Practical Image-Enhanced LBS for AR Applications

Antonio J. Ruiz-Ruiz(✉), Pedro E. Lopez-de-Teruel, and Oscar Canovas

University of Murcia, Murcia, Spain
{antonioruiz,pedroe,ocanovas}@um.es

Abstract. We have designed a multisensor indoor LBS suitable for augmented reality applications which, mainly based on computer vision techniques, provides precise estimations of both the 3D position and rotation of the device. Our proposal makes use of state-of-the-art IMU data processing techniques during the training phase in order to reliably generate a 3D model of the targeted environment, thus solving typical scalability issues related to visually repetitive structures in large indoor scenarios. A very efficient camera resection technique will then be used in the on-line phase, able to provide accurate 6 degrees of freedom estimations of the device position, with mean errors in the order of 5 cm and response times below 250 ms.

Keywords: IMU · SIFT · Computer vision · LBS · Augmented reality

1 Introduction

Nowadays smartphones are equipped with many types of sensors, which is simplifying the process of obtaining context information. Particularly, it is now possible to obtain accurate location estimations using visual information provided by the device camera. This is especially useful in augmented reality (AR) applications, which seamlessly integrate the sensor and the display. However, image processing is a time consuming task, which usually imposes the use of additional sensors to limit the amount of required computation. The accuracy and performance of a location based service (LBS) is closely related not only to the number of sensors being used, but also to the quality of the context model they are based on. For the last few years we have been publishing different approaches to address the challenge of indoor precise localisation, using all the available sensors in order to improve performance, accuracy and ease of deployment.

Our initial solution was a classical fingerprinting map based on 802.11 RSSIs which also used SIFT (Scale Invariant Feature Transform) descriptors from images obtained at different training points [18]. We found correspondences between the images being acquired during the on-line phase and those obtained during the training phase. Instead of examining the whole database of images we constrained the space search to those acquired at places where the RSSIs signals

I. Stojmenovic et al. (Eds.): MOBIQUITOUS 2013, LNICST 131, pp. 460–473, 2014.
DOI: 10.1007/978-3-319-11569-6_36

were similar to the current ones. This solution was limited by the granularity of our sampling procedure, since we just returned the location estimation which provided more matchings with the current image.

We later extended the system to provide precise estimations of the position and rotation of the device with respect to some conveniently chosen world coordinate system [19]. Using visual Structure from Motion techniques (SfM) [10], we built off-line 3D reconstructions of the scenario from the correspondences among SIFT descriptors of training images extracted from recorded video sequences of the environment. The generation of those 3D models had to be supervised by a human operator to deal with repetitive structures (windows, doors, etc.) typically found in indoor scenarios, and which could affect the automatic modelling process, generating arbitrary layouts not matching the real scenario. Model accuracy was crucial for our posterior location estimation, based on a resection process and using the images captured by the devices during the on-line phase.

In this paper we present several key improvements in relation to these and other authors' works. First, we introduce the use of IMU (Inertial Measurement Units) to generate accurate 3D models in a mostly automatic and unsupervised manner. An operator carrying a tablet or smartphone walks around the indoor environment taking time indexed pictures that will be used to build the 3D model. Once the walk is finished, our training application uses the inertial sensor measurements to infer rough locations of the places from where each image was taken. The only input which is required is an approximate vectorial map of the scenario. Images are then automatically geo-tagged using the inferred pathway, and this approximate spatial information will be paramount to rule out comparisons between images taken from very different angles or distant positions, thus avoiding the aforementioned model generation failures due to visually similar structures.

Second, we have noticeably improved the resection process to determine the device pose. The new technique demands a fewer number of correct matchings to perform the camera resection, which speeds up the procedure and provides response times closer to the requirements imposed by AR applications. Moreover, since our 3D model is divided into different zones according to the characterisation of the 802.11 signals, we perform our matching process only against the 3D submodel where the RSSI signals from the surrounding antennas are similar to the current RSSI measurements. That is, RSSIs act as a filter for the involved image processing.

Third, we have performed an experimental analysis using different indoor locations and testing several configuration parameters to validate our proposal. We will show 3D models of large real indoor scenarios containing repetitive structures that were built using our image geo-tagging technique. We will also analyse the accuracy of our location estimations during the on-line phase using the new camera resectioning procedure. Finally, we will make use of an AR prototype in order to validate the location estimations and to demonstrate that our proposal is suitable for real-time applications.

2 Related Work

Many types of data and methods have been used to infer location. Though there are important contributions making use of a single sensor, better results can be obtained by integrating the information captured by multiple sensors. For example, [11,14] use WiFi signals and image recognition to estimate positions. However, these works are based on two-dimensional visual landmarks that must be placed in the scenario of interest, which involves the inclusion of obtrusive elements. One of our main objectives is to avoid special tagging or ad-hoc landmarks. In [1], a preliminary analysis of how image processing techniques like SIFT can be used to improve the accuracy in landmark-free LBSs is performed. SIFT features [13] provide a powerful mechanism to robustly match different images of a given scene, which is fundamental not only for visual recognition of objects or places [6], but also in the context of simultaneous localisation and three-dimensional scene reconstruction [20]. Alternatively, some other works on localisation have used other kind of features and associated local image descriptors instead of SIFT, such as SURF [15] or ASIFT [12].

