Omnidirectional Egomotion Estimation From Back-projection Flow[*]

Omid Shakernia René Vidal Shankar Sastry
Department of Electrical Engineering & Computer Sciences, UC Berkeley
{omids,rvidal,sastry}@eecs.berkeley.edu

Abstract

The current state-of-the-art for egomotion estimation with omnidirectional cameras is to map the optical flow to the sphere and then apply egomotion algorithms for spherical projection. In this paper, we propose to back-project image points to a virtual curved retina that is intrinsic *to the geometry of the central panoramic camera, and compute the optical flow on this retina: the so-called* back-projection flow. *We show that well-known egomotion algorithms can be easily adapted to work with the back-projection flow. We present extensive simulation results showing that in the presence of noise, egomotion algorithms perform better by using back-projection flow when the camera translation is in the X-Y plane. Thus, the proposed method is preferable in applications where there is no Z-axis translation, such as ground robot navigation.*

1. Introduction

The panoramic field of view offered by omnidirectional cameras makes them ideal candidates for many vision-based mobile robot applications, such as autonomous navigation [17], localization [14], and formation control [16]. The egomotion estimation problem (the task of recovering the camera motion relative to the environment) is fundamental to most of these vision-based mobile robot applications.

The task of egomotion estimation typically consists of first estimating the *optical flow* (the 2D motion field in the image plane) and then extracting the 3D camera motion from the optical flow. Previously, [7] studied the problem of egomotion estimation using parabolic and hyperbolic catadioptric cameras by mapping image points to the sphere and mapping the optical flow to the sphere through the Jacobian of the transformation, and then applying well-known egomotion algorithms adapted for spherical projection. In [15] this approach was generalized by computing mapping of image points to the sphere and its Jacobian as a function of the parameters of the central panoramic projection model.

In this paper, we develop the notion of *back-projection flow* as a natural generalization of optical flow for central panoramic cameras. We show that the unified projection model for central panoramic cameras [5] can be considered as a projection onto a virtual curved retina that is *intrinsic* to the camera geometry. The so-called back-projection flow is obtained by lifting the optical flow from the image plane onto this retina. We show that egomotion algorithms [13] can be applied directly to this back-projection flow.

We present extensive simulation results comparing the performance of egomotion algorithms using back-projection flow and spherical flow under varying camera motions, central panoramic camera parameters, and noise levels in the optical flow. Our results show that in the presence of noise, egomotion algorithms perform better when using back-projection flow compared with spherical flow when the camera translation is mostly in the X-Y plane. Thus the proposed method is preferable in applications with small Z-axis translation, such as ground robot navigation.

2. Central Panoramic Imaging Model

A catadioptric realization of an omnidirectional camera combines a curved mirror and a lens. In [1], an entire class of catadioptric systems containing a single effective focal point is derived. A single effective focal point is necessary for the existence of epipolar geometry that is independent of the scene structure [12]. Camera systems that have a unique effective focal point are called *central panoramic cameras*.

It was shown in [5] that all central panoramic cameras can be modeled by a mapping of a 3D point onto a sphere followed by a projection onto the image plane from a point in the optical axis of the camera. By varying two parameters (ξ, m), one can model all catadioptric cameras that have a single effective viewpoint, *e.g.* parabolic mirror with orthographic lens, or hyperbolic mirror with perspective lens. The particular values of (ξ, m) in terms of the shape parameters of different types of mirrors are listed in [2].

According to the unified projection model [5], the image point $(x, y)^T$ of a 3D point $q = (X, Y, Z)^T$ obtained through a central panoramic camera with parameters (ξ, m) is given by:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{\xi + m}{-Z + \xi\sqrt{X^2 + Y^2 + Z^2}} \begin{bmatrix} s_x X \\ s_y Y \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \qquad (1)$$

[*]We thank the support of ONR grant N00014-00-1-0621.

