Pose Estimation with Radio-Controlled Visual Markers[‡]

Edwin Rijpkema[*] Kavitha Muthukrishnan[†] Stefan Dulman[†] Koen Langendoen[†]
[*]Alten PTS
The Netherlands
[†]Delft University of Technology
The Netherlands

***Abstract*—Camera pose estimation (i.e. determining the position and orientation of a camera) found its applications, traditionally, in the field of virtual/augmented reality, gaming and robotics. In this paper we propose an inside-out system that 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, which is based on Extended Kalman filtering (EKF). We compare the performance of our EKF algorithm against an algorithm based on Discrete Linear Transform (DLT). We also consider the effectiveness of the presented algorithm for different camera frame rates, varying noise levels and varying LED visibility conditions using a mix of simulated and experimental data. Our initial results are promising and show that the EKF algorithm gives an accuracy of a few millimetres in position and few degrees in orientation, even under sparse LED conditions, low frame rates and high noise levels.**

I. Introduction

Pose estimation (i.e., determining the position and orientation 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 [18]. Although vision-only solutions offer an unobtrusive way of tracking an object (without tagging them), it is often considered expensive in terms of processing cost. However, with the increased availability of small-sized cameras, with embedded digital signal processors (DSPs), the range of applications where cameras are being used is becoming significantly larger. An example of this is the Nintendo's Wii remote, which revolutionised the gaming industry with its innovative interaction technology [10] [14].

Vision-based tracking systems process image streams from cameras to locate or track people and objects. 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. 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.

In this paper we explore the feasibility of using vision in combination with *wireless sensor networks* (WSNs); two technologies that were previously considered incompatible because of the high costs associated with cameras and the limited resources provided by typical WSN platforms. The range of applications opened by this combination is impressive. Two examples, situated at the opposite sides of the spectrum of possibilities are: *(i) Pose estimation of UAVs* – there is a growing interest in flying unmanned aerial vehicles (UAVs) indoors for applications such as building surveillance and disaster/rescue support by utilising flying objects [6]. The information about the pose of the camera-equipped device can be used in stabilising the UAV or in estimating relative distances and angles between UAVs and *(ii) Mapping radio environments* – using a single sensor node with pose estimation capabilities allows easy and precise fingerprinting of the indoor radio environment [9]. Fast measurement of the radiation pattern of a node when a particular type of antenna is being used, or a specific casing, or a specific orientation in placement, etc. becomes feasible.

We propose an inside-out system that, for the first time, makes use of LED sightings gathered from wireless sensor nodes to estimate the pose of the camera. Incorporating visual markers in the system enforces point correspondences, and reduces the processing effort and is thus useful when compared to computer vision-only counterparts. Although there are commercially available optical tracking systems that utilise LED markers [17], the synchronisation infrastructure (between the LED markers and the cameras) can be very expensive, whereas we offer a low-cost solution.

The contributions of this paper are outlined as follows:

- We propose an inside-out system that uses LED sightings gathered from sensor nodes fixed at known positions in the environment (acting as markers) to estimate the pose of the camera.
- We develop an algorithm for pose estimation based on extended Kalman filtering.
- We compare the performance against a reference algo-

[‡]This work is part of the LoCare project (KWR09128) and is subsidised by the Dutch Ministry of Economic Affairs.

rithm based on Discrete Linear Transformation (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

Many approaches have been developed for tracking the pose of an object both within augmented reality and ubiquitous computing (including WSN) communities. In this section, we summarise the most relevant work in the context of this paper, describing vision-based systems with and without markers. For other approaches (i.e., inertial, ultrasound, etc.), we refer the reader to [1] [13].

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 gaming kind of applications. A real-time 3D tracker for use with head-mounted displays is described by Ward et al. [15]. 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 [7] 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 [17] 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. [8] provides a low-cost alternative to optical tracking using commodity hardware such as Wiimotes 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 [19]. 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 [11]. 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 [12].

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) [5]. 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 [2] is another fiducial marker-based system used in many tangible interface applications.

