

# Collaborative Pedestrian Mapping of Buildings Using Inertial Sensors and FootSLAM

Patrick Robertson<sup>†</sup>, Maria Garcia Puyol<sup>†\*</sup>, Michael Angermann<sup>†</sup>

<sup>†</sup> German Aerospace Center (DLR), Institute of Communications and Navigation, PO Box 1116, D-82230 Oberpfaffenhofen, Germany. E-Mail: *firstname.lastname@dlr.de*

\* University of Malaga, Spain



## Biography

Patrick Robertson received a Ph.D. from the University of the Federal Armed Forces, Munich, in 1995. He is currently with DLR, where his research interests are navigation, sensor based context aware systems, signal processing, and novel systems and services in various mobile and ubiquitous computing contexts.

Maria Garcia Puyol received her Diploma in Telecommunications Engineering from the University of Malaga in September 2011. She developed her Diploma Thesis on cooperative FootSLAM at DLR during 2010/2011, where she will continue with her Ph.D. studies focusing on indoor navigation for pedestrians.

Michael Angermann received a Ph.D. from the University of Ulm in 2004. He is currently with DLR, where his research interests are advanced and integrated communication and navigation systems.

## Abstract

The FeetSLAM technique builds on iterative processing of multiple sets of pedestrian odometry data, based on FootSLAM. The objective is to obtain maps of large areas based on many data sets. The central idea is that maps originating from other data sets are used as a so-called *prior map* for a given data set. We show that this follows from the optimal FeetSLAM derivation but is more suited to practical computation limitations such as limited memory. It also yields maps which are not overly dominated by one data set but rather balances the characteristics of each with the effect of averaging out errors. Over iterations, FootSLAM maps are gradually combined to yield a high-accuracy global map - the iteration speed is controlled by employing concepts from simulated annealing. We validate our approach using two data sets from two locations, consisting of four and five walks respectively.

## 1 Introduction and FeetSLAM Principle

Pedestrian navigation has been drawing significant research and development interest over the last few years and encompasses a wide range of research communities. In addition to using satellite navigation receivers, or signals of opportunity such as mobile radio or WLAN, the use of other low-cost sensors has become one of the addressed topics. For a number of years it has been known that foot mounted MEMS based inertial sensors (IMUs) can, in combination with known building plans, allow for stable positioning in two and three dimensions even in the absence of other signals [1, 2, 3].

As an extension to this work we recently presented FootSLAM - Simultaneous Localization and Mapping for pedestrians - using foot mounted IMUs as the main sensors [4, 5]. Developed by the robotics community, traditional SLAM for indoor and urban environments has drawn on sensors such as laser scanners and cameras whereas FootSLAM uses only the odometry - the noisy IMU-based measurements of a person's step vectors. We use the term odometry because we are in principle agnostic towards the actual mechanism used to obtain the step estimates. Researchers have used a wide array of approaches, ranging from foot mounted IMUs (e.g. [6]), stride detection with mobile phones, visual odometry, to electromyography based estimators. In a perfect world this odometry would be error free and the pedestrian's pose (location and orientation) could be estimated within the relative coordinate system for an unlimited distance travelled. Since state-of-the-art odometry suffers from the gradual increase in errors, FootSLAM must search over many different odometry error hypotheses finding one which best fits the previous pose history. Hypotheses in which the pedestrian revisits areas in the environments are rewarded and over time a reliable map of essentially "walkable areas" is constructed. Real data from people walking within office environments

at two locations has so far been used to validate the map building and relative localization abilities of FootSLAM. The approach can use GPS as a provider of reference position before and after entering a building, thus anchoring the map with reasonable position accuracy. FootSLAM, and the generalization presented in this paper are appealing because existing maps are often inaccurate, unavailable, outdated, proprietary, and do not reflect important features such as furniture, stalls, displays and other features of a place that significantly limit or channel pedestrian motion.

In this paper we will present an extension to the FootSLAM method to the collaborative or multi-user case, hence the term *FeetSLAM*. We shall address the problem whereby different data sets (walks) are to be processed to generate a common map, that is more accurate than any single map and encompasses the total area covered by all walks. First of all we distinguish these different cases of FeetSLAM:

1. A number of walks all starting at the same starting point and/or finishing at the same finishing point (or pose) and overlapping the explored area to a certain degree.
2. Walks not necessarily starting/finishing in the same point (or pose) but overlapping in the explored area to a certain degree.
3. Walks not necessarily starting in the same point and not necessarily overlapping in the explored area.

All these cases may be formulated as real-time or offline mapping problems. An interesting real-time usage scenario is mapping of a building by multiple collaborating pedestrians with the objective of providing immediate map and position information of all collaborating pedestrians or others. Such a scenario may occur in emergency situations where multiple teams of fire-fighters enter a building through the same or different entrances and carry out search and rescue tasks and want to avoid unnecessarily revisiting areas or involuntarily leaving out areas. In law enforcement applications, accurate determination of every team member's position and providing this information on a map may significantly improve mutual situation awareness and potentially reduce the risk of accidentally harming a team member. In this application the real-time requirements may be severe and no a priori map data may be available.

In this contribution we will focus on non-real time processing for the second case presented above, in the expectation that the techniques can be sped up to real-time capabilities over time. In offline applications we wish to derive a map that will later serve as basis for localizing

pedestrians by map-aided pedestrian dead reckoning. An example of this is collaborative mapping of airports, museums, shopping centers and other public buildings for use in tourism, travel, commerce, and any high-precision location based services. To support FeetSLAM, pedestrians roam through accessible rooms and areas on all levels of a building - perhaps as a deliberate mapping effort or during activities of everyday life. The pedestrians carrying out the mapping task needs to be equipped with some form of odometry-generating sensor, such as a foot-mounted IMU and most likely a GPS receiver for anchoring in an absolute coordinate system. In this scenario the measurement data needs to be recorded and will then be processed offline. The resulting map is then stored at a server or distributed to localization devices that use it to perform map-aided pedestrian dead reckoning. As more data are collected the maps can be refined to incorporate the new walks.

