Appeared in the Proceedings of CVPR'07

# Kinematics from Lines in a Single Rolling Shutter Image

Omar Ait-Aider
LASMEA (CNRS / UBP)
Clermont-Ferrand, France
Omar.Ait-Aider@univ-bpclermont.fr

Adrien Bartoli
LASMEA (CNRS / UBP)
Clermont-Ferrand, France
Adrien.Bartoli@gmail.com

Nicolas Andreff
LASMEA (CNRS / UBP)
Clermont-Ferrand, France
Nicolas.Andreff@ifma.fr

## Abstract

*Recent work shows that recovering pose and velocity from a single view of a moving rigid object is possible with a rolling shutter camera, based on feature point correspondences.*

*We extend this method to line correspondences. Owing to the combined effect of rolling shutter and object motion, straight lines are distorted to curves as they get imaged with a rolling shutter camera. Lines thus capture more information than points, which is not the case with standard projection models for which both points and lines give two constraints.*

*We extend the standard line reprojection error, and propose a nonlinear method for retrieving a solution to the pose and velocity computation problem. A careful inspection of the design matrix in the normal equations reveals that it is highly sparse and patterned. We propose a blockwise solution procedure based on bundle-adjustment-like sparse inversion. This makes nonlinear optimization fast and numerically stable. The method is validated using real data.*

## 1. Introduction

CMOS cameras offer several advantages: low cost, low power demand, easy region of interest selection, on-chip characteristics and high frame rate. This makes them a natural fit for wireless hand-held applications, visual servoing and some in-vehicle uses. In the last years, the performances of CMOS sensors in terms of signal-to-noise ratio have been considerably improved, reaching the level of CCD sensors. This has made CMOS cameras more and more used by the vision community in applications for which a high frame rate and accurate feature detection is necessary (fast robot control and identification, road traffic, balistic).

Standard and cheap CMOS cameras frequently use rolling shutter sensors. This shuttering mode enables adequate exposure time without reducing the frame rate by overlapping exposure and readout. It reduces the number of in-pixel transistors, improving the fill factor (percentage of the pixel array sensitive to light) and the signal-to-noise ratio. The drawback of rolling shutter cameras 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). This distortion may represent a major obstacle in tasks such as localization, 3D 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.

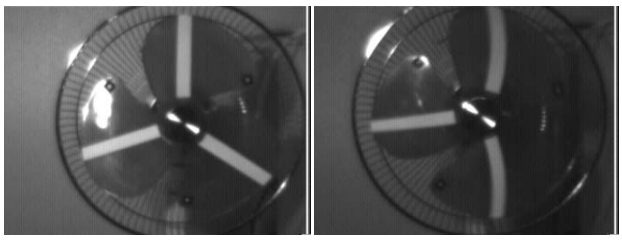

Figure 1. An example of distortion of a rotating ventilator observed with a rolling shutter camera: static object (left image) and moving object (right image).

The work done by Wilburn et al. [11] concerned the correction of image deformations due to rolling shutter by constructing a single image using several images from a dense camera array. Knowing 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. In [7] Meingast describes an approximated projection model of rolling shutter cameras which, in the case of fronto-parallel motion, is similar to a Crossed-Slits camera model [12]. Ait-Aider et al. [1] present a general and exact perspective projection model by removing the assumption of small motion during image acquisition. A nonlinear algorithm for simultaneous pose and velocity computation

using a single view is then developed. It extends bundle adjustment with point correspondences, to the case of moving objects observed with a rolling shutter camera. A linear algorithm is proposed in the particular case of planar objects. It provides an initial estimate of the pose and velocity parameters. To our knowledge, there is no other work in the vision community literature on taking into account effects of rolling shutter in pose recovery algorithms nor on exploiting them to compute the velocity parameters using a single view. Indeed, traditional pose recovery methods (for instance [5, 8, 2, 3, 10]) make the assumption that all image sensor pixels are exposed simultaneously.

