Camera Pose Estimation using Particle Filters

Fernando Herranz
Department of Electronics
University of Alcalá
Alcalá, Spain
Email: fernando.herranz@depeca.uah.es

Kavitha Muthukrishnan, Koen Langendoen
Embedded Software Group
Delft University of Technology
Delft, The Netherlands
Email: k.g.langendoen@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 also compare the performance of our pose estimation algorithm against two reference algorithms – (i) Extended Kalman filtering (EKF) and (ii) Discrete Linear Transform (DLT) based approaches. The performance of all the three algorithms are evaluated for different camera frame rates, varying level of 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 [12]. Vision-based tracking systems process image streams from cameras to locate or track people and objects [5]. 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.

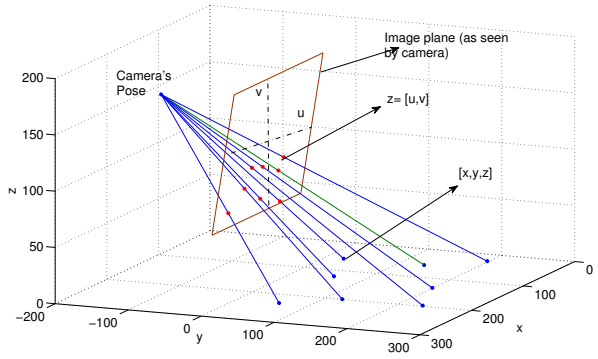


Fig. 1: Camera pose (position and orientation) estimation from observing eight LED markers.

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 compare the performance of particle filtering approach with two other reference algorithms – an Extended Kalman filtering (EKF) approach and Discrete Linear Transform approach (DLT). 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.

II. Related work

Vision-based tracking systems process images from cameras to locate or track people and objects. Several tracking systems have already been developed for augmented reality and virtual

978-1-4577-1804-5/11$26.00

gaming kind of applications. In this section, we summarize some of the vision-based systems with and without markers. A real-time 3D tracker for use with head-mounted displays is described by Ward et al. [10]. Three cameras, mounted on a helmet worn by the user, view an array of infrared-emitting LEDs fixed to the ceiling. A tracking controller individually illuminates the LEDs in turn. By determining which LEDs are actually seen by which camera, the helmet can be tracked with a resolution of around 2 mm in position and 0.1 degree in orientation. However, the camera arrangement is bulky and is tethered to a control unit, and a very large number of LEDs must be accurately placed on the ceiling rendering the system of little practical use.

Gottschalk et al. proposed and implemented an auto-calibration method [4] using rough LED location estimates, and thousands of observations from unknown locations. The system estimates the position at each test location, and calculates back the estimates of the LED positions. These two steps are repeated until the position estimates converge. Welch and Bishop [11] describe the complete HiBall tracking system, including novel optical, mechanical, electrical, and algorithmic aspects that enable the system to generate over 2000 head-pose estimates per second with less than one millisecond of latency, and less than 0.5 millimetres and 0.02 degrees of position and orientation error. The HiBall system uses Kalman filters to accomplish on-line auto-calibration, allowing the system to continually update the LED location estimates during normal operation. The HiBall tracking system is much of an inspiration to our work in terms of the inside-out design using fixed LEDs in the infrastructure and capturing the LED sightings to infer the camera's pose.

Recent work of Hay et al. [5] provides a low-cost alternative to optical tracking using commodity hardware such as Wii-motes to determine the pose of an object to an accuracy that is comparable to conventional high-cost systems. However, their system is an outside-in system and uses stereo vision-based methods for pose estimation. Using a commercial platform such as Wiimote is attractive as it is cheap, offers high update rate and runs the image processing in an embedded DSP. On the down side, the Wiimote can track only up to four infrared light sources and does not offer access to the raw image.

One example of a camera tracking system developed in the context of ubiquitous computing is the Pfinder system [13]. It uses 2D-models applied to images taken by fixed cameras to keep track of the movements of a user. Another system using stereo vision cameras is Easy Living [6]. One limitation of both these systems, and of vision-based tracking in general, is the inability to easily detect the identity of the tracked objects. Vision-based tracking requires more processing power, as detection and tracking algorithms tend to grow in complexity due to the requirement of achieving a robust detection. Marker-less approaches (using natural appearance such as colour, texture or features) are also being used in vision-based tracking. They however, require an off-line training phase and processing cost is generally high. For a detailed survey of 3D-model based detection and tracking refer to [7].

