A Monocular Camera Gyroscope

Ville Huttunen and Robert Piché
Tampere University of Technology

Abstract

We present a method for tracking the 3-axis orientation of a monocular camera using orthogonal vanishing points detected in individual frames of a sequence of images. Robust and real-time vanishing point detection is done using a standard line segment detection method and an adaptive RANSAC algorithm. Vanishing points and corresponding vanishing directions found in consecutive frames are associated with each other to produce a sequence of orientation quaternions, which is processed by an extended Kalman filter. Experiments with a consumer-level, handheld mobile device indicate that the accuracy of the proposed method is comparable with those of consumer-grade inertial motion sensors.

Keywords: Computer vision, indoor navigation, vanishing points, orientation estimation

1 Introduction

With the availability of cheap small low-power vision sensors, there is a surge of interest in using them in positioning and navigation applications in digital devices such as mobile phones. There is already an extensive body of work in the computer vision field on the so-called *egomotion* problem of inferring 3D motion (six degree-of-freedom rotation and translation) of a camera mounted on a robot or a vehicle from a sequence of images. Most methods are based on detecting and tracking primitive image features (points, lines, etc.) in the image stream. Methods such as simultaneous localization and mapping (SLAM) and structure from motion (SfM) use visual and other data to build up a three-dimensional model of the environment, relative to which the camera's motion is described [10].

In this work we focus on the development of a vision-based sensor for the 3-axis orientation, that is, a gyroscope. Because the measurement error of conventional gyroscopes that are based on inertial sensors accumulates (drifts) over time, a navigation system needs to include driftless data sources, such as vision-based sensors [19, 22]. In principle, a camera gyroscope is driftless and should thus well complement an inertial gyroscope.

Our system is based on tracking vanishing points that have been identified using line segments detected in images. Concerning ourselves only with vanishing points makes orientation tracking significantly easier for the following reasons:

1. Vanishing points arise from line features that are distinct and abundant in images of architectural scenes in indoor and dense urban environments, where Global Navigation Satellite System (GNSS) positioning is inadequate.

2. The number of interesting (orthogonal) vanishing points is limited, since any scene in the 3-dimensional world can contain at most three mutually orthogonal directions.

3. Vanishing points are dependent only on the camera's orientation, not its position.

4. Vanishing point based methods are robust with regard to non-stationary objects (people, vehicles) moving in the scene.

Vanishing points have been used in applications including autonomous vehicle navigation [17], architectural scene reconstruction [1, 12, 20, 21], and photo correction [8]. Similarly to our approach, most studies assume that the camera's instrinsic parameters are known and fixed, but vanishing points can also be used for intrinsic parameter calibration [3, 11, 16, 18]. The vanishing point based camera gyroscope of Kessler et al. [14] is similar to ours, except that they do not filter orientation estimates, and compute only the horizon line and the central vanishing point.

This paper is structured as follows. In Section 2, we define the pinhole camera model, derive vanishing points algebraically and describe their basic properties using projective geometry. Section 3 describes the combination of line segment detection and RANSAC-based line clustering used for vanishing point detection. The method for extracting and filtering accurate orientation measurements from detected vanishing points is presented in section 4. Finally in section 5, experiments are conducted using a mobile phone containing both a camera and inertial sensors. Conclusions and discussion of areas of future work are in section 6.

2 Geometry of Vanishing Points

The theory of vanishing points is usually presented in the context of *projective geometry*, the study of properties of geometric objects under projective transformations. A camera can be considered as such a transformation, since it takes a 3-dimensional scene and projects it onto a 2-dimensional image plane. In projective geometry, points are represented in *homogeneous* coordinates. For example, a point in 2-dimensional Euclidian space $(x, y) \in \mathbb{R}^2$ can be transformed into a homogeneous point in a projective 2-space $\mathbb{P}^2 \subseteq \mathbb{R}^3$ simply by appending component 1 to the end of the Euclidian representation, $(x, y, 1) \in \mathbb{P}^2$. In projective space $\mathbb{P}^2$, the points $(x, y, 1)$ and $(\alpha x, \alpha y, \alpha)$ are equivalent for any non-zero α. An important subset of $\mathbb{P}^2$ consists of points that have the form $(x, y, 0)$. These points are known as the *ideal points* and they correspond to points located infinitely far away from the origin. Similarly, in the projective space $\mathbb{P}^3$, ideal points have the form $(x, y, z, 0)$.

