2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Heterogeneous Map Merging Using WiFi Signals

Gorkem Erinc, Benjamin Balaguer, and Stefano Carpin

Abstract— **We propose a map merging algorithm that is capable of merging together heterogeneous maps independently built by different robots. Heterogeneous map merging is a crucially important problem for scenarios where multiple heterogeneous robots collaborate to provide situational awareness in urban search and rescue, patrolling, and explorations tasks, just to name a few. To remedy the lack of uniform representation between heterogeneous map models, we rely on the ubiquitous presence of WiFi signals in today's environments. Our solution consists of three steps. First, the overlap between the heterogeneous maps being merged is determined. Second, metric correspondences between overlapping parts are established. Third, the merging is improved by exploiting the structural properties inherent to graph-based maps. Our proposed system is validated using various occupancy grid and appearance-based maps built in real-world conditions, the results of which confirm its strengths. To the best of our knowledge, this is the first solution to the heterogeneous map merging problem.**

I. Introduction

In this paper we consider the problem of merging together spatial models relying on different representations. This problem is relevant when multiple heterogeneous robots are operating in the same environment and need to combine together their maps. The problem of merging occupancy grid maps can be considered solved [3], but the heterogeneous case has not been addressed. In many practical scenarios (e.g., urban search and rescue, patrolling, and exploration) robots are used as tools to provide information to humans, who then decide the proper course of action given that information. Evidently, the ability to fuse together multiple heterogeneous partial models can greatly improve the overall utility of the collected information, especially when considering each model's strengths and weaknesses.

Although the algorithm we propose extends to other map types, we focus our discussion on occupancy grid and appearance-based maps. The merging problem has the inherent challenge that different spatial models lack, by definition, a common representation. To overcome this obstacle, we assume that every robot, regardless of the type of map they produce, is equipped with a WiFi card. We believe this is a safe assumption considering that robots will need to communicate amongst each other and with human operators. As each robot builds its map, the Wireless Signal Strength (WSS) of all Access Points (APs) in range is recorded and later on used as a substrate to combine models relying on different representations. In this manuscript we offer the following contributions:

School of Engineering, University of California, Merced, CA, USA, {gerinc,bbalaguer,scarpin}@ucmerced.edu.

- we compute the amount of map overlap between multiple heterogeneous maps by casting the problem as One Class Classification;
- we develop, to the best of our knowledge, the first system capable of merging heterogeneous maps;
- we outline crucial design choices with respect to the algorithm's applicability and performance;
- we evaluate and compare numerous algorithms across different datasets.

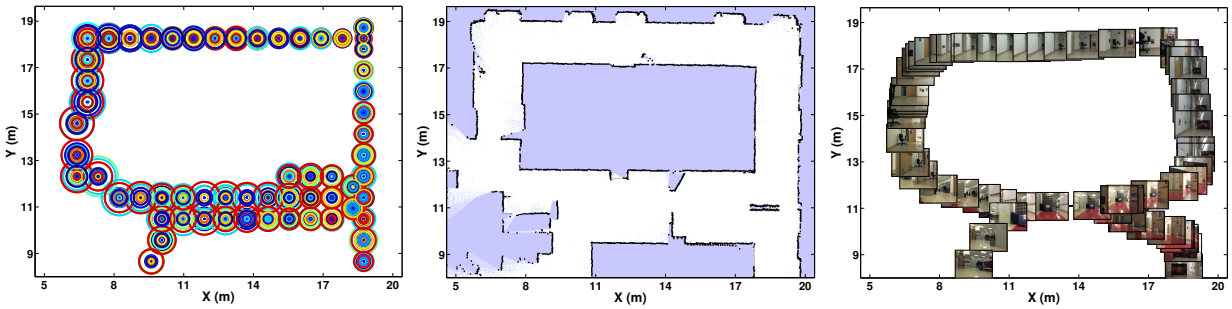
The rest of the paper is organized as follows. Section II highlights previous work on map merging and WiFi localization. In Section III, we describe the problem statement and introduce mathematical definitions used throughout the paper. The map merging algorithm consisting of three stages is described in Section IV. We conclude the paper by presenting, in Section V, results performed in real environments and making final remarks in Section VI.