As previously mentioned, fiducial marker-based systems are a valid alternative to the already presented systems. 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. Typically, fiducial markers are printed planar patterns (similar to barcodes) that a vision-only system can easily detect, track and decode. An example in this category is TRIP (Target Recognition using Image Processing) [3]. Users wear passive tags displaying 2D circular bar codes. Cameras in each room capture images that are analysed to identify tag wearers in the field of view. ReacTIVision [1] is another fiducial marker-based system used in many tangible interface applications.

Another recently developed system is CLIPS (Camera and Laser based Indoor Positioning System) [9]. CLIPS determine the pose of a mobile camera in respect to a laser rig. Since the rig emits laser-beams from a virtual central point, it can be regarded as an inverse camera. From the bright laser spots that are projected to any surface without any specific structure of the scene, the relative orientation between the camera and the laser rig can be computed.

There are systems that combine odometry with vision. In [14] the authors propose a particle filter based algorithm for monocular vision aided odometry for mobile robot localization. The algorithm fuses information from odometry with observations of naturally occurring static point features in the environment.

On the whole, there are two crucial differences between our approach and that of prior work on pose estimation: first, we utilize 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.

III. 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.

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

IV. Pose Estimation Algorithms: Overview

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 first present our algorithm based on Particle filtering. Subsequently, we present our two reference algorithms that are based on Extended Kalman filtering and Discrete Linear Transform.

A. Particle Filters for Camera Pose Estimation

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:

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

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.

B. Algorithms we compare:

In this section we present an overview of the two reference algorithms – an extended Kalman filtering method (EKF) and Discrete Linear Transform (DLT) method. EKF is a state estimation, which uses a form of Bayesian estimation to estimate the pose of the camera and many works within the computer-vision community have used Direct Linear Transform (DLT) method [7]. For more details on these algorithms refer to [8].

a) ***EKF algorithm****:* We maintain the camera's position, orientation and their derivatives as the state vector. The complete state is then represented by $state = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}] = [\vec{x}, \vec{v}, \boldsymbol{\alpha}, \boldsymbol{\omega}]$. Actually, 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 filter is initialized with a state estimate $\hat{state}$ and uncertainty or error covariance $\hat{P}$. We set the initial state to the ground truth value from the experimental set-up and added

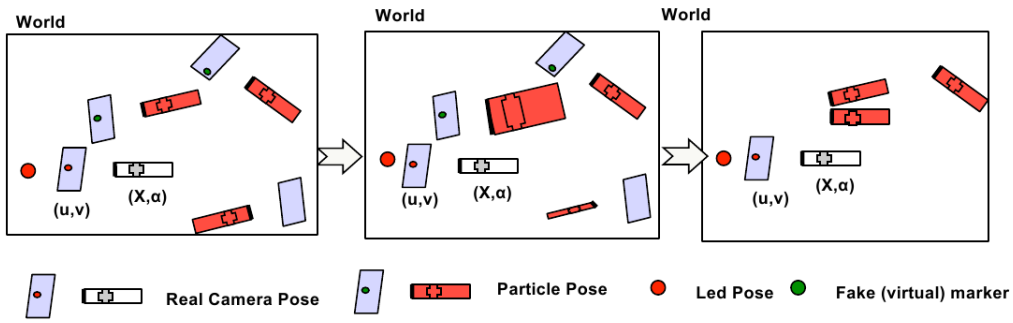


Fig. 2: Particle filters – (i) Initialisation: Each particle observes a virtual marker in the image plane, (ii) Importance weight: the particles are weighted based on the difference between the virtual and the real marker.

some noise to this value. This is to mimic an initial state estimate that would be obtained from a practical implementation, based on DLT (see below) for example.

We use a constant-velocity and constant-angular-velocity model. (i.e., the camera moves at constant speed and rotates at a constant speed between time steps). Note that the true mobility of the camera violates the constant velocity model, as the direction of the velocity vector continuously changes. This reflects reality, for which we would not have a perfect matching model available and accounts for the process noise in the Kalman filter.

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 filter's current estimate of the state (camera pose). Based on the measurement noise the filter can either weight measurements more or its predictions. The final step is the filter update to correct the state estimate per most recent measurement and to update the error covariance.