In this paper, we propose an extension of the pose and velocity computation algorithm presented in [1] to line correspondences. When observed with a rolling shutter camera, a moving 3D line is projected to a curve on the image. A typical example is the observation of a polyhedral object. Straight edges are detected by segmenting image into contours. Why is this extension worthwhile? First, in the point-based algorithm of [1] the local distorsions in the image point neighborhood are neglected so that interest point detectors and descriptors such as Harris or SIFT remain usable. When motion artefacts are too important, this approximation may result in both false negatives in correlation-based matching and bad point localization. Using curves, even local distortions are modelled with an accuracy only bounded by image resolution. Second, for a rolling shutter projection model, lines capture more information about the motion than points. This is not the case with standard projection models for which both points and lines give two constraints. Finally, using all the pixels of a contour provides redundant information which is exploited against noise. Note that the pixels are here used directly without high level image processing. Conversely, detecting a straight line (with a classical camera) implies finding the function which best fits aligned pixels.

The main difficulty is that one can not derive an algebraic formulation of the curve corresponding to the projection of a straight line for a general motion. It is also difficult to find a metric which measures the distance between two such curves. We write the error between an observed and a reprojected curve as the sum of distances between the observed contour pixels and the reprojected 3D line. The point-to-curve distance does not however have a closed-form solution in general. The error is thus computed in practice by introducing, for each contour point, a corresponding point moving along the 3D line, so as to minimize the distance. This is done by introducing additional unknowns called nuisance variables in addition to the desired pose and velocity parameters. This results in a large but sparse Jacobian matrix. The latter property is taken into account in the resolution process. The optimization is achieved thanks to a blockwise solution procedure based on bundle-adjustment-like sparse inversion [9]. This ensures fast and numerically stable nonlinear optimization.

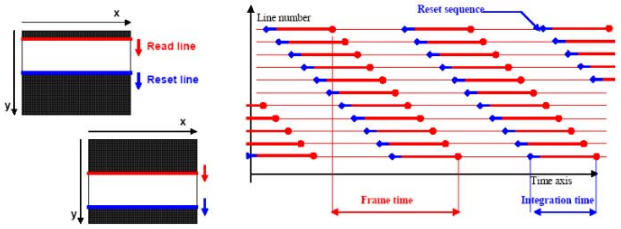


Figure 2. Reset and reading chronograms in rolling shutter sensor (Silicon Imaging documentation).

In section 2, we briefly recall the point projection model of a rolling shutter camera. In section 3, we formulate the pose and velocity computation problem under the form of a cost function derived from a set of line-to-curve correspondences. In section 4, we focus on improving efficiency and numerical stability of the nonlinear optimization of the cost function by exploiting the sparse structure of the Jacobian matrix. Finally, experimental results obtained with real images illustrate the performances of the method.

## 2. Rolling Shutter Camera Projection Model

In a CMOS camera operating in rolling shutter mode, the sensor pixels are exposed sequentially 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 row in the image has the same amount of integration, however the starting and ending time of integration are shifted in time as the image is scanned (rolled) out of the sensor array, as shown in Fig.2. If an observed object is moving during the integration time, some artefacts may appear and its image is distorted. The faster the object the larger the distortion. A simple case where the object undergoes a pure translational motion is illustrated on Fig.3.

Assume that an object of known geometry, modelled by a set of $n$ points $\mathbf{P}_i = [X_i, Y_i, Z_i, 1]^{\mathrm{T}}$, undergoing a motion with instantaneous angular velocity $\Omega$ around an instantaneous axis of unit vector $\mathbf{a} = [a_x, a_y, a_z]^{\mathrm{T}}$, and instantaneous linear velocity $\mathbf{V} = [V_x, V_y, V_z, 1]^{\mathrm{T}}$, is snapped with a rolling shutter camera at time $t_0$. Denoting $\mathbf{R}$ and $\mathbf{T}$ the instantaneous object pose at $t_0$, it was demonstrated in [1] that the 2D projection $\mathbf{m}_i = [u_i, v_i, 1]^{\mathrm{T}}$ of $\mathbf{P}_i$ can be expressed up to an arbitrary scale factor $s$ as follows:

$$s\mathbf{m}_i = \mathbf{K}\left[\mathbf{R}\delta\mathbf{R}_i \quad \mathbf{T} + \delta\mathbf{T}_i\right]\mathbf{P}_i \qquad (1)$$