## 2 Proposed Iterative Processing

It can be shown that the optimal (in the Bayesian estimation sense) FeetSLAM estimator is a trivial extension of FootSLAM. In this case a single run of a sequential Bayesian estimator would process all data sets sequentially or in parallel, and the state would include the unknown starting pose (starting conditions, SC) of each walk. The common element linking all the walks is the map of the environment. The relationship for two walks can be seen in the Dynamic Bayesian Network (DBN) in Figure 1, which is an extension of the DBN from [4]. The main variables are:

- Pose  $\mathbf{P}_k$ : the location and the orientation of the person in 2D at time step  $k$ .
- Step vector  $\mathbf{U}_k$ : the change of pose at time  $k - 1$  to pose at time  $k$ .
- Inertial sensor errors  $\mathbf{E}_k$ : all the correlated errors of the inertial system.
- Step measurement  $\mathbf{Z}_k$ : a measurement subject to correlated errors  $\mathbf{E}_k$  as well as white noise.
- The visual cues which the person sees at time  $k$ :  $\mathbf{Vis}_k$ .
- The Intention of the person at time  $k$ :  $\mathbf{Int}_k$ .
- The Map  $\mathbf{M}$ : it is time invariant and can include any features and information to let the pedestrian choose  $\mathbf{Int}$ .
- The Starting Conditions  $\mathbf{SC}$ : the starting pose of the pedestrian, heading angle and scale factor of the underlying step measurements.

The starting conditions may, of course, be different for both walks. These starting conditions are a vital component of the state space and need to be estimated if they are not known. In fact much of the work presented in this paper is devoted to estimation of these starting conditions.

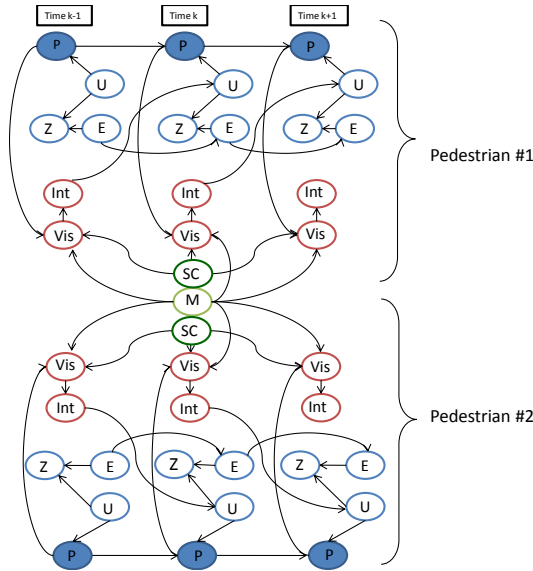
The goal of the Bayesian formulation for a two-Pedestrian scenario is to compute:

$$p(\mathbf{P}_{0:k}^1, \mathbf{P}_{0:k}^2, \mathbf{U}_{0:k}^1, \mathbf{U}_{0:k}^2, \mathbf{E}_{0:k}^1, \mathbf{E}_{0:k}^2, \mathbf{SC}^1, \mathbf{SC}^2, \mathbf{M} | \mathbf{Z}_{1:k}^1, \mathbf{Z}_{1:k}^2) = p(\{\mathbf{PUE}\}_{0:k}^{1:2}, \mathbf{SC}^{1:2}, \mathbf{M} | \mathbf{Z}_{1:k}^{1:2}), \quad (1)$$

which can be easily extended to a  $N_W$ -Pedestrian scenario as follows:

$$p(\{\mathbf{PUE}\}_{0:k}^{1:N_W}, \mathbf{SC}^{1:N_W}, \mathbf{M} | \mathbf{Z}_{1:k}^{1:N_W}). \quad (2)$$

Note that for this simple representation, the time indices  $k$  are the same for the walks. This is not a requirement for our map merging algorithm, in which the data are processed off-line, and hence can be obtained from walks occurring at different times.



**Figure 1.** Dynamic Bayesian Network (DBN) for the estimation problem with two pedestrians during three time slices. We have omitted an index that would differentiate the two segments for Pedestrian #1 and Pedestrian #2 for clarity.

In a particle filter implementation, particles would have to explore the state space of all odometry error sequences and all starting conditions. A further practical complication for a finite number of particles and the resulting particle depletion is that in a sequential approach the trajectories will tend to favor early data, which will bias the map

to data processed early in the sequential estimation process. Later data will then tend to follow the rut from early data. While this can be a problem for single-data FootSLAM, it will be confounded with the addition of more data sets.

The Dynamic Bayesian Network (DBN) of the two-pedestrian case from Figure 1 has some structural similarities to a family of error correction coding schemes from digital communications theory. In 1993 a family of codes called ‘‘Turbo’’ Codes were developed and are decoded iteratively at the receiver [7]. The codes are constructed by concatenating two or more simple component codes and are now used in a wide range of modern mobile communication standards. The optimal detector is prohibitively complex but a suboptimal, iterative variant exhibits very good error correction performance. From a Bayesian perspective this kind of iterative processing can be seen as a form of loopy belief propagation in a Bayesian network [8]. The name ‘‘Turbo’’ codes was chosen to reflect the nature of the iterative processing, as will now be explained in the context of FeetSLAM.

It can be shown by simple extension of the FootSLAM Bayesian Estimator derivation that other walks can be incorporated in a given FootSLAM estimation process in the form of prior counts in the FootSLAM maps. Ideally, we would start a FootSLAM run of a specific data set initialized with the *posterior distribution of the maps* computed from the other walks.

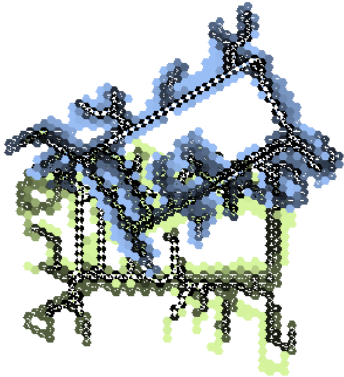
In FootSLAM, the transition map is learned by each particle  $p$  by counting each transition it makes across the edges of the hexagons that lie within the coordinate system. Operating in this manner, each particle stores its whole path through the hexagon grid. Learning the map by each particle  $p$  is based on Bayesian learning of multinomial distributions [4]. Each particle’s weight is updated as follows:

$$w_k^p = w_{k-1}^p \cdot \left\{ \frac{C_h^e + \alpha_h^e}{C_h + \alpha_h} \right\}^p, \quad (3)$$

where  $C_h^e$  are the transition counts for edge  $e$  of hexagon  $h$  and  $C_h = \sum_{e=0}^5 C_h^e$  stored in the individual map of the particle  $p$  that had been computed up to step  $k-1$ . The terms  $\alpha_h^e$  and  $\alpha_h = \sum_{e=0}^5 \alpha_h^e$  represent the *a priori* knowledge regarding the transition counts across the edges of hexagon  $h$  for particle  $p$ . Note that when no other prior information is available,  $\alpha_h^e$  has been empirically chosen to be  $0.8 \forall \{e, h, p\}$ .

