKF update equation. To overcome this, the residual of the observation $\mathbf{r}^j$ is projected to the left nullspace (here $\mathbf{A}$) of the linearized model of the feature observation:

$$\mathbf{r}^j \simeq \mathbf{H}_{\mathbf{X}}^j\widetilde{\mathbf{X}} + \mathbf{H}_f^j \; {}^G\widetilde{\mathbf{p}}_{f_j} \quad (8)$$

$$\mathbf{r}_o^j = \mathbf{A}^\top\left(\mathbf{z}^j - \hat{\mathbf{z}}^j\right) \simeq \mathbf{A}^\top\mathbf{H}_{\mathbf{X}}^j\widetilde{\mathbf{X}} + \mathbf{A}^\top\mathbf{n}^j \quad (9)$$

Where $\mathbf{H}_{\mathbf{X}}^j$ and $\mathbf{H}_f^j$ result from stacking the Jacobians of $\mathbf{z}_i^j$ w.r.t. the state and the feature position, respectively, and $\widetilde{\mathbf{p}}_{f_j}$ is the error of the position estimate of f_j. After the projection into the nullspace, we can now use r_o and $\mathbf{A}^\top\mathbf{H}_{\mathbf{X}}^j\widetilde{\mathbf{X}}$ to perform the regular EKF update. Since $\mathbf{A}$ is unitary, $\mathbf{A}^\top\mathbf{n}^j$ can also be simply written as $\mathbf{n}_o^j$, a diagonal matrix with the pixel noise as diagonal entries.

V. Vanishing point update

One could adopt a method that does not require to keep track of the VP in the state, for example, by simply using a voting scheme to determine the direction of the VP in the global frame. However, tracking the direction of the VP as part of the filter allows us to account for the uncertainty of the VP in the world frame and update the filter optimally (up to linearization errors).

However, in order to initialize a first estimate of the VP in the global frame we do use a voting scheme. This means that our system is capable of keeping track of more than one VP in the state. Thus, this method supports man-made environments with corridors or structures at non square angles w.r.t. each other, i.e. a superposition of Manhattan Worlds (known as "Atlanta world" [23]).

A. Initialization of new vanishing points

Since we are assuming an intrinsically calibrated camera, we simplify the discussion by assuming VP measurements as homogeneous vectors measured in the normalized image plane. Whenever a new VP measurement is registered, in the form of a unit vector in the camera frame $\mathbf{v}_C$, we do an angle check against a set $\mathcal{C}$ of VP candidates in the global frame to be added to the filter state:

$$v_G^{new} := \mathbf{C}\left({}_G^I\hat{q}\right)\mathbf{C}\left({}_C^Iq\right)\mathbf{v}_C \quad (10)$$

$$\theta_i = \mathbf{D}\left(P_{0,\pi/2}\cdot\mathbf{v}_G^{new},\;\; \mathbf{v}_G^i\right),\;\; \mathbf{v}_G^i \in \mathcal{C} \quad (11)$$

where $\mathbf{D}(\cdot)$ is a measure of angular distance and $\mathbf{C}(q)$ is the corresponding rotation matrix of a quaternion q. If the angle θ_i falls bellow a threshold, member i gains a vote and, if already present in the EKF's state, we proceed to use $\mathbf{v}_G^{new}$ to update the filter, otherwise $\mathbf{v}_G$ is added to $\mathcal{C}$.

Since we operate under the Manhattan assumption, we also do an angle check of $\mathbf{v}_C$ rotated around gravity by $\pi/2$ (as $P_{o,\pi/2}$ in (11)), this way any candidate and its corresponding orthogonal counterpart (around the direction of gravity) are considered as the same vanishing direction. This means that we can keep track of only one VP in the state that corresponds to both principal x, y directions of the Manhattan World.

At every update, only the highest voted candidate in $\mathcal{C}$ is chosen for state augmentation, provided that it has a minimum amount of votes.