In the pose estimation method that we have developed, we have chosen an inside-out configuration. On the whole, 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. Although there are commercially available optical tracking systems that utilise LED markers, the synchronisation infrastructure between the markers and the cameras can be prohibitively expensive, especially when the system must scale to larger tracking areas, where as we offer a low-cost solution using WSNs.

III. System Overview and Design Choices

Our tracking 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 static LED markers. The camera can in principle be fixed to each object to be tracked (e.g., micro aerial vehicle, UAV or a person). The *communication and synchronisation* between the markers is coordinated by the WSN.

The basic working principle is illustrated in Figure 1. 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

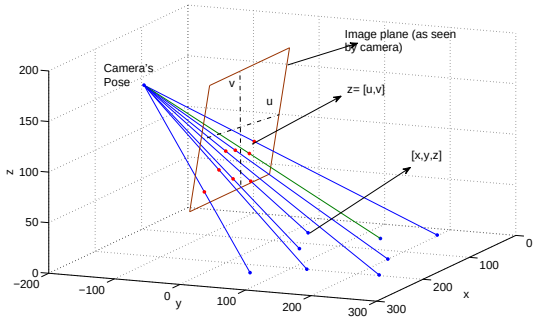

Fig. 1: Camera pose estimation from observing eight LED markers. The figure represents an over-constrained system.

this enforces point correspondence. The observation (or measurement) are the 2D image coordinates $[u, v]$ of 3D scene points $[x, y, z]$. Given the intrinsic camera parameters (from the calibration toolbox, refer Sect. IV-B), the location of the LED markers and their corresponding pixel coordinates $[u, v]$ the camera's pose can be computed. The hardware platform used for this work is described in Section VI.

Figure 2 illustrates how the distortion is corrected in the captured image before being used for pose estimation. We found that it works best to first locate the LEDs in the distorted image and then to undistort (or correct) these points. This also saves the computational load of undistorting the complete image. We used the camera calibration toolbox [3] for finding the distortion parameters.

A. Design Choices

In this subsection we describe three important choices we have taken when designing our system.

i) Inside-out configuration: By placing the cameras on the user rather than in the environment, our system can be scaled indefinitely. The system can evolve from using dense LED markers to fewer.

ii) Imaging sensor: There are a number of options available when looking for imaging sensors. CMOS/CCD cameras are cheap but operate at a low frame rate and require expensive pixel processing. *Lateral effect position sensitive devices* (LEPDs) are expensive, but offer a high frame rate. They work on the principle that they deliver the centroid of the incident light [17]. Another cheap, commercially available, imaging sensor that has attracted attention is the Wiimote, which runs all the image processing in an embedded DSP. It has already been used for outside-in designs [8] [10]. As reported earlier, the Wiimote has the limitation of tracking only four blobs of infrared light and does not report the raw image. Because of this reason, we opted to use a normal CCD camera, as it gives the flexibility to perform camera calibration, helps in robust LED detection and supports a larger number of markers for pose estimation.

iii) Radio-controlled LEDs as markers: We are particularly interested in using the LEDs already present on the WSN nodes as markers in our approach. We believe that using radio for transmitting the identifiers of the nodes can significantly

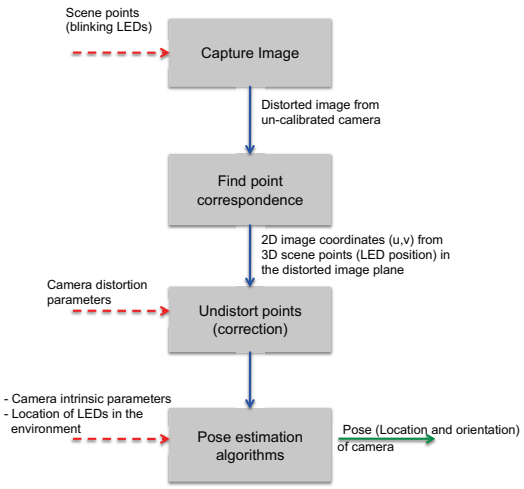

Fig. 2: Overview of inside-out pose estimation.

reduce the processing cost. However, one has to consider that the data rate of transmitting node IDs is directly related to the camera's frame rate. In our work we assume that nodes transmit their identities. Alternative scheme on how to transmit and retrieve node identities is discussed in Lee [10]. Comparing our approach with the fiducial marker approach, fiducial-based schemes can still be expensive in terms of processing. In our approach, we focus only on identifying the brightest pixel in the whole image and do the processing around it. Making use of radio communication also aids point correspondence (matching the intensity of the LED in pixel in view with the node that emitted it) and extracting timing information (when a particular node is blinking). In the future we plan to control the blinking sequence of the nodes that are in the field of view of the camera similar to the approach presented in Welch [17].