In order to define vanishing points algebraically, we specify the projective transformation used to model the camera. One of the simplest and most commonly used models for a finite projective camera is the *pinhole model* [10]:

$$P : \mathbb{P}^3 \mapsto \mathbb{P}^2, \quad P(\boldsymbol{X}) = KR\,[I_3, -\boldsymbol{t}]\,\boldsymbol{X}, \tag{1}$$

where $K \in \mathbb{R}^{3\times3}$ is the camera's calibration matrix, $R \in \mathbb{R}^{3\times3}$ is an orthogonal rotation matrix and $\boldsymbol{t} \in \mathbb{R}^3$ is the camera's translation vector in the world coordinate frame (Figure 1). Rotation R and translation $\boldsymbol{t}$ are known as the extrinsic camera parameters. The calibration matrix K is an upper-triangular matrix consisting of the intrinsic camera parameters

$$K = \begin{bmatrix} \alpha f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix},$$

where f is the focal length, (p_x, p_y) is the principal point, α is the pixel aspect ratio and s is the skew coefficient of the camera. Most CCD-based digital cameras have square pixels ($\alpha = 1$), zero skew ($s = 0$) and principal point close to the image center. It is also reasonable to assume that the intrinsic camera parameters do not change over time.

Consider a line in 3-space that goes through point $\boldsymbol{A} = (a_x, a_y, a_z, 1)^T$ with direction vector $\boldsymbol{D} = (\boldsymbol{d}^T, 0)^T, \boldsymbol{d} \in \mathbb{R}^3$. The line's equation is

$$\boldsymbol{X}(\lambda) = \boldsymbol{A} + \lambda\boldsymbol{D}, \quad \lambda \in \mathbb{R}.$$

When this line is projected through the pinhole camera P onto the image plane $\mathbb{P}^2$, we get

$$\boldsymbol{x}(\lambda) = P(\boldsymbol{X}(\lambda)) = P(\boldsymbol{A}) + \lambda P(\boldsymbol{D}) = \boldsymbol{a} + \lambda KR\boldsymbol{d},$$

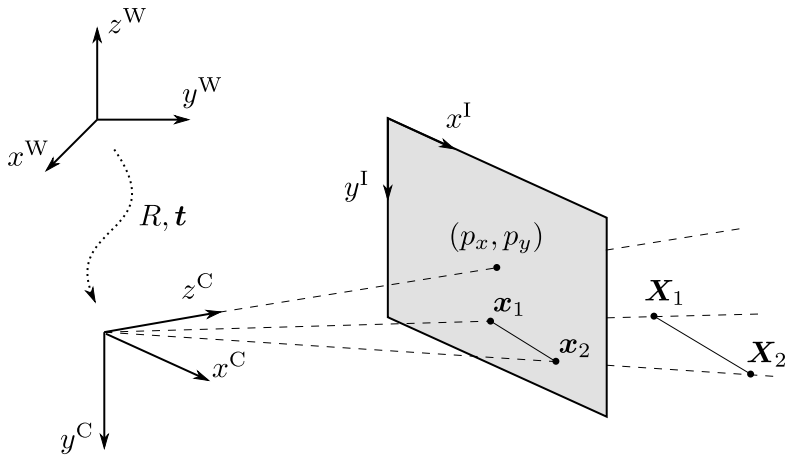

Figure 1: Pinhole camera model. Camera rotation R and translation $\boldsymbol{t}$ define a coordinate transformation from the world coordinate frame W to the camera frame C. A point in the camera frame C can be projected onto the image plane I if the camera calibration matrix K is known. Combining these two transformations results in the camera model (1) which takes a point $\boldsymbol{X} \in \mathbb{P}^3$ in the world frame W and projects it into a point $\boldsymbol{x} \in \mathbb{P}^2$ on the image plane I.

where $\boldsymbol{a} = P(\boldsymbol{A})$ is the image of $\boldsymbol{A}$. The vanishing point $\boldsymbol{v} \in \mathbb{P}^2$ corresponding to the direction $\boldsymbol{d}$ is the limit point of projected line $\boldsymbol{x}(\lambda)$ as λ tends to infinity,

$$\boldsymbol{v} = \lim_{\lambda \to \infty} \boldsymbol{x}(\lambda) = \lim_{\lambda \to \infty} (\boldsymbol{a} + \lambda K R \boldsymbol{d}) = K R \boldsymbol{d}. \tag{2}$$

When (2) is represented in the camera's own coordinate frame ($R = I_3, \boldsymbol{t} = \boldsymbol{0}$) we get

$$\boldsymbol{v} = K \boldsymbol{d}. \tag{3}$$

From (2) we can infer that the vanishing point $\boldsymbol{v}$ does not depend on the camera's position. Furthermore, equation (3) shows that there is an one-to-one relation between direction vectors in 3-space and vanishing points in the image plane. Two vanishing points $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are said to be orthogonal if the corresponding line directions $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ are orthogonal. A vanishing point $v \in \mathbb{P}^2$ which is also an ideal point is called an *infinite vanishing point*. Similarly, non-ideal vanishing points are referred to as *finite vanishing points*.

