EMBEDDED REAL-TIME LOCALIZATION OF UAV BASED ON AN HYBRID DEVICE

H. Chenini [*], *D. Heller*[+], *C. Dezan*[*], *JP. Diguet*[+] *and D. Campbell*[++]

[* +] Lab-STICC, Université de Bretagne Occidentale and Université de Bretagne Sud, France.
[++] ARCAA, Queensland University of Technology, Brisbane, Australia.

ABSTRACT

This paper presents a method for localizing an Unmanned Aerial Vehicle (UAV) in indoor or outdoor environments. The approach has the ability to estimate the 3D pose of the on-board camera by using a Harris corner detector and the Levenberg-Marquardt (LM) with the Random Sample Consensus (RANSAC) algorithm to perform detection. The implementation of such computational intensive tasks in embedded system is necessary for the autonomy of UAV. Accelerators implemented on FPGA provide a solution to reach required performances. In addition to the algorithm development, we present the embedding of a real time camera pose estimation algorithm on a Xilinx System on Programmable Chip (SoPC) platform. Partitioning of our embedded application into hardware and software parts on a Zynq Board has significantly reduced the execution time when compared with software implementation, while offering necessary reconfiguration capabilities.

Index Terms— Localization, UAV, 3D pose of the camera, Harris corner, Matching, Levenberg-Marquardt, Zynq board

1. INTRODUCTION

In the last few years, the development of UAVs attracts lots of research interests. Their utilization has significantly increased in many tasks such as navigation, recognition [1], surveillance [2] or military missions [3]. Our main goal is to develop a UAV with flight management system including multiple functionalities in the same design. To achieve UAV autonomy, one particular mission of interest involves computing UAV's location in unknown environment using the information gathered by the on-board sensors. Navigation systems are indispensable components of a UAV, enabling it accurately localizing the position and orientation, or pose of the UAV particularly when flying in bad conditions (weather, disaster, etc.) which decreases the quality of the Global Positioning System (GPS) navigation. This work focuses on the design and implementation of a localization algorithm from a set of images sequence taken from an on-board downward looking camera. The 3D pose estimation of moving camera from its images is one of the most important functions in a number of applications such as calibration [4], object recognition/tracking [5] and Simultaneous Localisation and Mapping (SLAM) [6]. However, these applications need accurate measurements of the position and orientation (pose) of the camera with respect to the scene. Previous researches have provided localization techniques with respect to vision-based SLAM for grounded-robot applications that has been largely explored. The use of such approaches for UAV still requires some improvements to obtain optimal UAV's position [7] [8]. Once we have a calibrated on-board camera, the problem consists of estimating its relative position and orientation by matching images (i.e. computing the inter-image point correspondences), and this is closely related to the epipolar geometry. Whereas, point correspondences, which have been extracted from images, always contain wrong matches, estimating relative pose from such data requires a robust algorithm. A new algorithm was originally proposed for computing the rotation and translation between the observed geometry (3D points) and the projected points (2D points). Combining RANdom SAmple Consensus (RANSAC) [9] with the Levenberg-Marquardt (LM) [10] method leads to bearing measurements of the relative pose of an object from an initial position since our problem can be formulated as a non-linear least squares problem. To track a set of points through a sequence of images, it is required to run a number of iterations and keep only the hypothesis that fits most of the data as the best one. The set of points, which totally miss their correct positions (outliers), can have a severe impact on the solution of the pose problem and it is important to detect and discard them. The input to the algorithm is simply the images and the output is the estimated movement together with a set of interest points in correspondence. The first step of the algorithm is to compute interest points in each image. This problem is resolved after by using robust estimation, here in our case RANSAC, as a search engine. The idea is first to obtain by some means a set of putative point correspondences. It is expected that the proportion of these correspondences will in fact be mismatches. RANSAC is designed to deal with exactly this situation-estimate the movement and also a set of inliers consistent with this estimate (the true correspondences), and outliers (the mismatches). The paper is structured as follows: In section 2, we present the proposed localization algorithm considered to get the camera location and orientation in the world. We then describe the hardware implementation used in the context of the experimental platform that we have chosen in section 3. We present also some performance results concerning resource costs and execution times. The section 4 concludes the paper.

