Title

Geometry of Vanishing Points and its Application to External Calibration and Realtime Pose Estimation

Author

Kim, ZuWhan

Publication Date

2006-07-01

Institute of Transportation Studies
University of California at Berkeley

Geometry of Vanishing Points and its Application to External Calibration and Realtime Pose Estimation

ZuWhan Kim

RESEARCH REPORT
UCB-ITS-RR-2006-5

July 2006
ISSN 0192 4095

Geometry of Vanishing Points and its Application to External Calibration and Realtime Pose Estimation

ZuWhan Kim
Institute of Transportation Studies
University of California, Berkeley, CA, USA
zuwhan@berkeley.edu

Abstract

Vanishing points of an image contain important information for camera calibration. Various calibration techniques have been introduced using the properties of vanishing points to find intrinsic and extrinsic calibration parameters. This paper revisits the vanishing points geometry and suggests a simple extrinsic parameter estimation algorithm which uses a single rectangle. The comparison with the Camera Calibration Toolbox for Matlab® shows that the proposed algorithm is highly competitive. The suggested technique is also applied to a realtime pose estimation for an unmanned air vehicle's navigation in an urban environment. We present a realtime vanishing point extraction algorithm and a pose estimation procedure. The experimental result on a real flight video clip is presented.

1. Introduction

When parallel lines are projected to an image coordinates, they meet at a single vanishing point. Vanishing points have useful information for camera calibration. For example, both intrinsic and extrinsic camera parameters can be recovered using three orthogonal vanishing points on an image. The image center (or the principal point) is the orthocenter of the triangle formed by the three vanishing points [7]. Various camera calibration techniques have been introduced based on such properties, [3], [8], [4], and used in practice, for example, in *Calibration Toolbox for Matlab®* [1] (the Toolbox, hereafter) for initial estimation.

Most of the previous vanishing point-based calibration has been focused on accurately extracting interinsic calibration parameters because, 1) extracting extrinsic calibration parameters is an easy, solved problem, and 2) the internal and the external calibrations do not have to be done separately in many applications. Therefore, a very simple extrinsic calibration approach presented in this paper has been overlooked. In many robotics applications (especially in robotics research), users need to move the camera mounts quite often, and need to perform the external camera calibration (with respect to another camera for stereo, or another sensor for sensor fusion) whenever the camera mount is moved. For example, when a camera is installed on the wings of an unmanned air vehicle (UAV), it has to be taken off every time when the UAV is maintained because the wings need to be taken off. The intrinsic camera parameters do not need to be extracted in every such practice. Instead, a simple external calibration algorithm with a simple calibration grid will be very useful in such cases.

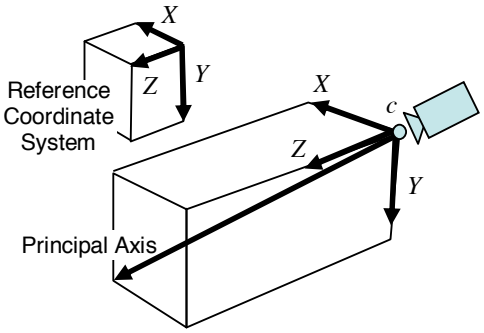
More importantly, such a calibration technique can be useful for realtime robot navigation in an urban environment, where many parallel lines are available. While vision-based robot navigation requires accurate pose (attitude) estimates, most sensors are based on a relative acceleration (open-loop estimation) and the error accumulates over time. Vanishing points-based estimation is uniquely useful because it gives absolute pose estimates with respect to the environment, where the errors are not accumulated over time.

In this paper, we revisit the vanishing point geometry (Section 2) and compare a simple external calibration algorithm based on it with a state-of-the-art calibration tehchnique (Section 3). In Section 4, we present its application to realtime pose estimation for uban robot navigation, and the conclusion is presented in Section 5.