1063-6919/03 $17.00 © 2003 IEEE

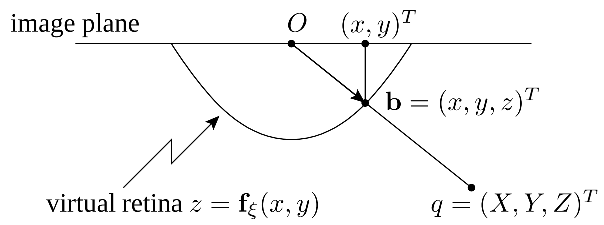

Figure 1: Showing the curved virtual retina in central panoramic projection and back-projection ray $\mathbf{b}$ associated with image point $(x,y)^T$.

where $0 \le \xi \le 1$, and (s_x, s_y) are scales that depend on the geometry of the mirror, the focal length and the aspect ratio of the lens, and $(c_x, c_y)^T$ is the mirror center.

Since central panoramic cameras for $\xi \neq 0$ can be easily calibrated from a single image of three lines [6, 2], in this paper, we assume that the camera has been calibrated, *i.e.* we know the parameters $(s_x, s_y, c_x, c_y, \xi, m)$. Therefore, without loss of generality, we consider the following *calibrated* central panoramic projection model:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{-Z + \xi\sqrt{X^2+Y^2+Z^2}} \begin{bmatrix} X \\ Y \end{bmatrix}, \qquad (2)$$

which is valid for $Z < 0$. It is direct to check that $\xi = 0$ corresponds to the case of perspective projection, and $\xi = 1$ corresponds to paracatadioptric projection (a catadioptric camera with a parabolic mirror and an orthographic lens).

3. Back-projection Flow

In this section, we introduce the notion of *back-projection flow*, which is the natural generalization of optical flow in the case of central panoramic cameras. We also review and discuss differences with an alternate approach [7, 15] which maps the optical flow to the sphere.

3.1. Mapping Optical Flow to a Curved Retina

Since central panoramic cameras have a unique effective focal point, one can efficiently compute the *back-projection ray* (a ray from the optical center in the direction of the 3D point being imaged) associated with each image point.

One may consider the central panoramic projection model in equation (2) as a simple projection onto an curved virtual retina whose shape depends on the parameter ξ. We define the *back-projection ray* as the *lifting* of the image point (x, y) onto this retina. That is, as shown in Figure 1, given an image $(x,y)^T$ of a 3D point $q = (X,Y,Z)^T$, define the back-projection rays as:

$$\mathbf{b} \triangleq (x,y,z)^T, \qquad (3)$$

where $z = \mathbf{f}_\xi(x,y)$ is the height of the virtual retina. We construct $\mathbf{f}_\xi(x,y)$ in order to re-write the central panoramic projection model in equation (2) as a simple scaling:

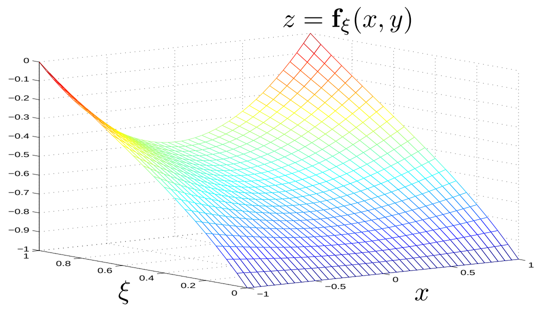

Figure 2: Showing cross sections $(y = 0)$ of the virtual retina for central panoramic camera parameters $0 \le \xi \le 1$.

$$\lambda \mathbf{b} = q, \qquad (4)$$

where the unknown scale λ is lost in the projection. Using equations (4) and (2), with $\lambda = -Z + \xi\sqrt{X^2+Y^2+Z^2}$, it is direct to solve for the virtual retina as:

$$z \triangleq \mathbf{f}_\xi(x,y) = \frac{-1 + \xi^2(x^2+y^2)}{1 + \xi\sqrt{1 + (1-\xi^2)(x^2+y^2)}}. \qquad (5)$$

Figure 2 shows a cross section $(y = 0)$ of the virtual retina surface $z = \mathbf{f}_\xi(x,y)$ for different catadioptric camera parameters $0 \le \xi \le 1$. Notice the retina becomes more planar as the model approaches perspective projection $(\xi \to 0)$.