On the other hand, there are also some works that try to track a device from a known initial position using dead-reckoning [2,5], where inertial sensor measurements are integrated over time. However, when dealing with noisy sensors, even small sensing errors are magnified by integration. This drawback can be addressed using map constraints and particle filtering [22]. In a recent work [17], authors present a proposal which is able to track users without any a priori knowledge about the user's initial location, stride-length or device placement. Though their paper is mainly focused on reducing the calibration effort required to build WiFi fingerprinting maps by means of crowdsourced data, some of their ideas have been inherited in our proposal. We will also make use of a particle filter augmented to estimate unknowns such as the stride length or the orientation. However, we will follow a different approach in order to estimate the orientation and the step detection. We start from a more constrained situation, since our operator will always behave in a known manner, and therefore we will be able to make some assumptions about the device placement and the human locomotion.

3 Building Accurate 3D Models Relying on Inertial Sensors

It is a well known fact that the accelerometers usually integrated in mobile devices are relatively noisy sensors. The great magnitude value of the always present gravity ($9.8\,\mathrm{m/s^2}$) with respect to interesting accelerations induced by user movements is indeed a problematic issue. Though, at least theoretically, the distance travelled could be calculated by integrating acceleration twice with respect to time, error accumulates rapidly after some seconds due to the presence of noise in readings.

To avoid this type of drifting we adopt the individual step count as the metric of walking distance, taking advantage of the fact that, when a person walks, their

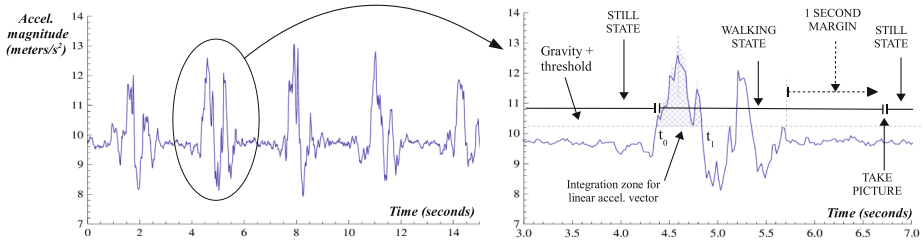

Fig. 1. Acceleration magnitude analysis.

feet periodically experiment a stationary stance phase of approximately 0.5 s in every step [5]. An intelligent integration scheme, that applies *zero velocity updates* (ZUPT) when detecting this stance phase, together with a carefully designed *Extended Kalman Filter* [21], is able to accurately estimate the position of a pedestrian for relatively long walks. However, this approach requires (*i*) better inertial sensors than those commonly found in typical mobile devices, and (*ii*) the sensor has to be mounted in the shoe of the operator, in order to take advantage of the ZUPT trick. In [17] the authors presented a proposal making use of the information from the compass, a particle filter, and an approximate vectorial map of the environment to correct the generated trajectories. But, once again, a basic requirement in that work is that users have to put their phones in the pocket or somewhere else where the periodic movement pattern is clearly noticeable.

Although none of these techniques are directly applicable in our case, since the operator must hold the tablet in their hands to take pictures of the environment, we somehow partially combine these two approaches in order to adapt them to our needs. Our training application suggests the kind of movements that the operator must perform while taking pictures of the environment in a way that (*i*) makes it easy to predict them, (*ii*) ensures that pictures are taken in the best imaging conditions, and (*iii*) simplifies the operator's job. Our application expects a movement pattern in which the operator stands still, takes a picture of the environment, moves to a nearby position (typically one meter away or so), and the process is repeated. These relative movements, though certainly noisy and suffering a considerable drift when integrated in the long term (see Fig. 2(a) for an example), will be the basis for a posterior particle filter that will fit these measurements to our scenario map, in a similar way to that performed in [17].

We analyse the magnitude $|\overrightarrow{a}|$ of the acceleration vector $\overrightarrow{a} = (a_x, a_y, a_z)$ in time, sampled at 50 Hz by the device and considering two main states of the operator: still and walking (see Fig. 1). The logic for changing from one to the other is: if the accelerometer signal exceeds the gravity $|\overrightarrow{g}| = 9.8\,\mathrm{m/s^2}$ by a predefined small threshold, we consider that the operator is walking. As there may be short periods of time where $|\overrightarrow{a}|$ falls below that threshold during movement, the application only gets back to the still state as long as $|\overrightarrow{a}|$ keeps under the threshold value for a period of at least one second. That instant the

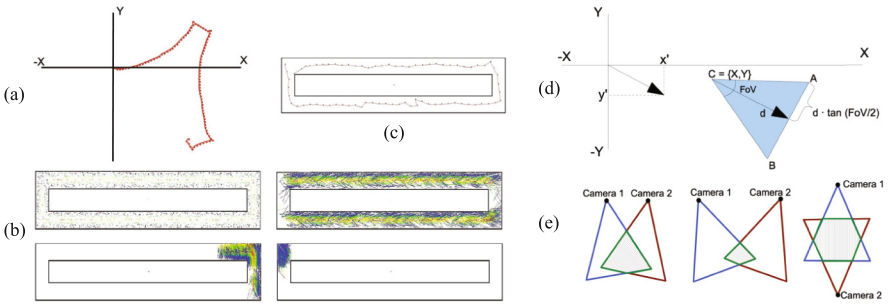

Fig. 2. (a) Raw input as obtained by the motion model. (b) Evolution of particles population as the operator moves, converging to a mostly monomodal state. (c) Corrected path estimation. (d) Approximate field of view covered by a camera. (e) Different examples of overlapping areas.

application automatically takes a picture, what helps in taking clear and sharp images since the mobile device is fairly still when the picture is taken.