*b) **DLT-based algorithm** (**RefAl**):* DLT is used to estimate the projection matrix P by solving a linear system of equations. However, DLT provides a homography that minimises the algebraic transfer error, which in our case, is equivalent to the geometric error. Instead of finding P=$\mathbf{K}$[R $\mathbf{t}$] it finds a homography P' such that $K^{-1}P'$ yields a matrix whose first part is *not* a rotation matrix. After reorthogonalization and converting the rotation matrix to Euler angles new errors are introduced. Hence, the camera parameters estimated by this method should be refined by iterative optimization of the non-linear reprojection error. Several methods for performing this iterative optimization are discussed in [7].

We use the Levenberg-Marquardt (LM) method to further refine the initial guess based on the DLT algorithm. We use this combination of DLT- and LM-based methods as our reference algorithm (*RefAl*).

V. Hardware platform and Experimental set-up

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.

A. Hardware platform

The camera is a Fire-i™ Digital Board Camera. It is a 1/4" CCD camera with a resolution up to 640 × 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. 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.

In this work, the WSN triggers the camera to capture the image when the LED is flashed, because the WSN runs a TDMA-based MAC layer that does not allow for external synchronization. Alternatively, one can use cameras with trigger modes, or create a system in which the camera triggers the wireless sensor nodes to blink for instance, using a master node connected to the camera unit. However, having the network trigger the image capture has the additional advantage of being able to use multiple cameras.

The experimental setup for the circular test is shown in Figure 3. It shows the rotating plate of a turn table (the turntable itself is not depicted). On top of this plate is a square fixture that holds eight radio-controlled markers; the rectangles are the sensor nodes and the small squares on the sensor nodes are the LEDs. The turn table is rotated manually,

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

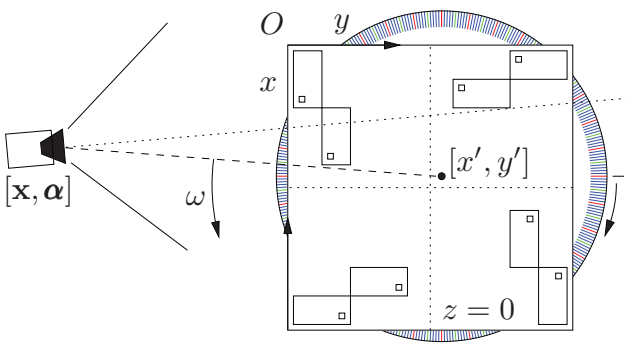


Fig. 3: Experimental setup used for data collection. A rotating plate supports eight radio-controlled markers (LEDs). The turn table is rotated manually, one degree at a time, indicated by the arrow next to the turn table. For each rotation we captured eight images; one for each LED sighting. Finally, we collected eight LED sightings for 360 camera poses.

indicated by the arrow next to the turn table. This emulates the rotation of the camera around the markers. To support the manual rotation we have put a 1 degree angular scale on the rotating plane. We carefully rotated the ground plate; one degree at a time. For each rotation we captured eight images; one for each LED sighting. The one degree measurements are done for one revolution of the turn table, that is, we collected eight LED sightings for 360 camera poses. The trajectory of the camera has a longitude of 2253 mm and the maximum distance between the camera and the LEDs is 539 mm.

VI. Performance Evaluation

We benchmark our particle filtering algorithm against EKF and DLT-based method 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. We particularly consider the effectiveness of our algorithms' for varying number of marker distribution, frame rates and measurement noises.

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

1. Effect of number of LEDs: Figure 4 shows the performance of PF, EKF and DLT for different number of markers (4 and 8) used. In general, the performance of the algorithm degrades slightly with less number of markers (seventieth percentile: position error 3.78 mm vs. 3 mm (4 and 8 markers) and angle error 0.55 deg vs. 0.5 deg (4 and 8 markers)). While EKF performs quite similar to the particle filters, DLT algorithm fairs slightly better when 8 markers are used. However, we also observed that when less than four markers were used (Figure 5), both EKF and particle filters did converge and produced meaningful results. Contrary to the minimum amount of markers that is required to compute a pose using the DLT (RefAl), EKF and PF is able to estimate the pose even when the measurements are under-constrained, even for a one-marker case (seventieth percentile: position error 8 mm and angle error 1.5 deg approximately). This is because both these algorithms use the predicted estimate to compute the pose of the camera.