Now suppose we have measurements of the *optical flow* $(\dot{x}, \dot{y})^T$ induced by the camera motion. Then, using the definition of the virtual retina surface in equation (5), the time derivative of the back-projection ray $\dot{\mathbf{b}} \triangleq (\dot{x}, \dot{y}, \dot{z})^T$ is obtained from the optical flow and the retina's shape using:

$$\dot{z} \triangleq \frac{\partial \mathbf{f}_\xi}{\partial x}\dot{x} + \frac{\partial \mathbf{f}_\xi}{\partial y}\dot{y} = \frac{\xi(x\dot{x} + y\dot{y})}{\sqrt{1 + (1-\xi^2)(x^2+y^2)}}.$$

We call $\dot{\mathbf{b}}$ the *back-projection flow*. It is direct to see that back-projection rays and back-projection flows are natural generalization of image points and optical flow in the case of perspective projection where $\xi = 0$.

3.2. Mapping Optical Flow to the Sphere

An alternative to the back-projection flow method was proposed in [7, 15], and consists of mapping the image points and their optical flow to the unit sphere.

The scene rays are described in spherical coordinates (ρ, θ, ϕ) about the center of projection, where ρ is the magnitude, θ is the azimuth angle in the X-Y plane, and ϕ is the polar angle between the ray and the Z-axis. Therefore, given an image point, we first compute its back-projection ray $\mathbf{b} = (x,y,z)^T$ using equation (5) and normalize it to unit length $\mathbf{s} \triangleq \mathbf{b}/\|\mathbf{b}\| = (x_s, y_s, z_s)^T$. The spherical coordinates of the "unitized" back-projection ray are given by:

$$\rho = 1, \qquad \theta = \arctan\frac{y}{x}, \qquad \phi = \arctan\frac{r}{z}, \qquad (6)$$

where $r \triangleq \sqrt{x^2+y^2}$. Using equation (5), we can derive the following Jacobian which relates partial derivatives from the image plane to the unit sphere:

$$J = \begin{bmatrix} \frac{\partial\theta}{\partial x} & \frac{\partial\theta}{\partial y} \\ \frac{\partial\phi}{\partial x} & \frac{\partial\phi}{\partial y} \end{bmatrix} = \begin{bmatrix} -y/r^2 & x/r^2 \\ x\delta & y\delta \end{bmatrix} \quad (7)$$

$$\delta \triangleq \frac{z\sqrt{1+(1-\xi^2)r^2}-\xi r^2}{r(r^2+z^2)\sqrt{1+(1-\xi^2)r^2}}. \quad (8)$$

Now, we need one more transformation which takes the partial derivatives on the sphere from spherical coordinates to rectangular coordinates:

$$S = \begin{bmatrix} \frac{\partial x_s}{\partial\theta} & \frac{\partial x_s}{\partial\phi} \\ \frac{\partial y_s}{\partial\theta} & \frac{\partial y_s}{\partial\phi} \\ \frac{\partial z_s}{\partial\theta} & \frac{\partial z_s}{\partial\phi} \end{bmatrix} = \begin{bmatrix} -\sin\theta\sin\phi & \cos\theta\cos\phi \\ \cos\theta\sin\phi & \sin\theta\cos\phi \\ 0 & -\sin\phi \end{bmatrix}. \quad (9)$$

Thus, for any central panoramic camera, the following equations define the mapping of an image point $(x,y)^T$ and its optical flow $(\dot{x},\dot{y})^T$ to the unit sphere:

$$\mathbf{s} = \mathbf{b}/\|\mathbf{b}\|, \qquad \dot{\mathbf{s}} = SJ\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix}. \quad (10)$$

As in [15], the derived SJ, which maps optical flow from the image plane to the sphere, is a generalization of the Jacobian derived in [7], and is valid for all central panoramic cameras. Our expression for SJ is simpler than the one in [15] because of our use of the back-projection ray $\mathbf{b}$.