Now we have to estimate the relative movement performed by the operator between two such instants. Double integration of linear acceleration to estimate displacement suffers from serious drifting issues, but the direction of movement is still acceptably estimated if the integration interval is short enough. Thus, we integrate only from the start of movement until the magnitude $\overrightarrow{a}$ falls under the threshold for the first time (instants t_0 and t_1 in Fig. 1). The output of this stage will be a unit norm direction vector $(\Delta x, \Delta y, \Delta z)$, leaving the stride length estimation to the subsequent particle filter:

$$\overrightarrow{\Delta x_{dev}} = (\Delta x, \Delta y, \Delta z) = \frac{\int_{t_0}^{t_1}(\overrightarrow{a} - \overrightarrow{g})dt}{|\int_{t_0}^{t_1}(\overrightarrow{a} - \overrightarrow{g})dt|} \tag{1}$$

However, since this vector refers to device coordinates, it still has to be corrected using an absolute rotation with respect to the world coordinates. Many mobile devices do in fact integrate compass, gravity and gyroscope readings to get a virtual sensor that provides a filtered output rotation $\mathtt{R}_{dev \to wrl}$. Thus, our final estimation of the relative direction vector in world coordinates will be $\overrightarrow{\Delta x_{wrl}} = \mathtt{R}_{dev \to wrl} \cdot \overrightarrow{\Delta x_{dev}}$.

To track the operator's movement we implemented a particle filter [21], which uses spatial restrictions of the a priori known indoor map to filter out long term drifting errors in the raw input trajectory. Figure 2(a–c) show the overall functioning of the particle filtering algorithm working in one of our scenarios (see also Sect. 5). A classical backward propagation indicates the final estimated path (Fig. 2(c)). It is worth noting that in the vectorial map modelling the 220 m^2 faculty hall we included a virtual rectangle in the center which, though non-existent in practice, models a zone avoided by the operator in their walks, and

helps to get a faster convergence to a mostly monomodal state (Fig. 2(b)) while keeping a low sample size.

One of the most usual problems we found during the SfM process, which partly motivated our approach, is the existence of repetitive visual structures in different parts of the scenario (i.e. doors, windows, benches, etc.). That implies incorrect matches that, though being visually correct, are incorrect from a geographic point of view, and thus induce invalid 3D models. Our main goal in the training phase is to achieve an error-free 3D model, since it is essential for the accuracy of the posterior on-line resection process. Using the previously obtained rough estimations of the position and orientation of the training images to guide the 3D model reconstruction, we increase the robustness of the SfM process eliminating the need of a careful manual supervision which we had to perform in previous works [19] in order to fix false positive correspondences.

More precisely, the method we use to build the 3D model in a more unsupervised but reliable way is based on the generation of a pairwise candidate matching list. We determine which pairs of images must be compared to look for coincidences between them taking into account the position and orientation from which they were taken. As shown in Fig. 2(e), two images will be tried to match only if (i) the areas of the scenario that they cover are fairly well overlapped, and (ii) they have been taken with a difference in this angle below 50° (otherwise it would degrade the repeatability of features, as indicated in the original paper describing SIFT [13]). Additionally, greater angles differences between the optical axis of the cameras (180° in the last example) makes it impossible to expect finding good visual matching of the corresponding SIFT features.

To get the pairwise candidate matching list we first estimate the area covered by each camera. From the results obtained after executing the particle filter over the sensors data we were able to approximate the 2D location in the scenario $C = (X_i, Y_i)$ at which each image was taken during the training. We also obtained the rotation matrix $\mathtt{R}$ that gave us the approximate orientation of the smartphone at that moment. Moreover, considering that the typical camera pose when capturing the images keeps the imaging plane approximately vertical, the unit norm vector $(x', y') = (v_2, v_3)/|(v_2, v_3)|$, with $v = \mathtt{R}^\top \cdot (0, 0, -1)\top$, gives us the approximate orientation of the optical axis projected on the floor. As detailed in Fig. 2(d), it is then possible to calculate the area covered by a camera estimating the points $A, B = (X, Y) + d \times ((x', y') \pm tan(FoV/2) \times (y', -x'))$, being d the maximum distance threshold beyond which we discard the obtained SIFT image features[1] and FoV the horizontal field of view of our camera. Finally, using the triangles generated for every camera position, we can compute their corresponding pairwise overlapping areas. These will provide us the list of matching candidate image pairs that will be used as input to the SfM process.

[1] Values for d in typical indoor scenarios are of a few meters (9 m in this work).

