Exploiting Rolling Shutter Distortions for Simultaneous Object Pose and Velocity Computation Using a Single View

Omar Ait Aider, Nicolas Andreff, Jean-Marc Lavest, Philippe Martinet

▶ **To cite this version:**

Omar Ait Aider, Nicolas Andreff, Jean-Marc Lavest, Philippe Martinet. Exploiting Rolling Shutter Distortions for Simultaneous Object Pose and Velocity Computation Using a Single View. ICVS06 - 4th IEEE International Conference on Computer Vision Systems, Jan 2006, New-York City, United States. hal-02468170

HAL Id: hal-02468170
https://inria.hal.science/hal-02468170

Submitted on 5 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Exploiting Rolling Shutter Distortions for Simultaneous Object Pose and Velocity Computation Using a Single View

Omar Ait-Aider, Nicolas Andreff, Jean Marc Lavest and Philippe Martinet
Blaise Pascal University
LASMEA UMR 6602 CNRS
Clermont Ferrand, France
firstname.lastname@lasmea.univ-bpclermont.fr

Abstract

An original method for computing instantaneous 3D pose and velocity of fast moving objects using a single view is presented. It exploits image deformations induced by rolling shutter in CMOS image sensors. First of all, a general perspective projection model of a moving 3D point is presented. A solution for the pose and velocity recovery problem is then described. The method is based on bundle adjustment and uses point correspondences. The resulting algorithm enables to transform a CMOS low cost and low power camera into an original velocity sensor. Finally, experimental results with real data confirm the relevance of the approach.

1 Introduction

In many fields such as robotics, automatic inspection, road traffic or metrology, it is necessary to capture clear images of objects undergoing high velocity motion without any distortion, blur nor smear. To achieve this task, there is a need for image sensors which allow very short exposure time of all the matrix pixels simultaneously. This functionality requires a particular electronic design that is not included in all camera devices. Full Frame CCD sensors, without storage memory areas, require mechanical obturator or stroboscopic light source, introducing more complexity in the vision system. Frame Transfer CCD sensors may not reach the desired frame rate or may be costly because of additional sillicium in storage areas [6].

Standard CMOS Rolling Shutter sensors are considered as low cost and low power sensors. They are becoming more frequently used in cameras. They enable adequate exposure time without reducing frame rate thanks to overlapping exposure and readout. Their drawback is that they distort images of moving objects because the pixels are not all exposed simultaneously but row by row with a time delay defined by the sensor technology. Fig.1 shows an example of distortions due to rolling shutter on the appearance of a rotating ventilator. This distortions may represent a major obstacle in tasks such as localization, reconstruction or default detection (the system may see an ellipse where in fact there is a circular hole). Therefore, CMOS Rolling Shutter cameras could offer a good compromise between cost and frame rate performances if the problem of deformations is taken into account.

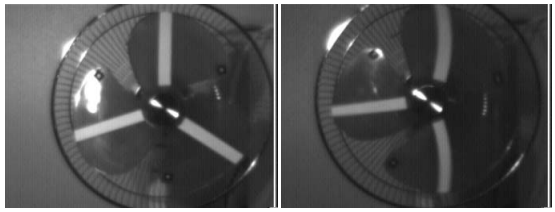

Figure 1. An example of distortion of a rotating ventilator observed with a Rolling Shutter camera: static object (right image) and moving object (left image).

To our knowledge, there is no works in the vision community literature on modelling effects of rolling shutter in pose recovery algorithms nor on computing velocity parameters using a single view. All pose recovery methods ([3, 5, 1, 4, 7]) make the assumption that all image sensor pixels are exposed simultaneously. The work done by Wilburn et al. [8] concerned the correction of image deformation by constructing a single image using several images from a dense camera array. Using the knowledge of the time delay due to rolling shutter and the chronograms of

release of the cameras, one complete image is constructed by combining lines exposed at the same instant in each image from the different cameras.