Comment 3.1. *In essence, both of the above methods map optical flow to the surface of a curved virtual retina. The difference is that the back-projection flow is on a virtual retina that depends on ξ and is intrinsic to the central panoramic camera, while the spherical flow is on the unit sphere regardless of ξ. In the absence of noise, both methods are equally valid. However, as we will see in Section 5, when there is noise in the optical flow measurements, the performance of egomotion algorithms varies depending on whether the input is back-projection flow or spherical flow, as well as on the direction of camera motion.*

4. Egomotion Estimation

Here, we show that the differential epipolar constraint at the heart of egomotion algorithms is in fact a constraint on the back-projection flow. Thus, one can use the back-projection flow directly in well-known egomotion algorithms.

4.1. Differential Epipolar Constraint

If the camera undergoes a linear velocity $v \in \mathbb{R}^3$ and an angular velocity $\omega \in \mathbb{R}^3$, then the coordinates of a static 3D point $q \in \mathbb{R}^3$ evolve in the camera frame as $\dot{q} = [\omega]_\times q + v$. Assume that we measure the back-projection ray $\mathbf{b}$ and the back-projection flow $\dot{\mathbf{b}}$ corresponding to the fixed point q. Now, using equation (4) we have:

$$\dot{\lambda}\mathbf{b} + \lambda\dot{\mathbf{b}} = \lambda[\omega]_\times\mathbf{b} + v. \quad (11)$$

By taking the inner product of both sides of equation (11) with $[v]_\times\mathbf{b}$, we can eliminate the unknown scales $(\lambda,\dot{\lambda})$, and obtain the following bilinear constraint on the motion parameters (v,ω), which is independent of scene structure:

$$\dot{\mathbf{b}}^T[v]_\times\mathbf{b} + \mathbf{b}^T[\omega]_\times[v]_\times\mathbf{b} = 0. \quad (12)$$

Equation (12) is the *differential epipolar constraint*, and is the constraint at the heart of the well-known ego-motion algorithms (see [13] for an overview).

It is clear that rather than being a constraint on *image points* and *optical flows* as it was first derived, the differential epipolar constraint is more generally a constraint on the *back-projection rays* and *back-projection flows*. In fact, because we eliminated λ and $\dot{\lambda}$, the constraint is valid whether $\mathbf{b}$ is on the curved retina as in Section 3.1 or on the unit sphere as in Section 3.2. What is important is that the back-projection flow is *consistent* with the back-projection ray: $\dot{\mathbf{b}}$ must be in the tangent space of the virtual retina at $\mathbf{b}$.

4.2. Egomotion Algorithms

Most egomotion algorithms were designed for perspective cameras and explicitly use the fact that in perspective projection the back-projection ray $\mathbf{b} = (x,y,z)^T$ has $z = 1$ and $\dot{z} = 0$. However, the algorithms can easily be adapted to the general case where z and $\dot{z}$ are arbitrary.

In this section, we give brief descriptions of the three most successful egomotion algorithms (see [13, 9]).

Bruss-Horn. The Bruss-Horn algorithm (BH) [3] uses the differential epipolar constraint in equation (12) to obtain a least squares estimate of ω in terms of v. This estimate of ω is substituted back into (12) resulting in a bilinear constraint on v. An estimate of v is then obtained by a numerical minimization, *e.g.* using Levenberg-Marquardt. Given the estimate v, one can solve linearly for ω (see [3] for details).

Heeger-Jepson. The Heeger-Jepson (HJ) algorithm [8] begins by re-writing (12) as:

$$v^T[\mathbf{b}]_\times\dot{\mathbf{b}} = v^T([\mathbf{b}]_\times)^2\omega.$$

Then, given measurements of back-projection flows at n points, it finds a linear combination of motion vectors that is *independent* of depth and rotation and *orthogonal* to translation. By defining a vector of coefficients $\boldsymbol{c} = (c_1,\ldots,c_n)^T$ and the vector $\boldsymbol{\tau}(\boldsymbol{c}) = \sum_{i=1}^n c_i[\mathbf{b}_i]_\times\dot{\mathbf{b}}_i$, it follows that:

$$v^T\boldsymbol{\tau}(\boldsymbol{c}) = v^T\sum_{i=1}^n c_i([\mathbf{b}_i]_\times)^2\omega.$$

One can solve linearly for coefficients $\boldsymbol{c}$ such that $\sum_{i=1}^{n} c_i([\mathbf{b}_i]_\times)^2 = 0$. With this choice of $\boldsymbol{c}$, we have $v^T \boldsymbol{\tau}(\boldsymbol{c}) = 0$, which one can use to solve linearly for v. Given v, one can solve linearly for ω (see [8] for details).

