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, ballistic).

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.

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 distortions 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 cannot 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.

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 \( P_i = [X_i, Y_i, Z_i, 1]^T \), undergoing a motion with instantaneous angular velocity \( \Omega \) around an instantaneous axis of unit vector \( a = [a_x, a_y, a_z]^T \), and instantaneous linear velocity \( V = [V_x, V_y, V_z, 1]^T \), is snapped with a rolling shutter camera at time \( t_0 \). Denoting \( R \) and \( T \) the instantaneous object pose at \( t_0 \), it was demonstrated in [1] that the 2D projection \( m_i = [u_i, v_i, 1]^T \) of \( P_i \) can be expressed up to an arbitrary scale factor \( s \) as follows:

\[
sm_i = K [R \delta R_i + T + \delta T_i] P_i
\] (1)
edges, defined in the object frame by their direction vectors. Assume that a set of straight edges are projected into the 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:

$$u_{ki} = \frac{\Delta}{\xi_{uki}} (R, T, \Omega, a, V, \Sigma)$$
$$v_{ki} = \frac{\Delta}{\xi_{vki}} (R, T, \Omega, a, V, \Sigma)$$

where $\Sigma$ 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:

$$\epsilon_{ki} = \left[ \hat{u}_{ki} - \xi_{uki} (R, T, \Omega, a, V, \Sigma) \right]$$
$$\epsilon_{vki} = \left[ \hat{v}_{ki} - \xi_{vki} (R, T, \Omega, a, V, \Sigma) \right]$$

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

### 4. Blockwise Nonlinear Minimization

Let $y$ be the vector of image projections in the left hand side of equation (7) and $x$ the vector of pose, velocity and nuisance parameters $\Pi = [R, T, \Omega, a, V, \Sigma]$. The relationship between these two vectors is denoted $y = \xi (x)$. Given
a set of noisy observations \( \hat{y} \), we want to converge toward the value \( \hat{x} \) so that the error \( \delta \) in the relation \( \hat{y} = \xi(\hat{x}) + \delta \) is minimal. The minimization starts from an initial guess \( x_0 \) and is updated iteratively by applying variations \( \delta \) to \( \hat{x} \). This is generally achieved by assuming local linearity of \( \xi \) under which one can write \( \xi(x_0 + \delta) = \xi(x_0) + J\delta \) with \( J \) the Jacobian matrix of \( \xi(x) \). This implies to solve at each iteration the so called normal equations:

\[
J^T J \delta = J^T e \tag{10}
\]

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

\[
U_{p,q} = \sum_k \sum_i \left( \frac{\partial m_{ki}}{\partial P_p} \right) \left( \frac{\partial m_{ki}}{\partial P_q} \right) \tag{11}
\]

\[
V_{p,q} = \sum_k \sum_i \left( \frac{\partial m_{ki}}{\partial \Sigma_p} \right) \left( \frac{\partial m_{ki}}{\partial \Sigma_q} \right) \tag{12}
\]

\[
W_{p,q} = \sum_k \sum_i \left( \frac{\partial m_{ki}}{\partial P_p} \right) \left( \frac{\partial m_{ki}}{\partial \Sigma_q} \right) \tag{13}
\]

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

\[
\begin{bmatrix}
U & W \\
W^T & V
\end{bmatrix}
\begin{bmatrix}
\delta_P \\
\delta_\Sigma
\end{bmatrix}
= 
\begin{bmatrix}
E_P \\
E_\Sigma
\end{bmatrix} \tag{14}
\]

where \( \delta_P \) and \( \delta_\Sigma \) contains the small variations of respectively \( P \) and \( \Sigma \). The blocks \( E_P \) and \( E_\Sigma \) form the vector on the right hand side of the normal equation (10). Their components are defined as follows:

\[
E_P = \sum_k \sum_i \left( \frac{\partial m_{ki}}{\partial P_p} \right) \epsilon_{ki} \tag{15}
\]

\[
E_\Sigma = \sum_k \sum_i \left( \frac{\partial m_{ki}}{\partial \Sigma_q} \right) \epsilon_{ki} \tag{16}
\]

Equation (14) can be rewritten as follows:

\[
\begin{bmatrix}
U - WV^{-1}W^T & 0 \\
W^T & V
\end{bmatrix}
\begin{bmatrix}
\delta_P \\
\delta_\Sigma
\end{bmatrix}
= 
\begin{bmatrix}
E_P - WV^{-1}E_\Sigma \\
E_\Sigma
\end{bmatrix} \tag{17}
\]

which can be decomposed into two separate equation systems:

\[
(U - WV^{-1}W^T) \delta_P = E_P - WV^{-1}E_\Sigma \tag{18}
\]

and

\[
\delta_\Sigma = V^{-1}(E_\Sigma - W^T \delta_P) \tag{19}
\]

Equation (18) can be solved very efficiently because \( V \) 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)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>R (deg)</th>
<th>T (m)</th>
<th>V (perc.)</th>
<th>Omega (perc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean value</td>
<td>1.4</td>
<td>0.015</td>
<td>1.55</td>
<td>2.60</td>
</tr>
<tr>
<td>stand. dev.</td>
<td>1.0</td>
<td>0.006</td>
<td>1.05</td>
<td>1.80</td>
</tr>
</tbody>
</table>

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 subpixel 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 row after the other the rows of the image, yielding optical distortions due to high object velocity. The approach extends
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