3 Vanishing Point Detection

As noted in the previous section, there is a direct correspondence between the camera's orientation in the world frame and the vanishing points found in the image. The problem of determining the camera's orientation can therefore be

treated as the problem of finding vanishing points. The camera's orientation has three degrees of freedom, so we need to detect three vanishing points in every image. In section 4.1 it is shown that if we assume the three major vanishing points to be mutually orthogonal, then detecting only two of these points is enough to describe the camera rotation uniquely.

Vanishing point detection can be regarded as the search for points in an image where projections of parallel scene lines intersect. Our method for vanishing point detection is based on line segment detection in the images and it can basically be separated into two parts:

1. *Line segment detection.* The line segment detector takes a grayscale image as an input and returns a list of line segments detected in the image.

2. *Line clustering.* Detected line segments are clustered into groups of lines with common intersection points. Clusters with most line segments are picked and detected vanishing points are chosen to be the intersection points within these clusters.

The next two sections describe these two steps in more detail.

3.1 Line Segment Detection

In the last few decades, numerous algorithms have been proposed for line detection. Most of the methods are based on the Hough transform [6] or connected component analysis of image gradient orientations [2]. We use a method of Gioi et al. [9] which belongs to the latter category. Although not as fast as some methods based on Hough transform, it has linear complexity with regard to the image size. With proper image scaling, we found Gioi's method to be able to perform under real-time constraints. Figure 2 shows typical output of the line detector [9] in an indoor scene.

3.2 Line Clustering

After line segments have been extracted from the image, the Random Sample Consensus algorithm [7] (RANSAC) is used to cluster the line segments into groups whose lines share a common intersection point. These intersection points are regarded as the detected vanishing points. RANSAC produces one line cluster at a time, so in order to detect three vanishing points, the RANSAC algorithm is applied three times. Between each application, line segments corresponding to already detected vanishing points are removed. We do not differentiate between finite and infinite vanishing points during the

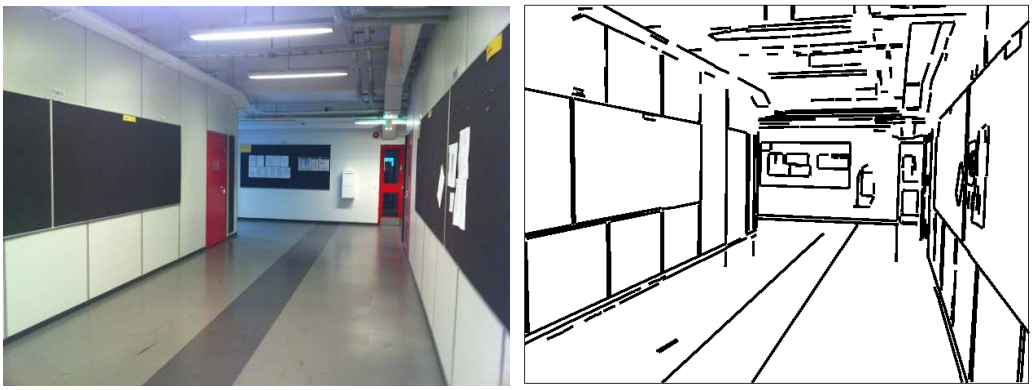

Figure 2: Line segment detection

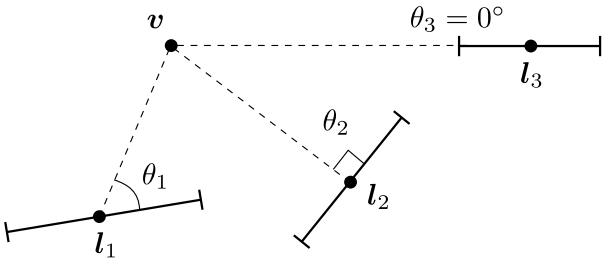

Figure 3: Angles between line segments and a vanishing point $\boldsymbol{v}$.

line clustering. For our purposes, infinite vanishing points are approximated sufficiently well by finite vanishing points located far away from the image center.