Ma-Kosecka-Sastry. The Ma-Kosecka-Sastry algorithm (MKS) [9] is a linear geometric algorithm in the spirit of the well-known "8-point algorithm" for discrete egomotion estimation. It is shown in [9] that the differential epipolar constraint (12) is equivalent to $(\dot{\mathbf{b}}^T, \mathbf{b}^T)E\mathbf{b} = 0$, where:

$$E \triangleq \begin{bmatrix} [v]_\times \\ \frac{1}{2}([\omega]_\times [v]_\times + [v]_\times [\omega]_\times) \end{bmatrix} \in \mathbb{R}^{6\times 3} \qquad (13)$$

is the *differential essential matrix*. The algorithm continues by linearly estimating E from the optical flow of at least 8 points using $(\dot{\mathbf{b}}_i^T, \mathbf{b}_i^T)E\mathbf{b}_i = 0$, then projecting E onto the space of differential essential matrices, and finally decomposing E into parameters ω and v (see [9] for details).

5. Experiments

In this section, we present simulation results on synthetic data, comparing the performance of egomotion algorithms using our proposed back-projection flow and spherical flow of [7, 15] under different motions, central panoramic camera parameters, and levels of noise in the optical flow.

5.1. Simulation Setup

We generated synthetic optical flow by adapting the method described in [13] to the case of omnidirectional cameras. We consider a central panoramic camera given by (1) with $s_x = s_y$ and define $(\xi + m)s_x$ as the focal length. The diameter of the image disk was considered to be 512 pixels.

We assume a fixed CCD size for all camera parameters ξ, which corresponds to fixing the image disk to unit radius for all cameras. Since the field of view (FOV) of a central panoramic camera is determined by the back-projection ray at the perimeter of the image disk, the FOV is determined by ξ. Figure 3 shows the experimental setup, from which it is clear that $\mathrm{FOV}(\xi) = \pi + 2\mathrm{atan}(\mathbf{f}_\xi(1, 0))$. Notice that $\mathrm{FOV} = 90°$ when $\xi = 0$, and $\mathrm{FOV} = 180°$ when $\xi = 1$. As Figure 3 shows, we also model the fact that catadioptric cameras have a blind spot in the center of the image disk due to the reflection of the CCD in the mirror. In these experiments, we do not image any 3D point whose projection is at a distance less than $r_{\min} = 0.25$ from the image center.

A cloud of 400 points was scattered uniformly within the camera field of view at random depths which varied from $Z_{\max} = -10$ to $Z_{\min} = -400$ focal lengths (see Figure 3).

The 3D points were projected onto the image plane through the central panoramic projection (2). In each experiment, we computed the optical flow induced by a camera translation of 5 focal lengths per frame and rotation of $1°$ per frame about the specified translation and rotation axes.

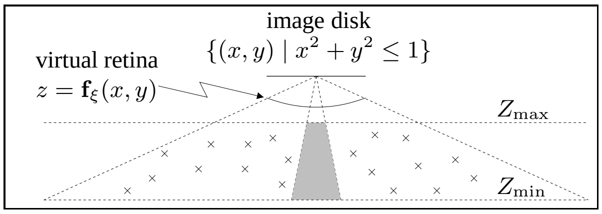
image disk
$\{(x, y) \mid x^2 + y^2 \leq 1\}$
virtual retina
$z = \mathbf{f}_\xi(x, y)$
$Z_{\max}$
$Z_{\min}$

Figure 3: Showing the experimental setup for the simulations. The field of view is determined by ξ through the back-projection ray at the perimeter of the image disk.

For a focal length of 8mm and a frame rate of 30Hz, this corresponds to translation of 1.2m/s and rotation of $30°$/s. This camera motion amounted to an average motion in the image plane of about 7 pixels. The true optical flow was corrupted by adding zero-mean Gaussian noise in the image plane where the standard deviation σ was specified in terms of pixel size and was independent of camera motion.