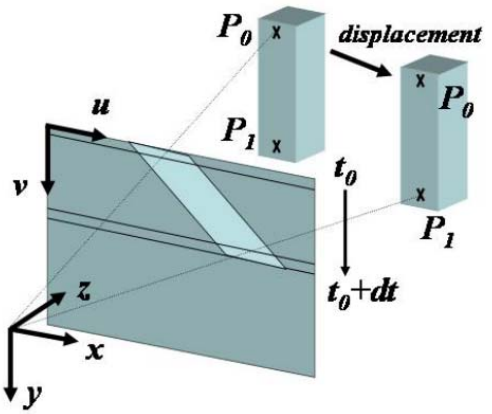


Figure 3. Perspective projection of a moving 3D object: due to the time delay, points $\mathbf{P}_0$ and $\mathbf{P}_1$ are not projected under the same pose

.

with:

$$\delta\mathbf{R}_i = \mathbf{a}\mathbf{a}^T\left(1 - cos\left(\tau v_i \Omega\right)\right) + \mathbf{I}cos\left(\tau v_i \Omega\right) + \hat{\mathbf{a}}sin\left(\tau v_i \Omega\right) \tag{2}$$

and:

$$\delta\mathbf{T}_i = \tau v_i \mathbf{V} \tag{3}$$

where $\mathbf{I}$ is the 3×3 identity matrix, $\hat{\mathbf{a}}$ is the antisymmetric matrix associated to $\mathbf{a}$, $\tau$ is the image scanning speed (in raws per second) and $\mathbf{K}$ contains the classical intrinsic parameters of a pinhole camera. Note that $\mathbf{V}$ is the sum of two vectors $\mathbf{V}_L$ and $\mathbf{V}_R$. The first component is due to the pure translational motion and is thus expressed in the camera frame. The second component is induced by the rotation (tangential velocity) and must be expressed in the object frame. Thus we have $\mathbf{V} = \mathbf{R}\mathbf{V}_R + \mathbf{V}_L$.

Equation (5) 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 $\tau$. Note that it contains the unknown $v_i$ on 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.

## 3. Pose and Velocity Computation with Lines

If a moving polyhedral object is observed with a rolling shutter camera, its straight edges are projected into the image as curved contours. Assume that a set of $N$ straight edges, defined in the object frame by their direction vectors $\mathbf{L}_k$, are matched with a set of curved image contours $\mathbf{l}_k$. Considering an arbitrary point $\mathbf{M}_{k0}$ on $\mathbf{L}_k$, any other point $\mathbf{M}_{ki}$ on the latter edge can be expressed in the object frame as follows:

$$\mathbf{M}_{ki} = \mathbf{M}_{k0} + \sigma_{ki}\mathbf{L}_k \tag{4}$$

Thus, for each pixel on the curve one can write the following projection equation:

$$s\mathbf{m}_{ki} = \mathbf{K}\left[\mathbf{R}\delta\mathbf{R}_i \quad \mathbf{T} + \delta\mathbf{T}_i\right]\left(\mathbf{M}_{ki} + \sigma_{ki}\mathbf{L}_k\right) \tag{5}$$

This means that each pixel of the contour yields a pair of contraints of the form:

$$\begin{aligned} u_{ki} &= \alpha_u \frac{\mathbf{R}_{1i}(\mathbf{M}_{ki}+\sigma_{ki}\mathbf{L}_k)+T_{xi}}{\mathbf{R}_{3i}(\mathbf{M}_{ki}+\sigma_{ki}\mathbf{L}_k)+T_{zi}} + u_0 \\ v_{ki} &= \alpha_u \frac{\mathbf{R}_{2i}(\mathbf{M}_{ki}+\sigma_{ki}\mathbf{L}_k)+T_{yi}}{\mathbf{R}_{3i}(\mathbf{M}_{ki}+\sigma_{ki}\mathbf{L}_k)+T_{zi}} + v_0 \end{aligned} \tag{6}$$

It is obvious that matching a 3D straight edge with an image curve does not tell us for each contour pixel which is the corresponding 3D edge point. In other words, the values of $\sigma_{ki}$ are unknown. Thus, equation (6) can be expressed as follows:

$$\begin{aligned} u_{ki} &\triangleq \xi_{uki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right) \\ v_{ki} &\triangleq \xi_{vki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right) \end{aligned} \tag{7}$$

where $\mathbf{\Sigma}$ is the vector of all the parameters $\sigma_{ki}$.