Including other walks in a given FootSLAM estimation process is only possible, however, if we are able to relate all walks within the same coordinate system. This has been illustrated in Figure 2, where two FootSLAM maps

(one in blue and one in green) from the same floor of a building have been generated using two data sets coming from two different walks and are framed within different coordinate systems.



**Figure 2.** Two FootSLAM maps (one in blue and one in green) of the same building that are computed within different coordinate systems.

Therefore, before we can apply a map as an a priori map for another FootSLAM process we need to ensure that both data sets are within the same coordinate system. When this is ensured, our proposed iterative “Turbo” FeetSLAM algorithm presented in detail later starts by processing all data in individual FootSLAM runs and then combines all counts to a new map, which is used as a prior map in the next iteration of FootSLAM runs. This is repeated for a number of iterations. It should be noted that when constructing the map we must not include the map contribution that arose from a given walk the next time we process that particular walk. This is in exact analogy to the “prior” construction in Turbo decoding.

In the next section we will describe how we align maps within the same coordinate system.

### 3 Aligning Maps

#### 3.1 Starting Conditions

At this point we must first differentiate two terms that have been used in the paper so far: *starting conditions* and the *coordinate system* in which two FootSLAM maps lie, since although related, these are different concepts. The starting conditions define the user’s pose that places the odometry measurements - which are always differential, into a defined coordinate system in two or three spatial dimensions, plus initial heading, as well as scale. It is the nature of FootSLAM with finite number of particles to tend to snap into a particular resulting map or small set of maps. Two runs of FootSLAM with the same starting conditions may still lead to different maps that differ by way of a translation,

rotation and scale difference. Of course, different starting conditions will also lead to different maps. To summarize, a part of the difference in the maps’ coordinate systems comes from different starting conditions, but also from the stochastic nature of FootSLAM as described above.

The starting conditions are specified by four Gaussian distributions, one for each one of the following starting parameters:  $x$  coordinate,  $y$  coordinate, heading and scale factor. Each Gaussian distribution is defined by the mean ( $\mu$ ) and the standard deviation ( $\sigma$ ). A smaller standard deviation indicates greater certainty of the starting conditions of the pedestrian.

Consider the case where three data sets (walks) are to be combined. The prior map for any one data set is the sum of the maps of the other two (see (3)); but to allow us to add the maps, they must be aligned to the same coordinate system. Transforming or aligning the maps to a common coordinate system is necessary in order to be able to use one map as the prior for another data set.

#### 3.2 Transformation and Projection

When a map is transformed (i.e. translated, rotated and scaled), its hexagons are not necessarily aligned with the hexagons of the underlying grid of hexagons in another FootSLAM map. In order to be able to combine the counts of two maps, one of which has been transformed, a further manipulation to the map is needed: projection of the counts to a common hexagon grid.

In a 2D context and for our application, a transformation is the combination of a translation along  $x$  and  $y$  axes, a rotation around a *center of rotation* and a uniform scaling around a *center of scaling*. In short, these four parameters involved in the transformation are referred to, respectively, as  $x$  shift,  $y$  shift, rotation, and scale factor.

The mathematical formula used for transformation can be described by the following equation:

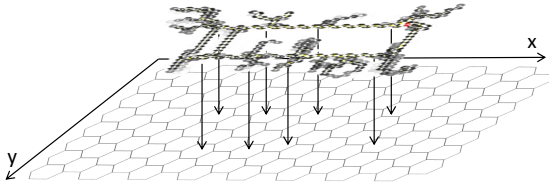
$$\begin{cases} x_t = (x - x_c) \cdot s \cdot \cos(r) - (y - y_c) \cdot s \cdot \sin(r) + x_c + \Delta x \\ y_t = (x - x_c) \cdot s \cdot \sin(r) - (y - y_c) \cdot s \cdot \cos(r) + y_c + \Delta y \end{cases} \quad (4)$$

where  $(x, y)$  are the Cartesian coordinates of a given point in 2D before the transformation,  $(x_t, y_t)$  the Cartesian coordinates after the transformation,  $(x_c, y_c)$  the Cartesian coordinates of the center for rotation and scaling and  $r, s, \Delta x$  and  $\Delta y$  the rotation, scale factor,  $x$  shift and  $y$  shift, respectively.

Note that the rotation and scaling use the mean  $x$  and  $y$  coordinates of the starting point of the walk as their center, that is,  $x_c = \mu_x$  and  $y_c = \mu_y$ .

After transforming every point it is projected onto a

target grid of hexagons, which serves as a common coordinate system for all the maps that are to be considered. The projection has been illustrated in Figure 3.

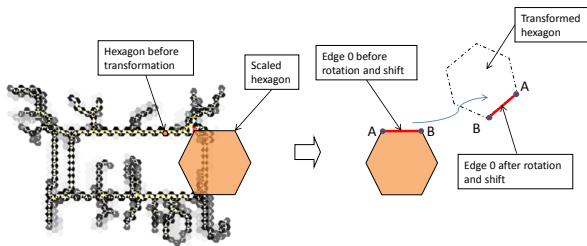


**Figure 3.** Illustration of the projection of a FootSLAM map onto a grid of hexagons.

The projection involves two steps: the projection of the vertices of every edge of the map and the projection of the transition counts of every edge. The projection of the vertices is trivial since it is done from a 2D surface (a plane) to another 2D surface that is parallel to the first, and hence the coordinates for the transformed and projected vertices are the same. The projection of the edges is more complex and will be explained later.

The transformation and projection of a map is performed on a hexagon per hexagon basis, and the following is applied:

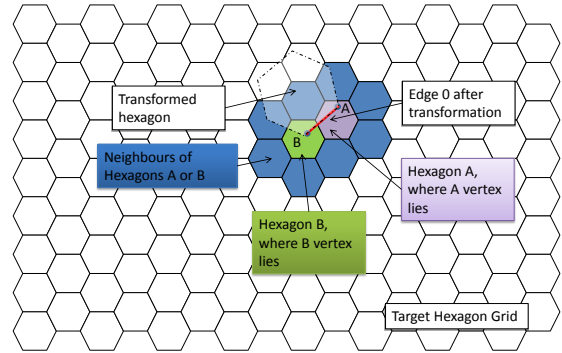
1. **Scaling:** the hexagon is scaled by multiplying its radius by the scale factor as shown on the left side of Figure 4. Then each of the edges of the scaled hexagon is rotated and shifted as shown on the right side of Figure 4.



**Figure 4.** Example of a transformation of the top edge of one hexagon. The hexagon is first scaled and then the edge is rotated and shifted.