4 Camera Resection

Having described our improvements in the offline construction of the 3D model, in this section we focus on the optimization of the online phase, i.e., the estimation of the position of the device from a given set of matching 3D model points and 2D pixels in an input image. The *projection matrix* $\mathtt{P}_{3\times4}$ defines the algebraic relationship $\mathtt{P}\mathbf{X}_i = \mathbf{x}_i$ for every $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ correspondence when both the pixels $\mathbf{x}_i = (x_i, y_i, 1)^\top$ and the 3D points $\mathbf{X}_i = (X_i, Y_i, Z_i, 1)^\top$ are given in homogeneous coordinates [10]. Matrix $\mathtt{P}$ can be factorized as $\mathtt{K}_{3\times3}\mathtt{R}_{3\times3}[\mathtt{I}|-C^\top]_{3\times4}$, where the relationship with the sought rotation $\mathtt{R}$ and 3D position C of the device in the 3D reference system is made explicit. This factorization depends also on $\mathtt{K}$, the so-called camera *calibration matrix*. We focus on the simplest *precalibrated* case, in which the camera focal in pixels f is known in advance, $\mathtt{K} = diag(f, f, 1)$, and $\mathbf{x}_i = \mathtt{K}^{-1}\mathbf{x}_i^p$ is the way to obtain the working coordinates $\mathbf{x}_i$ from the original, image centered pixel coordinates $\mathbf{x}_i^p$.

Camera resectioning consists on the estimation of $\mathtt{R}$ and C from potential matches $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$. However, the SIFT matching process is always contaminated with a variable proportion ρ of *outliers*, i.e. wrong correspondences [13]. To filter them out, the robust RANSAC algorithm [4] selects random minimal subsets of the original correspondences set, from which it computes tentative locations of the device. In previous works [19] we used two classical approaches, namely the DLT algorithm [10] for the uncalibrated case (i.e., unknown $\mathtt{K}$), and the Fiore algorithm [3] for the precalibrated case. These algorithms had the advantage of being based on simple linear algebra procedures, but unfortunately needed minimal subsets of 6 and 5 matchings, respectively. Since the RANSAC algorithm works iteratively by finding solutions from tentative random subsets of these minimal sizes until it eventually finds a subset free of outliers, the probability of finding a good subset diminishes exponentially with the size of such subsets. Thus, in this paper we propose to change the resection procedure by an *algebraic minimal solution* which only needs 3 correspondences. Though the algorithm gets slightly more complicated, as it is not linear any more, the advantages are far greater, because the theoretical probability of finding earlier a good minimal solution gets boosted, as we will also confirm experimentally in the next section. The general operation of the *P3P algorithm* (*pose from 3 points*) is summarized in Algorithm 1, though to implement the somewhat involved mathematical computations the reader is referred to [4].

Note that P3P first finds the depths (d_a, d_b, d_c) of the $\mathbf{X}_{i=a,b,c}$ 3D points, i.e., their euclidean distance to camera center C, by solving a 4^{th} degree polynomial. Thus, in fact it can produce up to 4 valid distinct solutions $(d_a(s_l), d_b(s_l), d_c(s_l))$, $l = 1 \ldots 4$. Once depths have been obtained, the estimation of the position C and rotation $\mathtt{R}$ of the device reduces to find the correct alignment of two triplets of 3D points, i.e. $(X_i, Y_i, Z_i) \leftrightarrow d_i \frac{(x_i, y_i, 1)}{\|(x_i, y_i, 1)\|_2}$ for $i = a, b, c$. If we first center both triplets on their respective means and then rescale them to a common average size, we are left with the simpler problem of estimating the unknown rotation between the two transformed sets of points. This is the well known *orthogonal Procrustes* problem, and the sought rotation is given by $\mathtt{R} = \mathtt{V} \cdot$

Algorithm 1. P3P algorithm

Require: Three 2D pixel $\leftrightarrow$ 3D model coordinates correspondences:
$\{(x_a, y_a), (x_b, y_b), (x_c, y_c)\} \leftrightarrow \{(X_a, Y_a, Z_a), (X_b, Y_b, Z_b), (X_c, Y_c, Z_c)\}$

(*) Compute five G_k ($k = 0 \ldots 4$) coefficients from (x_i, y_i) and (X_i, Y_i, Z_i), $i = a, b, c$. Solve the resulting 4^{th} degree polynomial in variable s, $\sum_{k=0}^{4} G_k s^k = 0$ (up to 4 solutions).

for all s_l, $l = 1 \ldots 4$, solution of polynomial **do** (One possible resection for each s_l):

(*) Obtain distances of 3D model points to camera center (depths $d_a(s_l), d_b(s_l), d_c(s_l)$).

(**) Obtain $\mathtt{R}^j$ and C^j from depths d_a, d_b, d_c.

end for

Ensure: Up to 4 different solutions for absolute orientation $\{\mathtt{R}^l, C^l\}$ of the device, $l = 1 \ldots 4$.

(*) These steps involve a fair ammount of nonlinear computations, see [4] for details.

(**) This step is performed using the *Procrustes algorithm* (explained in main text).

$diag(1, 1, \mathrm{D}et(\mathtt{UV}^\top)) \cdot \mathtt{U}^\top$, where $\mathtt{U}_{3\times3}\mathtt{D}_{3\times3}\mathtt{V}_{3\times3}{}^\top$ is the Singular Value Decomposition (SVD) [8] of the matrix $\mathtt{YW}^\top$, being $\mathtt{W}_{3\times3}$ and $\mathtt{Y}_{3\times3}$ the matrices formed by stacking the corresponding transformed triplets of 3D points [9]. Once we have calculated the rotation matrix $\mathtt{R}$, it is very easy to obtain the translation vector C by undoing the previously performed centering and scaling of both set of 3D points.

