# Simultaneous Object Pose and Velocity Computation Using a Single View from a Rolling Shutter Camera

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

Université Blaise Pascal Clermont Ferrand,
LASMEA UMR 6602 CNRS
Omar.AIT-AIDER@univ-bpclermont.fr
http://www.lasmea.fr

**Abstract.** An original concept for computing instantaneous 3D pose and 3D velocity of fast moving objects using a single view is proposed, implemented and validated. It takes advantage of the image deformations induced by rolling shutter in CMOS image sensors. First of all, after analysing the rolling shutter phenomenon, we introduce an original model of the image formation when using such a camera, based on a general model of moving rigid sets of 3D points. Using 2D-3D point correspondences, we derive two complementary methods, compensating for the rolling shutter deformations to deliver an accurate 3D pose and exploiting them to also estimate the full 3D velocity. The first solution is a general one based on non-linear optimization and bundle adjustment, usable for any object, while the second one is a closed-form linear solution valid for planar objects. The resulting algorithms enable us to transform a CMOS low cost and low power camera into an innovative and powerful velocity sensor. Finally, experimental results with real data confirm the relevance and accuracy 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 to 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 [9].

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

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part II, LNCS 3952, pp. 56–68, 2006.

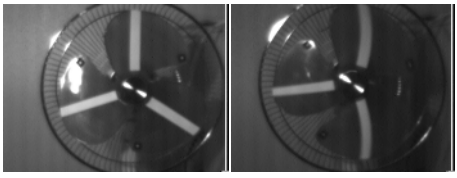

**Fig. 1.** An example of distortion of a rotating ventilator observed with a Rolling Shutter camera: static object (right image) and moving object (left image)

objects because the pixels are not all exposed simultaneously but row by row with a time delay defined by the sensor technology (figure 1). This distortion 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.

## 2 Related Works and Contributions

This work, is related to our previous one presented in [1], which focused on the development of a method which maintains accuracy in pose recovery and structure from motion algorithms without sacrificing low cost and power characteristics of the sensor. This was achieved by integrating, in the perspective projection model, 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 moving objects. Rolling shutter effects which are considered as drawbacks are transformed 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). In a parallel work by Meingast [7] (published after the submission of this paper), the projection in rolling shutter cameras is modelled in the case of fronto-parallel motion obtaining equations which are similar to those of Crossed-Slits cameras [13]. To our knowledge, there is no work in the vision community literature on taking into account effects of rolling shutter in pose recovery algorithms nor on computing velocity parameters using a single view. Indeed, all pose recovery methods ([6], [8], [2], [3], [10]) make the assumption that all image sensor pixels are exposed simultaneously. The work done by Wilburn et al. [11] concerned the correction of image deformation due to rolling shutter 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.

Two main contributions are presented in this paper. First, the perspective projection model of rolling shutter cameras presented in [1] is improved by removing the assumption of small motion during image acquisition. This makes the model more accurate for very fast moving objects. A novel non-linear algorithm for pose and velocity computation is then described. It generalizes the bundle adjustment method to the case of moving points. Indeed, 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). Second, a linear algorithm for pose and instantaneous velocity computation is developed in the particular case of planar objects. This linear solution provides an initial estimate of the pose and velocity parameters and serves to initialize the non-linear algorithm.

Section 3 of this paper describes the process of image acquisition using a CMOS Rolling Shutter imager. In section 4, 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 due to image scanning. Section 5 deals with the problem of computing pose and velocity parameters of a moving object, imaged by a Rolling Shutter camera, using point correspondences. Finally, experiments with real data are presented and analyzed in section 6.

## 3 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 (exposure time). The benefit of rolling shutter mode is that exposure and readout are overlapping, enabling full frame exposures without reducing the frame rate. 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.

