Monocular Visual Odometry in Robots for Agriculture using Fisheye Cameras

André Silva Aguiar (andres.aguiar@inesctec.pt); **Armando Jorge Sousa** (asousa@fe.up.pt); **Filipe Neves dos Santos** (fbsantos@inesctec.pt)
Faculdade de Engenharia da Universidade do Porto
INESC TEC – Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

L. FAST-FUSION: An Improved Accuracy Omnidirectional Visual Odometry System with Sensor Fusion and GPU Optimization for Embedded Low Cost Hardware. Appl. Sci. 2019, 9, 5516.

The main statement that this work pretends to defend is:
It is possible to track the motion of a robot with moderated accuracy using a single fisheye camera and common sensors on top of a low-cost microprocessor in an outdoor environment.

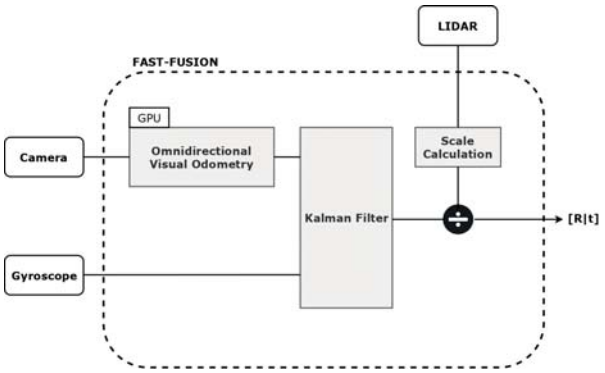

Figure 1. Global systemarchitecture [1].

The proposed system aims to localize in real-time a ground robot in an agricultural environment. As shows in Figure 1, a fusion of common sensors with a monocular omnidirectional Visual Odometry (VO) algorithm is performed. The entire system runs on top of a low-cost microprocessor, a Raspberry Pi 3B. An OpenCL-based optimization approach applied to the VO method is proposed recurring to the Raspberry Pi's GPU to overcome the performance limitations of this microprocessor. The central unity of the system is the VO method. This one is publicly available on the official Libviso2 repository (https://github.com/srv/viso2) and can work standalone, giving a primary estimation of the robot motion. A sensing system,constituted by a planar laser and a gyroscope,is also proposed to support and solve the main limitations monocular VO. The first is used to calculate the motion scale due to the unavailability of depth information resultant from a monocular VO system. The last is used as a support to VO in rotations and it is fused with VO recurring to a Kalman Filter (KF). The system output is a homogeneous transformation **[R|t]** between consecutive image frames.

Monocular Visual Odometry: As represented in Figure 2, the monocular VO system is composed of three main subsystems:

1. Application of a camera model that converts 2-D feature pixels in the omnidirectional image in3-D unit vectors.
2. A Random sample consensus (RANSAC) approach to select the inliers from the entire set of 3-Dunit vectors.
3. Motion estimation using the epipolar constraint and linear triangulation.

So, to deal with the fisheye camera high distortion, a state-of-the-art camera calibration toolbox was used, allowing to perform conversions such as, for example, from an image pixel into a 3-D unit vector, and from a 3-D world point into an image pixel, considering the lens distortion. With these capabilities,is proposed an adaptation of the state-of-the-art Libviso2 to es-