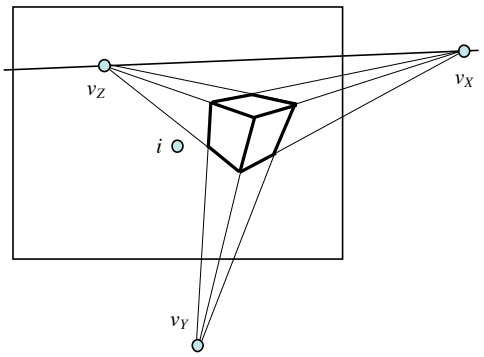
2. Revisiting the Vanishing Point Geometry

We assume that the intrinsic camera parameters are known. Therefore, we only deal with the coordinate conversion between the world coordinate (the *reference coordinate*) and the camera coordinate systems. Uppercase letters represent a point or a vector of the reference coordinate system and lowercase letters for the camera coordinate systems.

Camera geometry for vanishing points are illustrated in Figure 1. The three orthogonal vanishing points (v_X, v_X, and v_Z in Figure 1b) for the reference coordinate system in the bottom figure are the intersections of $\vec{X}$, $\vec{Y}$, and $\vec{Z}$ (originated from the principal point c) with the image plane ($z = 1$). In other words, $\vec{v_X}$, $\vec{v_Y}$, and $\vec{v_Z}$ (with $z = 1$) are the axes of the reference coordinate system.



(a)



(b)

Figure 1: Geometry of vanishing points: (a) camera geometry; and (b) geometry on the image plane ($z = 1$). The three orthogonal vanishing points (v_X, v_Y, and v_Z) for the reference coordinate system (shown in the bottom figure) are the intersections of $\vec{X}$, $\vec{Y}$, and $\vec{Z}$ (originated from the principal point c) with the image plane.

In fact, such a derivation loses generality especially when one or more axes of the reference coordinate system is parallel to the image plane. For example, when the camera is looking straightly forward, $\vec{X}$ and $\vec{Y}$ never intersects the image plane. Even when it is not perfectly aligned it can still cause numerical errors (both in the vanishing point estimation and the coordinate conversion) when an axis is roughly aligned with the image plane. Therefore, for both the vanishing point estimation and the coordinate conversion, we consider the following three cases:

Center: When a reference coordinate axis is roughly aligned with the z axis of the camera coordinate, then the axis vector is $(x_0, y_0, 1)^T$ in the camera coordinate system.

Vertical: When a reference coordinate axis is roughly aligned with the y axis of the camera coordinate, $|y_0|$ can be extremely large. Therefore, we use x_0/y_0 and $1/y_0$ instead of x_0 and y_0. Then, the axis vector is $(x_0/y_0, 1, 1/y_0)^T$ in the camera coordinate system.

Horizontal: When a reference coordinate axis is roughly aligned with the x axis of the camera coordinate, $|x_0|$ can be extremely large. Therefore, we use $1/x_0$ and y_0/x_0 to represent the vanishing point. Then, the axis vector is $(1, y_0/x_0, 1/x_0^T)$ in the camera coordinate system.

Given two projected parallel lines $\mathbf{x_1}s + \mathbf{v_1}$ and $\mathbf{x_2}t + \mathbf{v_2}$ ($\mathbf{v_1}$ and $\mathbf{v_2}$ are unit vectors), their vanishing point is estimated by the following equations:

$m_z = \mathbf{v_1} \cdot \mathbf{v_2}^{\perp}$, $a = (\mathbf{x2} - \mathbf{x1}) \cdot \mathbf{v_2}^{\perp}$, and $(m_x, m_y) = m_z\mathbf{x_1} + a\mathbf{v_1}$, where $(x, y)^{\perp} \equiv (-y, x)$.

Center: When $|m_z| = \max(|m_x|, |m_y|, |m_z|)$, $x_0 = m_x/m_z, y_0 = m_y/m_z$.

Vertical: When $|m_y| = \max(|m_x|, |m_y|, |m_z|)$, $1/y_0 = m_z/m_y, x_0/y_0 = m_x/m_y$.

