VISUAL ODOMETRY WITH AN OMNIDIRECTIONAL MULTI-CAMERA SYSTEM

Ana Rita Pereira*, Valdir Grassi Junior*, Helder Araújo†

* São Carlos School of Engineering, University of São Paulo, São Carlos, SP, Brazil
† Institute for Systems and Robotics, University of Coimbra, Coimbra, Portugal

Emails: anaritap@usp.br, vgrassi@usp.br, helder@isr.uc.pt

Abstract—This paper suggests a visual odometry approach for an omnidirectional multi-camera system. Monocular visual odometry is carried out for each camera and the best estimate is selected through a voting scheme. While this approach does not present significant improvements in the trajectory in the XY-plane, it outperforms any of the monocular odometries in the relative translation in Z.

Keywords—Visual odometry, multi-camera system, omnidirectional vision.

1 Introduction

Autonomous navigation has been a relevant research topic for a long time, leading to evident progress in the last years. One of the many approached topics is visual odometry, which aims at retrieving the relative motion of one or more cameras based solely on their images.

Visual odometry with one camera or a stereo pair facing the same part of the scene has a limited field of view. Of the 360° surroundings, only a small part is used for estimating the camera’s ego motion, potentially ignoring parts of the scenario which are rich in features and could provide a better motion estimate. In this paper we propose a visual odometry approach for an omnidirectional multi-camera system so that the whole surroundings of the scene can be explored.

A lot of work has been done on visual odometry with multiple cameras. Kim et al. (2007) determines the ego-motion of a set of non-overlapping cameras. Visual odometry is computed for each camera through the 5-Point algorithm and all estimates are averaged. If the motion contains rotation, the translation scale can be retrieved. Clipp et al. (2008) proposes a method for solving the motion of two non-overlapping cameras mounted on the roof of a vehicle, requiring only five correspondences in the first camera and one in the second. The 5-Point algorithm is carried out in the first camera, retrieving its rotation and translation up to a scale factor. Using one correspondence in the second camera, and as long as motion has enough rotation, scale is retrieved. This approach can be expanded for a larger number of cameras. Kazik et al. (2012) computes the 6 DoF motion of two opposite-facing cameras in real-time and challenging indoor environments. Monocular visual odometry is computed separately for each camera and, in every key-frame, bundle adjustment is performed, minimizing the reprojection error. The individual motion estimates are fused in an Extended Kalman Filter (EKF). The known transform between both cameras allows retrieving absolute scale as long as motion is not degenerate. Degenerate motions are detected and discarded, preventing from propagating wrong scale estimations. Lee et al. (2013) computes the non-holonomic motion of a vehicle with four cameras mounted on the roof. The authors simplify the Generalized Epipolar Constraint applying the Ackerman steering principle and allowing motion to be computed with two points. They also propose a method which allows retrieving scale in degenerate motions, requiring one additional inter-camera correspondence. Motion is computed in a Kalman filter framework, where velocity is assumed to be constant. Loop-closure and bundle adjustment are carried out.

The omnidirectional system used in this work is the spherical multi-camera system Ladybug2 by FLIR. This sensor consists of six cameras; five disposed as a ring, looking outwards, and the sixth is looking up. Only the five lateral cameras are considered here. The distance between cameras is relatively short, and the lack of a large baseline prevents an accurate depth estimation. The proposed approach for visual odometry consists of two main steps: in the first step, a monocular visual odometry algorithm is applied to each camera. The outputs of the first step are five relative motion estimates, obtained from a set of previous and current images of the five cameras. In the second step, feature matches found in each camera are used to test each of the five motion estimates - if a feature match agrees with the motion estimate being tested, it is considered an inlier. The motion estimate with the most inliers is selected. This process is repeated every time a new frame arrives, leading to a chain of voted motion estimates. A general overview of the proposed algorithm is depicted in Figure 1.

The remainder of this paper is as follows: section 2 explains each step of the proposed method. The experimental procedure and results are described in section 3 and section 4 adds closing remarks and future work suggestions.
for each pair of corresponding points

\[ x_{i,t}^{t+1}, y_{i,t}^{t+1}, 1 \]

The SVD of \( A \) is computed and \( F \) is obtained from the elements of the eigenvector corresponding to the lowest eigenvalue (Trucco and Verri, 1998).

Finally, the rank-2 constraint is imposed. Afterwards, \( F \) is tested against all points, by calculating the Sampson cost, defined as

\[
d_S = \frac{\tilde{p}_{i,t}^T F \tilde{p}_{i,t}}{(F \tilde{p}_{i,t})_x^2 + (F \tilde{p}_{i,t})_y^2 + (F^T \tilde{p}_{i,t+1})_x^2 + (F^T \tilde{p}_{i,t+1})_y^2} \tag{2}
\]

where the tilde indicates homogeneous coordinates and the subscripts \( x \) and \( y \) refer to the first and second components of the resulting vector. For a given point in the previous image, this cost is the distance from the actual corresponding point to the epipolar line given by \( F \) in the current image. Points are considered inliers if this distance is below a threshold. This process is repeated 400 times, and the \( F \) with the most inliers is afterwards refined using all agreeing points.