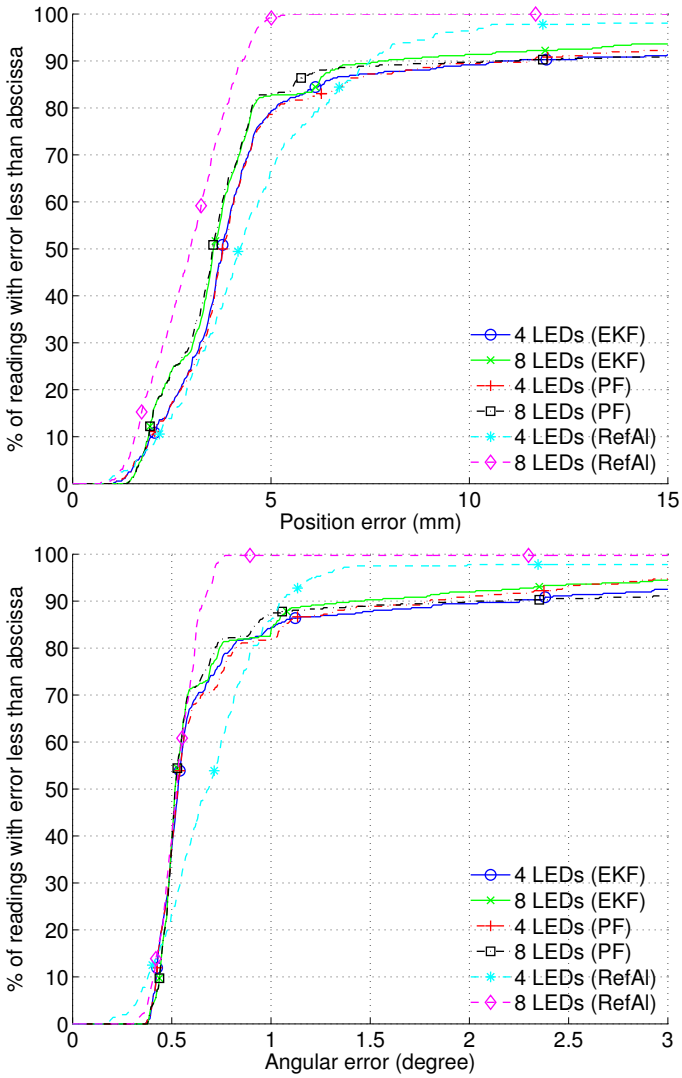
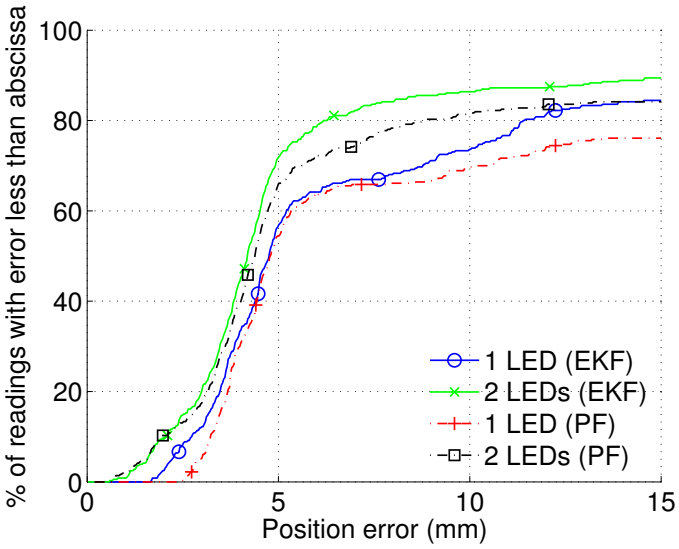


Fig. 4: Error distribution for different number of LEDs (experimental data).

2. Effect of camera framerates: Figure 6 shows the performance of the algorithm for frame rates in the set $\{30, 90, 300\}$ fps. As one would anticipate, the higher the frame rate, the better is the accuracy: seventieth percentile position error being 3.5 mm vs. 23 mm (30 fps and 300 fps respectively, for