Horizontal: When $|m_x| = \max(|m_x|, |m_y|, |m_z|)$, $1/x_0 = m_z/m_x, y_0/x_0 = m_y/m_x$.

3. Camera Calibration with a Single Rectangle

The vanishing point geometry described in the previous section suggests a very simple external calibration algorithm. When we know the intrinsic calibration parameters, all we need to know for the external calibration are the vanishing points corresponding to any two axes of the reference coordinate system. The other axis is simply the cross-product of the two.

For example, when we assume that the two orthogonal vanishing points are $\vec{v_X}$ and $\vec{v_Y}$ (Figure 2), $\vec{v_Z} = \vec{v_X} \times \vec{v_Y}$, and the rotation matrix

$$\mathbf{R} = (\mathbf{v_X v_Y v_Z}), \tag{1}$$

where

$$\mathbf{x_C} = \mathbf{R}\mathbf{X_W} + \mathbf{T}, \tag{2}$$

$\mathbf{x_C}$ and $\mathbf{X_W}$ are the world and the camera coordinates of a point (homogeneous transform), and $\mathbf{T}$ is the translation.

Note that we do not need to know the dimension of the calibration grid (the rectangle) to find the camera orientation. The dimension of the rectangle is used to find the translation. To find the translation, we need to know the world coordinates of at least two points. Given a point correspondences, Eq 2 gives two linear equations on $\mathbf{T}$ and we have three unknowns for $\mathbf{T}$. That means that we only need to know either the width or the height of the rectangle. For example, when we know the width, w, we can simply set the world coordinates of the upper left corner $(0, 0, 0)$ and the upper right corner $(w, 0, 0)$. For better estimation,

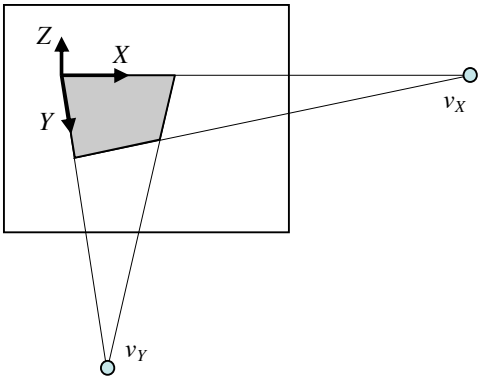


Figure 2: Camera calibration with a single rectangle. A set of two orthogonal vanishing points give a camera rotation matrix.

we can apply a least-square optimization on $\mathbf{T}$ given all the available correspondences:

$$\mathbf{T} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}, \tag{3}$$

where

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & -x_{C1} \\ 0 & 1 & -y_{C1} \\ \vdots & \vdots & \vdots \\ 1 & 0 & -x_{Cn} \\ 0 & 1 & -y_{Cn} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} x_{C1} \cdot z_{R1} - x_{R1} \\ y_{C1} \cdot z_{R1} - y_{R1} \\ \vdots \\ x_{Cn} \cdot z_{Rn} - x_{Rn} \\ y_{Cn} \cdot z_{Rn} - y_{Rn} \end{pmatrix}, \tag{4}$$

and $(x_R, y_R, z_R)^T = \mathbf{R}\mathbf{X}_\mathbf{W}$.

An experimental comparison of this algorithm with the Toolbox is shown in Figure 3. The experimental datasets were synthetically generated from a 5×5 grid of fixed dimensions (no camera distortion was modeled). For each dataset, ten random rotation and translation matrices were generated (the translation matrices were carefully selected that the calibration grids are "visible"), and a set of corresponding image coordinates were calculated. A discretization noise (of $1/5$ pixel) is added (most of the corner detectors have a good subpixel accuracy of about $1/10$ pixel).