5.2. Performance Metrics

We compare the performance of the egomotion algorithms when using as input back-projection flow or spherical flow computed from the synthetic optical flow data. Each experiment consists of 1000 trials for a given setting of parameters. The translational and rotational *bias* are computed as the average angle between the estimated and the true direction of translation and axis of rotation, respectively.

Motion dependency. Figure 4 shows the motion dependency of translational and rotational bias for the three egomotion algorithms. The optical flow had $\sigma = 1$ pixel standard deviation noise, and the camera parameter was $\xi = 1$.

The BH algorithm outperforms the HJ and MKS algorithms because it uses nonlinear minimization. In the subsequent experiments, we only show the results of the BH algorithm, since the other two algorithms have the same behavior, but with a larger bias. Notice that the MKS algorithm has large rotation bias when v and ω are aligned. This is because the decomposition of the differential essential matrix (13) is numerically less accurate when v and ω coincide [9].

From Figure 4, we observe that when the camera translates in the X-Y plane, using back-projection flow in the egomotion algorithms gives more accurate motion estimates than using spherical flow, and vice versa when the translation is in the Z axis. For the same noise and camera parameters as above, Figure 5 shows the bias of the BH algorithm as a function of the polar angle ϕ between the camera velocity and the Z-axis. Due to symmetry, performance of egomotion algorithms does not depend on the direction of camera velocity in X-Y plane. Hence, in this and all subsequent experiments, a velocity along the X axis was chosen to represent a velocity in the X-Y plane.

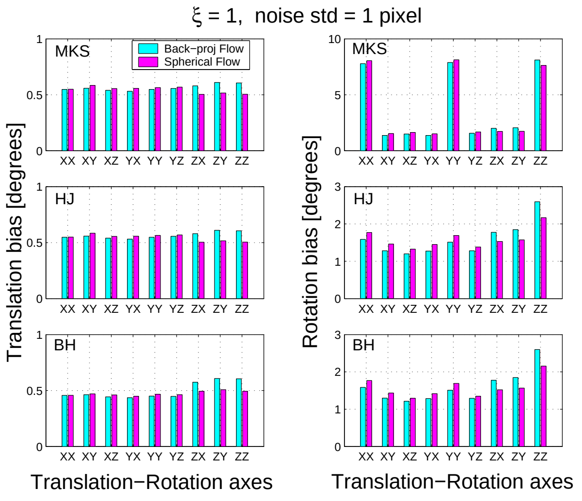

Figure 4: Bias dependency on the camera motion. "XY" means translation along X-axis and rotation about Y-axis.

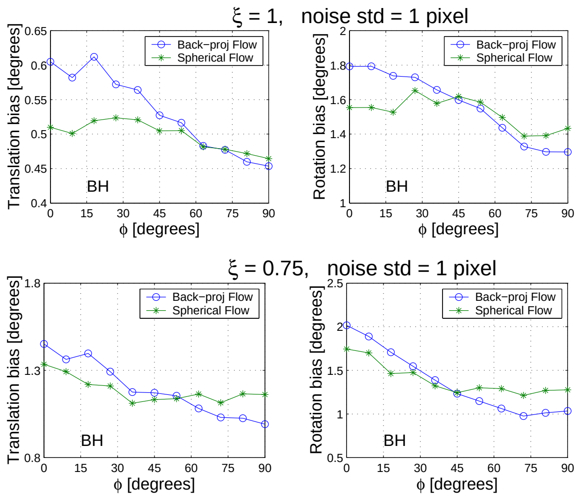

Figure 5: Bias dependency on the translation direction. ϕ is the polar angle between v and the Z-axis.

From Figure 5, we observe that it is preferable to use back-projection flow in the egomotion algorithms when the translation direction is roughly in the X-Y plane. Further, we see that the *cross-over point* where it is preferable to use back-projection flow over spherical flow is a function of ξ. The cross-over point is consistently 45° for rotation for both $\xi \in \{0.75, 1\}$, and shifts to smaller angles for translation as ξ decreases.