II. Related Work

Map merging research has been limited and most related works appeared relatively recently when compared to the mapping literature. All former attempts to solve this problem have considered instances where all maps to be merged are homogeneous. Specifically, map merging can be approached in two ways. On one hand, robots may combine their individual models during the mapping phase. On the other hand, the maps may be fused together after the individual mapping processes have terminated. We consider the second approach, which has been solved exclusively for the homogeneous merging of occupancy grid [2], [3], [6], [7], topological [10], and appearance-based maps [4]. Heterogeneous maps in general have scarcely been considered, although never in the context of merging, in [12], [16].

Since we exploit WSS as a shared medium available to every heterogeneous map, we briefly described robotics-related WiFi research. Literature in this area is growing and the reader is referred to [1] for a more comprehensive survey. We embrace a data-driven approach, where we exploit recorded WSS information to infer location. This technique enables various algorithms to solve WiFi localization problem as a machine learning problem [1], [5], [9], [11], [15]. It is important to note that WSS readings acquired with different hardware and at varying times of the day (and even weeks or months) were sufficiently consistent to produce accurate localization results [1], [9]. These findings were the first indication that exploiting WSS for map merging had potential, especially when considering heterogeneous robot teams.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE



(a) Wifi map (b) Occupancy grid map (c) Appearance-based map

Fig. 1: Illustrative examples for the three different types of maps presented in this paper.

III. Problem Formulation

As aforementioned, we focus on merging occupancy grid and appearance-based maps, but the approach is generic and supports merging different types of maps as long as WSS are collected during the map building process. An occupancy grid map, M_{OCC}, represents the environment as a uniformly spaced grid of binary random variables, each encoding the presence of an obstacle at that location [8]. Conversely to occupancy grid maps, there exists no unified definition of appearance-based maps. We define an appearance-based map as an undirected weighted graph $M_{APP} = (V, E, w)$ in which every vertex $v \in V$ represents an image acquired by a camera at a certain position in the workspace. An edge $e_{ij} \in E$ connects vertices $v_i, v_j \in V$ if the associated images are sufficiently similar according to a given similarity metric S. The weight w of an edge is set to the similarity between the vertices it connects, $w(e_{ij}) = S(v_i, v_j)$. Different S metrics have been proposed. In our implementation, the similarity between two images is defined by extracting SIFT features from each image and counting the number of common ones among both images. It is important to note that the graph-like appearance-based map structure does not include any metric information. Further details on building appearance-based maps are given in [4]. Figure 1 shows examples of the different types of maps presented herein.

Regardless of the map's type being considered, an observation $Z_i = [z_i^1, z_i^2, \ldots, z_i^a]$ is acquired using a WiFi card, where a denotes the total number of APs seen throughout the environment. Each z_i^k is the WSS of the k-th AP, measured in dBm, unless the AP cannot be seen from a particular location, in which case we set $z_i^k = -100$. Each observation Z_i is linked to a label $\hat{L}_i$ representing either a Cartesian point $C_i = [X_i, Y_i]$ in the case of an occupancy grid map or an image I_i for an appearance-based map. All acquired observations and their labels are then collected into a set $\hat{T} = \cup_{i=1}^{m}\{\hat{L}_i, Z_i\}$ where m is the total number of observations.

In an attempt to model the noise in WSS readings and increase the robustness of the algorithm, we partition the m WiFi readings using their labels $\hat{\mathbf{L}} = \{\hat{L}_1, \hat{L}_2, \ldots, \hat{L}_m\}$ into c clusters $\mathbf{L} = \{L_1, L_2, \ldots, L_c\}$ with $c \leq m$. For appearance-based maps no clustering is required since each label representing an image corresponds to only one observation. Therefore, labels are kept such that $L_i = \hat{L}_i$ and $c = m$. For occupancy grid maps, WiFi observations are clustered using their Cartesian coordinates. One of the most common clustering methods is k-means. This method aims at creating optimal cluster boundaries such that each cluster contains all the points closest to its centroid. This methodology fails to address some vital aspects required for successful WiFi signal classification. First, the algorithm does not guarantee a minimum number of points within each cluster. As experimentally shown in [1], the number of observations per location has a considerable impact on the WiFi classification error. The minimum number of observations per location s_{min} should not be less than a preset value in order to reach the targeted average classification error. Furthermore, we do not want observations acquired at locations that are far away from each other to be grouped in the same cluster. By increasing the number of clusters k the average cluster diameter can be decreased. The maximum cluster diameter d_{max}, however, cannot be controlled and depends on several other parameters such as the initialization method and data distribution. In our implementation, we set s_{min} and d_{max} to 3 and 1 meter, respectively. These values are motivated by the results presented in [1].