Considering the observation of $m$ pixels $[\hat{u}_{ki}, \hat{v}_{ki}]$ on each one of the $n$ image curves matched with a straight edge, and comparing them with the theoretical projections using (6) we obtain a $n \times m$ equation system representing the reprojection error:

$$\epsilon_{ki} = \begin{bmatrix} \hat{u}_{ki} - \xi_{uki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right) \\ \hat{v}_{ki} - \xi_{vki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right) \end{bmatrix} \tag{8}$$

From this, a cost function in the least square sense is expressed with respect to pose and velocity parameters $\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}$ and also with respect to the so-called nuisance variables $\mathbf{\Sigma}$:

$$\begin{aligned} \epsilon = \sum_{k=1}^{n}\sum_{i=1}^{m} &\left[\hat{u}_{ki} - \xi_{uki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right)\right]^2 \\ &+ \left[\hat{v}_{ki} - \xi_{vki}\left(\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}\right)\right]^2 \end{aligned} \tag{9}$$

This cost function is minimized using the Levenberg-Marquardt algorithm [6].

## 4. Blockwise Nonlinear Minimization

Let $\mathbf{y}$ be the vector of image projections in the left hand side of equation (7) and $\mathbf{x}$ the vector of pose, velocity and nuisance parameters $\mathbf{\Pi} = [\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}, \mathbf{\Sigma}]$. The relationship between these two vectors is denoted $\mathbf{y} = \xi\left(\mathbf{x}\right)$. Given

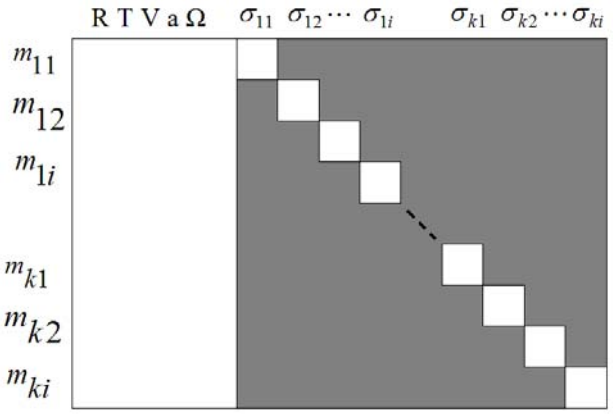


Figure 4. Structure of the Jacobian matrix $\mathbf{J}$.

a set of noisy observations $\hat{\mathbf{y}}$, we want to converge toward the value $\hat{\mathbf{x}}$ so that the error $\hat{\mathbf{e}}$ in the relation $\hat{\mathbf{y}} = \xi(\hat{\mathbf{x}}) + \hat{\mathbf{e}}$ is minimal. The minimization starts from an initial guess $\mathbf{x_0}$ and is updated iteratively by applying variations $\delta$ to $\hat{\mathbf{x}}$. This is generally achieved by assuming local linearity of $\xi$ under which one can write $\xi(\mathbf{x}_0 + \delta) = \xi(\mathbf{x}_0) + \mathbf{J}\delta$ with $\mathbf{J}$ the Jacobian matrix of $\xi(\mathbf{x})$. This implies to solve at each iteration the so called normal equations:

$$\mathbf{J}^{\mathrm{T}}\mathbf{J}\delta = \mathbf{J}^{\mathrm{T}}\mathbf{e} \tag{10}$$