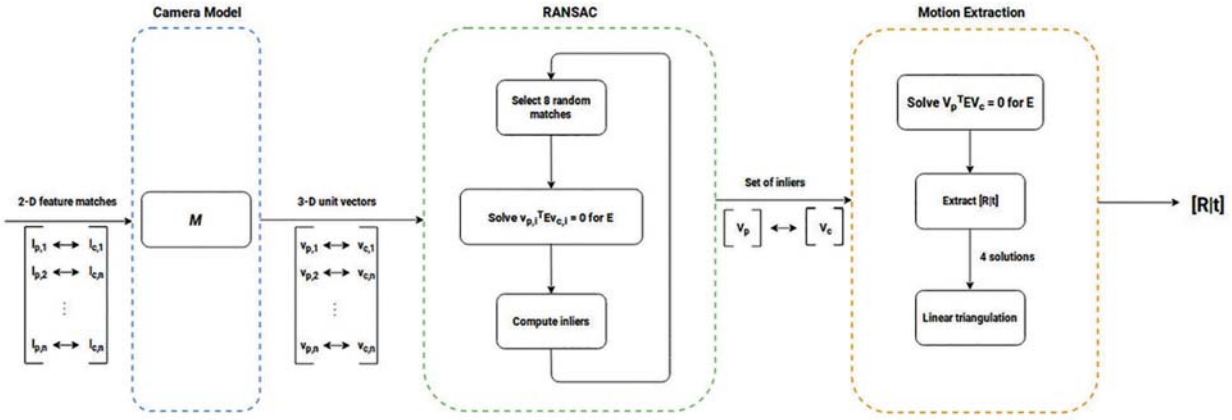

Figure 2. Omnidirectional Visual Odometry Scheme [1].

timate the camera motion. To do so, the matching procedure from the original approach is reused and the epipolar geometry approach is recreated to work with omnidirectional cameras. Our system, instead of using 2-D pixel-based feature matches, used 3-D unit vectors instead. In this, instead of projecting the line that contains the previous camera centre and the scene point into the current image plane – the so-called epipolar line – it is projected into the current unitsphere. With this configuration, epipolar curves are obtained, instead of epipolar lines. This means that a point on the unit sphere correspondent to the current image that matches a point on the unit sphere correspondent to the last image lies on an epipolar curve. With this,the RANSAC input is prepared, the entire set of 3-D unit vector matches, and this method can be used to solve for the essential matrix*E*. So, for each iteration, a random sample of size eight (eight-point algorithm) is selected, from the total set of unit vectors and the following equation is solved

$$v_{pi} E v_{ci} = 0 \quad (2)$$

for each one, where v_{pi} and v_{ci} represent a 3-D unit vector match between consecutive images Using single value decomposition (SVD), the solution of *E* is extracted. To calculate the set of inliers the original Libviso2 approach if followed – iterate through all the matches and use the Sampson Distance to filter the outliers. At the end of all the RANSAC iterations, the final set of matches is available and is used to refine the essential matrix*E*. Then we extract the camera motion**[R|t]**from it. However, for a given essential matrix *E* there are four different solutions for the current camera matrix. To extract the correct solution, a linear triangulation approach is used. Solving a linear equation for all the matches considering the four possible **[R|t]** solutions and choosing the one that presents the higher number of 3-D triangulated points with positive depth results in the final solution for the camera motion. However, since only information about one camera is being used, the notion of depth is not available. So, the extracted **[R|t]** solution is computed up to a scale factor *a* – the motion scale.

Motion Scale Calculation: To complement the camera motion estimation from our omnidirectional VO approach, aplanar LIDAR sensor is considered to recover the scale factor. This approach is divided into four essential steps:

1. Transformation of the range measurement of the LIDAR into the camera referential frame.
2. Projection of the LIDAR measures in the camera referential frame into the omnidirectional image.
3. Search for associations between image features and LIDAR measures in the omnidirectional image.
4. Scale calculation using the associations found.

To perform the transformation of the range measures to the camera referential frame, we measured the physical distance from the camera centre to the LIDAR. We apply a transformation *H* to the range measures to convert them to the desired referential. After that, to obtain the range measures as 2-D pixel points in the omnidirectional image, the camera model is applied. The final set of 2-D range measures are the ones who are mapped inside the omnidirectional image, i.e., the ones who are inside the camera field of view. The next step consists in associating the LIDAR measures projected into the image with 2-D feature points present in the current image frame. To do so, a search on the 2-D LIDAR measures neighbourhood is computed. Figure 3 shows the real projection of LIDAR measures in the omnidirectional image in black and the associated features in white. After matching 2-D feature points with 2-D range measures, it is possible to estimate the scale factor. These feature points were already triangulated using their respective matches in the previous image frame. Thus, we already have a set of matches between the 2-D LIDAR measures and 3-D triangulated feature points, and, consequently, between the 3-D world points extracted from the range measures and 3-D triangulated image feature points. So, the scale factor is the average of the relation between the norm of the triangulated matched features and the distances measured by the LIDAR that are matched. This factor is directly applied to the translation vector extracted from the essential matrix *E*.