The essential matrix \( E \) is computed out of \( F \) as shown below:

\[
E = K^T \cdot F \cdot K, \tag{3}
\]

where \( K \) is the camera matrix with the intrinsic parameters. The rank-2 constraint is also applied to \( E \), before decomposing it into a rotation matrix \( R \) and a translation vector \( t \). Assuming matrices

\[
W = \begin{bmatrix}
0 & -1 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

and

\[
Z = \begin{bmatrix}
0 & 1 & 0 \\
-1 & 0 & 1 \\
0 & 0 & 0
\end{bmatrix}, \tag{4}
\]

rotation matrix candidates \( R_a \) and \( R_b \) and the skew symmetric translation matrix \( S \) are computed as follows:

\[
R_a = U \cdot W \cdot V^T, \quad R_b = U \cdot W^T \cdot V^T, \quad S = U \cdot Z \cdot U^T, \tag{5}
\]

where matrices \( U \) and \( V \) are the outputs of the SVD of \( E \). If the determinant of \( R_a \) (or \( R_b \)) is negative, the matrix must be multiplied by \(-1\). As to translation vector \( t \), its components are extracted from \( S \). After decomposing \( E \), two possible rotations (\( R_a \) and \( R_b \)) and two possible translations (\( t \) and \(-t\)) are obtained, leading to four possible transforms. Each combination is subject to a chirality test, where points are triangulated and checked whether they lie in front of both previous and current cameras. The combination with the most valid points is selected. This implementation of the 8-Point algorithm was based on (Geiger, 2010) for the monocular case.

### 2.2 Voting scheme

The previous step outputs five motion estimates, obtained from the features found in each camera. These estimates undergo a voting scheme, where each camera \( n \) votes for each transform \( m \). The following steps are carried out: First, as the transform returned in the monocular visual odometry step for camera \( m \) is expressed in camera \( m \) coordinates, it must first be converted to the camera \( n \) reference frame. After extracting \( R \) and \( t \) of
the obtained transform, the essential matrix $E$ is computed.

$$E = S \cdot R$$  \hspace{1cm} (6)

where $S$ is the skew symmetric matrix of the translation vector. Matrix $F$ is then composed by

$$F = K^{-T} \cdot E \cdot K^{-1},$$  \hspace{1cm} (7)

followed by a rank-2 re-enforcement. Finally, the Sampson cost is computed for all features found in camera $n$, obtaining a number of inliers - these are the votes of camera $n$ to camera $m$’s motion estimate. The votes are summed up for all $n$’s and the estimate $m$ with the most votes is selected, being the final output of the multi-camera visual odometry algorithm.

2.3 Translation scale

Since we are using a multi-camera system with known extrinsic parameters, one could assume that scale could be retrieved, as many authors have done. However, due to the small distance between the cameras, contrasting with the large distance between sensor and scene points, the Ladybug2 system acts almost like a central sensor. In other words, the optical centers of all cameras can be approximated to be at the same place and, without a prominent baseline, scale cannot be reliably retrieved. In the graphs showing the results, the translation vector resulting from visual odometry has been scaled such that its norm is equal to the ground truth translation norm at corresponding instants.

3 Experiments and results

This section is dedicated to this project’s experiments and results. The experimental procedure is first described. Stepping into the results, the estimated trajectories of the five monocular odometry estimates and the voted camera estimate are presented. Afterwards, accuracy is studied in greater detail by analyzing graphs of the error of each component of the relative motion. Finally execution time is discussed. Color code of plots and graphs is as follows: Ground truth is represented in black, the voted odometry sequence in red and the monocular odometries of cameras 0, 1, 2, 3 and 4 in yellow, gray, blue, green and magenta, respectively.

3.1 Experimental procedure

Experiments were carried out by driving a vehicle along a path of approximately 1.6 km at a mean speed of 27.4 km/h, in an environment with few buildings and mainly composed by vegetation. The vehicle is part of an autonomous car project, Carina2, led by ICMC, USP.

The vehicle was equipped with a GPS, an IMU and the Ladybug2 system, fixed on a pole on the roof. Along this text, the individual cameras of Ladybug2 are referred according to Figure 2, right, where camera 0 points at the front of the vehicle. Ground truth is composed by the GPS positions and IMU’s roll and pitch. The yaw is obtained by computing the angle of the trajectory in the $XY$-plane. The GPS has RTK correction and a position precision of 0.4 m (horizontally) and 0.8 m (vertically). The framerate of the Ladybug2 sensor is 6 Hz and intrinsic and extrinsic calibration is provided by FLIR.