B. Vanishing point error representation

A vanishing point is considered a direction in 3-space, encoded as a unit vector in $\mathbb{R}^3$ and consisting of two degrees of freedom. In order to update the filter using the same amount of degrees of freedom, we will parametrize the error representation of the VPs. We use a tangent space to the unit sphere, which is a map $\mathbb{R}^3 \mapsto \mathcal{S}^2$, namely, $f(\mathbf{y}) = \hat{\mathbf{y}}/\left|\hat{\mathbf{y}}\right|$ with $\hat{\mathbf{y}} = [\mathbf{y}^\top\; 1]^\top$. So, f maps the origin to $[0\;0\;1]^\top$ and its Jacobian is $\partial f/\partial\mathbf{y} = [\mathrm{I} \mid 0]^\top$[19].

Consider now $\mathbf{x}$ to be a unit vector in $\mathbb{R}^3$, and let $\mathrm{H}_{\mathbf{v(x)}}$ be its Householder matrix such that $\mathrm{H}_{\mathbf{v(x)}}\mathbf{x} = [0\;0\;1]^\top$. We can now use this to parametrize a vector $\mathbf{x} \in \mathbb{R}^3$ to lie on a unit sphere by setting $\mathbf{x} = \mathrm{H}_{\mathbf{v(x)}}f(\mathbf{y})$, with $\mathbf{y} \in \mathbb{R}^2$. The Jacobian of this parametrization is:

$$\frac{\partial\mathbf{x}}{\partial\mathbf{y}} = \mathrm{H}_{\mathbf{v(x)}}\;[\mathrm{I} \mid 0]^\top. \quad (12)$$

In our particular case this means that we can use a plane tangent to the current estimate of the VP in the global frame, $\hat{\mathbf{v}}_G$, and encode its error in the state $\delta\hat{\mathbf{v}}_G$ by using

$$\hat{\mathbf{v}}_G = \mathrm{H}_{\mathbf{v}(\hat{\mathbf{v}}_\mathbf{G})}\cdot f\left(\delta\hat{\mathbf{v}}_G\right). \quad (13)$$

Whenever a VP is added to the state, the *augmented* error state vector becomes:

$$\widetilde{\mathbf{X}} = \left[\widetilde{\mathbf{X}}_{IMU}^\top \;\; \widetilde{\mathbf{X}}_{Clones}^\top \;\; \delta\hat{\mathbf{v}}_G^1 \;\; \delta\hat{\mathbf{v}}_G^2 \cdots\right]^\top. \quad (14)$$

C. Measurement model

The camera measures a VP as $\mathbf{z}_j$, which is a projection of the normalized vector $\mathbf{v}_G^j$, i.e.

$$\mathbf{z}_j = \mathbf{\Pi}\ \mathbf{C}\left({}^{C}_{I}q\right)\mathbf{C}\left({}^{I}_{G}q\right)\mathbf{v}_G^j + \eta_j \tag{15}$$

where $\mathbf{\Pi}$ represents a projection matrix, so that

$$\mathbf{z}_j = \mathbf{\Pi}\ \mathbf{v}_C^j + \eta_j, \quad \mathbf{\Pi} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \tag{16}$$

$$\mathbf{v}_C^j = \frac{1}{\sqrt{u^2+v^2+1}}\begin{bmatrix}u\\ v\\ 1\end{bmatrix} + \eta_j \tag{17}$$

with u and v being the coordinates of the observed vanishing point[2]. For the update, we need to relate the measurement error $\tilde{\mathbf{z}}_j$ to the state vector $\widetilde{\mathbf{X}}$,

$$\tilde{\mathbf{z}}_j = \mathbf{z}_j - \hat{\mathbf{z}}_j \simeq \mathbf{H}_j\widetilde{\mathbf{X}} + \eta_j, \tag{18}$$

where $\hat{\mathbf{z}}_j = \mathbf{h}(\widetilde{\mathbf{X}})$ is the expected measurement obtained by evaluating (15) at the current state estimate and the Jacobian $\mathbf{H}_j$ is equal to