In our case, the Jacobian matrix is very sparse because each image pixel $\mathbf{m}_{ki}$ on a matched contour depends on all the pose and velocity parameters $\mathbf{P} = [\mathbf{R}, \mathbf{T}, \Omega, \mathbf{a}, \mathbf{V}]$ but only on its own nuisance variable $\sigma_{ki}$. Thus $\frac{\partial \mathbf{m}_{ki}}{\partial \sigma_{lj}} \neq 0$ only for $k = l$ and $i = j$ but is null elsewhere. This results for $\mathbf{J}$ in the structure illustrated in Fig.4, which in turn produces the $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ pattern illustrated in Fig.5 with:

$$\mathbf{U}_{p,q} = \sum_k \sum_i \left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{P}_p}\right)\left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{P}_q}\right) \tag{11}$$

$$\mathbf{V}_{p,q} = \sum_k \sum_i \left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{\Sigma}_p}\right)\left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{\Sigma}_q}\right) \tag{12}$$

$$\mathbf{W}_{p,q} = \sum_k \sum_i \left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{P}_p}\right)\left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{\Sigma}_q}\right) \tag{13}$$

A similar strucutre is exploited in bundle adjustment, for instance in [9], to reduce the computational cost for solving the normal equations by rewriting it as follows:

$$\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \delta_{\mathbf{P}} \\ \delta_{\mathbf{\Sigma}} \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{\mathbf{P}} \\ \mathbf{E}_{\Sigma} \end{bmatrix} \tag{14}$$

where $\delta_{\mathbf{P}}$ and $\delta_{\mathbf{\Sigma}}$ contains the small variations of respectively $\mathbf{P}$ and $\mathbf{\Sigma}$. The blocks $\mathbf{E}_{\mathbf{P}}$ and $\mathbf{E}_{\mathbf{\Sigma}}$ form the vector on the right hand side of the normal equation (10). Their components are defined as follows:

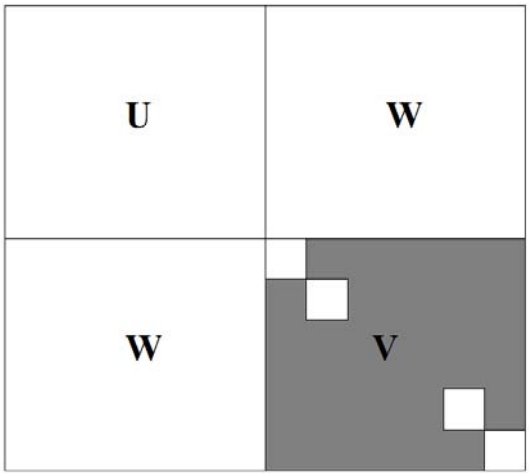


Figure 5. Structure of $\mathbf{J}^T\mathbf{J}$ in the normal equations.

$$\mathbf{E}_{\mathbf{P}q} = \sum_k \sum_i \left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{\Pi}_q}\right) \epsilon_{ki} \tag{15}$$

$$\mathbf{E}_{\mathbf{\Sigma}q} = \sum_k \sum_i \left(\frac{\partial \mathbf{m}_{ki}}{\partial \mathbf{\Pi}_q}\right) \epsilon_{ki} \tag{16}$$

Equation (14) can be rewritten as follows:

$$\begin{bmatrix} \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{W}^{\mathrm{T}} & \mathbf{V} \end{bmatrix} \begin{bmatrix} \delta_{\mathbf{P}} \\ \delta_{\mathbf{\Sigma}} \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{\mathbf{P}} - \mathbf{W}\mathbf{V}^{-1}\mathbf{E}_{\Sigma} \\ \mathbf{E}_{\Sigma} \end{bmatrix} \tag{17}$$

which can be decomposed into two separate equation systems:

$$\left(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^{\mathrm{T}}\right)\delta_{\mathbf{P}} = \mathbf{E}_{\mathbf{P}} - \mathbf{W}\mathbf{V}^{-1}\mathbf{E}_{\Sigma} \tag{18}$$

and

$$\delta_{\mathbf{\Sigma}} = \mathbf{V}^{-1}\left(\mathbf{E}_{\mathbf{\Sigma}} - \mathbf{W}^{\mathrm{T}}\delta_{\mathbf{P}}\right) \tag{19}$$

Equation (18) can be solved very efficiently because $\mathbf{V}$ is diagonal. Equation (19) is then solved by substituting the solution of (18).