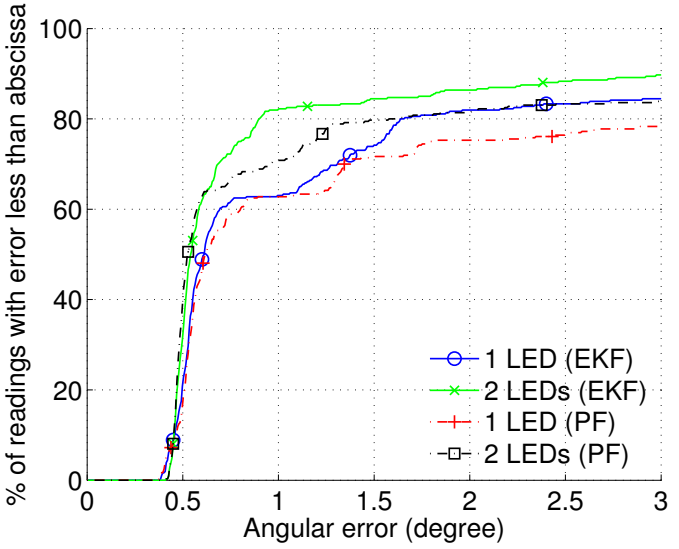
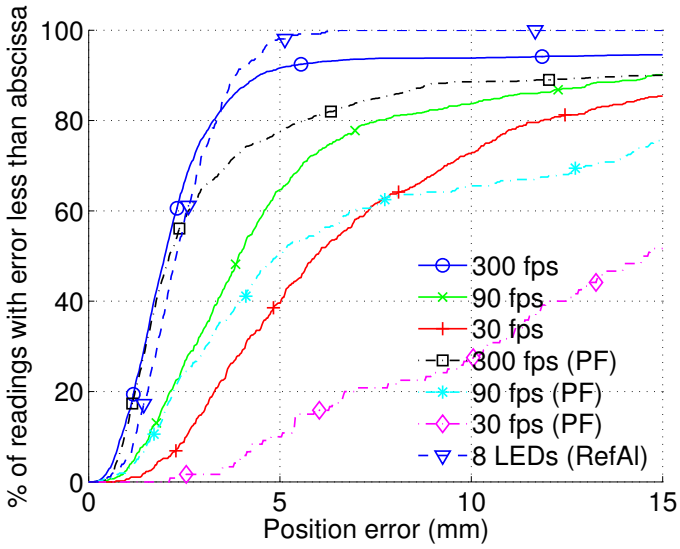


Fig. 5: Error distribution for 1 and 2 LEDs (experimental data).



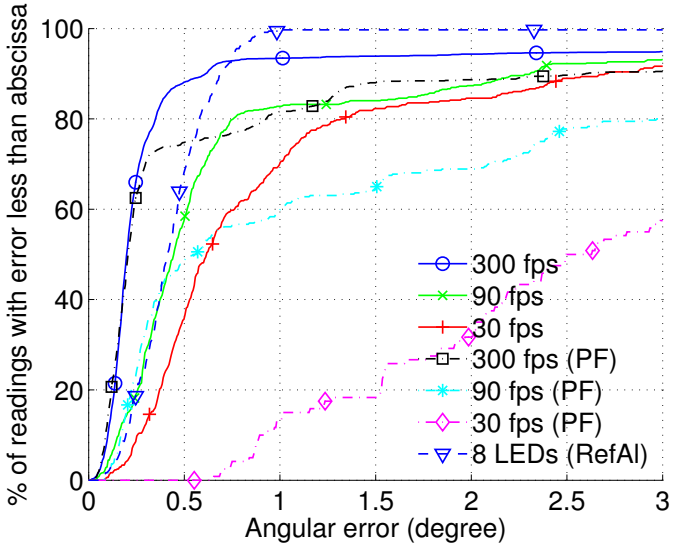


Fig. 6: Error distribution for different frame rates (simulated data).

particle filters) and angle error being 3.5 deg vs. 0.3 deg (30 fps and 300 fps respectively). We do observe that the EKF performs better in comparison to PF, and this is because of two reasons (i) with increased frame rates the accuracy of the EKF prediction improves and (ii) EKF requires an initialization, where as in the case of PF the particles are uniformly distributed and thus the error is slightly larger for the first few measurements.

3. Effect of measurement noise: Position and angle errors for different noise levels are shown in Figure 7. 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 slightly more robust to noise level when compared to the EKF. Figure 7 also shows that for the same noise level (1 pixel) PF performs better than EKF and RefAl.

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.