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 synchronisation. 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. Although an interesting option, so far we have not studied the possibility for asynchronous operation between the camera and the WSN.

IV. LED DETECTION AND CAMERA CALIBRATION

LED detection and camera calibration are the two main building blocks of our system. This section presents them in detail.

A. LED detection

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.

Let $\mathcal{I}(u, v)$ be the image patch around the $[u, v]$. The dynamic threshold is the mean value of the pixel intensities plus a static threshold.

$$\text{weight}(u, v) = \mathcal{I}(u, v) - (\mu_P + \tau) \tag{1}$$

where μ_P is the mean intensity of all the pixels in the patch and τ is the static threshold. We found that an image patch of size 7 and a static threshold of 30 performs well.

After finding the $\mathbf{z} = [u, v]$ coordinates of the LED in the image plane, undistortion is applied using the parameters of the camera model.

B. Camera model and calibration

In this work we use a standard pin-hole camera model. The 3D coordinates of a LED (sensor node) is defined as $[x, y, z]^T$ and the corresponding projection on the camera image plane $\mathbf{z}$ is $[u, v]^T$. These are related by $s\tilde{\mathbf{z}} = P\tilde{\mathbf{x}}$, 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 $\begin{bmatrix} R & \mathbf{t} \end{bmatrix}$. The latter transforms points from the world coordinate system to the camera coordinate system; R is a rotation matrix and $\mathbf{t}$ is a translation vector.

$$P = K \begin{bmatrix} R & \mathbf{t} \end{bmatrix} \tag{2}$$

We use the camera calibration method described in [3] to simultaneously estimate the intrinsic parameters and the lens distortion parameters. Basically, the OpenCV library provides the parameters that describe the distortion, not the undistortion. We use an iterative algorithm, similar to the undistortion of points in OpenCV, to undistort the individual points found by the LED detection. The undistorted locations of the LEDs in the captured image are the input to our pose estimation algorithms.

V. Overview of Pose Estimation Algorithms

Pose estimation involves calculating the rotation matrix R and translation vector $\mathbf{t}$ (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 a reference algorithm and subsequently present our algorithm based on Kalman filtering.

A. DLT-based algorithm (RefAl)

Many works within the computer-vision community have used Direct Linear Transform (DLT) method [12]. 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 $P = K \begin{bmatrix} R & \mathbf{t} \end{bmatrix}$ it finds a homography P' such that $K^{-1}P'$ yields a matrix whose first part is *not* a rotation matrix. After reorthogonalisation 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 optimisation of the non-linear reprojection error. Several methods for performing this iterative optimisation are discussed in [12].

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*).

B. Extended Kalman Filtering (EKF)

We use a form of Bayesian estimation – Extended Kalman Filtering – to estimate the pose of the camera. We maintain the camera's position, orientation and their derivatives as the state vector. The complete state is then represented by $\mathbf{s} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}] = [\mathbf{x}, \mathbf{v}, \boldsymbol{\alpha}, \boldsymbol{\omega}]$. 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.

a) Initialisation: The filter is initialised with a state estimate $\hat{\mathbf{s}}$ and uncertainty or error covariance $\hat{P}$. We set the initial state to the ground truth value from the experimental set-up and added some noise to this value. This is to mimic an initial state estimate that would be obtained from a practical implementation, based on DLT for example.

b) System model: 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.

c) Measurement model: 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). In order to predict the measurement, we will have to describe how measurements are related to the state. The measurement model is:

$$\hat{\mathbf{z}}_i = \mathbf{h}_i(\hat{\mathbf{x}}, \hat{\boldsymbol{\alpha}}) \tag{3}$$

where $\mathbf{h}()$ is the composition of two functions. The first one is the projection of the 3D location of LED marker i by the projection matrix P in Equation 2. P is parameterised in location and angle elements of the state vector, that is, $P = P(\mathbf{x}, \boldsymbol{\alpha})$. The second function in the composition of $\mathbf{h}()$ is the conversion of the resulting vector in homogeneous representation to normal representation. It is this second part that makes the measurement function non-linear and motivates us to use an extended Kalman filter.