Tentative solutions produced by successive executions of the P3P algorithm are then tested against the whole set of matches, measuring the *reprojection errors* ξ_i of $\mathtt{P}\mathbf{X}_i$ with respect to the corresponding $\mathbf{x}_i$. A given match $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ is considered an inlier if such ξ_i is below a given threshold ϵ. Once the number of inliers gets above the expected proportion $1-\rho$, the RANSAC iteration stops, as we have found a valid solution confirmed by a sufficient number of matchings. If, on the contrary, the number of P3P attempts exceeds a given maximum without having found a valid solution, the resection is considered unsuccessful, as it can't return a reliable device position.

Once the set of inliers has been identified, the final pose is obtained by refining the P3P solution using all the available valid correspondences. Starting from the initial values of $\mathtt{R}$ and C, we use the nonlinear Gauss-Newton optimization algorithm [16] to iteratively minimize the sum of all *alignment errors* between the unit vectors $\frac{(x_i, y_i, 1)}{\|(x_i, y_i, 1)\|_2}$ and the corresponding $\frac{X_i - C}{\|X_i - C\|_2}$ for all valid correspondences i. This minimization is based on solving a linear equation whose coefficient matrix depends on the derivatives of the individual components of this cost with respect to the six parameters (α, β, γ) of $\mathtt{R}$ and (c_x, c_y, c_z) of C. A thorough mathematical derivation of this procedure falls out of the scope of this paper, but the resulting algorithm is included in Algorithm 2 for completeness (again, the reader is referred to [16] for further details on the Gauss-Newton

technique). We can optionally make this procedure incremental by adding new correspondences according to the current $\{\tilde{\mathtt{R}}, \tilde{C}\}$ until no more inliers are found.

Algorithm 2. Nonlinear Gauss-Newton refinement of $\mathtt{R}$ and C

Require: $\mathtt{R}$, C and inlier set $\{\mathbf{X}_i = (X_i, Y_i, Z_i, 1) \leftrightarrow \mathbf{x}_i = (x_i, y_i, 1)\}_{i=1}^{n}$ estimated by P3P.

Initial solution: Note that $\hat{\mathbf{x}}_i$ always contain input $\mathbf{x_i}$ rectified by current rotation $\tilde{\mathtt{R}}$, $\forall i$:

$\tilde{C} = (\tilde{c}_x, \tilde{c}_y, \tilde{c}_z) \leftarrow C = (c_x, c_y, c_z)$; $\quad \mathtt{R} \leftarrow \tilde{\mathtt{R}}$; $\quad \hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i, 1) \leftarrow \mathtt{R}^\top \mathbf{x}_i$; $\quad \forall i = 1..n$;

while not convergence of $\{\tilde{\mathtt{R}}, \tilde{C}\}$ **and** $n_{iters} < max_{iters}$ **do**

Jacobian computation: matrices J_i of partial derivatives of alignment error costs with respect to rotation and center increments $(\Delta\alpha, \Delta\beta, \Delta\gamma, \Delta c_x, \Delta c_y, \Delta c_z)$:

$\delta X_i \leftarrow X_i - \tilde{c}_x$; $\quad \delta Y_i \leftarrow Y_i - \tilde{c}_y$; $\quad \delta Z_i \leftarrow Z_i - \tilde{c}_z$; $\quad \forall i = 1..n$

$l_i \leftarrow \sqrt{\delta X_i^2 + \delta Y_i^2 + \delta Z_i^2}$; $\quad l'_i \leftarrow \sqrt{\hat{x}_i^2 + \hat{y}_i^2 + 1}$; $\quad \forall i = 1..n$

$$J_i \leftarrow \begin{bmatrix} \frac{\delta Y_i}{l_i} & \frac{\delta Z_i}{l_i} & 0 & \frac{-(\delta Y_i)^2-(\delta Z_i)^2}{l_i^3} & \frac{(\delta X_i)(\delta Y_i)}{l_i^3} & \frac{(\delta X_i)(\delta Z_i)}{l_i^3} \\ \frac{-\delta X_i}{l_i} & 0 & \frac{\delta Z_i}{l_i} & \frac{(\delta X_i)(\delta Y_i)}{l_i^3} & \frac{-(\delta X_i)^2-(\delta Z_i)^2}{l_i^3} & \frac{(\delta Y_i)(\delta Z_i)}{l_i^3} \\ 0 & \frac{-\delta X_i}{l_i} & \frac{-\delta Y_i}{l_i} & \frac{(\delta X_i)(\delta Z_i)}{l_i^3} & \frac{(\delta Y_i)(\delta Z_i)}{l_i^3} & \frac{-(\delta X_i)^2-(\delta Y_i)^2}{l_i^3} \end{bmatrix} \quad \forall i = 1..n$$

Current residuals: alignment errors for current solution:

$r_i \leftarrow (\delta X_i/l_i - \hat{x}_i/l'_i, \delta Y_i/l_i - \hat{y}_i/l'_i, \delta Z_i/l_i - 1/l'_i)^\top$;

Solve Gauss-Newton step linear equation system: using all J_i jacobians and r_i residuals:

Stack J_i matrices and r_i vectors $\forall i$ to form matrix $J_{3n\times 6}$ and vector $r_{3n\times 1}$.