2. **Projection of the edge:** each transformed edge is then projected onto the target grid of hexagons by projecting its two vertices (projection of the vertices A and B) as can be seen in Figure 5.
3. **Projection of the transition counts:** the transition counts - from now on referred to as  $C$  - are shared

among some of the edges of the target grid. To do that, first the *target hexagons* ( $h_{tg}$ ) - the hexagons of the target grid that will receive some of the transition counts - have been identified: these are the two hexagons where the two vertices of the transformed edge lie (points A and B) along with all their neighboring hexagons. See Figure 5.



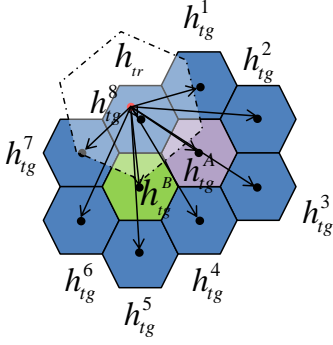
**Figure 5.** Target hexagons for transformed edge 0 (in red): hexagons A (in purple) and B (in green) where the two vertices of the transformed edge lie and their neighbors (in blue).

Once the target hexagons are defined, the counts  $C$  are shared among their edges. To compute how much of  $C$  each *target edge* receives, two different factors are used: a distance factor and an angular factor. The distance factor takes into account the distance between a transformed hexagon and a target hexagon (see Figure 6). The angular factor takes into account the relative orientation between a transformed edge and a target edge (see Figure 7) using an alternative representation for the edges - a semicircle on the outer part of the edge - that approximates the probability of the pedestrian crossing it at different angles. These two factors are then used to compute a weight that states how much each target edge receives counts from the transformed edge.

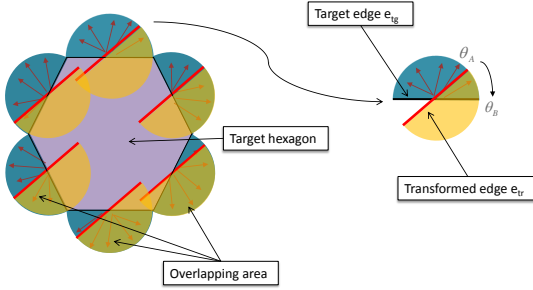
### 3.3 Correlating Two Maps

In order to compute the transformation that we should apply to one map so that it fits another we need to find a measure of how well these two maps fit. When combining two maps we can try all possible transformations on one map and choose the one that leads to the best fit.

The correlation between two random variables describes their statistical dependence. In the context of FootSLAM maps, a measure of how much one map *looks like* another map is needed. To this purpose, an appropriate function that reflects how well two maps fit each other has



**Figure 6.** Illustration of how the distance factor is computed. The red dot is the center of the transformed hexagon ( $h_{tr}$ ) with coordinates  $(x_{h_{tr}}, y_{h_{tr}})$ . The black dots are the centers of the target hexagons ( $h_{tg}$ ).



**Figure 7.** Example of a transformed edge (in red) that has been made to coincide with the six target edges (in black) of a hexagon and how their corresponding semicircles overlap (in green). The area in green is the angular factor for each edge.

been derived. We have named the two maps in terms of which one is transformed to fit the other:

- **“Underneath Map”:** This is the map that will stay fixed through the comparison of transformations, as a reference for the other map. It will be also referred to as  $M^U$ .
- **“Accounted for Map”:** This is the map that will be transformed to fit the underneath map and then projected onto its target grid of hexagons. From now on it will be referred to as “Accounted Map” or alternatively as  $M^A$ . We will use the term Transformed Map to refer to this map when it has been transformed.

We have chosen these names to reflect the roles of the two maps in the likelihood function presented as a justification of our correlation function. The counts of the underneath map are used to explain or account for the counts of the map that was transformed.

**Likelihood Function Choice:** Our justification of the correlation function relies on the DBN shown in Figure 1. We have two walks, and hence two histories of two pedestrian’s pose trajectories,  $\mathbf{P}_{0:k}^A$  and  $\mathbf{P}_{0:k}^U$ . We are interested in finding the transformation that will align the two pose trajectories. To do so, we will compute the posterior density function of a transformation  $\mathbf{T}$  conditioned on the pose trajectories:  $p(\mathbf{T}|\mathbf{P}_{0:k}^A, \mathbf{P}_{0:k}^U)$ . The transformation  $\mathbf{T}$  transforms the pose trajectory  $\mathbf{P}_{0:k}^A$  onto the map  $\mathbf{M}$ . We are assuming that  $\mathbf{P}_{0:k}^U$  is aligned with the map  $\mathbf{M}$ . Following the Foot-SLAM derivation of [4], we can apply Bayes Theorem and the chain rule to obtain:

$$p(\mathbf{T}|\mathbf{P}_{0:k}^A, \mathbf{P}_{0:k}^U) \propto \prod_{h \in \mathbf{P}^A} \prod_{e=0}^{e=5} \left( \frac{C_h^{eU} + \alpha_h^U}{C_h^U + \alpha_h^U} \right)^{T(C_h^{eA})}, \quad (5)$$

which can be used to compute how well  $\mathbf{P}_{0:k}^A$  fits  $\mathbf{P}_{0:k}^U$ . Note that  $T(C_h^{eA})$  is the number of transition counts for edge  $e$  of hexagon  $h$  of the Accounted Map when it has undergone transformation  $T$ .

As stated above, we must try all different transformations  $T$  and compute the likelihood value for each.

**The logarithmic form for the likelihood function:** So far we have been able to obtain a formula that states how well one map fits another. We can extend (5) to its symmetric form, that is, also taking into account how well the Underneath Map fits the transformed Accounted Map. So the likelihood value ( $LV$ ) between two maps can be computed as:

$$LV(M^A, M^U, T) = \prod_{h \in A} \prod_{e=0}^{e=5} \left( \frac{C_h^{eU} + \alpha_e}{C_h^U + \alpha_h} \right)^{T(C_h^{eA})} \cdot \prod_{h \in U} \prod_{e=0}^{e=5} \left( \frac{T(C_h^{eA}) + \alpha_e}{T(C_h^A) + \alpha_h} \right)^{C_h^{eU}}. \quad (6)$$