In case the camera detects multiple LEDs, the set of measured pixel locations $\mathbf{z}_i$ is stacked into one large measurement

$\mathbf{z}$. For n simultaneously detected LEDs, the dimension of $\mathbf{z}$ is $2n$. Using such a single measurement at a time is known as the *single-constraint-at-a-time* (SCAAT) approach [16]. The attractive aspect is that the filter also works for so called under constrained measurement, in our case with less than four LEDs at a time. Results on varying number of LEDs are reported in Section VII.

d) Prediction and Correction: We use the Kalman prediction equations at each time-step:

$$\hat{\mathbf{s}}^- = A(\Delta t)\hat{\mathbf{s}}(t-\Delta t) \tag{4}$$

Where $A(\Delta t)$ is the state transition matrix that projects the state forward at each time step. The predicted covariance is given by:

$$P^- = A(\Delta t)P(t-\Delta t)A^T(\Delta t) + Q(\Delta t) \tag{5}$$

where Q is the 12×12 process noise covariance matrix. The structure of Q is in accordance with [4].

$$Q = \begin{bmatrix} Q_{\mathbf{xx}} & Q_{\mathbf{xv}} & 0 & 0 \\ Q_{\mathbf{xv}} & Q_{\mathbf{vv}} & 0 & 0 \\ 0 & 0 & Q_{\boldsymbol{\alpha\alpha}} & Q_{\boldsymbol{\alpha\omega}} \\ 0 & 0 & Q_{\boldsymbol{\alpha\omega}} & Q_{\boldsymbol{\omega\omega}} \end{bmatrix} \tag{6}$$

where we assumed that the location and velocity noise are independent of the angular and angular-velocity noise. The non-zero 3×3 submatrices Q_{ij} in (6) are of the form $c_{ij}\Delta t^{n_{ij}} I_{3\times 3}$. This implies that we assume the process noise in the three elements of each subvector (i.e., $\mathbf{x}$, $\mathbf{v}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\omega}$) independent. The values n_{ij} are set in accordance with [16] and the c_{ij} are found empirically to work well on both the experimental and synthetic data sets.

$$Q_{\mathbf{xx}} = 1/3 \cdot \Delta t^3 I \qquad Q_{\boldsymbol{\alpha\alpha}} = 10^{-3}/3 \cdot \Delta t^3 I \tag{7}$$

$$Q_{\mathbf{xv}} = 10^5/2 \cdot \Delta t^2 I \qquad Q_{\boldsymbol{\alpha\omega}} = 10^{-3}/2 \cdot \Delta t^2 I \tag{8}$$

$$Q_{\mathbf{vv}} = 10^6 \cdot \Delta t I \qquad Q_{\boldsymbol{\omega\omega}} = 10^{-3} \cdot \Delta t I \tag{9}$$

We compute the Kalman gain to determine the proper weighting of the measurement innovation or residuals:

$$\Delta\mathbf{z} = \mathbf{z} - \hat{\mathbf{z}} \tag{10}$$

Here $\hat{\mathbf{z}}$ represents the measurement prediction and $\mathbf{z}$ represents the actual set of (noisy) $[u, v]$ coordinates of each observed LED in the image plane of the camera.

The Kalman gain K (not to be confused with the camera intrinsic matrix) is derived by:

$$K = P^- H^T (HP^- H^T + R)^{-1} \tag{11}$$

Here R is the measurement error covariance matrix and H is the Jacobian of the measurement function $\mathbf{h}()$ at point $\hat{\mathbf{s}}$.

With n LEDs detected R is a $2n \times 2n$ matrix. We model the errors in the pixel coordinates as independent zero-mean normally-distributed random variables. The standard deviation σ is set to 1 pixel for all our experiments but one. In the experiment with varying pixel noise we vary σ in the range from $\frac{1}{4}$ pixel to 4 pixels. R is set to $\sigma^2 I$ (see Section VII).