The focus of our work, in this paper, is to develop a method which maintains accuracy in pose recovery and structure from motion algorithms without sacrificing low cost and power characteristics of the sensor. This is achieved by integrating, in the algorithm, kinematic and technological parameters which are both causes of image deformations. The resulting algorithm, not only enables accurate pose recovery, but also provides the instantaneous angular and translational velocity of observed objects. Rolling shutter effects which are considered as drawbacks are transformed here into an advantage ! This approach may be considered as an alternative to methods which uses image sequences to estimate the kinematic between views since it reduces the amount of data and the computational cost (one image is processed rather than several ones).
Section 2 of this paper describes the process of image acquisition using a CMOS Rolling Shutter imager. In section 3, a general geometric model for the perspective projection of 3D point on a solid moving object is presented. Image coordinates of the point projections are expressed with respect to object pose and velocity parameters and to the time delay of CMOS sensor image row scanning. Section 4 deals with the problem of computing pose and velocity parameters of a moving object, imaged by a CMOS Rolling Shutter camera, using point correspondences. The approach generalizes the bundle adjustment method to the case of moving points. It is based on non-linear least-square optimization of an error function defined in image metric and expressed with respect to both pose and velocity parameters (rather than to only pose parameters in classical approaches). Finally, experiments with real data are presented and analyzed in section 5.

2 What is rolling shutter ?

In digital cameras, an image is captured by converting the light from an object into an electronic signal at the photosensitive area (photodiode) of a solid state CCD or CMOS image sensor. The amount of signal generated by the image sensor depends on the amount of light that falls on the imager, in terms of both intensity and duration. Therefore, an on-chip electronic shutter is required to control exposure. The pixels are allowed to accumulate charge during the integration time. With global shutter image sensors, the entire imager is reset before integration. The accumulated charge in each pixel is simultaneously transferred to storage area. Since all the pixels are reset at the same time and integrate over the same interval there is no motion artifacts in the resulting image. With a CMOS image sensor with rolling shutter, the rows of pixels in the image are reset in sequence starting at the top and proceeding row by row to the bottom. The readout process proceeds in exactly the same fashion and the same speed with a time delay after the reset. This time delay between a reset of a row and its reading out is the exposure time. Each line in the image has the same amount of integration, however the start and end time of integration is shifted in time as the image is scanned (rolled) out of the sensor array as shown in Fig.2. In this case, if the object is moving during the integration time, some artifacts may appear. The faster the object moves the larger is the distortion. The benefit of rolling shutter mode is that exposure and readout are overlapping, enabling full frame exposures without reducing frame rate.

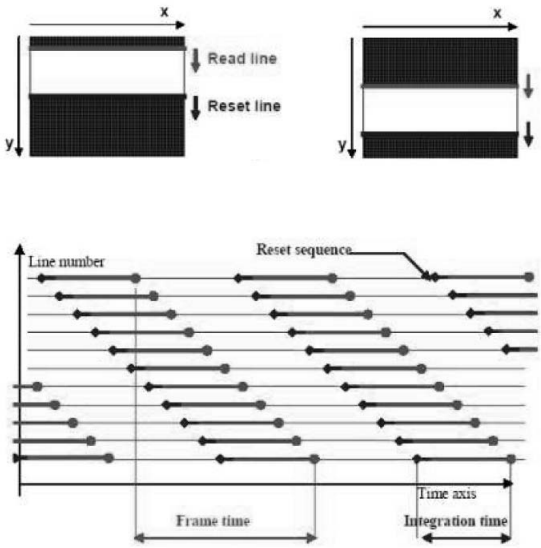

Figure 2. Reset and reading chronograms in rolling shutter sensor (SILICON IMAGING documentation).