2. POSE ESTIMATION METHOD

As already stated, our goal is to perform 6 degrees of freedom (translation and the orientation) localization of the UAV in indoor environment (Fig. 1). The principle is to acquire a sequence of images while the sensor (which will be embedded on a drone) moves. At each additional acquisition, a set of image features is detected and tracked in the image. The problem of localization and mapping consists in

[*]Email: hanen.chenini, Catherine.Dezan@univ-brest.fr
[+]Email: dominique.heller, jean-philippe.diguet@univ-ubs.fr
This work is supported by French CNRS (Project PICS SWARMS)

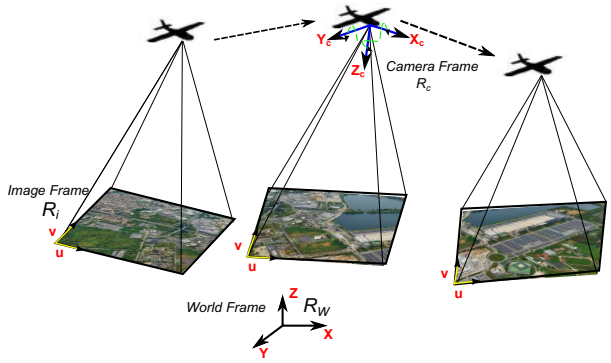

Fig. 1. Localization of the UAV using a set of real images.

3 steps. For each new image, we: (1) detect distinctive features in each new video image using Harris corner detector [11], (2) identify the detected significant features allows to match 2D image features in the current image with their correspondences in the key image features using ZNCC [12]. The goal of this step is to find correspondence between two images. Practically, the current images can be matched by taking the previous image as the reference image. In brief, the matching process involves computation of the similarity measure between two pixels windows and finally (3) estimate camera pose based on 2D correspondence using RANSAC.

2.1. Camera Pose Estimation Algorithm

We present here the camera pose estimation algorithm for a calibrated camera with known intrinsic camera parameters where 6 degrees of freedom of a camera's pose have to be calculated (i.e. orientation R=f(ϕ,θ,φ) and position $T = [tx, ty, tz]^t$ between camera and image coordinate system. As shown in Fig. 2, our estimation

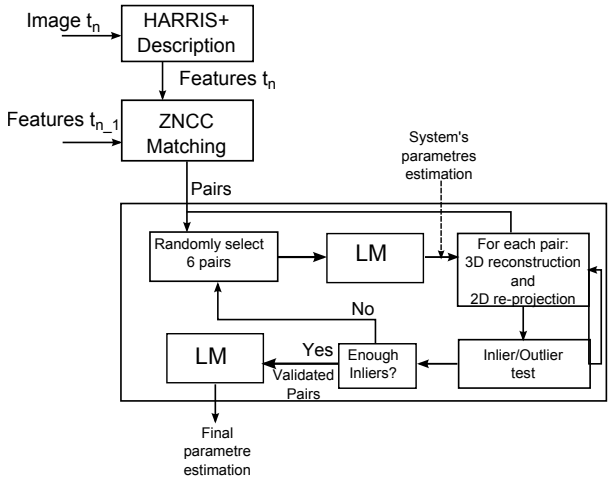

Fig. 2. The proposed localization algorithm.

methodology initially considers the corner locations of the current image extracted by the Harris corner detector (features at time t_n) and windowed image patches of the key image (features at time t_{n-1}). Given these two consecutive images I^l and I^r, our method directly finds a set of primary matches features using ZNCC (Zero-Mean Normalized Cross-Correlation) approach. These matched couples often contain many incorrect matches. In order to achieve a robust matching algorithm, the RANSAC algorithm is used in computing 3D position of the camera. We thus propose the following LM algorithm for refining the parameters:

I- We initialize the number of estimation **N**=100 performed by RANSAC before converging and the maximum number of inliers $MAX\_inlier = 0$. Then, we give the intrinsic parameter matrix **K** of the calibrated camera (i.e. with known intrinsic parameters):

$$\mathbf{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where u_0 and v_0 are the column and row where the optical axis (Z_C axis of the camera frame) intersects the image plan and α_u and α_v encode the scale change between metric coordinates and pixel coordinates. It is equal to the ratio of the focal length on the vertical pixel size.

Then, we give initial extrinsic parameters P =[ϕ, θ, φ, tx, ty, tz], the 3×3 rotation (orientation matrix $R_0(3,3)$ and the 3×1 translation (position) vector $T_0(3,1)$ where ϕ, θ, φ are rotations around the X_C, Y_C and Z_C axis of R_C respectively (see Fig. 1). To get the rotation matrix R_0, 3 × 3 rotation matrix from the 3-vector[ϕ,θ,φ] and the translation vector T_0, we use the following conversions:

$$R_0 = \begin{bmatrix} cos\varphi.cos\phi- & -sin\varphi.cos\theta.sin\phi- & sin\theta.sin\phi \\ sin\varphi.cos\theta.sin\phi & cos\varphi.cos\theta.sin\phi & \\ cos\varphi.cos\phi+ & -sin\varphi.cos\phi+ & -sin\theta.sin\phi \\ sin\varphi.cos\theta.sin\phi & cos\varphi.cos\theta.sin\phi & \\ sin\varphi.sin\theta & cos\varphi.sin\theta & cos\theta \end{bmatrix}$$

$$T_0 = \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix}$$

II- for $k^{th}(k = 1 : \mathbf{N})$ estimation

1. Applying Levenberg Marquardt algorithm (LM). For q from 1 to **N** estimation
   (a) Randomly choose 6 correspondences.
   (b) For the chosen matched pairs, we compute the error: ε=0 − $[K^{-1}.m_j^l]^t.E_{ij}^{lr}.K^{-1}.m_j^r$ where m_i^l and m_j^r are the projection of a $3D$ point P_k on the panoramic images I^l and I^r respectively. Noting E_{ij}^{lr} is the essential matrix of the two columns containing m_i^r and m_j^l, we can write: E_{ij}^{lr} = $[T]_X^q.R^q$ where $[T]_X$ is the antisymmetric matrix of the vector T.
   (c) Estimation process terminates when the iteration counter exceeds a pre-specified limit.
   (d) Compute the updated parameters vector P_c = P + δ_{ij}, where P_c=[ϕ_c,θ_c,φ_c,tx_c,ty_c,tz_c].
2. for all the **M** correspondences (x and y positions of an extracted corner in the right and left images):
   (a) Recalculate R = f(ϕ_c, θ_c, φ_c) and T = [tx_c, ty_c, tz_c] using the updated parameters of P_c obtained after minimization.

(b) Calculate x, y and z ($3D$ position) positions of the projection of the corresponding model point in the right image using the obtained extrinsic parameters (i.e. R and T).

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} R(1) & R(2) & R(3) & T(1) \\ R(4) & R(5) & R(6) & T(2) \\ R(7) & R(8) & R(9) & T(3) \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ 1 \\ 1 \end{pmatrix}$$

(c) Compute the error projection between the left image coordinates and projective coordinates and count the number of inliers **m**.

3. if(**m** $> MAX\_inlier$) Update best $P_{save} = P_{curr}$, MAX_inlier = **m** and record all the inliers.

III- refinement: re-estimate P_{save} from all the inliers using the LM algorithm.

IV- The 6 parameters (ϕ, θ, φ, X, Y, Z) of P_{save} define the orientation and the position parameters of a panoramic camera.