For each dataset, the Toolbox is applied to find the intrinsic and extrinsic camera parameters. For the proposed algorithm, only 4 point-correspondences for the upper-left, the upper-right, the lower-left, and the lower-right corners were used along with the intrinsic calibration parameters extracted from the Toolbox. For each image an average reprojection error is calculated for all 5×5 points (for both of the methods).

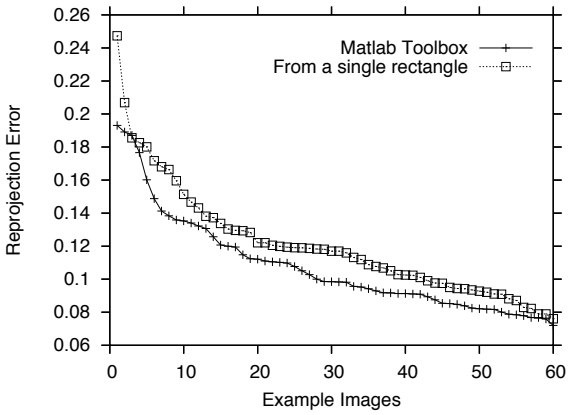


Figure 3: The vanishing point-based algorithm was compared with the *Calibration Toolbox for Matlab*®. The Toolbox used all 5×5 points while the proposed approach only used the 4 corner points. For both of the methods, average reprojection errors of all 5×5 points were calculated. The graph shows the sorted errors for all the examples. The Toolbox is slightly better but the difference is very small compared to the variations of the errors within a single method.

The result is very promising. Although the Toolbox is slightly better but the difference is much smaller than the variation of the errors within a single method. The reason behind its surprisingly high performance is that the vanishing point position estimate is more affected by the angles of the lines than by each point's absolute position, and the angles of the lines can robustly be estimated in most cases.

4. Realtime Pose Estimation of a UAV

In this section, we present an application of the proposed calibration algorithm to a realtime pose (attitude) estimation. In an urban environment, two or three orthogonal vanishing points (one vertical and one or two parallel to the ground) are available most of the time, so that the proposed approach can be useful.

One of the key steps is to find a set of orthogonal vanishing points with a small computation. A line detection procedure is a crucial preprocessing step to robustly find the vanishing points. In Section 4.1, our line detection algorithm is presented. The vanishing point extraction algorithm and the pose estimation algorithm is presented in Section 4.2. Finally, the experimental result is shown in Section 4.3.

4.1. Line Detection

An example image from an urban flight video is shown in Figure 4. An accurate extraction of the line segments is a crucial step for a reliable detection of vanishing points.

Figure 4: An example image from an urban flight video.

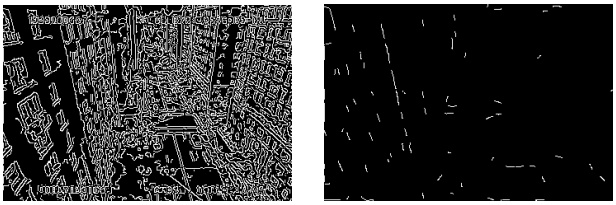


Figure 5: Edges extracted by a Canny detector (left) and lines detected by Guru et al.'s algorithm [5] (right). Guru et al's algorithm has two major problems: it does not disconnect L-junctions and the performance is degraded when there is more than one line present in the small eigen value computation window (7×7, in the above example).

Guru et al. [5] introduced a simple and robust line detection algorithm based on a *small eigenvalue* analysis. For each edge pixel (computed by a Canny detector [2]), the small eigen value, λ of the covariance matrix of S is computed by

$$\lambda = \frac{1}{2}\left[c_{11} + c_{22} - \sqrt{(c_{11} - c_{22})^2 + 4c_{12}^2}\right], \quad (5)$$

where

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

is the covariance matrix of the neighbor window. Then, each pixel is thresholded based on the smallest small eigen value of the windows it belongs to.