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.

$$\begin{aligned} \hat{\mathbf{s}}(t) &= \hat{\mathbf{s}}^- + K\Delta\mathbf{z} \\ P(t) &= (I - KH)P^- \end{aligned} \tag{12}$$

VI. Hardware platform and Experimental set-up

We evaluate the EKF algorithm presented in Section V-B using multiple sets of data gathered from small-scale experiments. In this section, we give a brief overview of the camera and sensing platform that we have used for our work, and subsequently, we explain our experimental set-up used for performing data collection.

A. Camera and WSN 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, 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. We use the MyriaNodes with their standard gMAC (TDMA-based) MAC-layer. This MAC layer does not allow the network to be triggered by an external resource. This is the main reason why we let the WSN trigger the network.

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. For our evaluation this set of sightings is considered a single measurement. By randomly selecting a subset of these eight LEDs per pose we can evaluate the performance of our algorithms as function of the number of LEDs simultaneously seen by the camera. This is reported in Section VII.

The experimental set-up for the circular test is shown in Figure 3. It shows the rotating plate of a turn table (the turn table itself is not depicted). On top of this plate is a square

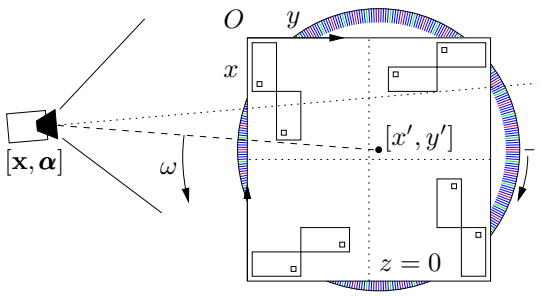

Fig. 3: Experimental set-up used for data collection.

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 dimensions of this set-up is given in the next subsection.

The turn table is rotated manually, 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.

C. Ground Truth

To allow for a quantitative evaluation of our algorithms we require precise ground truth – the exact location and orientation (in the order of 1 mm and 0.1 degree, respectively) – of the camera. Using other vision-based systems such as reacTIVision [2] is possible, however the error would be in the similar order of magnitude as our system.

Based on the knowledge we have about our experiment and the reference algorithm (RefAl) our approach is to use the set-up itself as the ground truth. Our approach relies on a number of assumption: (i) the rotating plate of the turn table rotates in a perfect plane (ii) manual rotation of the plate can be done accurate enough with zero mean random error (iii) the angular velocity of the plate (or the camera around the plate) is known (iv) the configuration of the LEDs in known accurately (v) systematic errors originate from the initial camera pose and centre of rotation

The angular velocity in known by construction. When we turn the table from one rotation to the next, we assign logical time stamps to the measurements we take. We take the difference of time stamps, δt, between two consecutive rotations constant. This models a rotating camera with constant angular velocity. We defined δt for the experiment to be $1/90s$. This basically models the angle of the camera as function of time, $\alpha(t) = \alpha_0 + \omega t$.

The LED configuration is set up carefully. For this work, we used a mechanical fixture (Lego$^{\text{TM}}$ bricks and plates) with known and accurate dimensions. This fixture holds the eight nodes in a planar ($Z = 0$) configuration. First each node is packaged in an identical housing build from Lego bricks, then these housings are fixed on a ground plate.

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

parameter	value
number of LED markers EKF	1
number of LED markers RefAl	4
standard deviation of pixel noise	1 pixel
frame rate	90 fps (frames per second)
rotation speed of camera	$\pi/2$ rad/s

The systematic errors are assumed from the initial camera pose and the actual centre of rotation. We use the static algorithm to eliminate these errors. The ground truth generator is parameterised in eight parameters; six parameters that define the initial camera pose and two parameters $[x', y']$ that define the actual centre of rotation of the camera. We use the Levenberg-Marquardt method to find the eight ground truth parameters that minimises an error vector $\mathbf{e}_{gt}$. The vector $\mathbf{e}_{gt}$ is a 1080-element vector that contains all the errors in the x, y, and z directions for all the 360 camera poses.