In each RANSAC iteration, a pair of line segments is chosen randomly. The intersection point of this pair is a candidate for the vanishing point. To measure the quality of the vanishing point candidate, each of the detected line segments is labelled either as an *inlier* or an *outlier*, based on how well they support the given candidate. We define the distance function $d(\boldsymbol{v}, \boldsymbol{l})$ between vanishing point $\boldsymbol{v}$ and line segment $\boldsymbol{l}$ to be the angle between line segment $\boldsymbol{l}$ and the line going through $\boldsymbol{v}$ and the midpoint of $\boldsymbol{l}$ (Figure 3). Line segment $\boldsymbol{l}_i$ is said to be an inlier if its distance from vanishing point $d(\boldsymbol{v}, \boldsymbol{l}_i) < \theta_t$ for some predeterminated threshold level θ_t.

The number of RANSAC iterations is determined adaptively, as follows. Let r denote the ratio of inliers to outliers corresponding to vanishing point $\boldsymbol{v}$ in the set of detected line segments. We want to calculate the number of iterations k needed to ensure with probability p that at least one of the random line pairs chosen by RANSAC is free of outliers. The probability of choosing k pairs of line segments with at least one outlier each is $(1 - r^2)^k$.

The number of iterations needed to ensure that this probability is below the threshold $1 - p$ is

$$k \geq \frac{\log(1-p)}{\log(1-r^2)}. \tag{4}$$

The true inlier ratio r is unknown but it can be approximated from below by the number of line segments in the largest cluster found in the earlier iterations divided by the number of all line segments.

RANSAC produces crude estimates of vanishing points and their inlier sets of line segments. The next section describes how the extra information within the set of inliers can be used to further refine the vanishing point estimates.

3.3 Vanishing Point Refinement

Given a set of inlier lines $\{\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n\},\ \boldsymbol{l}_i \in \mathbb{R}^3$, corresponding to a vanishing point $\boldsymbol{v}$ the following equation holds:

$$[\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n]^T \boldsymbol{v} = \boldsymbol{0}. \tag{5}$$

Overdetermined system (5) can be solved for the nontrivial $\boldsymbol{v}$ (basis of the null space of $[\boldsymbol{l}_1, \ldots, \boldsymbol{l}_n]^T$) using SVD. Instead of solving system (5) directly, Cipolla and Boyer [4] suggest that lines $\boldsymbol{l}_i$ should first be transformed into normalized image coordinates

$$\boldsymbol{L}_i = \frac{K^{-1}\boldsymbol{l}_i}{\|K^{-1}\boldsymbol{l}_i\|}, \quad i = 1, \ldots, n,$$

where K is the known camera calibration matrix. Normalized image coordinates correspond to a camera with focal length of 1 and principal point at the origin of the image. Instead of solving system (5) for the vanishing point $\boldsymbol{v}$, we solve the following system for the corresponding vanishing direction $\boldsymbol{d}$

$$[\boldsymbol{L}_1, \ldots, \boldsymbol{L}_n]^T \boldsymbol{d} = L\boldsymbol{d} = \boldsymbol{0}. \tag{6}$$

Vanishing directions can be transformed back into vanishing points by multiplying them from left with the camera calibration matrix K. Example of line clusters corresponding to refined vanishing point estimates found in an indoor scene is shown in Figure 4.

4 Orientation Tracking

In this section, we present a simple method for inferring the camera orientation from two or three mutually orthogonal vanishing points. We also present

