Camera Pose Estimation using Particle Filters

Fernando Herranz* and Kavitha Muthukrishnan** and Koen Langendoen**
*University of Alcalá, Department of Electronics, Alcalá, Spain. Email:fernando.herranz@depeca.uah.es
**Delft University of Technology, Embedded Software Group, Delft,The Netherlands. Email: k.muthukrishnan@tudelft.nl

***Abstract*—In this paper we propose a pose estimation algorithm based on Particle filtering which uses LED sightings gathered from wireless sensor nodes (WSN) to estimate the pose of the camera. The LEDs act as (visual) markers for our pose estimation algorithm. We show the effectiveness of our pose estimation algorithm for different camera frame rates, varying measurement noise and for different marker distribution. Our results (small-scale experimental and room-level simulation studies) show that the particle filtering algorithm gives an accuracy of a few millimetres in position and a few degrees in orientation.**

I. Introduction

Determining the position and orientation (pose) of an object found its application, traditionally, in virtual/augmented reality, gaming and robotics. There are many approaches and technologies to detect and track the pose of an object. For example, mechanical, magnetic, inertial, vision, and hybrid solutions exist, each having its pros and cons [7]. Vision-based tracking systems process image streams from cameras to locate or track people and objects [4]. One of the limitations of vision-based tracking is the inability to easily detect the tracked object's identity. It also has a higher processing cost as detection and tracking algorithms tend to be more complex, due to difficulty in achieving a robust detection. Alternatively, fiducial marker-based systems are available [3] [1]. Markers associated with objects make the task of finding and distinguishing objects easier, especially when the markers are encoded with identification information in some way. Most of the marker-based systems differ in the way the cameras and the markers (also known as landmarks or fiducials) are used as either (i) *Outside-in systems* – where a set of cameras are placed at static points in the environment to monitor objects within that environment, or (ii) *Inside-out systems* – where one or multiple cameras carried by an object can determine its position and orientation in relation to a set of static markers placed in the environment.

The system we present in this paper is an inside-out system that, makes use of LED sightings gathered from wireless sensor nodes to estimate the pose of the camera (shown in Figure 1). Our system consists of an outward looking *camera unit* (CCD camera) whose pose is to be estimated and a set of *static LED markers*. The camera unit observes a set of LEDs that are sequentially flashed (one-at-a-time). We flash the LEDs one-at-a-time as this enforces point correspondence. The *communication and synchronisation* between the markers is coordinated by the WSN. The observation (or measurement) are the 2D image coordinates $[u, v]$ of 3D scene points $[x, y, z]$. Given the intrinsic camera parameters, the location of the LED markers and their corresponding pixel coordinates $[u, v]$ the camera's pose (position and orientation) can be computed. There are two crucial differences between our approach and that of prior work on pose estimation: first, we utilise LEDs on the wireless sensor nodes for pose-estimation purposes (LEDs have been used in sensor nodes mostly for visual inspection and debugging purposes, and we are extending their usage to pose estimation); secondly, we use the radio transceiver on the nodes to transmit their identifiers, thus even further reducing the processing cost compared to fiducial-based vision systems. The main contribution of this paper is the formulation of a pose estimation algorithm based on particle filtering which, uses LED sightings gathered from wireless sensor nodes (WSN) to estimate the pose of the camera. We also consider the effectiveness of the presented algorithm for different camera frame rates, measurement noise and under different LED visibility conditions using a mix of experimental and simulated data.

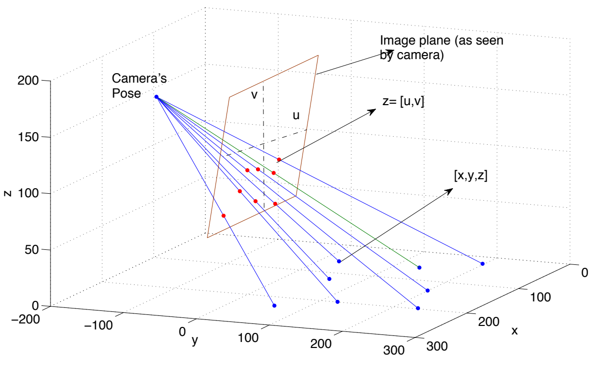

Fig. 1: Camera pose (position and orientation) estimation from observing eight LED markers.