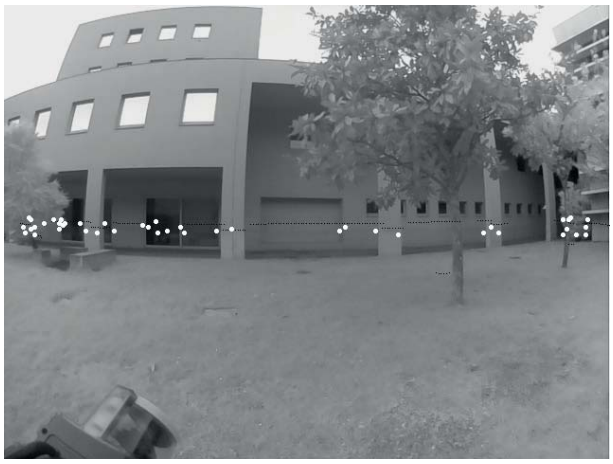

Figure 3. LIDAR measurements projection in the omnidirectional image and 2-D feature association with them [1].

Orientation Correction: After having a stable camera motion estimation, the need to support it in rotation-only motion types emerged due to high errors in the estimation in these cases. To do so, a gyroscope is used fusing it with the angular velocity resultant from the VO approach using a KF. Besides the angular velocity, the gyroscope bias b_i is also estimated in order improve the motion estimation accuracy. Considering the relative orientation $\Delta\theta_i$ obtained by VO as controls, and the gyroscope angular velocities ω_i as observations, with $i \in \{x, y, z\}$, the state vector $x = [\omega_x, \omega_y, \omega_z, b_x, b_y, b_z]$, is computed as follows:

$$\omega'_{i_k} = \Delta\theta_{i_k} \Delta t$$

$$b'_{i_k} = b_{i_{k-1}}$$

for each component $\{x, y, z\}$. We considered bias as a constant state ignoring flicker noise and temperature oscillations. Even so, this is a reasonable approximation due to the low impact of these two components in time-limited estimations. In addition, the observations model is:

$$\omega^G_i = \omega'_i + b'_i$$

with $i \in \{x, y, z\}$. This equation can be interpreted as: *the angular velocity state is equal to the gyroscope observation minus the bias estimation*. So, it is expected that this performs a correction of the angular velocity observation that is used in the computation of the state. The VO approach in some cases provides unrealistic estimations in pure rotations. Due to this, the covariance matrix of the state Q is dynamic. Both Q and the observations covariance R are initialized with constant values on their diagonals. The values of the diagonal of Q correspondent to the bias states are decreased over time, since bias is considered as a constant state. To detected and cancel the unrealistic peaks of angular velocity on the state, a non-linear approach was adopted. In this, asigmoid function that varies with the angular velocity states was calibrated. So, if a peak of angular velocity is detected, the sigmoid increases the covariance noise of the angular velocity which leads the filter to consider the gyroscope measure instead. In this way, it is possible to have in consideration two different sources of information to estimate the robot rotation.