It can be shown that the curvature of the sphere is greater than the curvature of the virtual retina everywhere, and for all $0 \leq \xi \leq 1$. Thus, the Jacobian of the mapping to spherical flow has larger singular values than the Jacobian of the mapping to back-projection flow. This implies that camera velocities in the X-Y plane induce a larger back-projection flow than spherical flow, while camera velocities along the Z-axis induce a larger spherical flow than back-projection flow. Thus, as shown in the above experiments, for a given noise level in the image plane, egomotion algorithms perform better using back-projection flow if the camera translation is mostly in the X-Y plane. Thus, we conclude that our proposed back-projection flow method is preferable in important special cases where there is little or no Z-axis translation, such as in ground robot navigation.

Noise dependency. Figures 6 and 7 shows the bias dependency on the noise level in the optical flow for a translation in the X-Y plane, and along the Z axis, respectively, for the camera $\xi = 1$. As expected, we observe that when the camera translates in the X-Y plane, the back-projection flow method gives slightly better motion estimates as the noise in the optical flow increases, and spherical flow gives improved estimates when the translation is along the Z axis.

Camera dependency. Figures 8 and 9 show the bias dependency on the camera parameter for translations in the X-Y plane and in the Z axis, respectively. The optical flow data had 1 pixel standard deviation noise. As expected from the previous experiments, we observe that for every central panoramic camera, using back-projection flow gives better motion estimates if the translation is in the X-Y plane, while the spherical flow gives better estimates when the translation is in the Z-axis.

From Figure 8, we observe the interesting phenomenon that as ξ increases, translation errors *increase* while rotation errors *decrease*. This is due to the following interesting interplay which occurs as the FOV increases. An increased FOV should improve the motion estimates because it is easier to distinguish between translational and rotational flows [7, 12]. However, as the FOV increases, the *spacial resolution* (the infinitesimal solid angle of the world viewed by an infinitesimal pixel) of the camera increases radially in the image plane [1]. Thus, for a fixed noise level in the image plane, higher FOV cameras have larger errors in the computed back-projection rays, and hence larger errors in the triangulation angles for egomotion estimation.

6. Conclusions and Future Work

We have developed the notion of *back-projection flow* as a natural generalization of optical flow for central panoramic cameras, and showed that well-known egomotion algorithms can be applied directly to this back-projection flow. Finally, through extensive simulation results, we showed that using back-projection flow gives better egomotion estimation results than using spherical flow in important special cases such as applications in ground robot navigation.

This work suggests many directions for future research: (1) Find a proper noise model for optical flow with central panoramic cameras; (2) Do a detailed analysis of the sta-

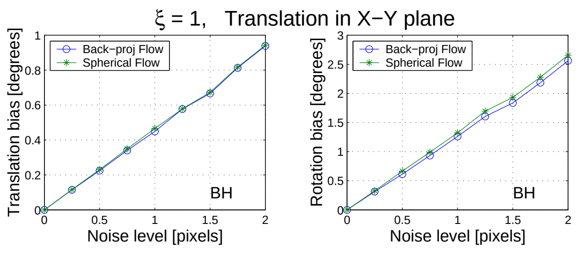

Figure 6: Bias dependency on noise in the optical flow. The translation was in the X-Y plane.

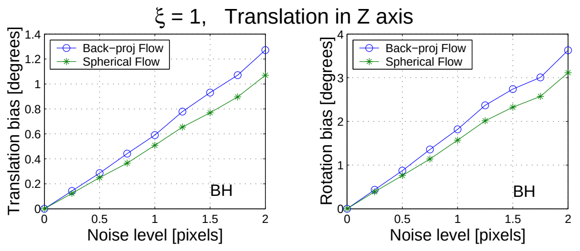

Figure 7: Bias dependency on noise in the optical flow. The translation was along the Z-axis.

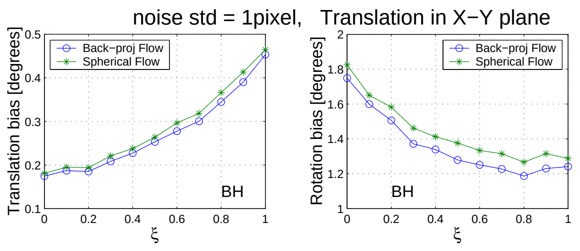

Figure 8: Bias dependency on type of catadioptric camera. The translation was in the X-Y plane.