## 5. Experimental Evaluation

The pose and velocity computation algorithm was tested on real image data. A reference 3D polyhedral object with both point and line features was used. A Silicon Imaging CMOS rolling shutter camera SI1280M-CL was first calibrated using the method described in [4] and then used to capture image sequences of the reference polyhedral object while undergoing rotational and translational motion at a high velocity. Fig.6 shows samples of images from these sequences.

Table 1. Differences between results of point-based algorithm and line-based algorithm (Mean value and standard deviation on the basis of 20 test images)

| Parameter | R (deg) | T (m) | V (perc.) | Omega (perc.) |
|---|---|---|---|---|
| mean value | 1.4 | 0.015 | 1.55 | 2.60 |
| stand. dev. | 1.0 | 0.006 | 1.05 | 1.80 |

Acquisition was done with a resolution of 640×480 square pixels and at a rate of 30 frames per second so that $\tau = 39.5 \times 10^{-6}$ s.

Point features served to generate groundtruth values for pose and velocity parameters. Indeed, since the point based algorithm was validated and evaluated in [1] using ground truth values, it was used here as a reference, simultaneously with the line-based algorithm and on the same image data. Image point coordinates were accurately obtained to sub-pixel accuracy estimation of the white spot centers and corrected according to the lens distortion parameters.

Thin image curves were detected thanks to Canny's criterion and chained to obtain contour curves. No additional processing was done on the contour pixels. The pixel coordinates were used directly in the algorithm.

For the nonlinear optimization, all nuisance and velocity parameters were initialized to zero. The position was initialized at $[0, 0, 1]^T$ (the object is in front of the camera) with a random orientation.

Both point and line correspondences with the model points and lines were established with a supervised method. The pose and velocity parameters were computed for each image using first our line-based algorithm, and compared with results obtained using the point-based algorithm and the classical pose recovery algorithm described in [4]. In the latter, an initial estimate of the solution is first computed using the algorithm of Dementhon [2] and then the pose parameters are refined thanks to a nonlinear method.

As shown in Fig.7, the trajectories and velocities computed by the line and the point-based algorithms are very close. The differences between the position, orientation, and velocity computed by the two algorithms are given in Table 1. Pose results obtained with a classical algorithm (which does not take into account the rolling shutter effects) show a shift proportional to the speed in the direction of the motion.

Fig.8 shows an example of correcting the object image by removing the velocity parameters in the projection equation. This corresponds to a global shutter image taken at $t_0$ (the instant of exposure of the first line of the sensor).

## 5.1. Conclusion and Perspectives

We presented a method for simultaneously computing the pose and instantaneous velocity (both translational and rotational) of rigid objects from a single rolling shutter image of straight lines. It benefits of 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. The approach extends

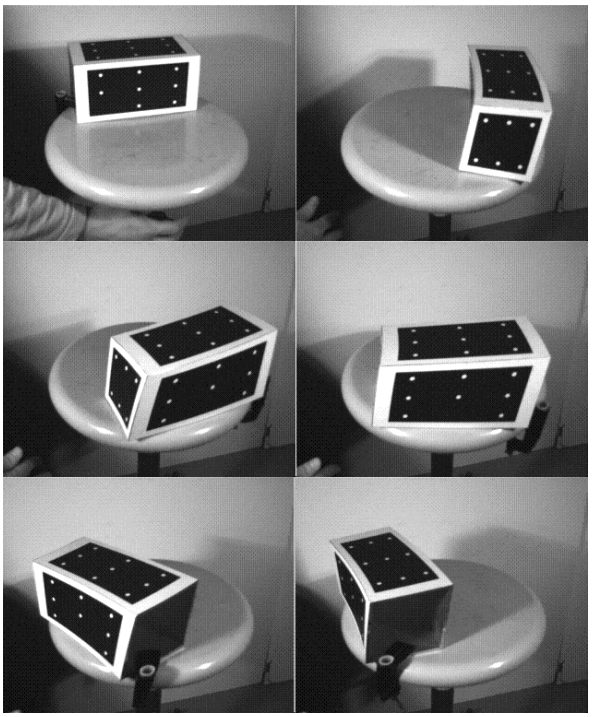

Figure 6. Samples of rolling shutter images of the moving reference object.