$$\mathbf{H}_p\left[0_{2\times 15}\mid 0_{2\times 3}\cdots\mathbf{H}_{\theta_I}^k\cdots 0_{2\times 3}\mid 0_{2\times 3}\cdots\mathbf{H}_{\mathbf{v}_G}\cdots 0_{2\times 3}\right], \tag{19}$$

where the index k corresponds to the pose that was cloned when the measurement j was scheduled (as will be discussed in Section V-E). The partial derivatives are:

$$\mathbf{H}_p = \frac{\partial\mathbf{h}}{\partial\mathbf{v}_C^j} = \mathbf{\Pi} \tag{20}$$

$$\mathbf{H}_{\theta_I}^k = \frac{\partial\mathbf{v}_C^j}{\delta\theta_I} = \mathbf{C}\left({}^{C}_{I}q\right)\lfloor\mathbf{C}\left({}^{I}_{G}\hat{q}\right)\mathbf{v}_G^j\rfloor_\times \tag{21}$$

$$\mathbf{H}_{\mathbf{v}_G}^j = \frac{\partial\mathbf{v}_C^j}{\delta\theta} = \mathbf{C}\left({}^{C}_{I}q\right)\mathbf{C}\left({}^{I}_{G}\hat{q}\right)\mathrm{H}_{\mathbf{v}(\hat{\mathbf{v}}_\mathbf{G})}\ [\mathrm{I}\mid 0]^\top. \tag{22}$$

The Jacobian $\mathbf{H}_j$ can now be used to update the filter or to augment the state along with the Jacobian w.r.t. the measurement noise.

D. Measurement noise

As it can be seen from (16)-(17) and if we consider the noise in the measurements to be corrupted with additive, zero-mean, white, Gaussian noise:

$$u_m = u + \eta_u \quad v_m = v + \eta_v, \tag{23}$$

then we have $\mathbf{z}_j$ as a nonlinear function of the vector in the camera frame $\mathbf{v}_C^j$. In order to compute the covariance of the innovation (as well as the Mahalanobis distance for data association) we need the derivative of the measurement model $\mathbf{h}$ w.r.t. the measurement noise, $\partial\mathbf{h}/\partial\mathbf{n}_{u,v}$, since

$$\mathbf{z}_j = \frac{1}{\sqrt{u_m^2+v_m^2+1}}\begin{bmatrix}u_m\\ v_m\end{bmatrix} \tag{24}$$

$$\simeq \frac{1}{\sqrt{u^2+v^2+1}}\begin{bmatrix}u\\ v\end{bmatrix} + \mathbf{\Gamma}_{u,v}\cdot\mathbf{n}_{u,v} \tag{25}$$

[2]Here we make the assumption that the camera directly observes the vanishing direction as a normalized image point, whereas in reality the camera observes only line segments. This simplifies the derivation while allowing to incorporate a more meaningful uncertainty into the measurement.

which results in the following Jacobian

$$\mathbf{\Gamma}_{u,v} = \frac{1}{(u^2+v^2+1)^{3/2}}\begin{bmatrix}(v^2+1) & -uv\\ -uv & (u^2+1)\end{bmatrix} \tag{26}$$

The resulting covariance calculated using (26) can be shown to be positive definite [24].

We can now compute the innovation covariance of the EKF as

$$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\top + \mathbf{\Gamma}\mathbf{R}_k\mathbf{\Gamma}^\top \tag{27}$$

with $\mathbf{R}_k$ as 2 dimensional zero-mean Gaussian noise measured in pixels. With $\mathbf{S}_k$ we can finally update the estimates of the system using the usual EKF gain.

E. Vanishing point detection scheduling

Because of its complexity, the line detection step might not be able to complete before a new frame is available to the system. We thus take advantage of the sliding window of clones in the state and adopt a delayed-measurement strategy that allows for real-time execution of the filter while being able to schedule line-detection tasks and use them when they become available.