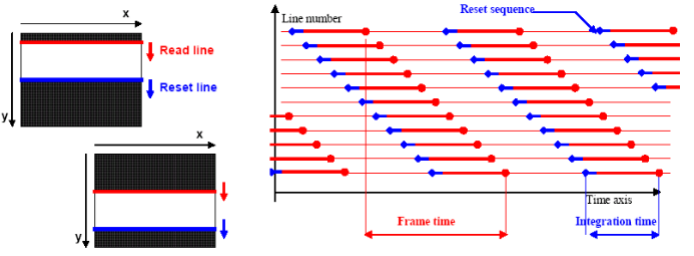


**Fig. 2.** Reset and reading chronograms in rolling shutter sensor (SILICON IMAGING documentation)

## 4 Projecting a Point with Rolling Shutter Camera

Let us consider a classical camera with a pinhole projection model defined by its intrinsic parameter matrix [10]

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

Let $\boldsymbol{P} = [X, Y, Z]^{\mathrm{T}}$ be a 3D point defined in the object frame. Let $\boldsymbol{R}$ and $\boldsymbol{T}$ be the rotation matrix and the translation vector between the object frame and the camera frame. Let $\boldsymbol{m} = [u, v]^{\mathrm{T}}$ be the perspective projection of $\boldsymbol{P}$ on the image. Noting $\tilde{\boldsymbol{m}} = \left[\boldsymbol{m}^{\mathrm{T}}, 1\right]^{\mathrm{T}}$ and $\tilde{\boldsymbol{P}} = \left[\boldsymbol{P}^{\mathrm{T}}, 1\right]^{\mathrm{T}}$, the relationship between $\boldsymbol{P}$ and $\boldsymbol{m}$ is:

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

where $s$ is an arbitrary scale factor. Note that the lens distortion parameters which do not appear here are obtained by calibration [5] and are taken into account by correcting image data before using them in the algorithm.

Assume now that an object of known geometry, modelled by a set of $n$ points $\boldsymbol{P_i} = [X_i, Y_i, Z_i]^{\mathrm{T}}$, undergoing a motion with instantaneous angular velocity $\Omega$ around an instantaneous axis of unit vector $\boldsymbol{a} = [a_x, a_y, a_z]^{\mathrm{T}}$, and instantaneous linear velocity $\boldsymbol{V} = [V_x, V_y, V_z]^{\mathrm{T}}$, 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 $\boldsymbol{P_i}$ will be collected with a delay $\tau_i$ proportional to the image line number on which $\boldsymbol{P_i}$ is projected. As illustrated in figure 3, $\tau_i$ is the time delay necessary to expose all the lines above the line which collects the light from $\boldsymbol{P_i}$. Therefore, to obtain the projection $\boldsymbol{m}_i = [u_i, v_i]^{\mathrm{T}}$ of $\boldsymbol{P_i}$, the pose parameters of the object must be corrected in equation 1 by integrating the motion during the time delay $\tau_i$. Since all the lines have the same exposure and integration time, we have $\tau_i = \tau v_i$ where $\tau$ is the time delay between two successive image line exposures. Thus $\tau = \frac{fp}{v_{max}}$ where $fp$ is the frame period and $v_{max}$ is the image height. Assuming that $\tau_i$

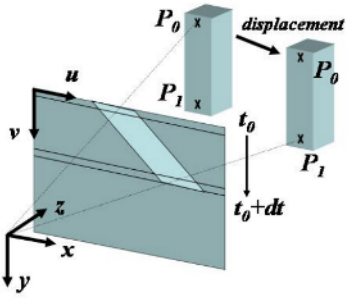


**Fig. 3.** Perspective projection of a moving 3D object: due to the time delay, points $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ are not projected from the same object pose

is short enough to consider uniform (but not necessarily small) motion during this interval, the object rotation during this interval is obtained thanks to the Rodrigues formula:

$$\delta \boldsymbol{R}_i = \boldsymbol{a}\boldsymbol{a}^T \left(1 - cos\left(\tau v_i \Omega\right)\right) + \boldsymbol{I} cos\left(\tau v_i \Omega\right) + \hat{\boldsymbol{a}} sin\left(\tau v_i \Omega\right)$$

where $\boldsymbol{I}$ is the 3×3 identity matrix and $\hat{\boldsymbol{a}}$ the antisymetric matrix of $\boldsymbol{a}$. The translation during the same interval, expressed in the static camera frame, is:

$$\delta \boldsymbol{T}_i = \tau v_i \boldsymbol{V}$$

Thus, equation 1 can be rewritten as follows:

$$s\tilde{\boldsymbol{m}}_i = \boldsymbol{K}\left[\delta \boldsymbol{R}_i \boldsymbol{R} \quad \boldsymbol{T} + \delta \boldsymbol{T}_i\right] \tilde{\boldsymbol{P}}_{\boldsymbol{i}} \tag{2}$$

where $\boldsymbol{R}$ and $\boldsymbol{T}$ represent now the instantaneous object pose at $t_0$. 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 $\tau$. One can note that it contains the unknown $v_i$ in its two sides. This is due to the fact that coordinates of the projected point on the image depend on both the kinematics of the object and the imager sensor scanning velocity.

## 5 Computing the Instantaneous Pose and Velocity of a Moving Object

In this section, we assume that a set of rigidly linked 3D points $\boldsymbol{P}_{\boldsymbol{i}}$ on a moving object are matched with their respective projections $\boldsymbol{m}_{\boldsymbol{i}}$ measured on an image taken with a calibrated rolling shutter camera. We want to use this list of 3D-2D correspondences to compute the instantaneous pose and velocity of the object at $t_0$.

### 5.1 Non-linear Method for 3D Objects

In the general case, the scale factor of equation 2 can be removed as follows:

$$\begin{aligned} u_i &= \alpha_u \frac{\boldsymbol{R}_i^{(1)}\boldsymbol{P}_i + T_i^{(x)}}{\boldsymbol{R}_i^{(3)}\boldsymbol{P}_i + T_i^{(z)}} + u_0 \triangleq \xi_i^{(u)}\left(\boldsymbol{R}, \boldsymbol{T}, \Omega, \boldsymbol{a}, \boldsymbol{V}\right) \\ v_i &= \alpha_v \frac{\boldsymbol{R}_i^{(2)}\boldsymbol{P}_i + T_i^{(y)}}{\boldsymbol{R}_i^{(3)}\boldsymbol{P}_i + T_i^{(z)}} + v_0 \triangleq \xi_i^{(v)}\left(\boldsymbol{R}, \boldsymbol{T}, \Omega, \boldsymbol{a}, \boldsymbol{V}\right) \end{aligned} \tag{3}$$

where $T_i^{(x,y,z)}$ are the components of $\boldsymbol{T}_i = \boldsymbol{T} + \delta\boldsymbol{T}_i$ and $\boldsymbol{R}_i^{(j)}$ is the $j^{th}$ row of $\boldsymbol{R}_i = \delta\boldsymbol{R}_i\boldsymbol{R}$. 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 $\tau$) parameters:

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

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

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

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 $\boldsymbol{R}$ is expressed by a unit quaternion representation $q(\boldsymbol{R})$. Thus, an additional equation, which forces the norm of $q(\boldsymbol{R})$ to 1, is added. It is obvious that this non-linear algorithm requires an initial guess to converge towards an accurate solution.

### 5.2 Linear Method for Planar Objects

In this section, a linear solution which may yield an initial guess of the pose and velocity parameters that can initialize the non-linear algorithm is developed. Assuming that $\tau_i$ is short enough to consider small and uniform motion during this interval, equation 1 can be rewritten, as in [7], as follows:

$$s\tilde{\boldsymbol{m}}_i = \boldsymbol{K}\left[\left(\boldsymbol{I} + \tau v_i\hat{\boldsymbol{\Omega}}\right)\boldsymbol{R} \quad \boldsymbol{T} + \tau v_i\boldsymbol{V}\right]\tilde{\boldsymbol{p}}_{\boldsymbol{i}} \tag{5}$$

where $\hat{\boldsymbol{\Omega}}$ is the antisymetric matrix associated to $\boldsymbol{\Omega} = \left[\Omega^{(x)}, \Omega^{(y)}, \Omega^{(z)}\right]^{\mathrm{T}}$. When points $\boldsymbol{p_i}$ are all coplanar, the projection equation 1 becomes a projective homography. By choosing an adequate object frame, all points can be written $\boldsymbol{p_i} = [X_i, Y_i, 0]^T$. Noting $\tilde{\boldsymbol{p}}_{\boldsymbol{i}} = [X_i, Y_i, 1]^T$, the classical projection equation is ([12]):