3. EXPERIMENTAL RESULTS

3.1. Experiment Result with Simulation Data

To show the validity of the proposed camera pose estimation, we applied it on simulated data. In order to investigate the efficacy of the proposed approach, we perform experiments with a calibrated sensor (HDR-CX240E). For accurate measurements, a sophisticated camera calibration is a crucial step before using the sensor for localization or 3D reconstruction. The calibration step consists in computing the values of geometrical model parameters u_0, v_0, α_u and α_v. The image resolution is 1280×720 pixels. We obtain u_0 and v_0 equal to 615.5 and 434.4 respectively. We record two different

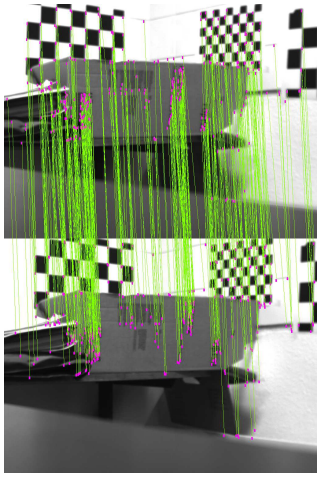

Fig. 3. RANSAC results: green points and lines represent inliers (number of inliers: 626).

sequences: forward motion (Translation along X axis see Fig. 3) and motion on a sphere around the scene in 45 degrees with the camera pointing to the center. In case of forward motion, the motion composed of ($T_{theo} = [1, 0, 0]^t$,$R_{theo} = I_{3x3}$) is the theoretical solution. The whole process is executed recursively until the result is stable. Once this has been done, the camera trajectory is estimated in a two processing steps : the first one focuses on the problem of recovering the camera orientation, R_{exp} and the second step returns an estimate for the camera translation, T_{exp}. Given enough points, the measurement errors also give good results as shown in Eq. 2.

Table 1. The average of computing time for each processing step per frame

Processing steps	number of points	Execution time(s)
Harris	715/643	0.7829308
Matching	256	0.035613
RANSAC	146	0.090461

$$R_{exp} = \begin{bmatrix} 0.988 & 0 & 0 \\ 0 & 0.974 & -0.007 \\ 0 & 0.004 & 0.998 \end{bmatrix}, T_{exp} = [0.926, 0.093, 0.184]^t \quad (2)$$

For the next experiment, the measurements error is also fine as shown in Eq. 3 and 4 and the RANSAC step is computationally expensive since it requires solving a nonlinear minimization problem and does not work well for few points.

$$R_{theo} = \begin{bmatrix} 0.7 & 0 & 0.7 \\ 0 & 0.999 & 0 \\ -0.7 & 0 & 0.7 \end{bmatrix}, R_{exp} = \begin{bmatrix} 0.634 & 0.125 & 0.611 \\ 0.145 & 0.988 & 0.037 \\ -0.678 & 0.169 & 0.607 \end{bmatrix} \quad (3)$$

$$T_{theo} = [0.7, 0, 0.7]^t, T_{exp} = [0.641, 0.059, 0.636]^t \quad (4)$$

The errors in rotation and translation depend on several factors such as the initial approximations of the rotation and translation vectors and LM parameters used for the optimization process. In addition, the motion estimation highly depends on interest point features density in each image. Now, we are trying to adapt the proposed approach to process real data taken from an urban environment in order to obtain accurate results without losing precision.