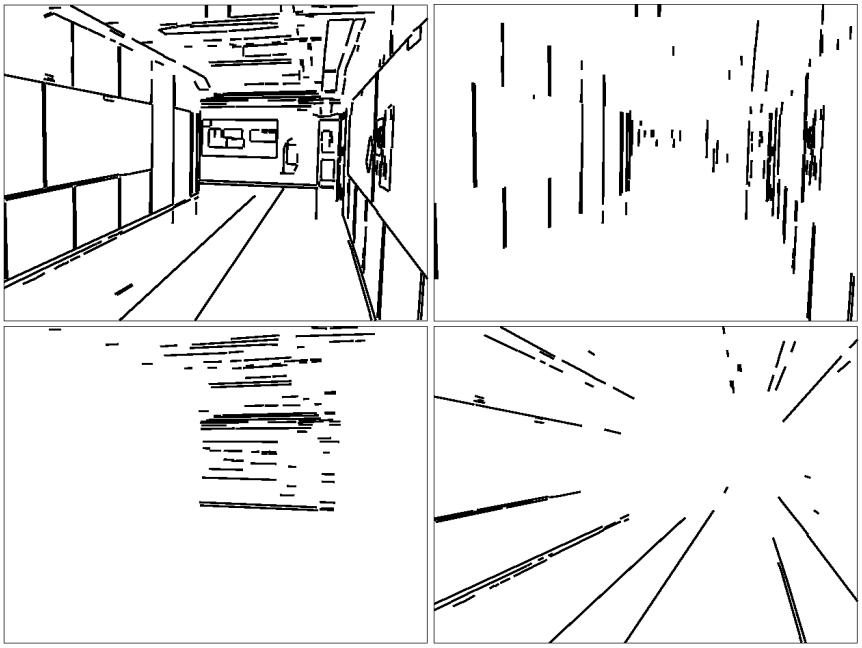

Figure 4: Results of the RANSAC-based line clustering. Upper left image displays all of the detected line segments. Other three images show line clusters corresponding to the three major orthogonal vanishing points detected in the scene. Upper right and lower left images correspond to the vertical and horizontal vanishing points located far away from the image center. Lower right image corresponds to the vanishing point at the end of the hallway.

a quaternion-based Kalman filter to estimate the real camera rotation from a sequence of orientation measurements.

4.1 Extracting Orientation from Vanishing Points

Equation (2) relates vanishing points with the camera orientation and actual line directions found in the scene. We assume that in the scene there exist three major line directions that are mutually orthogonal, and that the detected vanishing points correspond to these directions. This assumption is reasonable for indoor scenes based on rectangular architecture. The RANSAC-based vanishing point detection described in earlier sections generates three vanishing point candidates from an image regardless how many orthogonal line directions can be seen in the image (e.g. image of an wall can have at most two orthogonal vanishing points.). Under the orthogonality as-

sumption, these vanishing points should be orthogonal and the corresponding vanishing directions.

$$\boldsymbol{d}_i = \frac{K^{-1}\boldsymbol{v}_i}{\|K^{-1}\boldsymbol{v}_i\|}, \quad i = 1, 2, 3,$$

should form an orthonormal basis in $\mathbb{R}^3$. If only two of the vanishing directions are approximately orthogonal (one of the detected vanishing points is false), then the third direction can be inferred from them using cross-product.

Due to the approximative nature of the vanishing point detection, matrix $D = [\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3]$ is not necessarily orthogonal. It can be shown [13, pp. 431–432] that the closest (in sense of Frobenius-norm) orthogonal matrix to D is given by

$$\hat{D} = UV^T,$$

where $D = U\Sigma V^T$ is a singular value decomposition of D.

Given two orthonormal bases $D_1 = [\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3]$ and $D_2 = [\boldsymbol{d}'_1, \boldsymbol{d}'_2, \boldsymbol{d}'_3]$, their relative rotation matrix is given by

$$R = D_2 D_1^T.$$

For any image, the vanishing direction matrix D can be chosen in multiple ways. The signs and the order of detected vanishing directions $\boldsymbol{d}_i$ are arbitrary and likely to change between frames. To avoid ambiguity in the vanishing direction matrices D_i we swap and change the signs of its columns in such way that it resembles as closely as possible to the vanishing direction matrix found in the previous frame. The details are as follows.

Assume that the matrix $D_1 = [\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3]$ consists of the vanishing directions found in the previous frame $i-1$ and matrix $D_2 = [\boldsymbol{d}'_1, \boldsymbol{d}'_2, \boldsymbol{d}'_3]$ is constructed from the vanishing directions found in the current frame i. To resolve the ambiguity in D_2 and to ensure that it is a proper rotation matrix, we associate it with the matrix D_1. To do this we compute a score matrix S as follows

$$S = D_2^T D_1 = \begin{bmatrix} \boldsymbol{d}_1'^T\boldsymbol{d}_1 & \boldsymbol{d}_1'^T\boldsymbol{d}_2 & \boldsymbol{d}_1'^T\boldsymbol{d}_3 \\ \boldsymbol{d}_2'^T\boldsymbol{d}_1 & \boldsymbol{d}_2'^T\boldsymbol{d}_2 & \boldsymbol{d}_2'^T\boldsymbol{d}_3 \\ \boldsymbol{d}_3'^T\boldsymbol{d}_1 & \boldsymbol{d}_3'^T\boldsymbol{d}_2 & \boldsymbol{d}_3'^T\boldsymbol{d}_3 \end{bmatrix}.$$

We want the order of columns of D_2 to be such that it corresponds to the order of columns in D_1. To determine the proper order we search for the maximum absolute values in the rows $i = 1, 2$ of S. For each row, if the maximum absolute value of row i is found in the column j, we swap the columns i and j in matrix D_2, or if $i = j$, do nothing. Also, if the sign of S_{ij} is negative, we multiply the column i of the re-arranged matrix D_2 with

-1. Changing the order and signs of the columns of D_2 does not change the fact that it is an orthogonal matrix. If the D matrix found in the first frame was associated with the identity matrix I_3 we know that it has determinant $\det D = +1$. The association process ensures that this property carries on in the subsequent frames.