3 Projecting a point with a rolling shutter camera

Let us consider a classical camera with a pinhole projection model defined by its intrinsic parameter matrix [7]

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let $\mathbf{P} = [X, Y, Z]^{\mathrm{T}}$ be a 3D point defined in the object

frame. Let $\mathbf{R}$ and $\mathbf{T}$ be the rotation matrix and the translation vector between the object frame and the camera frame. Let $\mathbf{m} = [u, v]^{\mathrm{T}}$ be the perspective projection of $\mathbf{P}$ on the image. Noting $\tilde{\mathbf{m}} = \left[\mathbf{m}^{\mathrm{T}}, 1\right]^{\mathrm{T}}$ and $\tilde{\mathbf{P}} = \left[\mathbf{P}^{\mathrm{T}}, 1\right]^{\mathrm{T}}$, the relationship between $\mathbf{P}$ and $\mathbf{m}$ is:

$$s\tilde{\mathbf{m}} = \mathbf{K}\left[\mathbf{R} \quad \mathbf{T}\right]\tilde{\mathbf{P}} \tag{1}$$

where s is an arbitrary scale factor. Note that the lens distortion parameters which do not appear here are obtained by calibrating [2] and are taken into account by correcting image data before using them in the algorithm.

Assume now that an object of a known geometry modelled by a set of n points $\mathbf{P_i} = [X_i, Y_i, Z_i]^{\mathrm{T}}$, undergoing a motion with angular and linear velocities $\mathbf{\Omega} = [\Omega_x, \Omega_y, \Omega_z]^{\mathrm{T}}$ and $\mathbf{V} = [V_x, V_y, V_z]^{\mathrm{T}}$ respectively, is snapped with a rolling shutter camera at an instant t_0. In fact, t_0 corresponds to the instant when the top line of the sensor is exposed to light. Thus, the light from the point $\mathbf{P_i}$ will be collected with a delay τ_i proportional to the image line number on which $\mathbf{P_i}$ is projected. As illustrated in Fig.3, τ_i is the time delay necessary to expose all the lines above the line which collects the light from $\mathbf{P_i}$. Therefore, to obtain the projection $\mathbf{m}_i = [u_i, v_i]^{\mathrm{T}}$ of $\mathbf{P_i}$, the pose parameters of the object must be corrected in equation 1 by integrating the motion during the time delay τ_i. Since all the lines have the same exposure and integration time, we have $\tau_i = \tau v_i$ where τ is the time delay between two successive image line exposure. Thus $\tau = \frac{fp}{v_{max}}$ where fp is the frame period and v_{max} is the image height. Assuming that τ_i is short enough to consider small and uniform motion during this interval, equation 1 can be rewritten as follows:

$$s\tilde{\mathbf{m}}_i = \mathbf{K}\left[\left(\mathbf{I} + \tau v_i\hat{\mathbf{\Omega}}\right)\mathbf{R} \quad \mathbf{T} + \tau v_i\mathbf{V}\right]\tilde{\mathbf{P}}_\mathbf{i} \tag{2}$$

where $\mathbf{R}$ and $\mathbf{T}$ represent now the instantaneous object pose at t_0, $\hat{\mathbf{\Omega}}$ the antisymetric matrix of vector $\mathbf{\Omega}$ and $\mathbf{I}$ is the 3×3 identity matrix. Equation 2 is the expression of the projection of a 3D point from a moving solid object using a rolling shutter camera with respect to object pose, object velocity and the parameter τ. One can note that it contains the unknown v_i in its two sides. This is due to the fact that the coordinates of the projected point on the image depend on both the kinematics of the object and the imager sensor scanning velocity. This equation can be solved for v_i as a second degree equation with two solutions (only one is realistic, the second is purely analytic). u_i is obtained by substituting the value of v_i.

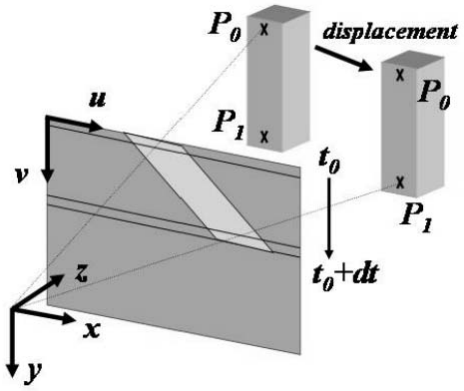

Figure 3. Perspective projection of a moving 3D object: due to the time delay, points P_0 and P_1 are not projected from the same object pose

.