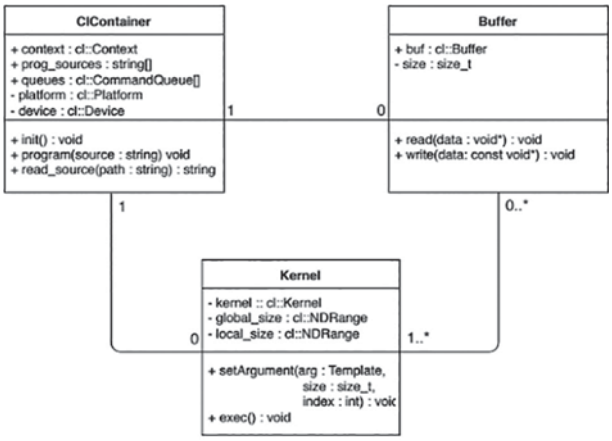

Figure 4. Unified modelling language (UML) diagram of the OpenCL abstraction layer implemented.

Heterogeneous Computing Optimizations: After having the previously described fusion working on top of a standard computer, we needed it to be fast in an embedded configuration to run on the robot in real-time. So, we chose a low-cost microprocessor – Raspberry Pi 3B – and tried to optimize the developed code to run on this platform. To do so, we use both Raspberry Pi's CPU and GPU with parallel computing technique. To access Raspberry Pi's GPU, the VC4CL (https://github.com/doe300/VC4CL) driver was used. This is an open-source OpenCL 1.2 implementation for Raspberry Pi's GPU that allows the use of OpenCL C++. To facilitate OpenCL usage, an additional layer of abstraction consisting of an OpenCL-wrapper for C++ and ROS was developed. This allowed a communication between the host CPU and the device GPU using simple write and read routines. The implementation layout is represented in Figure 4. The iterative RANSAC approach was parallelized, since it is the most computationally expensive block of the system. For the GPU-based optimizations, a 16-way single instruction multiple data (SIMD) kernel architecture was adopted since each quad processing unit (QPU) of this device uses 16-way SIMD, executing an instruction with four-way data parallelism, four cycles in a row. This way, our approach for each routine follows the following pattern:

1. Load the routine input data correspondent to all the RANSAC iterations.
2. Write all the data to the correspondent kernel at once using 16-way vector types.
3. Execute the kernel to all the data in a 16-way vectorized way and load it to a single output array.
4. Read all the output data at once and label the corresponding RANSAC iteration to it.

In this way, we maximize the GPU performance using vectorized kernels that match with its architecture. Also, we minimize the data transfer delays between the host and the device by performing the communication only once for writing and once for reading.

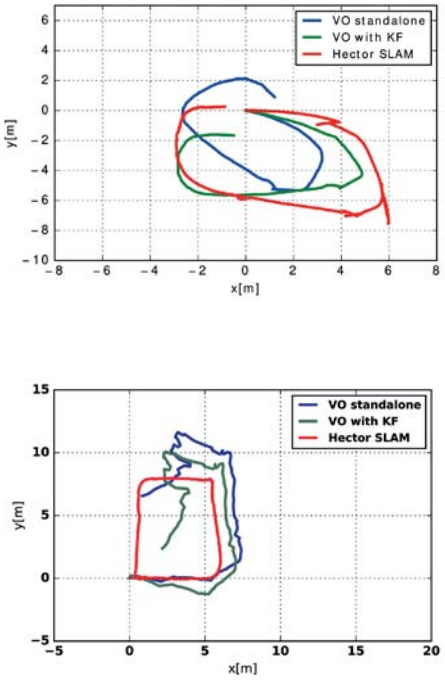

Figure 5. Motion estimation with and without the orientation correction.

Results: Figure 5 shows the motion estimation results performed our improved version of Libviso2, with all the sub-systems described. The images show the performance of Hector SLAM, a state-of-the-art accurate localisation algorithm (in red), our system without the KF (in blue), and with the KF (in green). Our system presents moderate accuracy, approximate with Hector SLAM, using low-cost hardware and running on an embedded device with several performance limitations.

CONCLUSION

Our system is composed of an omnidirectional extension of the state-of-the-art monocular VO method Libviso2 that uses raw omnidirectional images, a LIDAR to calculate the motion scale, and a gyroscope to support the estimation in pure rotations. In short, we achieved real-time performance in an embedded configuration, and our system presented higher accuracy than the original Libviso2 approach.