$$s\tilde{\boldsymbol{m}}_{\boldsymbol{i}} = \boldsymbol{H}\tilde{\boldsymbol{p}}_{\boldsymbol{i}} \tag{6}$$

where $\boldsymbol{H} = \boldsymbol{K}\left[\boldsymbol{r_1} \;\; \boldsymbol{r_2} \;\; \boldsymbol{T}\right]$ with $\boldsymbol{r_j}$ the $j^{th}$ column of $\boldsymbol{R}$. As for the 3D object case, the velocity parameters are integrated in the projection equation as follows:

$$s\tilde{\boldsymbol{m}}_{\boldsymbol{i}} = \boldsymbol{H}\tilde{\boldsymbol{p}}_{\boldsymbol{i}} + \tau v_i\boldsymbol{D}\tilde{\boldsymbol{p}}_{\boldsymbol{i}} \tag{7}$$

where $\boldsymbol{D} = \boldsymbol{K}[\boldsymbol{\omega_1}\ \ \boldsymbol{\omega_2}\ \ \boldsymbol{V}]$ with $\boldsymbol{\omega_j}$ the $j^{th}$ column of $\boldsymbol{\omega} = \hat{\boldsymbol{\Omega}}\boldsymbol{R}$. From equation 7 one can derive a cross product which must be null:

$$\tilde{\boldsymbol{m_i}} \times (\boldsymbol{H}\tilde{\boldsymbol{p_i}} + \tau v_i \boldsymbol{D}\tilde{\boldsymbol{p_i}}) = 0$$

which yields the following equation:

$$\boldsymbol{A}\boldsymbol{x} = 0 \tag{8}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \tilde{\boldsymbol{p_i}}^T & \boldsymbol{0}^T & -u_i\tilde{\boldsymbol{p_i}}^T & \tau v_i\tilde{\boldsymbol{p_i}}^T & \tau v_i\boldsymbol{0}^T & -\tau v_i u_i\tilde{\boldsymbol{p_i}}^T \\ \boldsymbol{0}^T & \tilde{\boldsymbol{p_i}}^T & -v_i\tilde{\boldsymbol{p_i}}^T & \tau v_i\boldsymbol{0}^T & \tau v_i\tilde{\boldsymbol{p_i}}^T & -\tau v_i^2\tilde{\boldsymbol{p_i}}^T \end{bmatrix}$$

is a $18 \times 2n$ matrix and $\boldsymbol{x} = \left[\boldsymbol{h_1}^T\boldsymbol{h_2}^T\boldsymbol{h_3}^T\boldsymbol{d_1}^T\boldsymbol{d_2}^T\boldsymbol{d_3}^T\right]^T$ is the unknown vector with $\boldsymbol{h_j}$, $\boldsymbol{d_j}$ being the $j^{th}$ columns of respectively $\boldsymbol{H}$ and $\boldsymbol{D}$. Equation 8 can be solved for $\boldsymbol{x}$ using singular value decomposition (SVD) as explained in [4].

Once $\boldsymbol{x}$ is computed, the pose parameters are derived, following [12], as follows:

$$\boldsymbol{r_1} = \lambda_h\boldsymbol{K}^{-1}\boldsymbol{h_1}, \quad \boldsymbol{r_2} = \lambda_h\boldsymbol{K}^{-1}\boldsymbol{h_2}, \quad \boldsymbol{r_3} = \boldsymbol{r_1} \times \boldsymbol{r_2}, \quad \boldsymbol{T} = \lambda_h\boldsymbol{K}^{-1}\boldsymbol{h_3} \tag{9}$$

where $\lambda_h = \frac{1}{||\boldsymbol{K}^{-1}\boldsymbol{h_1}||}$.