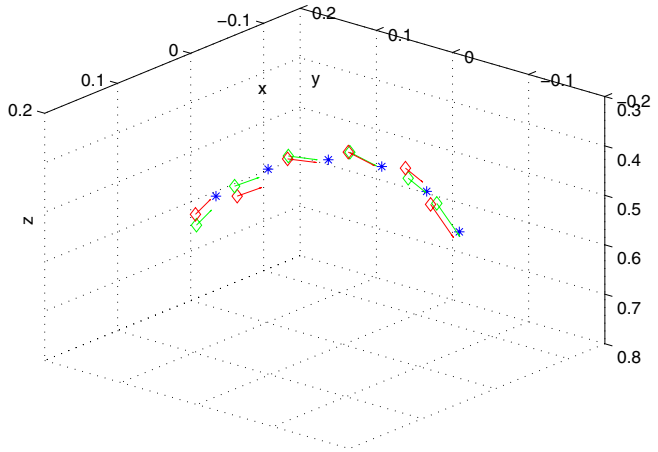


Figure 7. A comparison between two sets of trajectories and velocities computed by the line-based (red curve and arrows) and the point-based (green curve and arrows) algorithms respectively. The '*' symbols represent results obtained with a classical algorithm which does not take into account rolling shutter distorsions

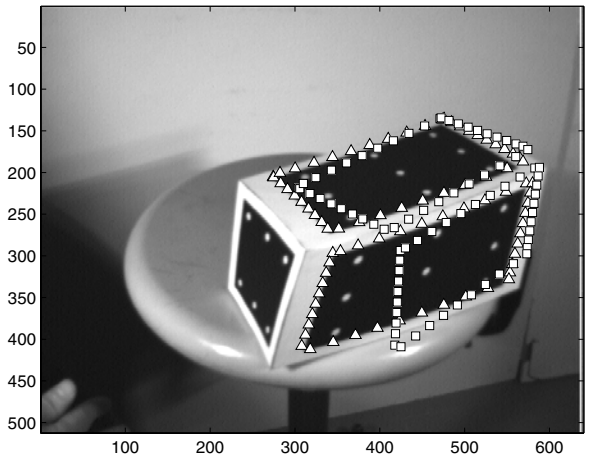


Figure 8. An example of reprojecting edge points using a rolling shutter model (triangles). Image correction: lines represented by square symbols are obtained by removing rolling shutter distorsions.

previous point-based methods to line correspondences. This offers, in the case of rolling shutter projection, real advantages (more information about motion, redundant information). An efficient optimization procedure was also proposed to improve numerical stability and computational cost of the approach.

The approach was validated on real data showing its relevance and feasibility. Hence, the proposed method is as accurate as similar classical algorithms in the case of static objects, but also preserves the accuracy of pose estimation when the object moves. 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 a 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. In the latter domain our approach can make the difference when image processing leaves little time to other tasks (control, data fusion) by reducing drastically the amount of data necessary for motion analysis by using a single view instead of image sequences.

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 could be easily extended 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). This certainly will make this work relevant to a broader range of scenes (where the identity of lines is not known a-priori).

## References

[1] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Simultaneous object pose and velocity computation using a single view from a rolling shutter camera. In *Proc. European Conference on Computer Vision, Vol. 2, pp. 56-68*, Graz, Austria, January 2006. 1, 2, 5

[2] 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, 5

[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. 2

[4] J. Lavest, M. Viala, and M. Dhome. Do we really need an accurate calibration pattern to achieve a reliable camera calibration. In *Proc. European Conference on Computer Vision ECCV, pp. 158-174*, Freiburg, Germany, June 1998. 4, 5

[5] D. G. Lowe. Fitting parameterized three-dimensional models to image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):441–450, May 1991. 2

[6] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(6):431–441, 1963. 3

[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. 1

[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. 2

[9] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment-a modern synthesis. In *Proc. of the International Workshop on Vision Algorithms: Theory and Practice, pp. 298-372*, London, UK, 1999. 2, 4

[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. 2

[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. 1

[12] 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. 1