Figure 5 shows the line detection result by Guru et al.'s algorithm. 7×7 windows (the smallest suggested window size) were applied to compute the small eigen values. Guru et al.'s algorithm has two major problems. First, it does not disconnect L-junctions, which causes problems in accurately extracting line segments. In addition, its performance is significantly degraded when there is more than one line present in the small eigen value computation window. Therefore, we augmented the algorithm:

- To handle the case of more than one line in a window, only the edge pixels, which have linear paths to the center point, are used to calculate the small eigen value (instead of using all the edge pixels in the window).
- To prevent L-junctions from being connected, we use only the small eigen value of the window where the target pixel is in the center instead of all the windows the target pixel belongs to.
- In addition, the threshold value was raised (to 0.3) because the above fixes eliminates many false alarms.

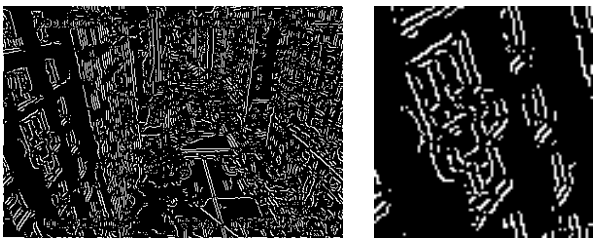


Figure 6: Lines detected by the proposed algorithm. Many more lines are detected and the L-junctions are disconnected (see the elarged image to the right).

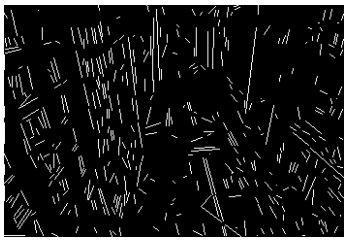


Figure 7: Fitted lines of 10 pixels or longer.

The resulting line pixels are shown in Figure 6. We see that many more lines are detected while L-junctions are disconnected. Once the line pixels are detected, a connected component analysis is applied to find the line segments. The resulting line segments of 10 pixels or longer are shown in Figure 7.

4.2. Vanishing Point Extraction and Pose Estimation

The next step is to extract vanishing points from the line segments. The vanishing point extraction needs to be fast and robust. We apply a RANSAC-style algorithm suggested in [6] to detect vanishing points. For all possible line pairs of, say, 25 pixels or longer, vanishing point hypotheses are generated. Each vanishing point hypothesis is tested by checking its compatibility with all the other line segments. The number of compatible lines (of one pixel or less errors) is used to score the hypotheses.

Once a highest scoring vanishing point hypothesis is selected, all the line segments that are compatible with the

Figure 8: Extracted vanishing point pairs (a white line segment connects the two vanishing points).

hypothesis are removed, and another vanishing point is extracted from the remaining line segments. A small number of vanishing point hypotheses (three in our implementation) is extracted for each frame. For each vanishing point hypotheses, the corresponding axis vector is calculated (cf. Section 2). Among all the possible vanishing point pairs, the most orthogonal one (where the dot-product of the axis vectors is the smallest) is chosen as a final vanishing point pair. Example vanishing point pairs are shown in Figure 8. For each image, the line segment connecting the two vanishing points is drawn, but only one vanishing point is visible.

The final vanishing point pair suggests two of the reference frame axes. The third axis is estimated by the cross-product of the two. From the axis vectors, the rotation matrix is estimated by Eq 1.

4.3. Experimental Results

An experiment was performed on a real flight video clip of total 566 frames (30 fps). In 437 frames of the 566 frames, two or more orthogonal vanishing points are found. The total computation was about 50ms per frame on an Intel® Pentium® M 2GHz processor. Currently, a Pentium® M processor of up to 1.6GHz is available for embedded computing (PC/104).

Since there is no ground truth available, we rotated the image to the reference coordinate system for a visual validation of the result. Example rotated images are shown in Figure 9. Only a few noticeable false alarms and misclassification of the axes due to an excessive roll angle (the third example in the figure) were reported. Out of the 437 frames, only 9 of them were totally misplaced and $5 \sim 10$ were roughly correct but noticeably misaligned. Eight of them had axis misclassification. The rest of them (over 400 frames) were reliably estimated. When we do not count the axis misclassification case, the detection rate was 73% and the false alarm rate was only $2 \sim 3\%$.