II. Preliminaries: LED detection and Camera model

The LED detection mechanism operates on the raw distorted image. The image coordinates $[u, v]$ of the brightest pixel in the image are considered as a first estimate for the pixel coordinates of the LED. This first estimate is improved by a sub-pixel analysis phase. This is done by taking a weighted average of the pixel locations around $[u, v]$. The weights themselves are just intensities of the pixels minus some dynamic threshold.

In this work we use a standard pin-hole camera model [2]. The 3D coordinates of a LED (sensor node) is defined as $[x, y, z]^T$ and the corresponding projection on the camera image plane $\boldsymbol{z}$ is $[u, v]^T$. These are related by $s\widetilde{\boldsymbol{z}} = P\widetilde{\boldsymbol{x}}$, where the tilde on the vectors indicate they are in homogeneous coordinates, s is a scale factor and P is a 3x4 projection matrix defined up to scale. The projection matrix P is the composition of the camera intrinsic matrix K and the extrinsic parameter matrix $[R \quad \boldsymbol{t}]$. The latter transforms points from the world coordinate system to the camera coordinate system; R is a rotation matrix and $\boldsymbol{t}$ is a translation vector.

$$P = \boldsymbol{K}[R \quad \boldsymbol{t}] \tag{1}$$

III. Pose Estimation Algorithms: Overview

A. Particle Filters for Camera Pose Estimation

Pose estimation involves calculating the rotation matrix R and translation vector $\vec{t}$ (i.e the extrinsic parameters of the camera), given the camera intrinsic matrix K, the locations of the LED markers, and the measured pixel coordinates of the sighted LEDs together with their identities. In this section we present our algorithm based on Particle filtering.

The particle filter (PF) is a form of Bayesian estimation which is used to track the pose of the camera. The PF is a recursive state estimator which has the ability to deal with non-gaussians and multimodal probability density function (pdf). We maintain the camera's position, orientation and their derivatives as the state vector. The complete state is then represented by $state = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}] = [\vec{x}, \vec{v}, \boldsymbol{\alpha}, \boldsymbol{\omega}]$. Instead of storing the Euler angles $\boldsymbol{\alpha}$ (i.e., the orientation of the camera) we store the rotation matrix $R_{\boldsymbol{\alpha}}$ that represents this angle. In the prediction phase of the filter we incorporate the knowledge of the system model and in the measurement phase, we incorporate the pixel coordinates of the detected LEDs in the image plane.

The key idea of PF is to represent the pdf by a set of random samples with associated weights and to compute the estimates based on these samples and weights. In the initialisation phase, the particles are uniformly distributed around the environment in order to cover all the space because it is assumed that the system does not have any previous knowledge about the initial pose of the camera. If the system has some knowledge about the initial pose of the camera the particles can be smartly distributed decreasing the number of particles needed and hence decreasing the computational complexity of the algorithm.

A set of particles are usually denoted $\chi := \{x_t^{(1)}, w_t^{(1)}\}, \ldots, \{x_t^{(j)}, w_t^{(j)}\}, \ldots, \{x_t^{(M)}, w_t^{(M)}\}$ where $x_t^{(j)}$ represents the state and $w_t^{(j)}$ the importance factor or weight of the particles. Here M denotes the total number of particles. So, having a set of particles the PF is capable of following several hypothesis at the same time.

The particles are moved during the prediction step in order to generate a hypothetical state $\{x_t^{(M)}\}$ for time t based on the previous state $\{x_{t-1}^{(M)}\}$. This step involves sampling from the state transition distribution $p(x_t|x_{t-1})$. Subsequently, the importance factors are computed to incorporate the measurement z_t into the particle set. The measurement model is used to predict the ideal noise-free response for each of the LED's 3D position projection in the image plane given the state of each particle. In order to predict the measurement, it is needed to describe how the measurements are related to the state. The measurement model is:

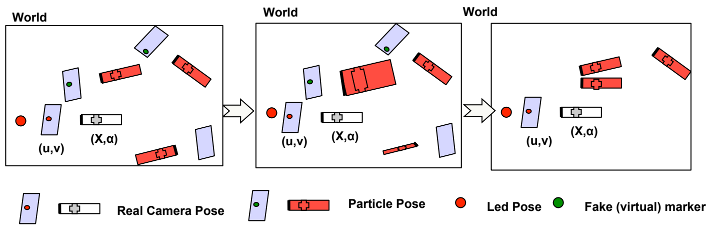

Fig. 2: Particle filters – (i) Initialisation: Each particle observes a virtual marker in the image plane, (ii) Importance weight: the particles are weighted based on the difference between the virtual and the real marker.

$$\hat{\boldsymbol{z}}_i^{(j)} = \boldsymbol{h}_i(\hat{\boldsymbol{x}}^{(j)}, \hat{\boldsymbol{\alpha}}^{(j)}) \tag{2}$$

where $\boldsymbol{h}()$ is the composition of two functions. The first one is the projection of the 3D location of the LED marker i by the projection matrix P in Eq. 1. P is parameterised in location and angle elements of the state vector, that is, $P = P(\boldsymbol{x}, \boldsymbol{\alpha})$. The second function in the composition of $\boldsymbol{h}()$ is the conversion of the resulting vector in homogeneous representation to normal representation.