The ground truth values for our experiment are: $[x, y, z] = [110.3, -230.3, 161.3]$ mm, $[\theta, \phi, \psi] = [0.82, 1.44, -101.79]$ degree, and $[x', y'] = [126.1, 129.4]$ mm.

VII. Evaluation

In this section we evaluate our EKF'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.

To avoid convergence effects ending up in the plots for the EKF, the algorithm is run for two revolutions and only the data pertaining to the second revolution is used for the analysis. An exception is the experiment with the varying frame rates, there an integral number of revolutions is taken such that the second till the last revolution in total have at least 360 data points.

*1) **Effect of number of LEDs**:* Figure 4 shows the performance of EKF for different number of markers used. The performance of EKF is comparable to the results of RefAl when all the eight markers are used. As one might expect, the accuracy decreases with reduced number of markers. Contrary to the minimum amount of markers that is required to compute a pose using the RefAl, EKF is able to estimate the pose even when the measurements are under-constrained. This is because the EKF algorithm uses the predicted estimate to compute the pose of the camera. The dashed lines in the figure show how the RefAl performs for different number of markers. Similar to EKF's performance that of RefAl also degrades as the number of measurements are reduced.

*2) **Effect of camera frame rates**:* To quantify the effect of different camera frame rates we used synthetic datasets. Fig. 5 shows the performance of the algorithm for frame rates in the set $\{30, 90, 300, 900\}$ fps. As one would expect, the higher the camera frame rate, the better is the accuracy. This is because with increased frame rates the accuracy of the EKF prediction

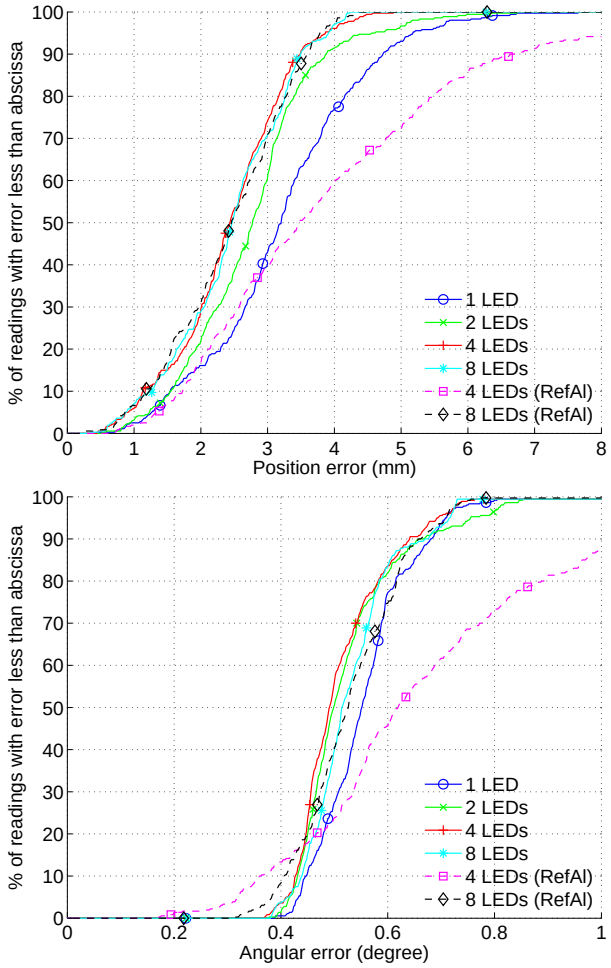

Fig. 4: EKF error distribution for different number of LEDs (experimental data).