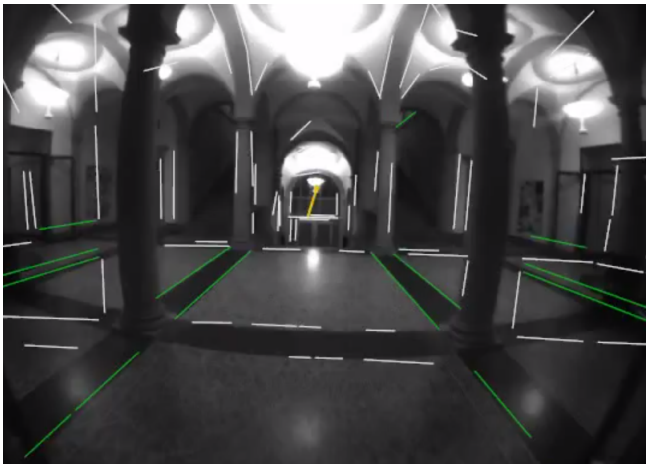
Whenever a new pose $\mathbf{T}_k$, corresponding to frame I_k, becomes cloned in the state, we schedule the frame for vanishing point detection. This is executed on a separate thread, and the cloned pose is kept on the state until the thread finishes and returns the VPs detected. We keep track of the poses waiting for VPs to be computed, so we keep $\mathbf{T}_k$ from being marginalized if we require to clone a new pose. At this point the clone $\mathbf{T}_k$ is updated with $\mathbf{v}_k$, the VP detected on frame I_k as detailed on Section V-C.

VI. Results

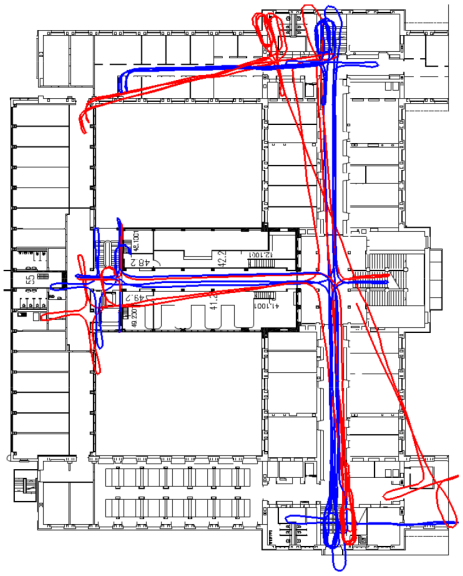
The experiments were carried out in man-made environments, such as indoor scenarios. In order to test the system's suitability for real-time operation on computationally constrained systems, a mobile device was employed. The testbed is equipped with an NVIDIA Tegra K1 processor with 4GB of RAM, a global-shutter fisheye camera and an IMU. Each frame processed was first undistorted before running LSD on them.

Given the high accuracy of the underlying system [2], large datasets had to be gathered in order to make the angular drift more noticeable. For instance, in Fig. 2c the length of the trajectory was over 1.4 kilometers in length. In order to quantize the error, the building plans were used. Fig. 3a shows the error of the trajectory at the control points situated in the locations shown as yellow dots in Fig. 2c Finally, Fig. 3b shows the decrease in uncertainty when using VPs to update the system. Although the uncertainty decreases, one would expect the standard VIO uncertainty to be higher, as we have no global observation of it. This is due to an inconsistency in the linearized model used in VIO, as discussed in [25].