4 Computing the instantaneous pose and velocity of a moving object

In this section, we assume that a set of rigidly linked 3D points $\mathbf{P_i}$ on a moving object are matched with their respective projections $\mathbf{m_i}$ measured on an image taken with rolling shutter camera. We want to use this list of 3D-2D correspondences to compute the instantaneous pose and velocity of the object at instant t_0. The scale factor of equation 2 can be removed as follows:

$$\begin{aligned} u_i &= \alpha_u \frac{\left(\mathbf{R_1}+\tau v_i\hat{\mathbf{\Omega}}_1\right)\mathbf{P_i}+T_x+\tau v_i V_x}{\left(\mathbf{R_3}+\tau v_i\hat{\mathbf{\Omega}}_3\right)\mathbf{P_i}+T_z+\tau v_i V_z} + u_0 = \xi_i^{(u)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right) \\ v_i &= \alpha_v \frac{\left(\mathbf{R_2}+\tau v_i\hat{\mathbf{\Omega}}_1\right)\mathbf{P_i}+T_y+\tau v_i V_y}{\left(\mathbf{R_3}+\tau v_i\hat{\mathbf{\Omega}}_1\right)\mathbf{P_i}+T_z+\tau v_i V_z} + v_0 = \xi_i^{(v)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right) \end{aligned} \tag{3}$$

where $\mathbf{R_i}$ and $\hat{\mathbf{\Omega}}_\mathbf{i}$ are respectively the i^{th} rows of $\mathbf{R}$ and $\hat{\mathbf{\Omega}}$. Subsiding the right term from the left term and substituting u_i and v_i by image measurements, equation 3 can be seen as an error function with respect to pose and velocity (and possibly τ) parameters:

$$\begin{aligned} u_i - \xi_i^{(u)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right) &= \epsilon_i^{(u)} \\ v_i - \xi_i^{(v)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right) &= \epsilon_i^{(v)} \end{aligned}$$

We want to find $(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V})$ that minimize the following error function:

$$\epsilon = \sum_{i=1}^{n} \left[u_i - \xi_i^{(u)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right)\right]^2 + \left[v_i - \xi_i^{(v)}\left(\mathbf{R},\mathbf{T},\mathbf{\Omega},\mathbf{V}\right)\right]^2 \quad (4)$$

This problem with 12 unknowns can be solved using a non-linear least square optimization if at least 6 correspondences are available. This can be seen as a bundle adjustment with a calibrated camera. Note that, in our algorithm, the rotation matrix $\mathbf{R}$ is expressed by a unit quaternion representation $q(\mathbf{R})$. Thus, an additional equation, which forces the norm of $q(\mathbf{R})$ to 1, is added. It is obvious that this non-linear algorithm requires an initial guess to converge towards an accurate solution.

5 Experiments

The presented algorithm was tested on real image data. A reference 3D object with white spots was used. Sequences of the moving object at high velocity were captured with the Silicon Imaging CMOS Rolling Shutter camera SI1280M-CL, calibrated using the method described in [2]. Acquisition was done with a 1280×1024 resolution and at a rate of 30 frames per second so that $\tau = 7.15 \times 10^{-5}$ s. Image point coordinates were accurately obtained by a sub-pixel accuracy computation of the white spot centers and corrected according to the lens distortion parameters. Correspondences with model points were established with a supervised method. The pose and velocity parameters were computed for each image using first our algorithm, and then using the classical pose recovery algorithm described in [2] where an initial guess is first computed by the algorithm of Dementhon [1] and then the pose parameters are accurately estimated using a bundle adjustment technique.

Figure 4 shows image samples from a sequence where the reference object was moved following a straight rail forcing its motion to be a pure translation. In the first and last images of the sequence the object was static. Pose parameters corresponding to these two views were computed accurately using the classical algorithm. The reference object trajectory was then assumed to be the 3D straight line relating the two extremities.