Figure 2: Left: Carina2. Right: Schematic drawing and reference frame of the Ladybug2 sensor.

3.2 Trajectories

Figure 3 shows the monocular visual odometry estimates obtained from each camera and the trajectory formed by the voted estimates in the $XY$-plane, i.e., as seen from above. None of the estimates is very accurate, but, visually guessing, estimates from cameras 2, 3, as well as the voted odometry are the ones that most resemble the real trajectory. Regarding the odometry of the best cameras, the initial straight line drifts towards negative $Y$ and some of the curves are estimated to be less tight than reality.

Figure 4 shows the same trajectories as the figure above, but in the $XZ$-plane, providing a vertical view. While cameras 0, 1 and 4 drift towards negative $Z$, cameras 2 and 3 tend to drift up. The voted odometry is the one that stays closer to $Z = 0$, keeping the most realistic height.

The sequence of the best (voted) odometry estimates is again shown in Figure 5, where the best camera in a particular instant is shown by a marker of the corresponding color. Although some tendencies are observed along different parts of the trajectory, all cameras are, from time to time, voted as the best camera.

3.3 Relative motion errors

This section compares the relative motion errors of the voted odometry towards each monocular estimate. The relative motion error is computed for each component of the trajectory (translation in $X$, $Y$ and $Z$ and rotation about $X$, $Y$ and $Z$).
Figure 3: Monocular visual odometry estimates (camera 0 to 4). Sequence of voted estimates (best camera). XY-plane.

Figure 4: Monocular visual odometry estimates (camera 0 to 4). Sequence of voted estimates (best camera). XZ-plane.

Figure 5: Left: Sequence of best cameras. Right: Detail.

as follows:

\[ \Delta Err = (\alpha_{vo,t+1} - \alpha_{vo,t}) - (\alpha_{gt,t+1} - \alpha_{gt,t}), \]  

where \( \alpha \) can be any of the mentioned motion components. \( gt \) stands for ground truth and \( vo \) for
visual odometry. As relative rotation errors are similar for every odometry, only relative translation errors are presented.

To facilitate the temporal correspondence of the error graphs and the trajectory, some points on the path are marked and named, both in the ground truth trajectory (Figure 6) and in the error graphs.

Figures 7, 8, 9, 10 and 11 show the relative translation errors in $X$, $Y$ and $Z$ to the ground truth. The errors of the voted odometry are shown together with the errors of each of the monocular odometries in separate plots. Although the trajectory plots suggest that the voted odometry is the most similar to the ground truth, its relative $X$ and $Y$ translation estimates are, in general, outperformed by the monocular odometries, especially by the ones provided by cameras 0, 2 and 3. For instance, the relative translation error in $X$ and $Y$ for camera 3 stays around a low value with few large variations between markers G and I. In that same interval the voted odometry varies frequently around positive values, visibly larger than the one of camera 3. For cameras 1 and 4, monocular visual odometry is not significantly more accurate than the voted odometry in $XY$, which can be exemplified with camera 4’s error, between markers D and E. The error in $Z$ is considerably smaller for the voted odometry, comparing to any monocular estimate. This is, for instance, visible for camera 0, between markers G and H, and camera 3, between markers B and C. Overall, the relative motion errors change in sign and magnitude along the trajectory. This behavior could be due to changes in the environment, at the different parts of the path.

3.4 Execution time

The program ran on a Dual-Core i5-3317U laptop. Processing of most frames does not take longer than 0.5 seconds. This means that the bagfile containing the 6 Hz image sequence collected on the road had to be played at 0.3 times the original speed. Therefore, this implementation is not suitable for real-time applications, with the current hardware.

4 Conclusion

This paper proposes a two-step visual odometry method for an omnidirectional multi-camera system. In a first step, monocular visual odometry is carried out for each camera and, in a second phase, the best estimate is selected through a vot-
Figure 10: Errors for relative translation in $X$, $Y$, $Z$. Voted odometry (red) and camera 3 (green).

Figure 11: Errors for relative translation in $X$, $Y$, $Z$. Voted odometry (red) and camera 4 (magenta).

The proposed method can be further improved and some ideas are left as future work. In order to overcome the low framerate of the sensor, more robust feature extraction and tracking techniques can be used. Furthermore, better state-of-art monocular visual odometry algorithms can be explored. Execution time can be decreased by down-scaling images before processing, using a faster feature extraction method or considering heuristics to facilitate feature matching.

Acknowledgments

The authors would like to thank CNPq for the scholarship given to the first author. Furthermore, special thanks go to the LRM (Mobile Robotics Laboratory) ICMC/USP for providing Carina2 and Ladybug2 sensor for the experiments of this work.

References