The translational velocity vector is obtained by:

$$\boldsymbol{V} = \lambda\boldsymbol{K}^{-1}\boldsymbol{d_3} \tag{10}$$

and angular velocity parameters are obtained by first computing columns 1 and 2 of matrix $\boldsymbol{\omega}$:

$$\boldsymbol{\omega_1} = \lambda_d\boldsymbol{K}^{-1}\boldsymbol{d_1}, \qquad \boldsymbol{\omega_2} = \lambda_d\boldsymbol{K}^{-1}\boldsymbol{d_2} \tag{11}$$

where $\lambda_d = \frac{1}{||\boldsymbol{K}^{-1}\boldsymbol{d_1}||}$, and then extracting $\boldsymbol{\Omega}$ as follows:

$$\Omega^{(x)} = \frac{\boldsymbol{\omega_{12}}R_{12} - \boldsymbol{\omega_{22}}R_{11}}{R_{32}R_{11} - R_{31}R_{12}}, \quad \Omega^{(y)} = \frac{\boldsymbol{\omega_{11}}R_{22} - \boldsymbol{\omega_{21}}R_{21}}{R_{31}R_{22} - R_{32}R_{21}}, \quad \Omega^{(z)} = \frac{\boldsymbol{\omega_{11}}R_{32} - \boldsymbol{\omega_{21}}R_{31}}{R_{31}R_{22} - R_{32}R_{21}} \tag{12}$$

## 6 Experiments

The aim of this experimental evaluation is first to illustrate our pose recovery algorithm accuracy in comparison with classical algorithms under the same acquisition conditions, and second, to show its performances as a velocity sensor. The 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 a Silicon Imaging CMOS Rolling Shutter camera SI1280M-CL, calibrated using the method described in [5]. 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 estimation 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

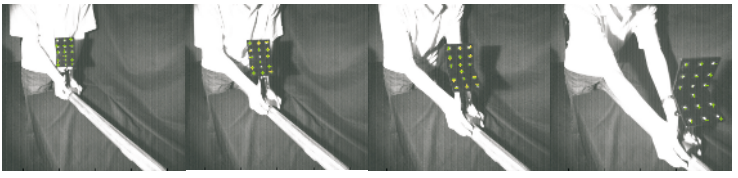

**Fig. 4.** Image samples of pure translational motion

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

| | Linear algorithm | | Classical algorithm | | Non linear algorithm | |
|---|---|---|---|---|---|---|
| Image number | RMS-$u$ | RMS-$v$ | RMS-$u$ | RMS-$v$ | RMS-$u$ | RMS-$v$ |
| 1 | 9.30 | 16.00 | 0.14 | 0.12 | 0.15 | 0.13 |
| 2 | 14.08 | 17.95 | 1.71 | 1.99 | 0.10 | 0.09 |
| 3 | 5.24 | 8.06 | 3.95 | 4.18 | 0.11 | 0.09 |
| 4 | 11.33 | 14.21 | 7.09 | 7.31 | 0.09 | 0.07 |
| 5 | 9.25 | 11.02 | 5.56 | 6.73 | 0.13 | 0.12 |
| 6 | 12.26 | 13.04 | 1.87 | 3.02 | 0.18 | 0.11 |
| 7 | 9.85 | 11.56 | 0.25 | 0.12 | 0.25 | 0.17 |

algorithm, and compared with results obtained using the classical pose recovery algorithm described in [5]. In the latter, an initial guess is first computed by the algorithm of Dementhon [2] 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 static views were computed accurately using the classical algorithm. They serve as ground-truth values to validate our algorithm when velocity is null. 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 algorithms and the rolling shutter projection model from the other hand-side. Column 2 shows results obtained with the linear algorithm using only nine coplanar points of the pattern. Note that these results are obtained using the minimum number of correspondences required from the linear algorithm and can thus be improved. Anyhow, even under these conditions, the method remains accurate enough to correctly initialize the non-linear algorithm. Results in columns 3 and 4 show errors obtained using respectively a classical algorithm and our non-linear algorithm. One can see 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.