The importance factor is given by $w_t^{(j)} = p(x_t|z_t^{(j)})$. In order to compute the weight of the particles a gaussian function is used. This step is one of the most important point in this work and is illustrated in Figure 2. Imagine that the position of the LEDs are known and the image of the camera is measured based on the known position of these LEDs and the unknown pose of the camera. A set of particles are randomly distributed around the environment and each particle computes a fake or virtual image based on its state and the known position of the LEDs. When the particles are weighted, the PF compares the fake image of each particle with the real image by $\Delta\boldsymbol{z}^{(j)} = \boldsymbol{z}_t - \hat{\boldsymbol{z}}_t^{(j)}$ and the PF uses this value and a gaussian function to compute the importance factor. Finally, the resampling step is executed. It refocuses the particle set to regions in state space with high posterior probability. By doing so, it focuses the computational resources to the regions that are more valuable. So, resampling draws with replacement M particles that are going to approximate the pdf. The probability of drawing each particle is given by its importance weight. Thus, it transforms a set of M particles into a new set with the same size in which particles with low weight are not copied into the new set and the particles with high weight (close to the camera pose) are drawn and copied into the new set.

IV. Results and Discussion

In this section, we give a brief overview of the camera and sensing platform that we have used for our work, we then explain our experimental set-up used for performing data collection and subsequently, evaluate the performance of our pose estimation algorithm.

TABLE I: Values for the fixed parameters in the experiments.

parameter	value
Standard deviation of pixel noise	1 pixel
Frame rate	90 fps (frames per second)
Rotation speed of camera	$\pi/2$ rad/s
Number of particles	5000

A. Hardware platform and Experimental set-up

a) Hardware platform: The camera is a Fire-i$^{\text{TM}}$ Digital Board Camera. It is a 1/4" CCD camera with a resolution up to 640 $\times$ 480 pixels and a frame rate up to 30 Hz. It has a focal length of 2.1 mm and a horizontal viewing angle of 80.85 degrees. The wireless sensor nodes are of type MyriaNode V3[1], they are based on an Atmel XMega micro controller, a Nordic nRF24L01 radio, and contain by default a number of LEDs in the visible spectrum.

b) Experimental set-up: In our set-up we use a single camera, eight fixed sensor nodes serving as radio-controlled markers, and one sensor node to interface the network to the PC. We performed an experiment to estimate a circular trajectory of the camera's position together with the orientation of the camera (for more information on the experimental setup refer to [6]). Instead of physically moving the camera around the markers we emulate this movement by letting the markers rotate while the camera is fixed. There are two reasons for doing this: (1) the experiment is easy to conduct, leading to greater ground truth accuracy and (2) the effect of changing lighting conditions is reduced. In a real scenario LED sightings might be missed, but for the purpose of our experiment we like to collect a complete data set. The sensor nodes run software that let the LEDs blink in sequence; at any point in time at most one LED is flashed. We construct a dense data set by freezing the camera until all eight LED sightings have been captured.

B. Evaluation

We evaluate our PF's performance using a mix of experimental and simulated data. The metrics used are *position error*, the length of the vector from the estimated location to the true location; and *angular error*, the magnitude of the angle of the single rotation from the estimated orientation to the true orientation. The experiments are done by sweeping over one parameter and fixing all the others. The fixed parameters always have the same values in all the experiments and are shown in Table I.

1. Effect of number of LEDs: Figure 3 shows the performance of PF for different number of markers (4 and 8) used. In general, the performance of the algorithm degrades with less number of markers (ninetieth percentile: position error 22.93 mm vs. 9.12 mm (4 and 8 markers) and angle error 2.85 deg vs. 1.71 deg (4 and 8 markers)). However, we also observed that when less than four markers were used, particle filters did not converge. This minimum set of markers are also required by other batch estimation method such as direct linear transform (DLT) as used by the computer-vision community [5] [6].