Table 1 shows the RMS pixel re-projection error obtained using the pose computed with the classical algorithm and a classical projection model from the one hand-side, and the pose computed with our algorithm and the rolling shutter projection model from the other hand-side. Results show that errors obtained with static object views are similar. However, as the velocity increases, the error obtained with the classical algorithm becomes too important while the error obtained with our algorithm remains small.

Let us now analyze pose recovery results shown in Fig.5. The left-hand side of this figure shows 3D translational pose parameters obtained by our algorithm and by the classical algorithm (respectively represented by square and *-symbols). Results show that the two algorithms give appreciably the same results with static object views (first and last measurements). When the velocity increases, a drift appears in the classical algorithm results while our algorithm remains accurate (the 3D straight line is accurately reconstructed by pose samples) as it is illustrated on Table 2 where are represented distances between computed poses with each algorithm and the reference trajectory. Table 3 presents computed rotational pose parameters. Results show the deviation of computed rotational pose parameters from the reference orientation. Since the motion was a pure translation, orientation is expected to remain constant. As one can see, a drift appears on classical algorithm results while our algorithm results show a very small deviation due only to noise on data.

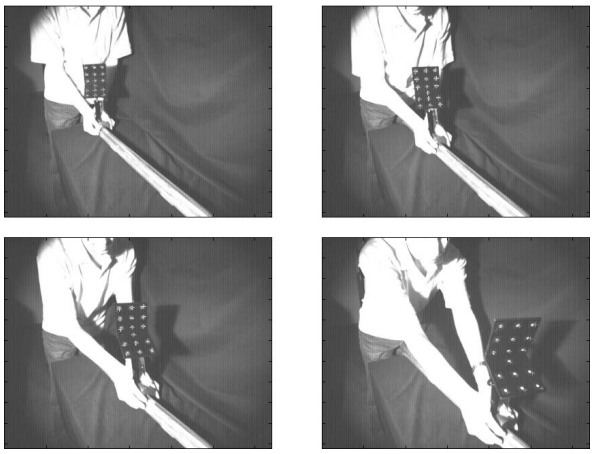

Figure 4. Image samples of pure translational motion.

Another result analysis concerns the velocity parameters. Figure 5 shows that the translational velocity vector is clearly parallel to translational axis (up to noise influence). Table 4 represents magnitude of computed velocity vectors in comparison with measured values. These reference values were obtained by dividing the distance covered between each two successive images by the frame period. This gives estimates of the translational velocity magnitudes. Results show that the algorithm recover correctly acceleration, deceleration and static phases. Table 5 represents computed rotational velocity parameters. As expected, the velocity parameter values are small and only due to noise.

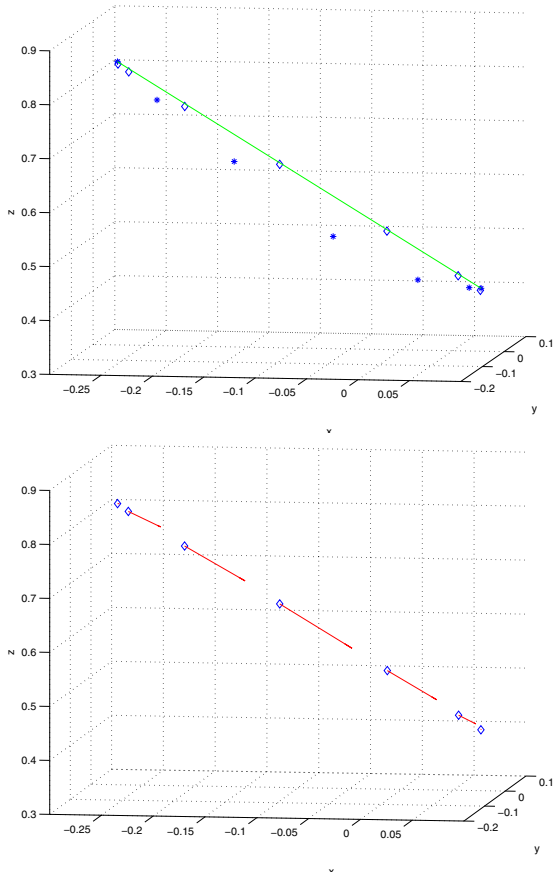

Figure 5. Pose and velocity results: reconstructed trajectory (top image), translational velocity vectors (bottom image).