3.2. Implementation of localization algorithm on a ARM

The software implementation of the proposed algorithm with a minimum memory footprint is a straightforward solution for the deployment of the application in our flight management system. ARM Cortex A9 dual core processor is a popular embedded processor with a SIMD NEON coprocessor. It has a low cost, low power consumption and small footprint. With the Zynq, we can port our existing application to run on the ARM processor, without the need for an FPGA designer. The pose estimation code is written in embedded C. The development software is Xilinx Vivado 2014.2 under Windows7. The ARM cores run at 667MHz and the off-chip 1-GB DDR3 memory at 533MHz. The time required for execution of the different processing step is calculated. Table 1 gives the time delay to estimate the camera pose between two consecutive frames. The total time required for pose estimation execution is more than 909 ms. Here, the time required for RANSAC execution is calculated for 100 iterations to generate hypotheses and then keep the best solution with the smallest number of outliers. As we can remark in Table 1, this application contains sequential tasks with different degrees of complexity. In each frame, the different processing phases (Harris, Matching and RANSAC) are computationally expensive processes

specially for Harris step. We will try then to get the best performance out of our code by implementing this part in hardware. The Zynq SoC offers the possibility to accelerate the implementation through exploring the possibilities for hardware acceleration.

3.3. Hardware implementation of the proposed approach

To accomplish even better performance than the pure software version, we have designed the hardware accelerator for carrying out Harris detector using the programmable logic (PL) and then integrated it to the complete system (see Fig. 4). Without affecting the estimation performance, the software parts matching and RANSAC are executed by an embedded processor (ARM processor), while the hardware part Harris detector is implemented as an accelerator. By

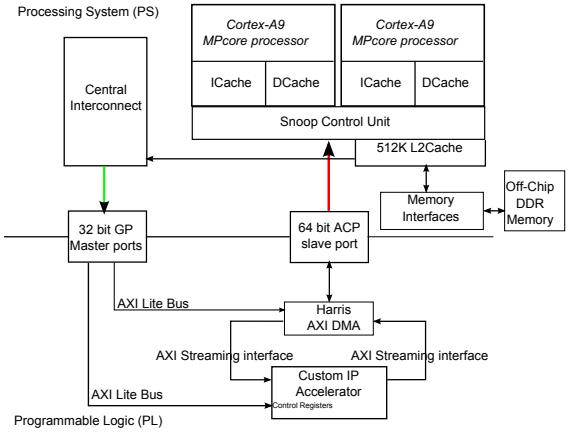

Fig. 4. HW Accelerator connected to the processing system (PS).

using AXI-Stream bus to receive the incoming image (or to send the output image) to the custom IP via DMA module, ACP (Accelerator Coherency Port) AXI can provide maximum bandwidth which can be benefit for image processing applications. Control signals of DMA module along with memory addresses are sent by ARM processor via general purpose AXI-Lite bus which is connected to central interconnect of the Processing System (PS). Fig. 4 shows the connectivity of ACP AXI port with the hardware accelerator. The generated Harris engine PCore is connected to the system using one AXI4-Lite interface (AXI GP), two AXI-Stream interfaces (AXI DMA module), and one dedicated port each for clock, reset, and interrupt. The synthesis target is Xilinx XC7Z020 evaluation Board. After synthesis, the resource requirement for Harris design and the complete design for Harris implementation is reported in Table 2.

For the pure software version, the Harris stage takes alone more than 782.93 ms. When the hardware accelerator is used to perform this stage, the time is reduced to 172.85 ms achieving a 4.54x speedup. Moreover, we measured the power consumption of each component when the hardware accelerator is implemented on programmable logic or not. In both cases, we observe that the CPU is consuming about 0.28 watts and the processing system consumes 1.31 watts when no hardware is implemented in programmable logic. In the first case, the complete design consumes around 1.331 watts with energy consumption equal to 1,04 J ($1,331 \times 0,782 =$ 1,04 J).