The non-uniform binary split algorithm [13] (also known as bisecting divisive partitioning or hierarchical clustering) addresses these issues at the expense of optimality. Specifically, it is possible for the binary split algorithm to create clusters in which some points are actually closer to a neighboring centroid. The method starts with one cluster containing all of the data and recursively bipartitions the cluster with largest distortion into two sub-clusters until the required number of clusters is reached. The method essentially structures clustering into a hierarchical binary tree. Our modified version of the algorithm uses k-means clustering with $k = 2$ to split the data into two partitions. More importantly, instead of fixing the final number of clusters, we define a stopping condition for splitting a branch in the binary tree. According to this condition, we only approve a cluster split if its diameter is greater than d_{max} and both resulting sub-clusters contain at least s_{min} points. These conditions guarantee that after the final iteration of the algorithm we have at least s_{min} observations in each cluster and the diameter of a cluster only exceeds d_{max} in parts

of the environment where not enough WSS readings were acquired. The algorithm is fast, taking less than 4 seconds to cluster over 1000 points.

For each cluster we store the Cartesian coordinate of its centroid as a label L_i along with a vector of observations located inside the cluster $\mathbf{Z_i} = \{Z_i^1, Z_i^2, \ldots, Z_i^{s_i}\}$ where s_i is the total number of observations inside cluster i. Put differently, s_i observations are made for cluster i, whose label is L_i. It is worthwhile to note that the clusters may not and do not need to be uniform (i.e., s_i is not necessarily equal to $s_j \ \forall_{i,j} : i \neq j$). In other words, the number of observations for one cluster can be different than the number of observations for another cluster. The WiFi map can mathematically be represented as a set T of observations and their labels, $T = \cup_{i=1}^{c}\{L_i, \mathbf{Z_i}\}$.

We define the heterogeneous map merging problem as follows. Given two maps M_1 and M_2, their corresponding WiFi maps T_1 and T_2 with the list of corresponding labels $\mathbf{L_1} = \{L_1^1, L_2^1, \ldots, L_{c_1}^1\}$ and $\mathbf{L_2} = \{L_1^2, L_2^2, \ldots, L_{c_2}^2\}$ where c_1 and c_2 are the number of clusters in T_1 and T_2, determine the mapping $f : \mathbf{L_2} \rightarrow \{\mathbf{L_1} \cup \xi\}$ where ξ is the *null label*. Since the maps do not necessarily fully overlap, some labels in $\mathbf{L_2}$ will not have a corresponding label match in $\mathbf{L_1}$. These labels will then be assigned to the null label ξ. Note that in general the roles of $\mathbf{L_1}$ and $\mathbf{L_2}$ can be swapped. For the sake of space and simplicity, the problem statement considers only the case where two maps are merged and, without loss of generality, we describe the process for merging an occupancy grid map $M_1 = M_{OCC}$ with an appearance-based map $M_2 = M_{APP}$. The idea can naturally be extended to multiple maps by merging them in pairs, other types of map (e.g., topological), and identical map types (e.g., merging two appearance-based maps). In the case considered in this paper, merging these two maps narrows down to assigning a Cartesian coordinate to each image in the appearance-based map. The merged map will consequently consist of images overlaid on the occupancy grid map with edges connecting the similar ones.

IV. Map Merging

Our map merging algorithm consists of three steps, as shown in Figure 2. Given an occupancy grid and an appearance-based map, we first deduce the amount of map overlap by identifying and separating the labels that map to the null label (i.e., $L_i \rightarrow \xi$). In the second stage, for each of the remaining labels in the appearance-based map $L_i \in \mathbf{L_2}$ the probability distribution over the set of labels in the occupancy grid map $\mathbf{L_1}$ is computed. The most similar Cartesian label can be determined using the probability distribution. Such an estimation, however, would be solely based on the WiFi readings linked to each image in the appearance-based map. In the last step, we use the additional information inherently encoded by the appearance-based map's edges to refine the probability distribution and improve the precision of the estimate by applying regression in the space of Cartesian labels.