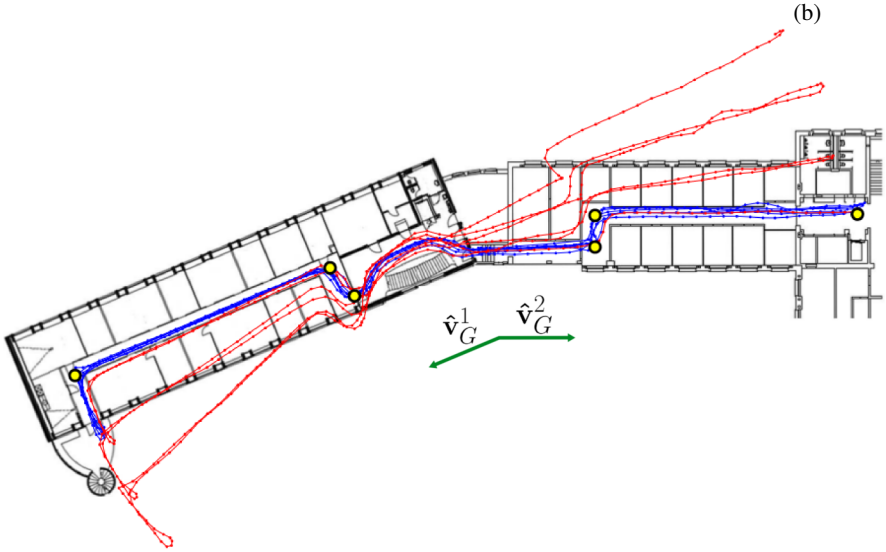
It should be emphasized that the system is capable of keeping track of several vanishing points, accommodating thus to superpositions of Manhattan-like environments. That



(a)



(b)



(c)

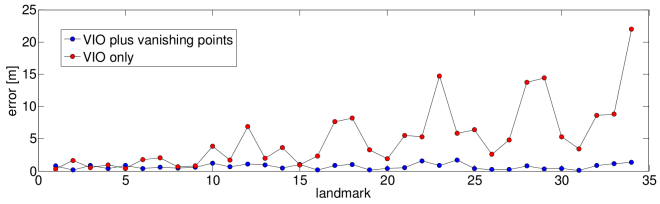
Fig. 2: Fig. a shows a sample of the VP detection with inlier lines depicted in green. Fig. b shows a top-down view of a trajectory comparison spanning 1.4 kilometers and four floors performed inside the Computer Science building at ETH Zürich. In red the VIO-only solution is shown and in blue the trajectory using VPs. Fig. c shows a top-down view of a 640 meter long dataset which exhibits two (non-orthogonal) vanishing points, shown in green. For this dataset, six control points were situated at key locations in order to quantify the error (Fig. 3a).

is, if an environment has sections of its construction at non-right angles w.r.t. each other, the system will simply initialize a new vanishing point to account for this new principal direction. Whenever a new VP is set to be initialized as a state in the filter but we have reached the maximum number of VP allowed in the filter, we check if the new VP has more votes than any of the VPs currently in the filter. In order to show this capability, the dataset shown in Fig. 2c exhibits two groups of orthogonal VP directions.

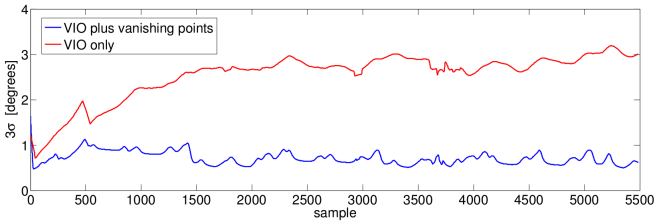
As mentioned, special care was put into making the system run at framerate regardless of the computational burden of the VP detection step. This means that the scheduler would not be relaying tasks at framerate on most mobile systems, but in fact runs in the background continuously and process the most recent frame whenever a VP detection cycle is finished. For the platform used, VP were detected at an average of every 3.5 frames. More specific timings for the platform used are listed in Table (I).

VII. Conclusions

In this paper we have shown that by carefully leveraging the available information in a Camera-IMU system, and by exploiting the types of structures in man-made environments, we can efficiently and reliably detect and keep track of vanishing directions. It was shown that the information provided



(a)



(b)

Fig. 3: Fig. a compares the error of the trajectory at the control points (see Fig. 2) of the proposed system vs VIO. Fig. a compares the covariance of the rotation around the direction of gravity

Cost of step	Mean
State propagation	0.48 ms
Point features update	0.82 ms
Line detection	**31.2 ms**
1-line RANSAC classification	1.4 ms
LLS VP solution	0.23 ms
VP voting and update	0.41 ms

TABLE I: Timings for the system. Notice that the timing for KLT is not present, since it was executed on the GPU.

by such measurements is complimentary information to the classical VIO framework, and effectively eliminates angular drift, increasing the overall accuracy of the estimates.