4.2 Orientation Filtering

Previous sections described how we can obtain direct, but noisy measurements of the camera's orientation relative to the scene. To improve the orientation estimation, we use an extended Kalman filter (EKF). In addition to attenuating noise, filtering also makes it possible to continue camera orientation estimation when vanishing point detection fails to detect at least two orthogonal vanishing points.

For the purpose of filtering, we represent the camera orientation as a four-dimensional orientation quaternion. Orientation quaternions can readily be transformed into rotation matrices and Euler angles [5], but are more compact than rotation matrices and do not have the singularities and discontinuities that Euler angles can have.

For a moving camera, it is necessary that we also model higher order motion parameters such as velocity and acceleration. We include the estimate of the camera's angular velocity in the state vector and model angular accelerations as random process noise. The complete state vector of the camera consists of the camera orientation quaternion $\boldsymbol{q} \in \mathbb{R}^4$ and angular velocity vector $\boldsymbol{\omega} \in \mathbb{R}^3$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\omega} \end{bmatrix}.$$

For sequential motion filtering, we represent the continuous motion of the camera using a sequence of discrete states which we label using subscript k. We assume the timestep Δt between any two consecutive camera states to be constant and equal with the inverse of the camera's framerate. Our camera motion model assumes constant angular velocity with unknown random zero-mean Gaussian angular accelerations $\boldsymbol{\alpha}_k$ occurring in each timestep k

$$f(\boldsymbol{x}_k, \boldsymbol{n}) = \begin{bmatrix} \boldsymbol{q}_k * \boldsymbol{q}((\boldsymbol{\omega}_k + \boldsymbol{n})\Delta t) \\ \boldsymbol{\omega_k} + \boldsymbol{n} \end{bmatrix},$$

where the impulse of angular velocity $\boldsymbol{n} = \boldsymbol{\alpha}\Delta t \sim N(\boldsymbol{0}, Q)$ is used to model the uncertainty in the camera's motion. We use $\boldsymbol{q}(\cdot)$ to denote conversion from angle-axis representation of a rotation (3-vector) to orientation quater-

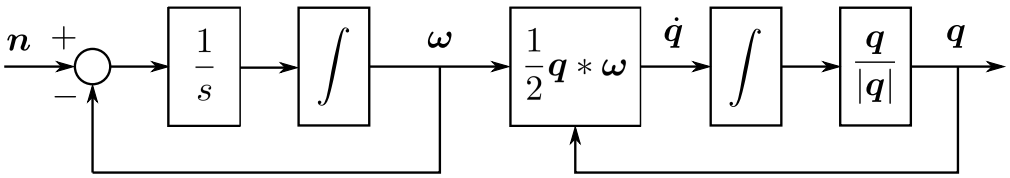

Figure 5: Process model of the orientation filtering using EKF.

nion. The measurement model h returns the quaternion part of the state vector plus noise,

$$h(\boldsymbol{x}_k) = H\boldsymbol{x}_k + \boldsymbol{r}_k = \boldsymbol{q}_k + \boldsymbol{r}_k,$$

where $\boldsymbol{r}_k \sim N(\boldsymbol{0}, R_k)$ and $H = [I_3, 0_{4\times 3}]$.

The standard EKF algorithm consists of *prediction* and *update* steps. The prediction step is

$$\begin{aligned} \boldsymbol{x}_k &= f(\boldsymbol{x}_{k-1}, \boldsymbol{0}) \\ P_k &= F_{k-1}P_{k-1}F_{k-1}^T + G_{k-1}QG_{k-1}^T \end{aligned}$$

where

$$\begin{aligned} F_{k-1} &= \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k-1},\boldsymbol{n}=\boldsymbol{0}} \\ G_{k-1} &= \left.\frac{\partial f}{\partial \boldsymbol{n}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k-1},\boldsymbol{n}=\boldsymbol{0}} \end{aligned}$$

Formulas for these matrices are lengthy and for the sake of brevity will not be presented here.

The update step is

$$\begin{aligned} \boldsymbol{z}_k &= \boldsymbol{y}_k - h(\boldsymbol{x}_k) \\ S_k &= H_k P_k H_k^T + R_k \\ K_k &= P_k H_k^T S_k^{-1} \\ \boldsymbol{x}_k &= \boldsymbol{x}_k + K_k \boldsymbol{z}_k \\ P_k &= (I - K_k H_k) P_k \end{aligned}$$

When vanishing point detection fails to produce a reasonable orientation, we skip the update step and just use the prediction.