We have chosen to implement this function in its logarithmic form because it is numerically more robust, and we have normalized it with the total number of counts in each

map:

$$\begin{aligned} \log LV(M^A, M^U, T) &= \frac{\sum_{h \in A} \sum_{e=0}^{e=5} T(C_{h,e}^A) \cdot \log\left(\frac{C_{h,e}^U + \alpha_e}{C_h^U + \alpha_h}\right)}{\sum_{h \in A} \sum_{e=0}^{e=5} T(C_{h,e}^A)} \\ &+ \frac{\sum_{h \in U} \sum_{e=0}^{e=5} C_{h,e}^U \cdot \log\left(\frac{T(C_{h,e}^A) + \alpha_e}{T(C_h^A) + \alpha_h}\right)}{\sum_{h \in U} \sum_{e=0}^{e=5} C_{h,e}^U}. \end{aligned} \quad (7)$$

**Hexagon Correlation Factor:** We found that the likelihood value formula needed to be augmented by incorporating a heuristic term ( $\gamma$ ) that takes into account also the correlation of the total counts of each hexagon. We found that this increases the robustness of the likelihood function:

$$\gamma = \beta \cdot \sum_{h \in A} T(C_h^A) \cdot C_h^U, \quad (8)$$

where the parameter  $\beta$  - a *hexagon correlation factor* - has been empirically chosen to have a small value, in our experiments we chose  $\beta = 0.04$ . Finally, we obtain the augmented likelihood value:

$$\begin{aligned} \log LV^a(M^A, M^U, T) &= \frac{\sum_{h \in A} \sum_{e=0}^{e=5} T(C_{h,e}^A) \cdot \log\left(\frac{C_{h,e}^U + \alpha_e}{C_h^U + \alpha_h}\right)}{\sum_{h \in A} \sum_{e=0}^{e=5} T(C_{h,e}^A)} \\ &+ \frac{\beta \cdot \sum_{h \in A} T(C_h^A) \cdot C_h^U}{\sum_{h \in A} \sum_{e=0}^{e=5} T(C_{h,e}^A)} \\ &+ \frac{\sum_{h \in U} \sum_{e=0}^{e=5} C_{h,e}^U \cdot \log\left(\frac{T(C_{h,e}^A) + \alpha_e}{T(C_h^A) + \alpha_h}\right)}{\sum_{h \in U} \sum_{e=0}^{e=5} C_{h,e}^U} \\ &+ \frac{\beta \cdot \sum_{h \in A} C_h^U \cdot T(C_h^A)}{\sum_{h \in U} \sum_{e=0}^{e=5} C_{h,e}^U}. \end{aligned} \quad (9)$$

### 3.4 Searching for the Best Transformation

Once we have the capability of comparing two maps, we need to look for the best fit between the maps. To do so,

one of the maps is transformed using different values for the  $x$  and  $y$  shifts, rotation and scale factor, and the corresponding likelihood value is computed using (9). To do this in practice, we need to limit the range of transformations (i.e. the search space):

**Restriction of the search space** An automatic restriction of the area of search is needed in order to develop a completely automated algorithm with manageable complexity. The area is easily restricted using the Starting Conditions of the two maps involved in the comparison, using the worst case situation in which each map could be located at the opposite ends of their Gaussian distributions for each one of the four Starting Conditions Parameters.

**Automated Search:** The search is undertaken over the space of transformations in a discrete manner. These step sizes along with the limits of the area of search determine which values for each one of the four transformation parameters (rotation, scale,  $x$  shift and  $y$  shift) are used to transform one of the maps to make it fit the other.

The transformation that allows the Transformed Map and the Underneath Map to have the largest likelihood value (equation (9)) is called the *winning* transformation. This transformation is then used to transform the Accounted Map so that it can be used as a prior for the data corresponding to the Underneath Map or to compute a combined map.

### 3.5 Combining Maps

Combining the maps after transformation and projection of the counts is simple: we just add the counts of the contributing maps and we can do this because after projection they are now aligned to the same hexagon grid.

### 4 Controlling the convergence of FeetSLAM

As mentioned above we want to process the individual data sets in such a way that a single data set does not dominate the resulting joint or total map. We propose that modifications of the prior maps can be beneficial in alleviating such asymmetries. In particular, we want the iteration process to be gradual, allowing improvements of individual maps to propagate to others through the priors. In effect, we want the prior to gradually herd the particles into the region of the state space that is close to the true state. This means that particles need not explore areas of the state space that are extremely unlikely, allowing for more diversity in the likely regions. We achieve this by starting with a prior that is both weakened and smoothed, and gradually reduce these two modifications as iterations progress.

The weakening of a map consists of the division of its transition counts by a *weakening factor*  $>1$ , that is, making

the map less strong. This can be used to control the influence the prior map has on the update of the particles' weight.

The smoothing of a map is achieved by spreading the transition counts of each edge of each hexagon of the map among that same edge of the six neighboring hexagons and itself. This filtering process has the effect of making the map wider, giving more freedom to the particles that will explore the area using it as a prior.

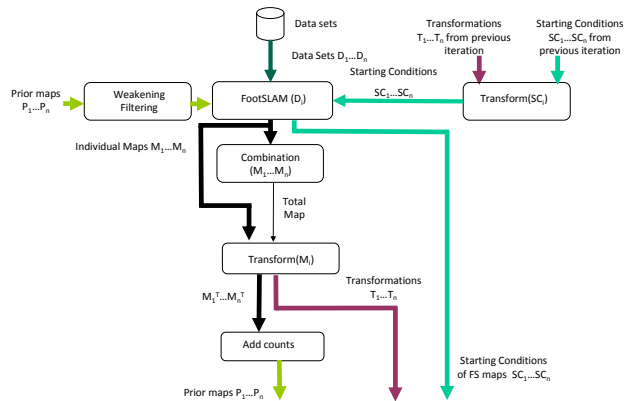
## 5 Formal Description of the Iterative Algorithm: "Turbo" FeetSLAM

Simply put, the Turbo FeetSLAM technique builds on iterative processing of odometry data, using maps originating from other data sets as a prior map for a given data set. The algorithm serves two main goals:

1. Obtaining a complete map, or total map, of the walkable areas.
2. Obtaining more accurate individual maps.

### 5.1 The Algorithm

In Figure 8 the basic structure of the algorithm is presented:



**Figure 8.** Schematic illustration of the proposed algorithm for automatic map computation from several walks.

The algorithm operates as follows: at the zeroth iteration FootSLAM is run for each of the data sets to obtain the Individual Maps  $\{M_1 \cdots M_n\}$ , with  $n$  being the number of data sets that is being considered at that iteration. Then, these maps are properly combined to obtain the Total Map,  $M^C$ . Next, the Individual Maps are transformed one by one to fit the Total Map to generate the Individual Transformed Maps  $\{M_1^T \cdots M_n^T\}$ . The Individual Transformed Maps are then used to generate the prior maps  $\{M_1^P \cdots M_n^P\}$

for the next iteration: here, for each data set, the combination of the *other* Individual Transformed Maps is used to generate its prior map (except for the map from that data set). The transformations  $\{T_1 \cdots T_n\}$  found for the Individual Maps to fit the Total Map along with the Starting Conditions  $\{SC_1 \cdots SC_n\}$  at the end of each FootSLAM process are used for the next iteration to obtain the new Starting Conditions for each data set. The prior maps are also used in the next iteration, appropriately weakened and filtered.

