Rotation Estimation of a Complete Omnidirectional Camera using Line Features

Sarthak Pathak* Alessandro Moro* Atsushi Yamashita* Hajime Asama*

1. Introduction

Rotation estimation with cameras is important for flying robot navigation. Usually, it is performed using a single camera by keeping track of some strong features such as corners or lines within the environment and observing their motion. However, with a normal camera, the field of view is quite limited and this can lead to an error because information is obtained from a small section of the scene. Also, features can leave the frame of the camera and can no longer be used for estimating the rotation. If we make use of an *Omnidirectional* camera which has a wide field of view (up to 4π steradians), we can overcome these issues.

There has been much similar research in the past about estimating the motion and 3D structure of the environment of a moving omnidirectional camera. For example, [2] used a catadioptric camera to perform visual servoing for human-robot interaction. Back in 2004, [3] presented a method to use catadioptric cameras with optical flow estimation to provide visual odometry for rover. Similarly, [9] showed a very accurate technique for control of a vehicle using 1-point RANSAC, while [5] also talked about Structure from Motion in textureless environments with omnidirectional images using parallel lines. However, all these methods are mostly suitable for ground robots while flying robots have more complicated motion. The method in [4] showed how to determine the altitude, roll, and pitch of a flying robot by detecting parallel lines in the catadioptric omnidirectional image. The technique used in [1] described a way to obtain the yaw angle using vanishing points in catadioptric images. These methods assumed the existence of vanishing points and parallel lines, although they may not always be present in every scene. Meanwhile, [6] talked about estimating the translation using only lines in images obtained by multiple central cameras. However, the line matching was done manually. Furthermore, most of the described methods use catadioptric cameras and are not truly 'Omnidirectional' as they do not deal with the *complete* 4π steradians. They still retain the problem of features leaving the frame of the camera and thus being rendered of no use. In contrast, [8] used complete omnidirectional images, wherein the authors introduced and evaluated several reprojection error models for solving motion estimation, and matched keypoints in order to densely reconstruct a scene. However, the technique seems to be quite computationally heavy and not suited for a realtime robotics platform. Thus, to build on the plethora of previous work, our aim is to formulate a rotation estimation technique using the information present in all directions of the robot so that the accuracy and robustness is maintained. We have decided to use line features for this approach as they provide a more stable matching and localization as compared to point features. Moreover, they are not affected much by occlusion because we make use of the entire 4π steradians to perform feature detection and matching, as we will show in later sections.

Fig.1 The Ricoh Theta camera (a) with its side view (b) showing two Fisheye cameras with 4π steradians (sr) field of view (solid angle).

To obtain the omnidirectional image, we can use two oppositely pointed *fisheye* cameras (each of which can provide a 180°, or 2π steradians field of view) and stitch them into a complete image of 4π steradians. For this work, we used the Ricoh Theta camera which already has perfectly aligned cameras (Fig. 1). The basic outline of the algorithm involves obtaining a pair of fisheye images at two different times, stitching the images to form an omnidirectional image, estimating and matching line features in the stitched image, and finally determining the rotation matrix. As mentioned before, using line features has several advantages such as a more stable matching and accurate localization of camera position (as a larger number of points are used) without an increase in processing time. Furthermore, since the number of line features detected and needed is usually quite less, we can perform a full brute force search across all the lines without reducing the search area or other such limitations imposed when point features are used. Thus, it can be

*Department of Precision Engineering
The University of Tokyo

made stable even for large changes and complicated motion patterns.

We capture such an omnidirectional image and treat it as a spherical image. We base all our processing on this spherical image, thus avoiding the distortions that usually affect normal fisheye images. The two other major advantages of this are that the problem of features exiting the frame is eliminated, and secondly, since no information is going out of the frame, constructing a matching tecnhique also becomes easier.

In contrast with [6], we attempt to perform line matching in omnidirectional images by taking advantage of the fact that the spherical image is continuous in all directions. If we project a line segment in this spherical image to infinity in both directions, it will meet itself around the sphere. This ensures that we don't need to locate the end points of the line segments in order to match them. (Using a similar technique, [7] estimated the visual odometry of a ground robot.)

2. Obtaining Complete Omnidirectional Images

The first step is to obtain the complete omnidirectional image. We make use of two fisheye cameras (each of which can capture 180° or 2π steradians). The camera shown in Fig. 1, is mounted on a flying robot. Images are captured at the same time from each camera. Each fisheye image (Fig. 2) forms a half sphere when reprojected on the surface of the unit sphere, according to the *Omnidirectional Camera Model* [10]. The two halves can be combined together to obtain the complete omnidirectional image.