5 Experimental Results

Experiments were conducted using a Nokia N900 mobile phone. The N900 shares similar hardware with many other modern smartphones: it has 5

megapixel camera for taking still images, it can capture 720p video at 25 fps and it has internal inertial measurement unit (IMU) with 3-axis accelerometer, 3-axis gyroscope and magnetometer.

The calibration matrix K for the camera was determined using method similar to Zhang [23]. The calibration process involved taking multiple pictures of planar calibration pattern shown in Figure 6.

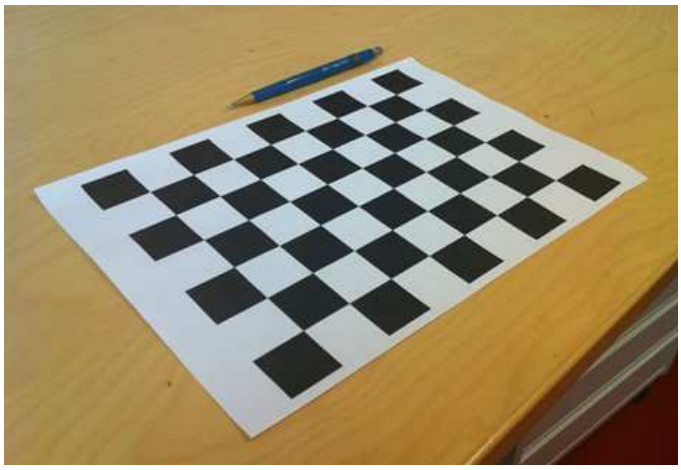

Figure 6: Calibration pattern

In our experiments the phone was used to capture handheld video in an indoor scene while direct orientation measurements from the IMU were recorded in the background. The video was afterwards divided into 680 individual frames. Line segments and vanishing points were detected in each frame offline using a desktop computer and our proposed method was used to track the camera's orientation through the whole image sequence.

Videos taken with the phone had native resolution of 848x480 pixels but in order to speed up the line segment detection, all of the frames were scaled down by 50% before processing. Line segment detection code was written in C-language, RANSAC-based line clustering, orientation computation and filtering were written as unoptimized MATLAB code. Line segment detection using detector [9] was clearly the bottleneck of the algorithm in terms of speed. Using 2.27 GHz dual core desktop computer we were able to process the test video with average rate of 20 fps which was very close to the native framerate of the camera (25 fps). The resulting orientation estimates were similar to those of the IMU (Figure 7).

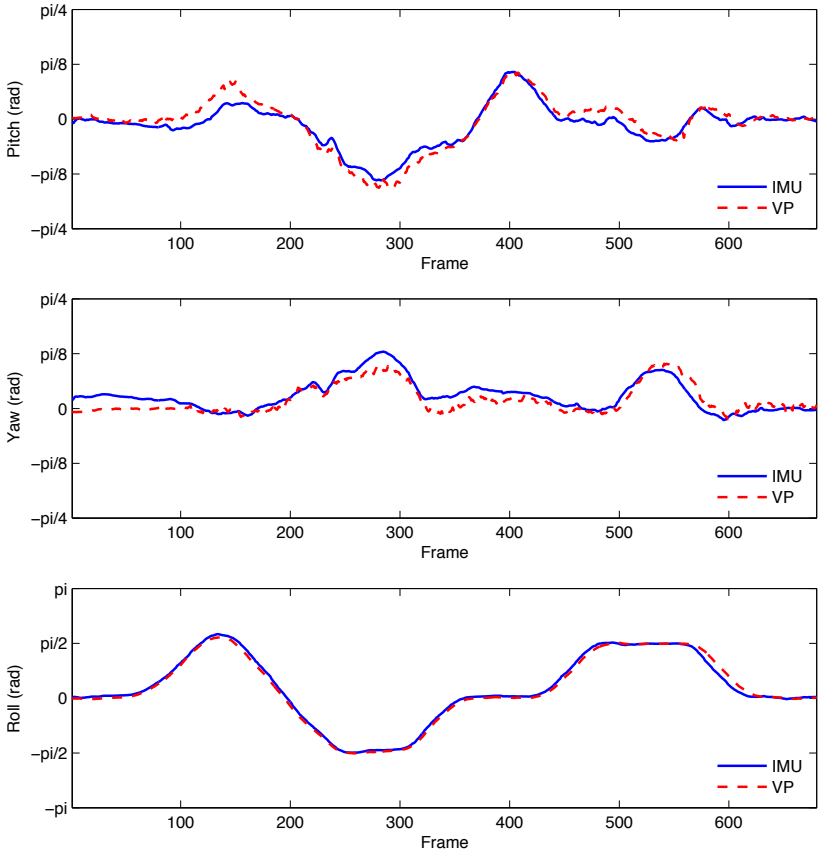

Figure 7: Results from an experiment with handheld camera. Dashed red line represents rotation angles measured using our proposed method, the solid blue line is the output of the built-in IMU.