[1] http://wsn.chess.nl/

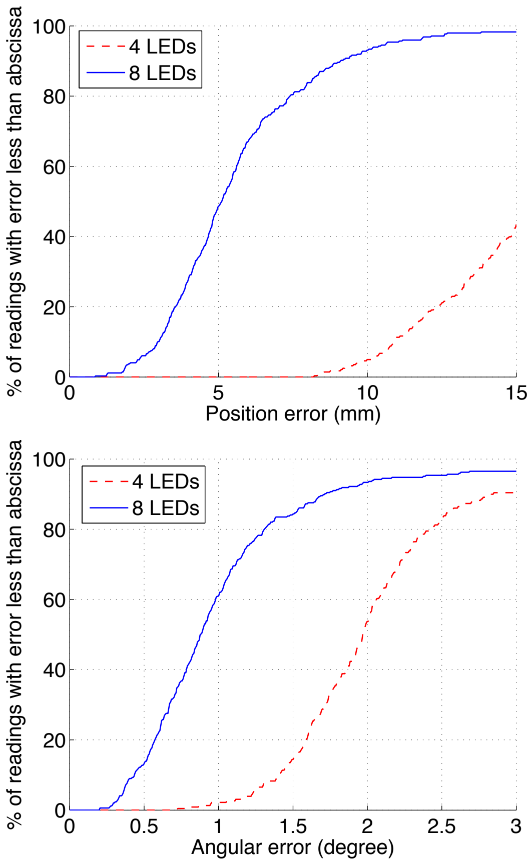

Fig. 3: PF error distribution for different number of LEDs (experimental data).

2. Effect of camera framerates: Figure 4 shows the performance of the algorithm for frame rates in the set $\{30, 90, 300\}$ fps. There is clearly a tradeoff between accuracy and camera frame rates: ninetieth percentile position error being 31.2 mm vs. 6.03 mm (30 fps and 300 fps respectively) and angle error being 10.8 deg vs. 0.5 deg (30 fps and 300 fps respectively).

3. Effect of measurement noise: Position and angle errors for different noise levels are shown in Figure 5. By measurement noise we refer to the difference between detected location of LED in image plane and its true location in image plane. Our results show that the particle filtering algorithm is more robust to noise level as it obtains almost the same results independently of the noise level.

4. Effect of particle size: Table II shows how the performance of PF can be impacted by varying the number of particles. As we can expect, increasing the number of particles results in improved results, however this also significantly increases the computational time. We measured the execution times per pose estimate (in matlab). PF requires 211.1 ms (1000 particles), 958 ms (5000 particles) and 3672 ms (20000 particles). We observe that a good compromise can be achieved between accuracy and computational cost using 5000 particles.

5. Room-scale simulation:

To quantify the effect of our algorithms performance over large area (room-level) we perform simulations over an area of

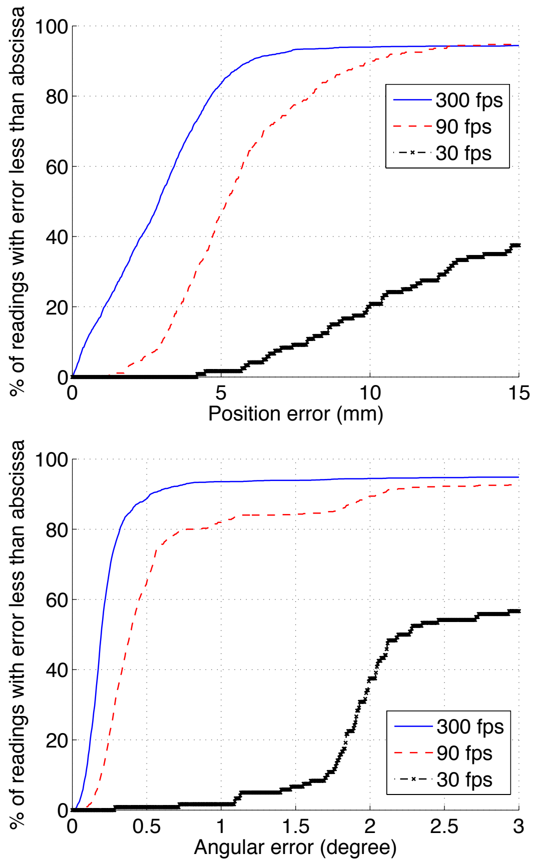

Fig. 4: PF error distribution for different frame rates (simulated data).