The system proposed is an inexpensive solution to the localization problem, specially for low-cost or computationally constrained platforms, such as mobile robots or handheld devices. Because of the nature of vanishing points, the method is a good alternative to full-scale SLAM, since its complexity does not increase with the length of the trajectory, nor it requires the same place to be re-visited. However, in order to also constrain drift in x, y and z, a loop closing step is suggested. Naturally, such step would also benefit from the increased accuracy of the pose estimates of this method to use as a prior for loop closure, being particularly beneficial after long trajectories.

REFERENCES

[1] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *The International Journal of Robotics Research*, vol. 30, no. 1, pp. 56–79, 2011.

[2] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3565–3572, IEEE, 2007.

[3] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *The International Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011.

[4] A. Martinelli, "Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *Robotics, IEEE Transactions on*, vol. 28, no. 1, pp. 44–60, 2012.

[5] J. Kim and S. Sukkarieh, "Improving the real-time efficiency of inertial slam and understanding its observability," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 1, pp. 21–26, IEEE.

[6] M. Bryson and S. Sukkarieh, "Observability analysis and active control for airborne slam," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 44, no. 1, pp. 261–280, 2008.

[7] D. G. Kottas and S. I. Roumeliotis, "Exploiting urban scenes for vision-aided inertial navigation.," in *Robotics: Science and Systems*, 2013.

[8] F. Fraundorfer, P. Tanskanen, and M. Pollefeys, "A minimal case solution to the calibrated relative pose problem for the case of two known orientation angles," in *Computer Vision–ECCV 2010*, pp. 269–282, Springer, 2010.

[9] Z. Kukelova, M. Bujnak, and T. Pajdla, "Closed-form solutions to minimal absolute pose problems with known vertical direction," in *Computer vision–ACCV 2010*, pp. 216–229, Springer, 2011.

[10] J. Lobo and J. Dias, "Vision and inertial sensor cooperation using gravity as a vertical reference," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 12, pp. 1597–1608, 2003.

[11] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics Research*, pp. 201–212, Springer, 2011.

[12] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 957–964, IEEE, 2012.

[13] S. Lynen, S. Omari, M. Wueest, M. W. Achtelik, and R. Siegwart, "Tightly coupled visual-inertial navigation system using optical flow," in *Research, Education and Development of Unmanned Aerial Systems*, vol. 2, pp. 251–256, 2013.

[14] F. M. Mirzaei and S. I. Roumeliotis, "Optimal estimation of vanishing points in a manhattan world," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2454–2461, IEEE, 2011.

[15] J.-C. Bazin and M. Pollefeys, "3-line ransac for orthogonal vanishing point detection," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 4282–4287, IEEE, 2012.

[16] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[17] J.-C. Bazin, Y. Seo, C. Demonceaux, P. Vasseur, K. Ikeuchi, I. Kweon, and M. Pollefeys, "Globally optimal line clustering and vanishing point estimation in manhattan world," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 638–645, IEEE, 2012.

[18] J. Montiel and A. J. Davison, "A visual compass based on slam," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1917–1922, IEEE, 2006.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[21] B. D. Lucas, T. Kanade, *et al.*, "An iterative image registration technique with an application to stereo vision.," in *IJCAI*, vol. 81, pp. 674–679, 1981.

[22] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, 2005.

[23] G. Schindler and F. Dellaert, "Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1, pp. I–203, IEEE, 2004.

[24] N. Trawny and S. Roumeliotis, "Sun sensor model," tech. rep., University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep, 2005.

[25] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Towards consistent vision-aided inertial navigation," in *Algorithmic Foundations of Robotics X*, pp. 559–574, Springer, 2013.