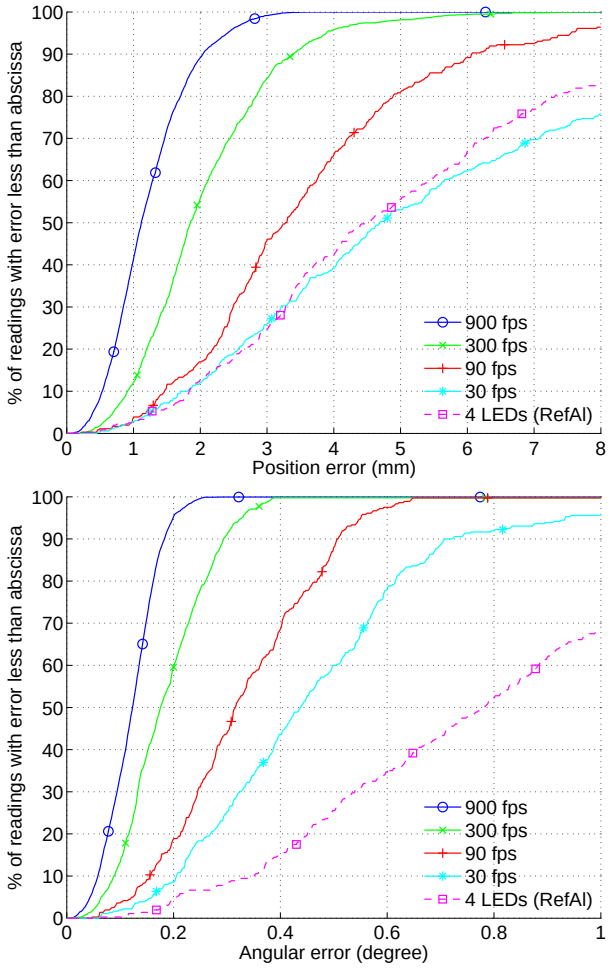

Fig. 5: EKF error distribution for different frame rates (simulated data, using one LED per measurement).

improves (linearisation of the model prediction is valid). The figure shows that for 30 fps EKF and RefAl perform equally well, and for higher frame rates EKF outperforms RefAl.

3) ***Effect of measurement noise****:* It is imperative to analyse the performance for varying degree of measurement noise. Measurement noise here refers to the difference between the detected location of the LED in the image plane and its true location in the image plane. We use simulated data to study the performance of EKF by adding noise (in terms of pixels) for the case when only one LED is observed per measurement. Position and angle errors for different noise levels are shown in Fig. 6. With increasing measurement noise, the performance of the EKF degrades. The figure also shows that for the same noise level (1 pixel) EKF performs better than RefAl.

We deliberately tested for one-LED case mainly for two reasons (i) it is more challenging to work when measurements are severely under-constrained and (ii) we want to study the potential of the SCAAT approach.

VIII. Discussion

Here we discuss some practicalities of using EKF for pose estimation.

a) ***Computational complexity****:* We evaluated the performance of EKF by comparing it with RefAl. Our analysis shows that both the algorithms perform comparable starting from 4 LEDs per frame.

We measured the execution times per pose estimate (in Matlab). EKF requires 1.6 ms (1 LED) and 1.7 ms (4 LEDs), while RefAl requires 27.2 ms (4 LEDs). As one may notice there is a clear computational advantage of the EKF-based approach over RefAl. This makes EKF an attractive option for implementation on resource-constrained platforms.

b) ***Tradeoff between accuracy and frame rates****:* We showed that EKF performs with fewer measurements (LEDs). This however, comes at a price of reduced accuracy. When the application demands more accuracy, either the number of measurements used must be increased or the underlying camera should provide higher frame rates to cater to the requirement. The option of favouring camera frame rates is expensive in terms of processing cost, hence it is better to use more measurements. This could be done for example, by incorporating more LEDs on each sensor node and make them all flash simultaneously. For instance, using four LEDs on a