Table 2. On-chip area occupation of Harris design and the complete design

Resource type	Harris design	Complete design
BRAM18k	120 /280 (43%)	134 /280 (47%)
DSP48E	27 /220 (12%)	30 /220 (14%)
FF	4127 /106400 (3%)	7446 /106400 (7%)
LUT	6918 /53200 (13%)	11179 /53200 (21%)

When the hardware accelerator is implemented on PL, it consumes more power but if we consider the speedup, it results a significant energy efficiency improvement. When the hardware is running, the whole design consumes around 1.357 watts with energy consumption equal to 0,233 J ($1,357 \times 0,172 = 0,233$ J). The hardware version with the direct access to memory through ACP AXI port can reduce the energy consumption by 4.46x compared to the pure software implementation. This point is crucial for small UAV where the payload and the size of the battery are critical. So the use of hardware accelerator connected to the ARM processor on a Zynq can achieve significant performance improvement for our localization application. But a programmable chip also allows to update the embedded system according to the objectives of the UAV mission. Moreover dynamic and partial reconfiguration (DPR) at runtime during the mission is also offering interesting opportunities. It is for instance possible to replace the Harris detector by a smaller Fast detector according to luminosity conditions. DPR applied to the Fast module requires 10ms when a Linux OS running on the ARM processor (5ms in a standalone version), it means that runtime adaptation of the UAV embedded system is a valid solution.

4. CONCLUSION

We have demonstrated the implementation of a new camera pose estimation method in an embedded system based on an hybrid device. We first describe the Harris corner detector, which is the most common algorithm for detecting and describing point features. Secondly, we consider a similarity measure based on ZNCC method. To avoid the matching issues, the idea is to directly eliminate the wrong matched couples between an image template and the current image. Then, the RANSAC is used to estimate the best fitting epipolar geometry using LM algorithm. Finally, we extract the camera movement from the essential matrix by directly comparing the whole current and desired images. The design of an efficient embedded system is the second objective of this work. The implementation results on a Zynq Board demonstrate that the hardware accelerator is an effective way to achieve good performances. Furthermore, the usage of an hardware accelerator also contributes to improve the power efficiency of the embedded system, this is a key factor in the context of a small UAV with limited payload. Moreover we experienced DPR as an effective solution to update the embedded system according to mission objective before the take off or even during the flight. We are currently working on enhancing the precision of the proposed localization approach. In future work, we also plan to evaluate the performance of our proposal in urban environments.

5. REFERENCES

[1] Sung. Chen-Ko and Florian. Segor, "Onboard pattern recognition for autonomous uav landing," *Proc. SPIE 8499, Applications of Digital Image Processing XXXV*, 2012.

[2] Michael Kontitsis, Kimon P. Valavanis, and Nikos Tsourveloudis, "A uav vision system for airborne surveillance.," in *ICRA*. 2004, pp. 77–83, IEEE.

[3] U. S. Army UAS Center of Excellence, "U.s. army roadmap for unmanned aircraft systems," pp. 2010–2035, 2010.

[4] Tomas Izo and W. Eric L. Grimson, "Simultaneous pose estimation and camera calibration from multiple views," *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2, pp. 439, 2004.

[5] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele, "Monocular 3d pose estimation and tracking by detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010.

[6] Hugh Durrant-Whyte and Tim Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms," *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, vol. 2, pp. 2006, 2006.

[7] Jorge Artieda, José M. Sebastian, Pascual Campoy, Juan F. Correa, Iván F. Mondragón, Carol Martínez, and Miguel Olivares, "Visual 3-d slam from uavs," *J. Intell. Robotics Syst.*, vol. 55, no. 4-5, pp. 299–321, Aug. 2009.

[8] F. Caballero, L. Merino, J. Ferruz, and A. Ollero, "Vision-based odometry and slam for medium and high altitude flying uavs," *J. Intell. Robotics Syst.*, vol. 54, no. 1-3, pp. 137–161, Mar. 2009.

[9] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[10] Jorge J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*, G. A. Watson, Ed., pp. 105–116. Springer, Berlin, 1977.

[11] Chris Harris and Mike Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[12] Luigi Di Stefano, Stefano Mattoccia, and Federico Tombari, "Zncc-based template matching using bounded partial correlation," *Pattern Recogn. Lett.*, vol. 26, no. 14, pp. 2129–2134, Oct. 2005.