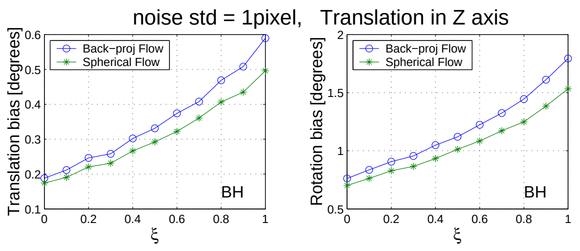

Figure 9: Bias dependency on type of catadioptric camera. The translation was along the Z-axis.

tistical correlation between back-projection flow and spherical flow, and the dependency of the corresponding signal-to-noise ratios on the camera motion; (3) Apply the back-projection flow to the recent algorithms for multi-frame and multi-body infinitesimal motion estimation and segmentation [10, 11]; (4) Investigate the computation of optical flow for central panoramic cameras. In [4], the sphere is suggested as the natural underlying space for image processing for omnidirectional cameras because of the significant radial distortion. Our work suggests that the correct space for image processing may depend on the camera motion. For example, the proposed virtual retina is more appropriate for motion in the X-Y plane.

References

[1] S. Baker and S. Nayar. A theory of single-viewpoint catadioptric image formation. *Int. Journal on Computer Vision*, 35:175–196, 1999.

[2] J. Barreto and H. Araujo. Geometric properties of central catadioptric line images. In *ECCV*, pages 237–251, 2002.

[3] A. Bruss and B.K.P. Horn. Passive navigation. In *Computer Vision, Graphics, and Image Processing*, volume 21, pages 3–20, 1983.

[4] K. Daniilidis, A. Makadia, and T. Bulow. Image processing in catadioptric planes: Spatiotemporal derivatives and optical flow computation. In *OMNIVIS*, pages 3–10, 2002.

[5] C. Geyer and K. Daniilidis. A unifying theory for central panoramic systems and practical implications. In *ECCV*, pages 445–461, 2000.

[6] C. Geyer and K. Daniilidis. Paracatadioptric camera calibration. *IEEE Transactions on PAMI*, 4(24):1–10, 2002.

[7] J. Gluckman and S.K. Nayar. Ego-motion and omnidirectional cameras. In *ICCV*, pages 999–1005, 1998.

[8] D.J. Heeger and A.D. Jepson. Subspace methods for recovering rigid motion. *Int. Journal on Computer Vision*, 7(2):95–117, 1992.

[9] Y. Ma, J. Kosecka, and S. Sastry. Linear differential algorithm for motion recovery: A geometric approach. *International Journal on Computer Vision*, 36(1):71–89, 2000.

[10] O. Shakernia, R. Vidal, and S. Sastry. Infinitesimal motion estimation from multiple central panoramic views. In *IEEE Workshop on Motion and Video Computing*, pages 229-234, 2002.

[11] O. Shakernia, R. Vidal, and S. Sastry. Multi-body motion estimation and segmentation from multiple central panoramic views. To appear: *ICRA*, 2003.

[12] T. Svoboda, T. Pajdla, and V. Hlavac. Motion estimation using panoramic cameras. In *IEEE Conference on Intelligent Vehicles*, pages 335–350, 1998.

[13] T. Tian, C. Tomasi, and D. Heeger. Comparison of approaches to egomotion computation. In *CVPR*, pages 315–320, 1996.

[14] I. Ulrich and I. Nourbakhsh. Appearance-based place recognition for topological localization. In *ICRA*, pages 1023–1029, 2000.

[15] R.F. Vassallo, J. Santos-Victor, and J. Schneebeli. A general approach for egomotion estimation with omnidirectional images. In *OMNIVIS*, pages 97–103, 2002.

[16] R. Vidal, O. Shakernia, and S. Sastry. Formation control of non-holonomic mobile robots with omnidirectional visual servoing and motion segmentation. To appear: *ICRA*, 2003.

[17] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omni-directional vision for robot navigation. In *OMNIVIS*, pages 21–28, 2000.