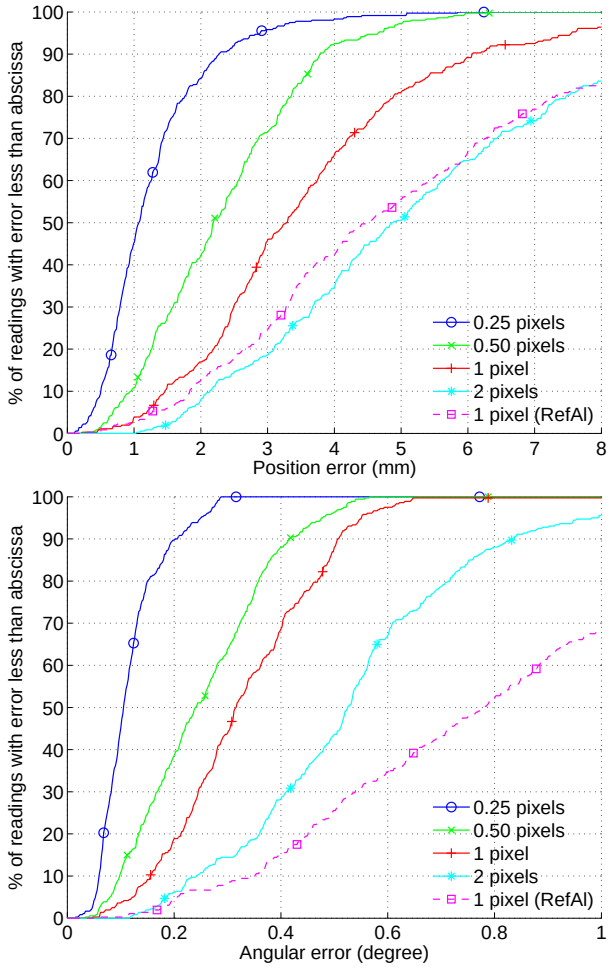

Fig. 6: EKF error distribution for different degrees of measurement noise (simulated data, using one LED per measurement).

sensor node will provide eight constraints and, hence, the pose can be estimated with greater accuracy.

*c) **Filter variations**:* We have used for our EKF implementation a constant-velocity and constant-angular velocity model. Knowing the exact motion of the camera in reality is very difficult. However, it could be circumvented by using other sensing modalities such as inertial sensors (as available on Wiimotes) to better predict the state. Typically, inertial sensors offer higher update rates (over 100 Hz), hence using multiple modalities can improve the overall update rate of the system. Alternatively, we could also adapt our approach by using a particle filter instead of the EKF. Although this is computationally intensive, it might result in better estimates as particle filters have the advantage of being able to model non-linear non-Gaussian systems.

IX. Conclusions and Future work

We have proposed an inside-out system that uses LED sightings gathered from wireless sensor nodes (WSN) to estimate the pose of the camera. We presented a pose estimation based on Extended Kalman filtering (EKF).

We evaluated the performance of the algorithm using small-scale experimental data. We showcased the effectiveness of EKF with simulated data for different camera frame rates, varying noise levels and under different LED visibility conditions. Our initial results show that our EKF algorithm gives an accuracy of about few millimetres and few degrees even in sparse conditions with just 1 LED per pose.

The EKF also has been compared with RefAl, an algorithm that is based on the Discrete Linear Transform (DLT). The EKF has shown to outperform RefAl, both in terms of accuracy and complexity. Contrary to EKF, the DLT-based algorithm needs more measurements and more iterations to estimate a pose.

In the future, we plan to analyse the practical performance of our algorithm by running a large-scale experiment using more sensor nodes and for different camera mobility models.

References

[1] R. T. Azuma. A Survey of Augmented Reality. *Presence*, 6:355–385, 1997.

[2] 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.

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

[4] Brown and Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. 1997.

[5] 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.

[6] Delfly. http://www.technovelgy.com/ct/Science-Fiction-News.asp?NewsNum=1780.

[7] 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.

[8] 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.

[9] Immaterials. http://www.nearfield.org/2009/10/immaterials-the-ghost-in-the-field.

[10] J.C.Lee. Hacking the Nintendo Wii Remote. *Pervasive Computing*, 7:39–45, 2008.

[11] 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.

[12] 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.

[13] K. Muthukrishnan. *Multimodal Localisation: Analysis, Algorithms and Experimental Evaluation*. PhD thesis, University of Twente, Enschede, September 2009.

[14] Nintendo. Consolidated Financial Highlights. www.nintendo.co.jp/ir/pdf/2008/080124e.pdf, Jan 2008.

[15] 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.

[16] G. Welch. *SCAAT: Incremental Tracking with Incomplete Information* . PhD thesis, UNC-Chapel Hill Chapel Hill, NC 27599-3175, 1996.

[17] 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.

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

[19] 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.