$(\Delta\alpha, \Delta\beta, \Delta\gamma, \Delta c_x, \Delta c_y, \Delta c_z) \leftarrow (J^\top J)^{-1} J^\top \cdot r$;

Update solution: using obtained solution for rotation and center increments:

$$\tilde{\mathtt{R}} \leftarrow \tilde{\mathtt{R}} \begin{bmatrix} \cos(\Delta\alpha) & \sin(\Delta\alpha) & 0 \\ -\sin(\Delta\alpha) & \cos(\Delta\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\Delta\beta) & 0 & \sin(\Delta\beta) \\ 0 & 1 & 0 \\ -\sin(\Delta\beta) & 0 & \cos(\Delta\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\Delta\gamma) & \sin(\Delta\gamma) \\ 0 & -\sin(\Delta\gamma) & \cos(\Delta\gamma) \end{bmatrix}$$

$\tilde{C} \leftarrow \tilde{C} + (\Delta c_x, \Delta c_y, \Delta c_z)$; $\quad \hat{\mathbf{x}}_i = (\hat{x}_i, \hat{y}_i, 1) \leftarrow \tilde{\mathtt{R}}^\top \mathbf{x}_i, \forall i = 1..n$;

end while

Ensure: Final polished solution $\{\tilde{\mathtt{R}}, \tilde{C}\}$

5 Experimental Analysis

Experimental Environment: We conducted all our experiments in the ground floors of the Computer Science (scenario 1) and Veterinary Science (scenario 2) faculties in our campus, covering two areas of 220 m^2 and 445 m^2 respectively (Fig. 3(a–b)). During the training phase we used a custom sensor capture application running on a Samsung Galaxy Tab+ 7.0, to acquire all the relevant sensor data (WiFi, images, accelerometer, gravity, gyro and compass). RSSIs measurements were used to build a fingerprinting map, while images were postprocessed on an Intel Core i7 3.4 GHz server equipped with a Nvidia GeForce GTX580 GPU, using the SiftGPU library [23] to extract the SIFT features, as well as the VisualSfM software [24] in order to get the full 3D reconstructions from the matching SIFTs. The IMU and compass data were used as described in Sect. 3 in order to automatically guide these reconstructions. The same server acted also as the cloud in the posterior on-line phase, with the client mobile device sending

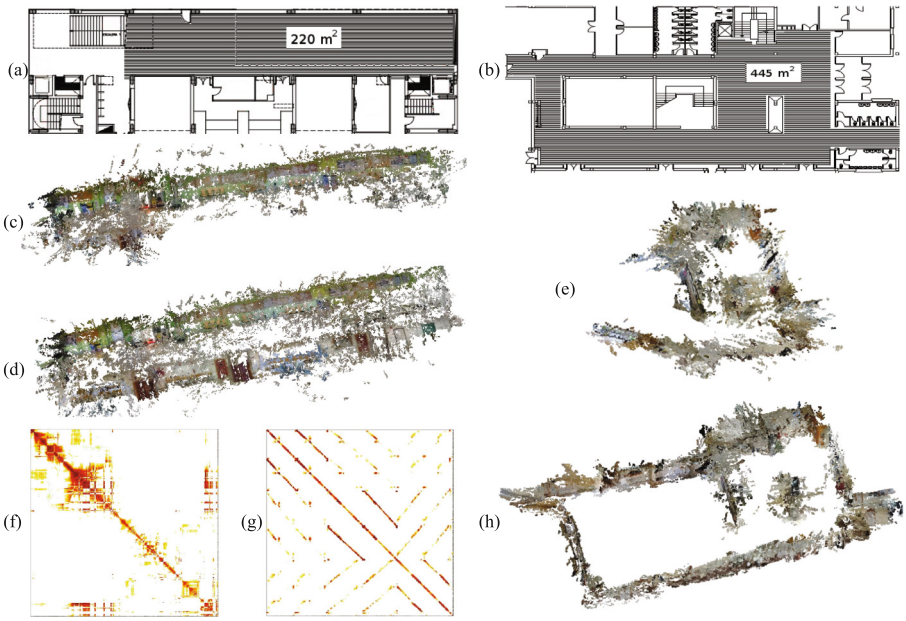

Fig. 3. (a–b) Ground floor plans of scenarios 1 and (2. c–e,h) 3D models obtained using both unguided and guided matching. (f–g) Unguided and guided matching matrices for scenario 1.

all its sensor readings to it in order to get its precise localisation. SiftGPU was used again for feature extraction in the client input image, while matching with features in the model was performed using an adapted version of the CUDA ENN (*Exhaustive Nearest Neighbour*) software [7]. It is key in our approach that this exhaustive search is performed only on a small subset of the whole set of features of the 3D model, using an initial coarse-grained localisation based on the RSSI and the available WiFi fingerprinting map [19]. Finally, all stages of the resection algorithm described in Sect. 4 were implemented using the QVision library [25].