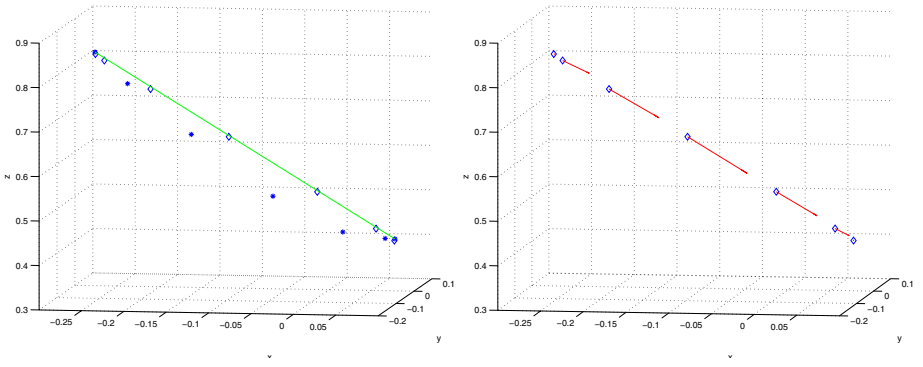


**Fig. 5.** Pose and velocity results: reconstructed trajectory (left image), translational velocity vectors (right image)

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

| Image number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Classical algorithm | 0.00 | 0.19 | 0.15 | 1.38 | 3.00 | 4.54 | 0.00 |
| Our algorithm | 0.28 | 0.34 | 0.26 | 0.32 | 0.32 | 0.11 | 0.10 |

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

| Image number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Dementhon's algorithm | 0.00 | 2.05 | 4.52 | 6.93 | 6.69 | 3.39 | 0.30 |
| our algorithm | 0.17 | 0.13 | 0.17 | 0.34 | 1.09 | 0.91 | 0.40 |

Let us now analyze pose recovery results shown in figure 5. The left-hand side of this figure shows 3D translational pose parameters obtained by our non-linear 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 due to the distortions 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.

Another result analysis concerns the velocity parameters. Figure 5 shows that the translational velocity vector is clearly parallel to the translational axis (up to noise influence). Table 4 represents magnitude of computed velocity vectors

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

| Image number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Measured values | 0.00 | 1.22 | 2.02 | 2.32 | 1.55 | 0.49 | 0.00 |
| Computed values | 0.06 | 1.10 | 1.92 | 2.23 | 1.54 | 0.50 | 0.02 |

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

| Image number | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| our algorithm | 0.04 | 0.07 | 0.05 | 0.01 | 0.15 | 0.11 | 0.12 |

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 mean value during each frame period. Results show that the algorithm recovers 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.

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 $\boldsymbol{N} = [0.05, 0.01, -0.98]^{\mathrm{T}}$. Thus, $\boldsymbol{N}$ 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, where rolling shutter effects appear clearly. The left part of figure 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 reconstructed 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). Moreover, the manifold of instantaneous rotation axis vectors was also correctly oriented. Indeed, the mean value of the angles between

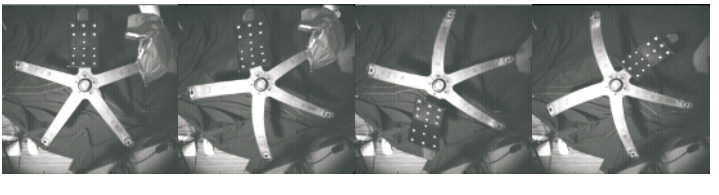

**Fig. 6.** Image samples of coupled rotational and translational motions

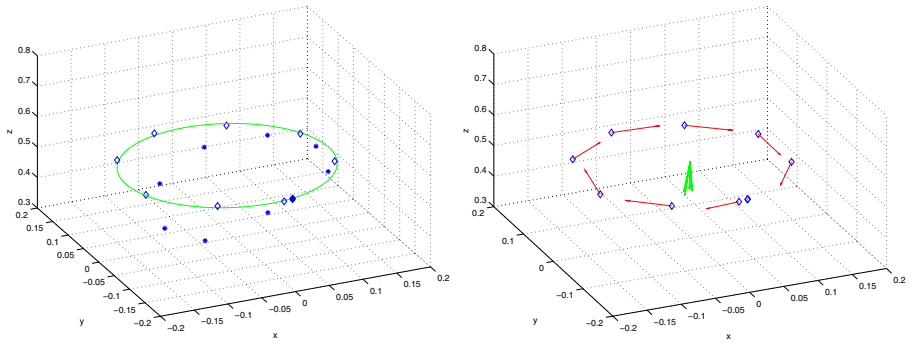

**Fig. 7.** Pose and velocity results for coupled rotational and translational motion: reconstructed trajectory (left image), rotational and translational velocities (right image)