6 Conclusions

We have presented a framework for a 3-axis gyroscope that works by tracking vanishing points detected in individual frames of a sequence of images taken by a monocular camera. Initial experiments indicate that the system can in principle track orientation in real time with an accuracy comparable to that of consumer-grade IMUs. Because the vision-based system is in principle driftless, it has good potential as a component of an integrated navigation system.

The weakest link in our camera gyroscope system is line detection. First, the images need to contain sufficient line features to infer vanishing points, and so the system can be expected to work well in well-lit man-made environments, such as indoors and in urban outdoor settings, but not so well

elsewhere. Lines are difficult to detect in images that are blurred by rapid camera motion; in this case the motion-blur based camera gyroscope of Klein and Drummond [15] could be a good alternative or complement. Finally, as line detection is the most computationally demanding stage of our system, the most important area for further work is the development of faster algorithms for vanishing point detection in an image stream.

Acknowledgements

This work was partly financed by Nokia corporation.

References

[1] M. Antone and S. Teller. Automatic recovery of relative camera rotations for urban scenes. In *Proceedings of IEEE Conference of Computer Vision and Pattern Recognition (CVPR'00)*, pages 282–289, Hilton Head, SC, USA, 2000.

[2] J. Burns, A. Hanson, and E. Riseman. Extracting straight lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):425–455, 1986.

[3] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4:127–140, 1990.

[4] R. Cipolla and E. Boyer. 3D model acquisition from uncalibrated images. In *Proceedings of the 1998 IAPR Workshop on Machine Vision Applications (MVA'98)*, pages 559–568, Chiba, Japan, 1998.

[5] J. Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. Technical report, Stanford University, Palo Alto, CA, 2006.

[6] R. Duda and P. Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15:11–15, 1972.

[7] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[8] A. Gallagher. Using vanishing points to correct camera rotation in images. In *Proceedings of the Second Canadian Conference on Computer and Robot Vision (CRV'05)*, pages 460–467, Victoria, Canada, 2005.

[9] R. Gioi, J. Jakubowicz, J. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, second edition, 2003.

[11] B. He and Y. Li. Camera calibration from vanishing points in a vision system. *Optics and Laser Technology*, 40(3):555–561, 2008.

[12] F. Heuvel. Vanishing point detection for architectural photogrammetry. *International Archives of Photogrammetry and Remote Sensing*, 32(5):652–659, 1998.

[13] R. Horn and C. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[14] C. Kessler, C. Ascher, N. Frietsch, M. Weinmann, and G. Trommer. Vision-based attitude estimation for indoor navigation using vanishing points and lines. *IEEE ION/PLANS 2010*, pages 310–318, 2010.

[15] G. Klein and T. Drummond. A single-frame visual gyroscope. In *Proceedings of British Machine Vision Conference (BMVC'05)*, volume 2, pages 529–538, Oxford, 2005.

[16] J. Košecká and W. Zhang. Video compass. In *Proceedings of European Conference on Computer Vision (ECCV'02)*, pages 657–673, 2002.

[17] Y. Lee, C. Nam, K. Lee, Y. Li, S. Yeon, and N. Doh. VPass: Algorithmic compass using vanishing points in indoor environments. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'09)*, pages 936–941, St. Louis, MO, USA, 2009.

[18] J. Lobo and J. Dias. Vision and inertial sensor cooperation using gravity as a vertical reference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1597–1608, 2003.

[19] F. Mirzae. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5), 2005.

[20] J. Montiel and A. Zisserman. Automated architectural acquisition from a camera undergoing a planar motion. In *Proceedings of International Symposium on Virtual and Augmented Architecture (VAA'01)*, pages 207–218, Dublin, Ireland, 2001.

[21] C. Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 20(9–10):647–655, 2002.

[22] S. Roumeliotis, A. Johnson, and J. Montgomery. Augmenting inertial navigation with image-based motion estimation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'02)*, pages 4326–4333, Washington D.C., USA, 2002.

[23] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, pages 666–673, Corfu, Greece, 1999.