3D Model Generation: Figs. 3(c,e) show two sample final results of the SfM process when the matching is not guided. It is clear that false positive matches caused by visually repetitive structures produce models that do not fit well with the corresponding scenarios. In previous versions of our system, these erroneous matchings –extremely frequent in indoor scenarios– had to be removed interactively by an operator in order to get fully correct reconstructions, in a cumbersome an tedious manual and incremental process. In contrast, when images were coarsely geotagged using the technique described in Sect. 3, we easily avoided those situations, since distant cameras were early discarded in the pairwise matching. This resulted in a smaller but also much more reliable set of matches. To illustrate this, Figs. 3(f–g) show two different *matching matrices* for

the 510 images taken from our first scenario. The first, more dense matrix, corresponds to a total of 260 K initial tentative matchings when all possible images pairs are considered. The second one, which uses the coarse location obtained by our particle filter, greatly reduces the candidate image pairs, resulting in a much more sparse matrix containing only 30 K initial matchings. This not only involves a great reduction in the computing time of the 3D model, but, what is more important, it also remarkably improves the reliability of the finally obtained 3D model. In spite of the much lesser number of initial correspondences, the number of finally correct reconstructed 3D points is clearly larger in the automatically guided reconstruction (Fig. 3(d), 150 K 3D features) than in the unguided case (Fig. 3(c), 86 K 3D features). Similar results were consistently obtained for scenario 2 (Fig. 3(h) vs. 3(e)) and different executions –which is also relevant since the SfM process involves a good amount of randomness in the incremental reconstructions–. Finally, we also evaluated *a posteriori* the accuracy of the initial coarse estimations performed by the particle filter using the final refined estimation performed by VisualSfM, resulting in a mean error of 206 ± 113 cm for the whole set of input pictures. As discussed above, this is more than enough to correctly guide the matchings, leaving the refinement of this initial rough solution to the much more precise visually based SfM process.

Camera Resection: In order to perform an in-depth evaluation of the new camera resection technique presented in this paper, we have exhaustively tested the different configuration parameters mentioned in Sect. 4, in order to estimate their influence on the results. These parameters were (i) the reprojection error threshold ϵ that decides when a 2D↔3D correspondence is considered a tentative inlier, (ii) the minimum ratio of required inliers $1 - \rho$ over the total amount of correspondences used to validate a RANSAC triplet, and (iii) the mean residual $\sum_i \xi_i / N$ value obtained after resection that indicates the error in alignment for the final number N of inlier correspondences, which is a metric of the goodness of the obtained solution. Figures 4(a–b) illustrate the estimation error in *cm* that we have obtained using each set of parameters and running the process 50 times over a set of 30 640×480 images taken from different viewpoints in scenario 1. From these results we see that it is possible to obtain an average accuracy[2] down to 5 cm with a percentage of success of 97 % using the configuration $\epsilon = 0.01$, $1 - \rho = 0.3$, and $max(\sum_i \xi_i / N) = 0.002$. If we compare these results with those obtained in our previous work [19], where we obtained a mean error of 14.6 cm, we can confirm that we have notably improved the localisation accuracy. Furthermore, it is worth mentioning the reduction of the resection time, taking now less than 10 ms instead of 119 ms it took using the previous system (Fig. 4(c)). That time reduction allows to complete an entire cycle (from sensor data acquisition to 3D position estimation) in less than 250 ms, making the proposal ideal for applications requiring high accuracy and rapid response, like AR-based applications.

[2] The ground-truth considered to get these numbers was obtained using the 3D model of the scene, adding the test images and reestimating their reconstruction coordinates within it.

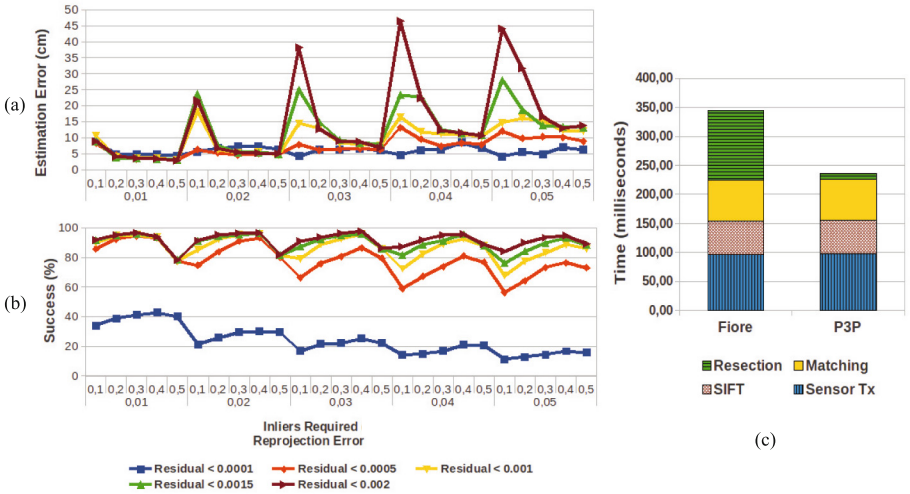

Fig. 4. (a–b) Resection error results and percentage of success after 1250 executions for each parameter set. (c) Performance comparison with previous proposal.

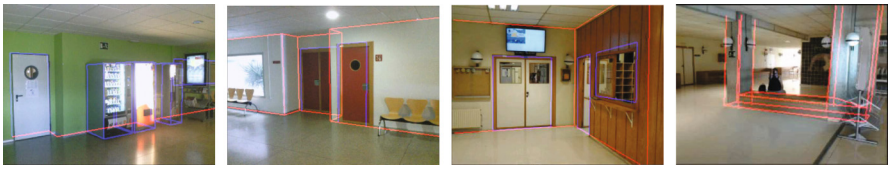

Fig. 5. AR results: (a–b) Scenario 1. (c–d) Scenario 2.