Our goal now is to explain the algorithm in greater detail, together with the remaining parameters and inputs. The index  $i$  will refer to the iteration number.

### 5.2 Data

To run the Turbo FeetSLAM Algorithm with  $N_W$  walks at iteration  $i$  the following are needed:

1. Data Sets for the walks  $\mathbf{D} = \{D_1, D_2, \cdots, D_{N_W}\}$ . These data sets are just the result of converting the raw data from the walks into odometry. This odometry data sets do not change over the iterations.
2. Starting Conditions  $\mathbf{SC}^i = \{SC_1^i, SC_2^i, \cdots, SC_{N_W}^i\}$ . The Starting Conditions for each walk.
3. Transformations for the Starting Conditions  $\mathbf{T}^i = \{T_1^i, T_2^i, \cdots, T_{N_W}^i\}$ . These transformations are computed at iteration  $i - 1$  and transform the Starting Conditions so that the map is located according to the Total Map computed at the previous iteration.
4. Prior Maps  $\mathbf{M}^i = \{M_1^i, M_2^i, \cdots, M_{N_W}^i\}$ . The Prior Maps are computed at iteration  $i$  so that at iteration  $i + 1$ , each one of the Data Sets  $D_d$  uses the information provided by the other  $N_W - 1$  walks.

### 5.3 FootSLAM Map Computation

FootSLAM is run for each one of the  $N_W$  data sets with the help of the Data Sets  $\mathbf{D} = \{D_1, D_2, \cdots, D_{N_W}\}$ , the Starting Conditions  $\mathbf{SC}^i = \{SC_1^i, SC_2^i, \cdots, SC_{N_W}^i\}$ , the Prior Maps  $\mathbf{M}^i = \{M_1^i, M_2^i, \cdots, M_{N_W}^i\}$ . As a result, the Individual Maps  $\mathbf{M}^i = \{M_1^i, M_2^i, \cdots, M_{N_W}^i\}$  are obtained.

At the end of each FootSLAM process, a new set of Starting Conditions is computed and is referred to as the *winning* Starting Conditions.

### 5.4 Computing the combined map

This is the most important part of the algorithm, where the  $N_W$  Individual Maps are processed to generate a Total Map.

**Comparing Maps Pairwise:** We form a pool of maps that contains all the  $N_W$  Individual Maps that were obtained

with FootSLAM for a single data set. In  $N_W - 1$  stages a Total Map that encompasses the information provided by all the maps can be generated as follows:

- At each stage, the maps are taken two at a time and compared. The comparison is performed as explained in 3.4 for every possible combination of maps.
- At the end of each stage, the two maps that best fit together - that is, the ones that had the greatest likelihood value (9)- are removed from the pool and their combined map is added. This means that after every stage there is one fewer map in the pool.
- The comparisons that were already run in previous stages are not computed again.

The number of combinations  $N_c$  that need to be tried is

$$N_c = \binom{N_W}{2} + \sum_{k=1}^{k=N_W-2} k = (N_W - 1)^2. \quad (10)$$

The first part of equation 10,  $\binom{N_W}{2}$ , is the combinatorial number of the  $N_W$  individual maps taken two at a time, (i.e. the number of handshakes between  $N_W$  people).

The second part of the equation,  $\sum_{k=1}^{k=N_W-2} k$  accounts for the possible combinations that still need to be computed every time a new map is added to the pool, that is, the comparisons between the new map and the other available maps in the pool. Note that there is no need to recompute the comparison between the maps that were already in the pool, since they were computed at the previous stage.

### 5.5 Transformations for Individual Maps

The transformations for the Individual Maps are obtained by running the search to make each of the  $N_W$  Individual Maps fit the Total Map. The resulting Transformed Individual Maps  $\mathbf{M}^{\mathbf{T}^i} = \{M_1^{T^i}, M_2^{T^i}, \dots, M_{N_W}^{T^i}\}$  will be used to generate the Prior Maps for the next iteration.

We use the transformation between the Individual Map and the Total Map and not the transformation that was already computed to generate the Total Map for each one of the Individual Maps because this transformation takes into account a richer total map.

### 5.6 Prior Map Computation

The prior maps for the next iteration for each one of the  $N_W$  data sets are very easily computed: for each Data Set, the other  $(N_W - 1)$  Individual Transformed Maps are combined. This is done so that when FootSLAM is run for a given Data Set  $D_d$  its own map is not explicitly included, but only the information provided by the rest of data sets.

So the prior map can be seen as the transition counts of the other maps, properly combined.

### 5.7 Weakening and Filtering

This block adjusts the parameters that control the influence of the Prior Maps on the FootSLAM estimation process. The parameters that are readjusted from one iteration to the next are:

- Prior map weakening factor: this factor is set at the zeroth iteration to a certain value greater than 1 and is slowly decreased to 1 over the iterations to make the prior map stronger. In our experiments 1.9 was empirically chosen as the starting value.
- Prior map filter factor: this factor is smaller than 1 and it is slowly increased to 1 over the iterations. This is because the prior map will be more accurate over the iterations, and can be given more importance during the FeetSLAM process. For our experiments it was chosen to start at 0.8.

### 5.8 Starting Conditions Computation

The Starting Conditions for the next iteration for each of the  $N_W$  data sets are computed by taking the winning Transformation computed in the Transformation of the Individual Maps block and applying it to the winning Starting Conditions for the FootSLAM process at the last iteration.

### 5.9 Zero'th Iteration Initialization

Some characteristics of the zeroth iteration are:

- No use of prior: no previous knowledge of the transition counts is available.
- Manually written Starting Conditions: a description of the Starting Conditions for the walk is needed, when no information with absolute SC is available.
- One might not include all the  $N_W$  data sets: data sets that do not converge without a prior might be included in in the algorithm at a later stage, in the expectation that it will converge when a prior map is available after processing the other data sets. We have not implemented this, and deliberately chose one of our experiments to have a data set that did not converge at iteration zero.