5. Summary and Conclusion

We revisited the vanishing point geometry and showed an experiment of comparing a simple vanishing-point-based algorithm to a state-of-the-art one which suggested that the former is highly competitive to the latter. We applied it to a realtime pose estimation for urban robot navigation. The experimental result was promising. The vehicle pose was reliably estimated in realtime for over 70% of the total video with a small false alarm ratio. The future work is to combine the vanishing-point-based pose estimates with those of an ego-motion estimation algorithm or gyroscope sensor readings for more reliable results.

References

[1] J.-Y. Bouguet. Camera calibration toolbox for Matlab®. http://www.vision.caltech.edu/bouguetj/calib_doc.

[2] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[3] B. Caprile and V. Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–140, 1990.

[4] L. Grammatikopoulos, G. Karras, and E. Petsa. Camera calibration combining images with two vanishing points. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 35(5):99–104, 2004.

[5] D. S. Guru, B. H. Shekar, and P. Nagabhushan. A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognition Letter*, 25(1):1–13, 2004.

[6] Z. Kim. Realtime road detection by learning from one example. In *IEEE Workshop on Applications of Computer Vision*, pages 455–460, 2005.

[7] E. L. Merritt. *Analytical Photogrammetry*. Pitman Publ. Co., New York, 1958.

[8] L.-L. Wang and W.-H. Tsai. Camera calibration by vanishing lines for 3-d computer vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(4):370–376, 1991.

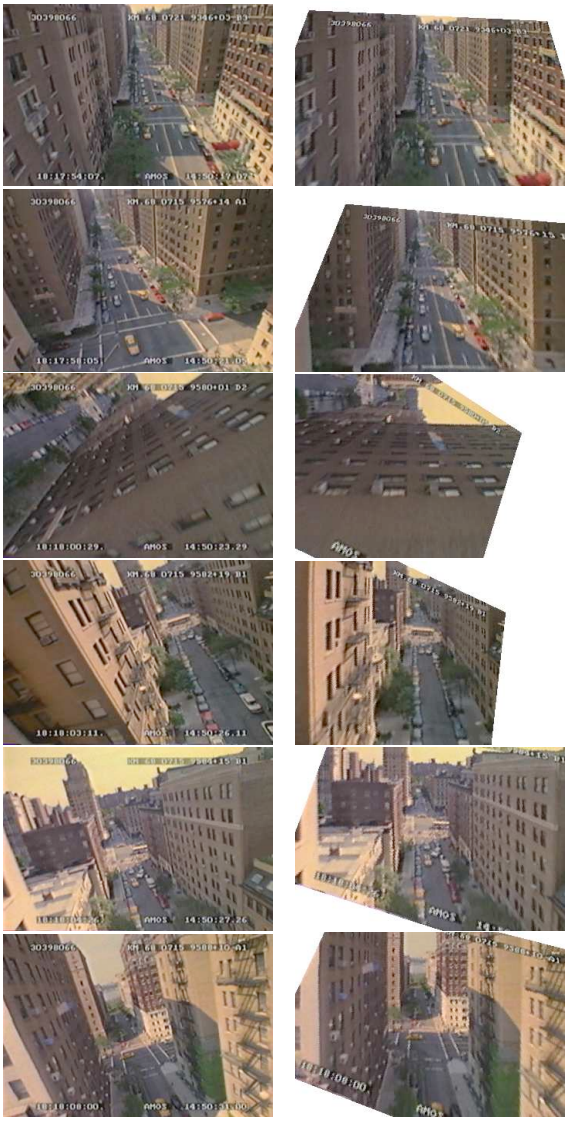


Figure 9: Example results. The images were rotated to the reference coordinate system for visual validation.