Augmented Reality Application: However, there is not best way to test our proposal than using it for AR purposes within a real scenario. We have developed an AR application for Android devices and have tested it in our two modelled scenarios. In Fig. 5 we can see how the AR perfectly matches to different sample input images, demonstrating the high accuracy reached by our resection process. We are aware that, though the response time of our resection service by the cloud is typically limited to less than 250 ms, this refreshment rate is still insufficient to fully cope with video input rates of mobile devices (25 fps = 40 ms in typical cameras). This problem can be alleviated using the gyroscope sensor to continuously detect small rotation changes of the device between successive resections, thus reducing the communication with the cloud.

6 Conclusion

In this work we have presented two key improvements for practical image-based LBSs. First, a robust method for building visual 3D models in a mostly unsupervised way using images which are geo-tagged by integrating data from inertial sensors during the training phase has been described. Second, we have adapted an efficient camera resectioning technique to infer location estimations which, based on the algebraically minimal P3P algorithm, drastically speeds up the procedure by requiring a fewer number of inliers correspondences. Our experiments show that the performance, liability, and accuracy provided by our solution is suitable for fast response AR services requiring very precise overlaid information.

Acknowledgments. This work was supported by the Spanish MINECO, as well as European Commission FEDER funds, under grant TIN2012-38341-C04-03. Additionally, this work was also supported by the Seneca Foundation, a Science and Technology Agency of Region of Murcia, under the Seneca Program 2009.

References

1. Arai, I., Horimi, S., Nishio, N.: Wi-foto 2: heterogeneous device controller using wi-fi positioning and template matching. In: Proceedings of Pervasive'10 (2010)
2. Beauregard, S., Haas, H.: Pedestrian dead reckoning: a basis for personal positioning. In: Proceedings of WPNC'06, pp. 27–35 (2006)
3. Fiore, P.D.: Efficient linear solution of exterior orientation. IEEE Trans. PAMI **23**(2), 140–148 (2001)
4. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Comm. ACM **24**(6), 381–395 (1981)
5. Foxlin, E.: Pedestrian tracking with shoe-mounted inertial sensors. IEEE Comput. Graphics Appl. **25**(6), 38–46 (2005)
6. Fraundorfer, F., Wu, C., Frahm, J.M., Pollefeys, M.: Visual word based location recognition in 3D models using distance augmented weighting. In: Proceedings of 4th 3DPVT (2008)
7. Garcia, V., Debreuve, E., Barlaud, M.: Fast k-nearest neighbor search using GPU. In: IEEE Proceedings of CVPRW'08, pp. 1–6 (2008)
8. Golub, G., Van Loan, C.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1996)
9. Gower, J.C., Dijksterhuis, G.B.: Procrustes Problems. Oxford University Press, Oxford (2004)
10. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2003)
11. Hattori, K., et al.: Hybrid indoor location estimation system using image processing and WiFi strength. In: IEEE proceedings of WNIS 2009, pp. 406–411 (2009)
12. Li, X., Wang, J.: Image matching techniques for vision-based indoor navigation systems: performance analysis for 3D map based approach. In: IEEE Proceedings of the IPIN'12, pp. 1–8 (2012)
13. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2), 91–110 (2004)

14. Mulloni, A., Wagner, D., Barakonyi, I., Schmalstieg, D.: Indoor positioning and navigation with camera phones. IEEE Pervasive Comput. **8**(2), 22–31 (2009)
15. Murillo, A.C., Guerrero, J., Sagues, C.: SURF features for efficient robot localization with omnidirectional images. In: IEEE Proceedings of ICRA'07, pp. 3901–3907 (2007)
16. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
17. Rai, A., Chintalapudi, K.K., Padmanabhan, V.N., Sen, R.: Zee: zero-effort crowdsourcing for indoor localization. IN: ACM Proceedings of MobiCom12, pp. 293–304 (2012)
18. Ruiz-Ruiz, A.J., Canovas, O., Rubio Muñoz, R.A., Lopez-de-Teruel Alcolea, P.E.: Using SIFT and WiFi signals to provide location-based services for smartphones. In: Puiatti, A., Gu, T. (eds.) MobiQuitous 2011. LNICST, vol. 104, pp. 37–48. Springer, Heidelberg (2012)
19. Ruiz-Ruiz, A.J., Lopez-de-Teruel, P.E., Canovas,O.: A multisensor LBS using SIFT-based 3D models. In: IEEE Proceedings of IPIN'12, pp. 1–10 (2012)
20. Se, S., Lowe, D., Little, J.: Vision based global localization and mapping for mobile robots. IEEE Trans. Robot. **21**(3), 364–375 (2005)
21. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
22. Woodman, O., Harle, R.: Pedestrian localisation for indoor environments. In: ACM Proceedings of Ubicomp'08, pp. 114–123 (2008)
23. Wu, C.: SiftGPU: a GPU implementation of scale invaraint feature transform (SIFT) (2012). http://cs.unc.edu/ccwu/siftgpu
24. Wu, C.: Towards Linear-time Incremental Structure from Motion. In: 3DV'13 (2013)
25. QVision Qt CV library, University of Murcia (2013). http://qvision.sourceforge.net