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

| Image number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Measured values | 0.00 | 1.50 | 9.00 | 11.20 | 10.50 | 10.20 | 10.10 | 10.00 | 10.00 | 7.50 |
| Computed values | 0.06 | 1.20 | 8.55 | 10.38 | 10.32 | 10.30 | 9.80 | 9.90 | 9.73 | 8.01 |

the computed rotation axis and $\boldsymbol{N}$ is 0.50 degrees. Results in table 6 shows a comparison of the computed rotational velocity magnitudes and the values estimated from each two successive images.

## 7 Conclusion and Perspectives

An original method for computing simultaneously the pose and instantaneous velocity (both translational and rotational) of rigid objects was presented. It profits from an inherent defect of rolling shutter CMOS cameras consisting in exposing one after the other the rows of the image, yielding optical distortions due to high object velocity. Consequently, a novel model of the perspective projection of a moving 3D point onto a rolling shutter camera image was introduced. From this model, an error function equivalent to collinearity equations in camera calibration was defined in the case of both planar and non-planar objects. In the planar case, minimizing the error function takes the form of a linear system, while in the non-planar case it is obtained through bundle adjustment techniques and non-linear optimization. The approach was validated on real data showing its relevance and feasibility. Hence, the proposed method in the non planar case is not only as accurate as similar classical algorithms in the case of static objects, but also preserves the accuracy of pose estimation when the object is moving. However, in the planar case, the experimental results were only accurate enough to initialize the non-planar method but these results were obtained with the minimal number of points.

In addition to pose estimation, the proposed method gives the instantaneous velocity using a single view. Thus, it avoids the use of finite differences between successive images (and the associated constant velocity assumption) to estimate a 3D object velocity. Hence, carefully taking into account rolling shutter turns a low cost imager into a powerful pose and velocity sensor. Indeed, such an original tool can be useful for many research areas. For instance, instantaneous velocity information may be used as evolution models in motion tracking to predict the state of observed moving patterns. It may also have applications in robotics, either in visual servoing or dynamic identification. However, in the latter case, accuracy needs to be quantified by independent means on accurate ground-truth values within an evaluation framework, such as laser interferometry or accurate high-speed mechanisms, before the proposed method can serve as a metrological tool.

From a more theoretical point of view, several issues open. First, the proposed method uses a rolling shutter camera model based on instantaneous row exposure, but it should be easily extendable to more general models where each pixel has a different exposure time. One could also imagine that an uncalibrated version of this method could be derived for applications where Euclidean information is not necessary (virtual/augmented reality or qualitative motion reconstruction, for instance). Finally, another point of interest could be the calibration of the whole system (lens distortion + intrinsic parameters + rolling shutter time) in a single procedure.

## References

[1] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. In *Proc. IEEE International Conference on Computer Vision Systems*, New York, USA, January 2006.

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

[3] M. Dhome, M. Richetin, J. T. Lapreste, 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.

[4] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2000.

[5] JM. 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.

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

[7] M. Meingast, C. Geyer, and S. Sastry. Geometric models of rolling-shutter cameras. In *Proc. of the 6th Workshop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras*, Beijing, China, October 2005.

[8] T. Q. Phong, R. Horaud, 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.

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

[10] R. Y. 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.

[11] B. Wilburn, N. Joshi, V. Vaish, M. Levoy, and M. Horowitz. High-speed videography using a dense camera array. In *IEEE Society Conference on Pattern Recognition (CVPR'04)*, 2004.

[12] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[13] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall. Mosaicing new views: The crossed-slits projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(6):741–754, 2003.