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. Room-scale simulation:

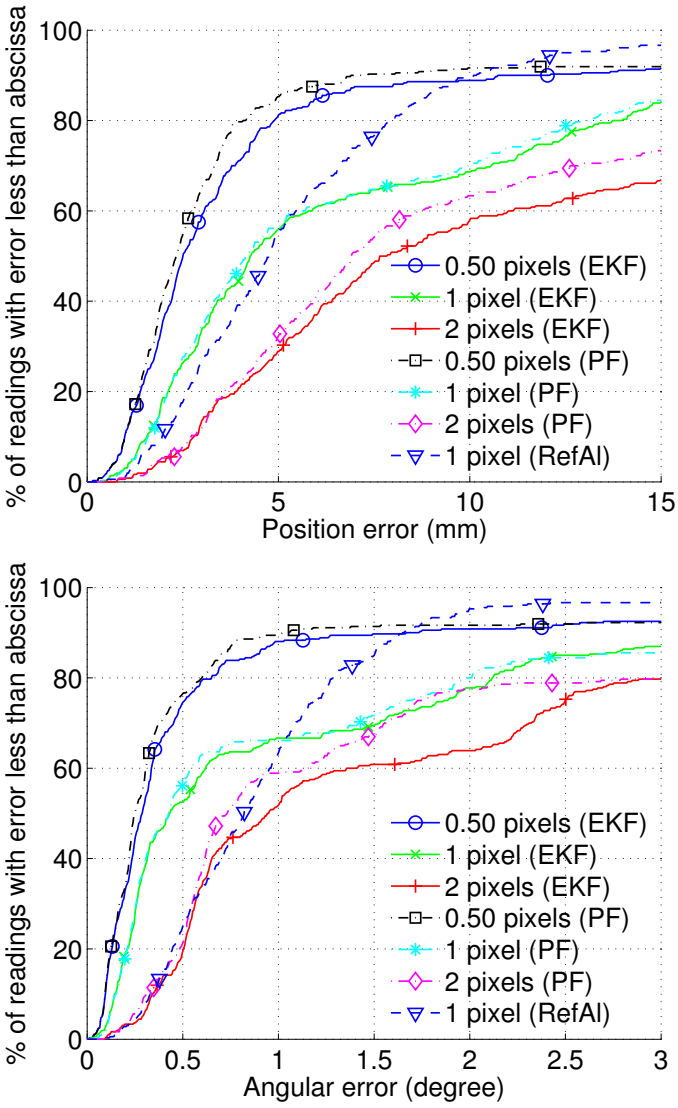


Fig. 7: Error distribution for different measurement noise (experimental).

To quantify the effect of our algorithms' (PF) performance over large area (room-level) we perform simulations over an area of 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. One can observe that with increased markers, the error (position and angular) improves significantly.

	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.

VII. 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. The PF has also been compared with RefAL, an algorithm that is based on Discrete Linear Transform (DLT) and an Extended Kalman filtering approach. In most cases, the performance of PF is similar to EKF. We also tested the algorithms when measurements are under-constrained (less than four markers). Contrary to the EKF and PF, the DLT based algorithm needs more measurements and more iterations to estimate a pose.

In future we plan to analyze the practical performance of our algorithm by running a large-scale experiment using more sensor nodes and use additional types of sensor data such as from accelerometer and gyroscope. While we have used constant-velocity and constant-angular velocity model in both our EKF and PF implementation, we would like to analyze the impact of other motion models.

Acknowledgment

We are grateful to Dr Edwin Rijpkema for providing insightful suggestions on this work and for extending his support for performing the experiments and data collection.

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 Ipiña, 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. Gottschalk and J. F. Hughes. Autocalibration for virtual environments tracking hardware. In *Proceedings of the Twentieth Annual Conference on SIGGRAPH 1993*, pages 65–72. ACM, 1993.

[5] 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.

[6] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer. Multi-Camera Multi-Person Tracking for EasyLiving. In *Proceedings of the Third IEEE International Workshop on VS*, pages 3–10. IEEE Computer Society, 2000.

[7] 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.

[8] 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.

[9] T. S and M. R. Development of a new optiocal indoor positioning sysem. In *ISPRS Commission V Midterm Symposium*, volume 98, pages 575–580, 2010.

[10] M. Ward, R. Azuma, R. Bennett, S. Gottschalk, and H. Fuchs. A Demonstrated Optical Tracker with Scalable Work Area for Head-Mounted Display Systems. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics*, pages 43–52, Cambridge, MA, March 1992.

[11] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. The HiBall Tracker: high-performance wide-area tracking for virtual and augmented environments. In *Proceedings of the ACM symposium on VRST '99*. ACM, 1999.

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

[13] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-Time Tracking of the Human Body. In *Proceedings of SPIE*, volume 2615, pages 89–98, 1996.

[14] T. N. Yap, Jr., M. Li, A. I. Mourikis, and C. R. Shelton. A particle filter for monocular vision-aided odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.