We used the omnidirectional camera toolbox [10] to calibrate each fisheye camera. After obtaining the calibration parameters, we can project each point on the fisheye image to its location on the unit sphere, as shown in Fig. 3. Once we obtain both half spheres, we can combine them to form the complete spherical image. To show this image on paper, it has been unrolled in an *Equirectangular Projection* (Fig. 4).

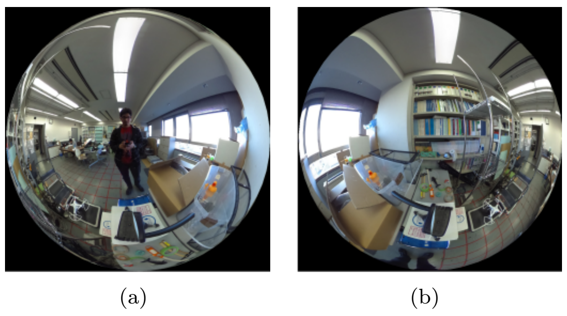


Fig.2 Individual Fisheye Images from the oppositely placed Fisheye cameras

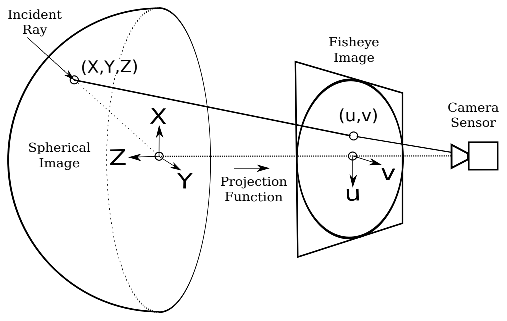


Fig.3 Omnidirectional Camera Model: Projecting a point on the unit half sphere from one fisheye image. Two of these placed opposite to each other can give us the complete sphere.

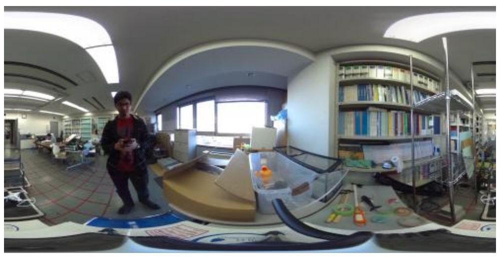


Fig.4 Spherical Image constructed from the individual fisheye images from Fig. 2 (Equirectangular Projection)

3. Line Detection and Matching

As mentioned before, line features are more stable as compared to point features for matching and localization. We use the popular probabilistic hough transform directly on the spherical image to detect lines. Before that, we have to first look at how real world line segments are projected on the spherical image.

As can be seen in Fig. 5, a line in the real world is projected as part of a *Great Circle* on the Spherical Image. Each great circle has a unique normal vector from the center of the sphere and can be described by its direction cosines $\cos(a)$, $\cos(b)$, and $\cos(c)$ (where a, b, and c are the respective euler angles).

The motion of any flying robot is such that it needs to rotate in order to change its direction of translation. When the flying robot is rotating mid air, these great circles and the associated normal vectors change their orientation. (Note that this motion is not restricted to rotation. Translation will cause the distances between the lines and the center of the camera to change but will affect the orientation of the great circle only slightly, hence it can be ignored.) If we are able to obtain the rotation of two of these great circles, we can estimate the complete rotation of the flying robot (Fig. 6).

In order to build the accumulator for the hough

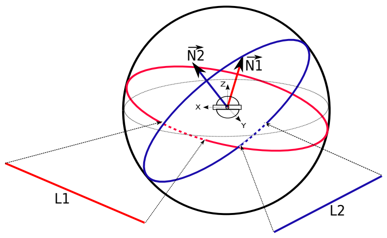


Fig.5 Projection of a line on the Spherical Image. L1 and L2 are the real world lines projected as great circles on the sphere. *N1* and *N2* are the respective normal vectors from the centers of the circles