Table 1. RMS re-projection error (pixel).

Image number	Dementhon's algorithm RMS-u	Dementhon's algorithm RMS-v	Our algorithm RMS-u	Our algorithm RMS-v
1	0.20	0.12	0.13	0.05
2	1.71	1.99	0.13	0.06
3	3.95	4.18	0.14	0.11
4	7.09	7.31	0.10	0.05
5	5.56	6.73	0.23	0.12
6	1.87	3.02	0.27	0.12
7	0.30	0.12	0.33	0.18

In the second experiment, the algorithm was tested on coupled rotational and translational motions. The previously described reference object was mounted on a rotating mechanism. Its circular trajectory was first reconstructed from a set of static images. This reference circle belongs to a plan whose measured normal vector is $\mathbf{N} = [0.05, 0.01, -0.98]^{\mathrm{T}}$. Thus, $\mathbf{N}$ represents the reference rotation axis. An image sequence of the moving object was then captured. Figure 6 shows samples of images taken during the rotation.

The left part of Fig.7 represents the trajectory reconstructed with a classical algorithm (*-symbol) and with our algorithm (square symbol). As for the pure translation, results show that the circular trajectory was correctly recon-

Table 2. Distances from computed poses to reference trajectory (cm).

Image num.	Dementhon's algorithm	Our algorithm
1	0.00	0.28
2	0.19	0.34
3	0.15	0.26
4	1.38	0.32
5	3.00	0.32
6	4.54	0.11
7	0.00	0.10

Table 3. Angular deviation of computed poses from reference orientation (deg.).

Image numb.	Dementhon's algorithm	Our algorithm
1	0.00	0.17
2	2.05	0.13
3	4.52	0.17
4	6.93	0.34
5	6.69	1.09
6	3.39	0.91
7	0.30	0.40

Table 4. Computed translational velocity magnitude in comparison with measured velocity values (m/s)

Image num.	Measured values	Computed values
1	0.00	0.06
2	1.22	1.00
3	2.02	1.82
4	2.32	2.23
5	1.55	1.54
6	0.49	0.50
7	0.00	0.02

Table 5. Computed rotational velocities (rad/s).

Image num.	Rotational velocity
1	0.06
2	0.09
3	0.05
4	0.01
5	0.35
6	0.11
7	0.12

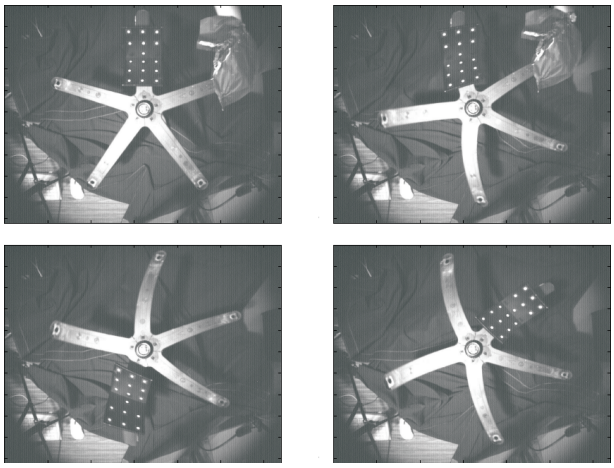

Figure 6. Image samples of coupled rotational and translational motions.