## 6 Qualitative Performance Assessment of FootSLAM and FeetSLAM maps

In this section, a novel quantitative metric of performance evaluation will be presented that counts the number of violations of FeetSLAM or FootSLAM maps against a known ground truth map.

The ratio  $R$  of crossed walls and furniture for a map  $\mathbf{M}$  has been defined as follows:

$$R = \frac{\sum_{h \in \mathbf{M}} \sum_{e \in V} C_h^{eV}}{\sum_{e=0}^5 \sum_{h \in \mathbf{M}} C_h^e}, \quad (11)$$

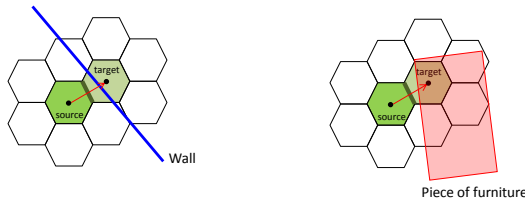
where  $\sum_h \sum_{e \in V} C_h^{eV}$  is the sum of all the transition counts that cross a wall or a piece of furniture ( $V$  stands for violation) and  $\sum_{e=0}^5 \sum_{h \in \mathbf{M}} C_h^e$  is the total transition counts in the map.

Since walls are lines and furniture are polygons two different criteria can be differentiated to determine whether a transition count crosses a wall or a piece of furniture or not.

A transition from a source hexagon  $h_s$  to a target hexagon  $h_t$  across edge  $e$  represents the probability of a pedestrian crossing it when walking from  $h_s$  to  $h_t$ . The center of the hexagons can be used as starting and finishing points for the transition across the edge as an approximation:

**Criteria for the Walls:** A wall between two hexagon centers indicates a crossed (or violated) wall. The left side of Figure 9 illustrates an example of a transition that crosses a wall.

**Criteria for the Furniture:** The center of the target hexagon  $h_t$  lying inside a piece of furniture indicates a wrong transition. The right side of Figure 9 illustrates an example of a transition count that would essentially allow the pedestrian to step over a piece of furniture.



**Figure 9.** Illustration of transition counts that cross a wall (on the left) and a piece of furniture (on the right).

Note that in the case when we have a transition count crossing a wall and a piece of furniture at the same time, it is only counted as one violation.

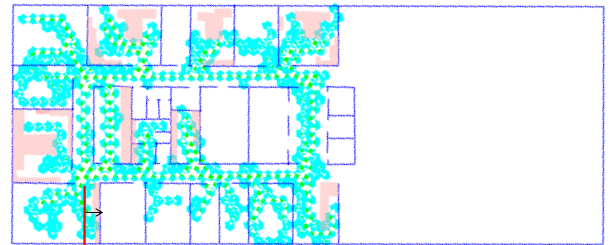
We used an XML representation of the walls and pieces of furniture for the DLR scenario when we evaluated our Turbo FeetSLAM algorithm quantitatively. The walls and pieces of furniture can be easily transformed (using the

same formula that we use when transforming a FootSLAM map, (4), to fit any given FootSLAM map and compute the corresponding value for the ratio of crossed walls and furniture. We used a computer program to search for the smallest violation ratio for a given map since FootSLAM maps are rotation, scale and translation invariant.

## 7 Results

Two sets of data were processed. One smaller set of five walks of about 6 to 15 minutes of data, called DLR. The other set, called MIT, was collected in a larger building (at the CSAIL campus of MIT) and consisted of four walks of roughly 15 minutes duration. The second data set contained more complex and diverse geometric regions and the walks were not aligned to a main corridor region. For one of the MIT data sets we discarded the last ca. 15% of the odometry data since it was affected by a large singular error. Real world implementations of FeetSLAM will have to identify strong deviations of a map from the total map automatically.

Figure 10 shows the results for DLR experiments after nine iterations. The FootSLAM map that has been represented is that of the particle with the highest posterior likelihood. Our results show that FeetSLAM can reduce the FootSLAM hexagon transition error rate from roughly 20% (iteration zero) to less than 2% (FeetSLAM after nine iterations), as shown in Figure 14. We chose one of the DLR walks to be very short and provide almost no loop closure by itself. As expected, the map for this data set (without a prior map) did not converge to a single map. However, in combination with the other maps this individual map converged after one iteration and helped the overall mapping process.



**Figure 10.** Total Map after nine iterations for the DLR data and ground truth and furniture arrangement. The red line represents the wall in the original plan designed by the architect. The black arrow points to the real location of one of the walls, which was erroneous in our original ground truth map.

Figures 11(a) to 11(d) show the aggregated FootSLAM

maps of all the particles for each one of the four data sets of the MIT data for the zeroth iteration - that is, when no prior was available. Figure 11(e) shows the best combination of those four maps at the end of that iteration and Figure 12 shows the total maps after iteration 1 (Figure 12(a)) and 2 (Figure 12(b)). The improvements of the quality of the Total Map from iteration 0 to iteration 1 are clearly visible. Videos of the FeetSLAM algorithm can be found in [9].

Figure 13 shows the results for the MIT experiments after 10 iterations. The FootSLAM map that has been drawn is the total map with the highest posterior likelihood. The ground truth has been manually transformed to fit the FootSLAM map.

With FeetSLAM we have even exposed an error in our building plan ground truth - the actual layout of the drywall construction had been recently changed and not reflected in the map, as shown in Figure 10, where we show that one of the walls had been incorrectly represented in the original reference map.

Both of our experiments have been run on the same processor, with six dual cores and a clock speed of 3.46 GHz. It took 37 and 42 hours to run, respectively, ten iterations for the 5 DLR data sets and the 4 MIT data sets with 90000 particles.

## 8 Conclusion and Outlook

We have presented a collaborative form of 2D FootSLAM (FeetSLAM) that allows multiple data sets to be combined in order to map larger areas. The proposed method significantly improves the mapping accuracy of a single data set in addition to providing maps for the entire area. This is because maps from all data sets support each other in the convergence process of FootSLAM.

Our approach is based on iterative processing, because the optimal estimator is expected to suffer from particle depletion due to the large state space for many data sets. We control the iteration process by gradually increasing the strength of the other maps (the prior) over iterations.

In a future application, the maps are expected to be useful not only for greatly improved positioning of pedestrians, but also as a basis for semantic maps where places have meanings and these can be learnt automatically from data.