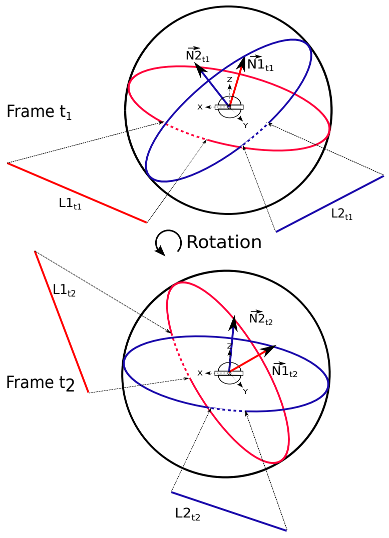


Fig.6 Effect of rotation on the line projections in two frames (t1 and t2). Lines $L1$, $L2$ and their normals $N1$, $N2$ rotate when the camera rotates (with respect to the camera's frame of reference.)

transform, every line can be parameterized by the normal vector of its great circle according to the equation $X\cos(a)+Y\cos(b)+Z\cos(c)=0$, where X, Y, and Z are the coordinates of the pixels on the surface of the sphere. We know that $\cos^2(a)+\cos^2(b)+\cos^2(c)=1$. Thus, we only need two of these angles to obtain the third one and the hough parameterization of the line can be performed with only two parameters, a and b. Since these parameters are angles that have the range $[0°, 180°]$, the hough accumulator has only 180×180 cells which makes processing efficient. (Usually, the size of the accumulator array is a major limitation in the use of the hough transform.) Thus, the probabilistic hough transform is performed directly on the sphere using its Canny edges to generate votes. Fig. 7 shows the result of this operation.

Since it is a complete spherical image, each line segment extended in both directions forms a great circle as shown in Fig. 5. We can use all the points on this great circle for matching. Thus, we can eliminate the usual line matching problem - detecting the line segment ends. We only need to match the pixels on the circle. Since it is difficult to find a particular starting point for ordering the data on the circle, we have to make use of statistical, order independent methods. Taking the histogram of the intensities of all the pixels on the circle provides a good feature vector for the line and matching these vectors using the *L2-norm* which works quite well (Fig. 8). Moreover, since we use the entire 360° circle, occlusion becomes a non-issue for the matching.

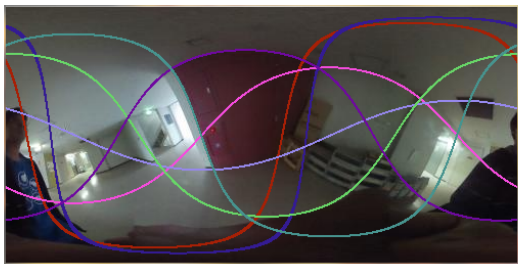
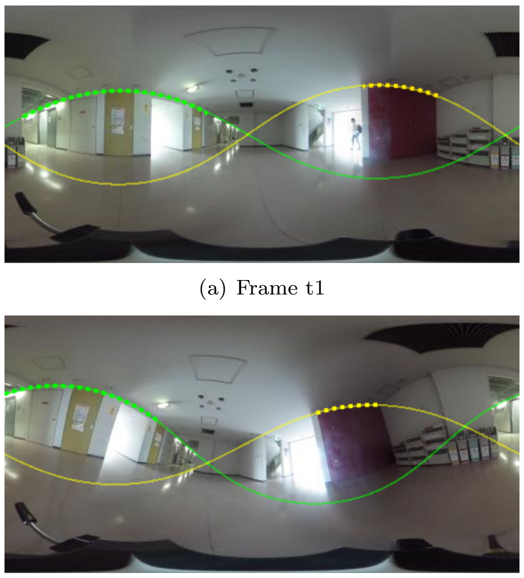


Fig.7 Detection of lines in the omnidirectional image using the probabilistic hough transform (equirectangular projection). The great circles subtended by each line segment are shown in the figure.



(a) Frame t1

(b) Frame t2

Fig.8 Line Matching between Frames t1 (a) and t2 (b) using Histograms. The dotted areas show the line segments corresponding to the great circles. The camera was manually rotated by 30° (yaw) and 15° (pitch).

4. Rotation Estimation

As mentioned earlier, the rotation can be estimated from two line matches. This requires a simple transformation. The normal vectors $\vec{N}1$ and $\vec{N}2$ (for lines L1 and L2) are taken to be of unit length. Thus, $\vec{N}1 + \vec{N}2$ and $\vec{N}1 - \vec{N}2$ are always perpendicular, and along with $\vec{N}1 \times \vec{N}2$, they form an orthogonal basis. Using this orthogonal basis, we can easily calculate the angles between corresponding axes of the bases formed by the two pairs of lines in frames $t1$ and $t2$ (as shown in Fig. 9) and transform them to the robot frame of reference. Using this technique, the rotation between the two frames (Fig. 8) was calculated to be 28.7° of yaw and 13.5° of pitch, which is quite close to the groundtruth of 30° and 15° respectively. (More results in Table 1.)