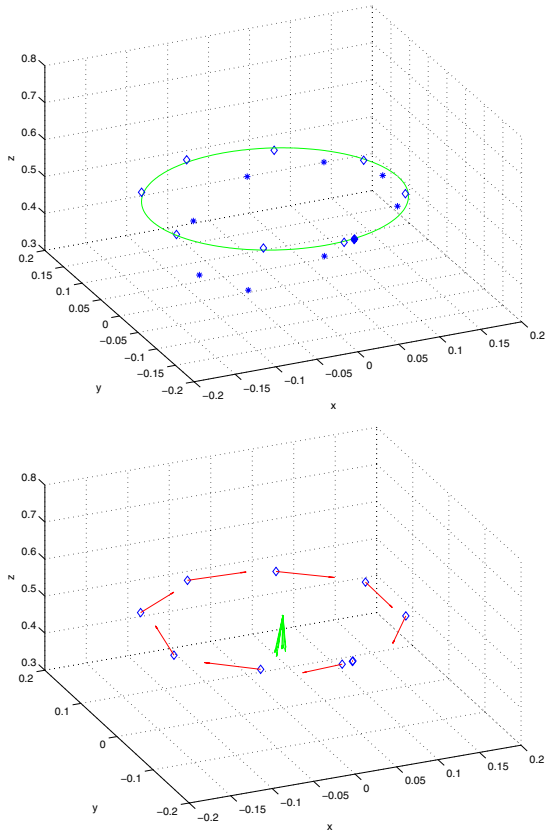

Figure 7. Pose and velocity results for coupled rotational and translational motion: reconstructed trajectory (top image), rotational and translational velocities (bottom image).

structed by the poses computed with our algorithm, while a drift is observed on the results of the classical algorithm as the object accelerates. The right part of the figure shows that translational velocity vectors were correctly oriented (tangent to the circle) and that the manifold of instantaneous rotation axis vectors (computed thanks to the Rodrigues formula applied on computed velocities) was also correctly oriented. The mean value of the angles between the computed rotation axis and $\mathbf{N}$ is 0.50 degrees. Results in table 6 shows a comparison of the computed rotational velocity magnitudes and the measured values.

Table 6. Computed and measured rotational velocity magnitudes (rad/s)

Image num.	Measured values	Computed values
1	0.00	0.06
2	1.50	0.50
3	9.00	7.55
4	11.20	10.48
5	1.55	10.32
6	10.50	10.50
7	10.20	9.41
8	10.10	9.67
9	10.00	9.43
10	7.50	8.71

6 Conclusion and perspectives

An original method for computing pose and instantaneous velocity of rigid objects using a single view from a rolling shutter camera was presented. The perspective projection equations of a moving 3D point was first established. Then, an error function equivalent to collinearity equations in camera calibration was defined and minimized numerically to obtain the object pose and velocity param-

eters. The approach was evaluated on real data showing its feasibility. The method is not less accurate than similar classical algorithms in case of static objects, and improves pose accuracy in case of fast moving objects. In addition, the method gives the instantaneous velocity parameters using a single view. This last property makes it an original tool for many fields of research. For example, it may avoid numerical derivation to estimate the velocity from several poses. Instantaneous velocity information may also be used as evolution models in motion tracking, as well as in visual servoing, to predict the state of observed moving patterns.

Our future works, will focus on improving accuracy of velocity computation by developing an evaluation framework with accurate ground truth values. A linear solution to initialize the non-linear optimization may also be investigated.

References

[1] D. Dementhon and L. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1/2):123–141, June 1995.

[2] J. Lavest, M. Viala, and M. Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration. In *Proceedings of ECCV98*, pages 158–174, Freiburg, Germany, June 1998.

[3] D. G. Lowe. Fitting parameterized three-dimensional models to image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991.

[4] J. T. L. M. Dhome, M. Richetin and G. Rives. Determination of the attitude of 3-d objects from a single perspective view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1265–1278, December 1989.

[5] R. H. T. Q. Phong and P. D. Tao. Object pose from 2-d to 3-d point and line correspondences. *International Journal of Computer Vision*, pages 225–243, 1995.

[6] A. J. P. Theuwissen. *Solid-state imaging with chargecoupled devices*. Kluwer Academic Publishers, 1995.

[7] Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, 1986.

[8] B. Wilburn and al. High-speed videography using a dense camera array. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington DC, USA, June - July 2004.