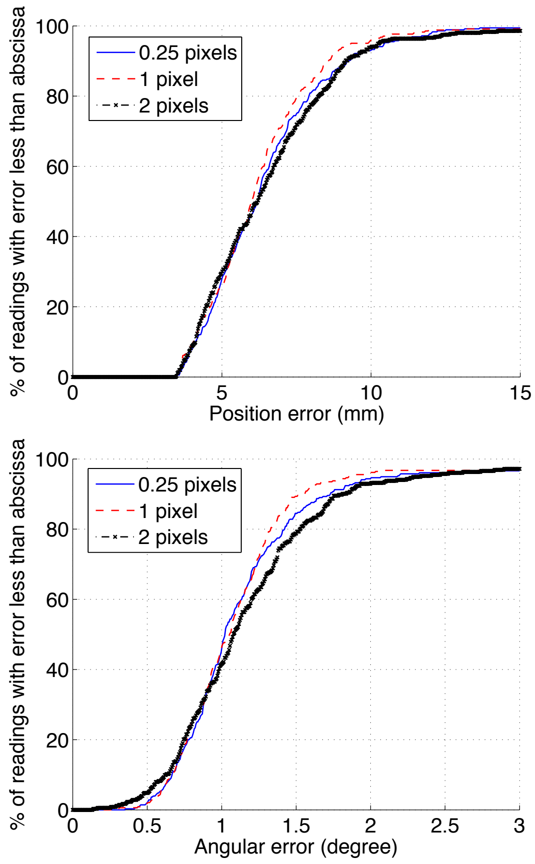

Fig. 5: PF error distribution for different measurement noise (experimental).

Metric/Particle density	1000	5000	10000	20000
Position error (mm)	13	6.65	6.4	5
Angle error (deg)	3.25	2.25	1.6	1.55

TABLE II: Particle filters performance (ninetieth-percentile values) for various particle densities using experimental data.

5 x 5m. The marker distribution is assumed to be random. We used two different types of camera movement in our simulations (i) camera moving randomly from the center of the arena to one side of the room and (ii) camera moving in random fashion over the whole deployment area. In Table III we plot the results of our simulation studies which are averaged over 100 simulation runs for varying marker density.

	Position Error (in mm)		Angular Error (in deg)	
Marker density	50% conf.	90% conf.	50% conf.	90% conf.
30	21.5	61.8	1.85	5.78
20	37.8	83.5	2.14	7.53
10	67.4	172.95	3.09	11.05

TABLE III: Performance summary of particle filter (using simulated data). Results averaged over hundred simulations.

V. Conclusions and Future work

In this paper we described a pose estimation algorithm based on Particle filters. Our algorithm uses LED sightings gathered from wireless sensor nodes (WSN) to estimate the pose of the camera. We evaluated the performance of the algorithm using a mix of experimental (small-scale) and simulated (large-scale) data. We showcased the effectiveness of PF with simulated data for different camera frame rates, varying noise levels and under different LED visibility conditions. In future, we would like our particle filter implementation to be compared with extended Kalman filtering (EKF) approach.

References

[1] R. Bencina and M.Kaltenbrunner. The Design and Evolution of Fiducials for the reacTIVision System. In *Proceedings of the Third International Conference on Generative Systems in the Electronic Arts, Melbourne (Australia)*, 2005.

[2] G. Bradski and A. Kaehler. *Learning OpenCV*. OReilly Media Inc, 2008.

[3] D. L. de Ipĩna, P. R.S.Mendonca, and A. Hopper. TRIP: A Low-Cost Vision-Based Location System for Ubiquitous Computing . *Personal and Ubiquitous Computing*, 6:206–219, 2002.

[4] S. Hay, J. Newman, and R. Harle. Optical tracking using commodity hardware. In *Proceedings of the Seventh IEEE and ACM International Symposium on ISMAR 2008*, 2008.

[5] V. Lepetit and P. Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. In *Foundations and Trends in Computer Graphics and Vision*, pages 1–89, 2005.

[6] E. Rijpkema, K. Muthukrishnan, S. Dulman, and K. Langendoen. Pose estimation with radio-controlled visual markers. In *In Third International Workshop on Mobile Entity Localization and Tracking (MELT 2010)*, 2010.

[7] G. Welch and E. Foxlin. Motion Tracking: No Silver Bullet, but a Respectable Arsenal. *IEEE Comput. Graph. Appl.*, 22(6):24–38, 2002.