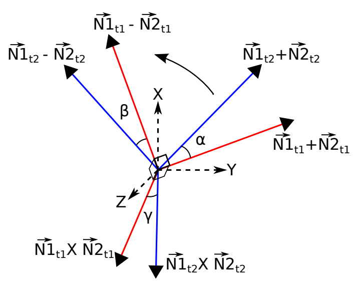
5. Conclusion

We have shown a method to extract the rotation for an omnidirectional camera, which can take advantage of features in all directions. We detect line segments (as they are more stable features) and match them efficiently using complete great circles projected on the spherical images. Using lines, we need a much lesser number of features (as opposed to using points). Hence, we can search completely over all the features without limiting the motion patterns or search space, thus making it stable even for complicated or large motions (30°,45°, etc.). In addition, the method has no assumptions about the environment other than the existence of straight line segments, and is fast enough for robot navigation (0.5s per omnidirectional image of 500×250 pixels). It can be made realtime by parallelizing the probabilistic hough transform.)

However, this is still a work in progress and the rotation will not be accurate because the orientation of the great circles is also slightly affected by large translations (several metres). Usually, a flying robot undergoes rapid rotation and slower translation. Thus, this method can work well if the frame rate is fast enough to prevent large translations (5-10 FPS). To be completely accurate, we must obtain the rotation and translation simultaneously, perhaps by utilizing vanishing points [1], and translation extraction as shown in [6]. We will try to extend this work along similar lines and formulate a real-time technique for complete, accurate motion estimation of a flying robot.

Table 1 Results of Rotation Estimation in 3 different scenarios of pitch and yaw. (Groundtruth in parenthesis)

Yaw	28.7° (30°)	21.6° (20°)	31.8° (30°)
Pitch	13.5° (15°)	46.3° (45°)	28.3° (30°)



$\vec{N1}_{t1} - \vec{N2}_{t1}$
$\vec{N1}_{t2} - \vec{N2}_{t2}$
$\vec{N1}_{t2}+\vec{N2}_{t2}$
X
β
α
$\vec{N1}_{t1}+\vec{N2}_{t1}$
Y
Z
γ
$\vec{N1}_{t1} X \vec{N2}_{t1}$
$\vec{N1}_{t2} X \vec{N2}_{t2}$

Fig.9 Rotation Estimation from line matches by forming two orthogonal bases

Acknowledgements

This work was in part supported by the SIP Program (Cross-ministerial Strategic Innovation Promotion Program).

References

[1] J.C. Bazin, C. Demonceaux, P. Vasseur, and I. Kweon. “Rotation estimation and vanishing point extraction by omnidirectional vision in urban environment”. *The International Journal of Robotics Research*, 31(1):63–81, 2012.

[2] P. Chang and M. Herbert. “Omnidirectional visual servoing for human-robot interaction”. *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3:1801–1807 vol.3, 1998.

[3] P. Corke, D. Strelow, and S. Singh. “Omnidirectional visual odometry for a planetary rover”. *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4:4007–4012 vol.4, 2004.

[4] C. Demonceaux, P. Vasseur, and C. Pégard. “UAV attitude computation by omnidirectional vision in urban environment”. *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 2017–2022, 2007.

[5] R. Kawanishi, A. Yamashita, T. Kaneko, and H. Asama. “Parallel line-based structure from motion by using omnidirectional camera in textureless scene”. *Advanced Robotics*, 27(1):19–32, 2013.

[6] S. Ly, C. Demonceaux, and P. Vasseur. “Translation estimation for single viewpoint cameras using lines”. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*, pages 1928–1933, 2010.

[7] O. Munteanu, R. Pronk, and A. Visser. “Visual odometry with the ricoh theta”. *Project Report, Universiteit van Amsterdam, February*, 2014.

[8] A. Pagani and D. Stricker. “Structure from Motion using full spherical panoramic cameras”. *Proceedings of the 2011 IEEE International Conference on Computer Vision Workshop*, pages 375–382, 2011.

[9] D. Scaramuzza, F. Fraundorfer, and R. Siegwart. “Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC”. *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, pages 4293–4299, 2009.

[10] D. Scaramuzza, A. Martinelli, and R. Siegwart. “A Toolbox for Easily Calibrating Omnidirectional Cameras”. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5695–5701, 2006.