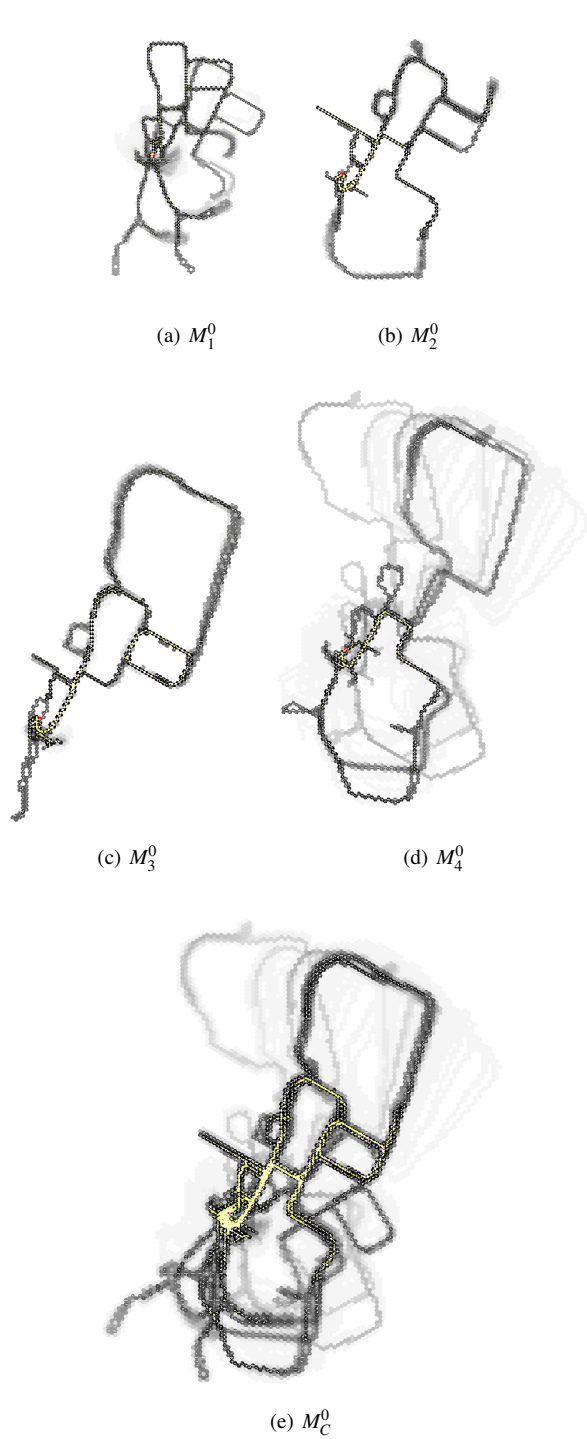
Future work will address mapping in three dimensions, robust mapping of larger areas and complexity analysis.

## Acknowledgements

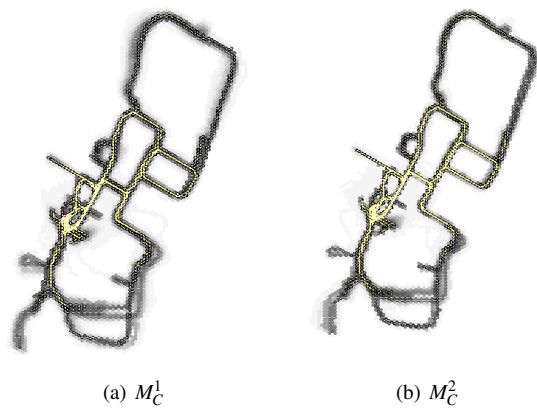
We would like to thank Daniela Rus, Brian Julian, John Leonard and Michael Kaess from the MIT CSAIL for their kind support and fruitful discussions.

## References

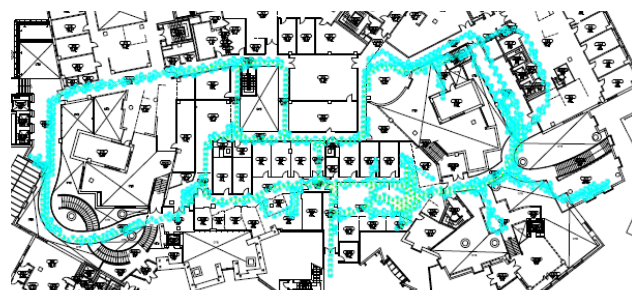
- [1] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments," in *Proc. of the UbiComp 2008*, Seoul, South Korea, Sep. 2008.
- [2] S. Beauregard, Widyawan, and M. Klepal, "Indoor PDR performance enhancement using minimal map information and particle filters," in *Proc. of the IEEE/ION PLANS 2008*, Monterey, USA, May 2008.
- [3] B. Krach and P. Robertson, "Cascaded estimation architecture for integration of foot-mounted inertial sensors," in *Proc. of the IEEE/ION PLANS 2008*, Monterey, USA, May 2008.
- [4] P. Robertson, M. Angermann, and B. Krach, "Simultaneous localization and mapping for pedestrians using only foot-mounted inertial sensors," in *Proc. UbiComp 2009*, Orlando, Florida, USA.
- [5] B. K. P. Robertson, M. Angermann and M. Khider, "Inertial systems based joint mapping and positioning for pedestrian navigation," in *Proc. ION GNSS 2009*, Savannah, Georgia, USA, Sep. 2009.
- [6] E. Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *IEEE Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, Nov. 2005.
- [7] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. ICC '93*, May 1993, pp. 1064–1070.
- [8] R. McEliece, D. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, 1998.
- [9] "FootSLAM videos and reference data sets download," <http://www.kn-s.dlr.de/indoornav>.



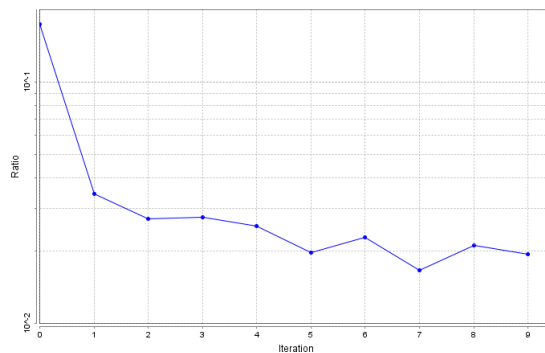
**Figure 11.** (a) to (d) Individual maps obtained running FootSLAM with no prior (zeroth iteration) for the “MIT data”. (e) Total combined map at the end of the zeroth iteration.



**Figure 12.** Best combination of the FootSLAM maps of the MIT experiment after (a) iteration 1 and (b) iteration 2.



**Figure 13.** Total Map after 10 iterations for the MIT data and an overlay of the ground truth map of the building where the walks took place.



**Figure 14.** Ratio of crossed walls and furniture for the “DLR data” FootSLAM map over the iterations.