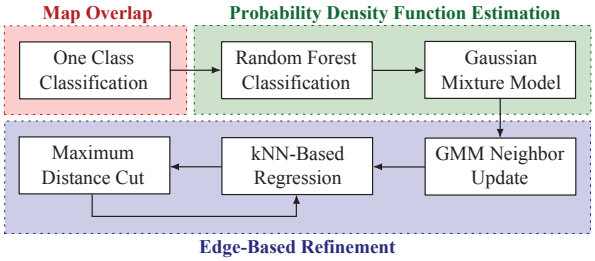


Fig. 2: Flowchart describing three stage of heterogeneous map merging algorithm

A. Computing Map Overlap

The issue of determining map overlap is a One-Class Classification (OCC) problem, whose aim is to distinguish one class of objects from all other possible objects. Given a set of observations in T_1 (i.e., the metric map) and we want to classify unknown observations inside T_2 (i.e., the appearance-based map) to see whether or not they belong to T_1. With principles related to outlier detection, OCC has stimulated the design of many algorithms. Since these algorithms are often data-dependent, we have implemented and contrasted five alternatives. Interested readers are directed to [14] for a more detailed survey.

One-Class Support Vector Machine (OC SVM): OC SVM is a modified version of the two-class SVM where only one class is necessary for training. Similarly to SVM, a kernel function maps the training data into a feature space. The origin of that feature space is then treated as the only member of the "second" class and the techniques of a binary SVM classifier can be applied to produce a hyperplane that maximizes the functional margin. For the kernel of the OC SVM, we use a Gaussian Radial Basis Function that requires a parameter h describing the bandwidth of the kernel. Once the OC SVM is trained on the first map's data T_1, a new observation $Z_i \in T_2$ is classified by computing its relation to the hyperplane, $f(Z_i) = (w \cdot \Phi(Z_i)) - \rho$, where $f(x)$ represents the hyperplane equation, w is the normal vector to the hyperplane, $\Phi(x)$ describes the kernel function, and ρ corresponds to the hyperplane's offset. If $f(Z_i) > 0$ then $Z_i \in T_1 \cap T_2$ and if $f(Z_i) < 0$ then $Z_i \notin T_1$.

Principal Component Analysis (PCA): PCA orthogonally projects a dataset from one subspace to another while retaining as much variance into as few components as possible. PCA in the context of OCC is performed as follows. First, the training data T_1 is used to compute the a components, of which only the $x < a$ with the greatest variance are retained. The x components are then used to project the training data into a lower dimensional subspace, resulting in $\tilde{T_1}$. A new observation to classify $Z_i \in T_2$ is projected into the same subspace (resulting in $\tilde{Z_i}$) and the Euclidean distance to the nearest point in $\tilde{T_1}$ is computed by $D(\tilde{Z_i})$. If $D(\tilde{Z_i}) < d$, where d is a pre-computed distance threshold, we deduce that $Z_i \in T_1 \cap T_2$ (otherwise, $Z_i \notin T_1$).

k-Means: the k-Means algorithm for OCC attempts to characterize the training data T_1 as a set of similar objects. The observations of the first map T_1 are partitioned into a set

of k clusters that can then be exploited to classify unknown observations. The k cluster centers are consequently a-dimensional objects. An observation to classify $Z_i \in T_2$ is compared against its closest cluster center, where the Euclidean distance between the two is given by $D(Z_i)$. The new observation is considered part of the first map (i.e., $Z_i \in T_1 \cap T_2$) when $D(Z_i) < d$ (otherwise, $Z_i \notin T_1$), where d is a pre-determined distance threshold.

Nearest Neighbors Ratio (NN Ratio): the NN ratio is a modification of the simple NN search, where the distances of the NN and second-NN are compared. Since NN searches operate directly in the space of the training data T_1, they possess the distinct advantage of not requiring a training phase. To classify a new observation $Z_i \in T_2$, its two NN $Z_{N1}, Z_{N2} \in T_1$ are computed, along with their Euclidean distances $D(Z_{N1})$ and $D(Z_{N2})$. If $D(Z_{N1})/D(Z_{N2}) < d$, where d is a pre-computed ratio threshold, then we can assert that $Z_i \in T_1 \cap T_2$ (otherwise, $Z_i \notin T_1$).

Gaussian Model: this method models the density of the training data by fitting a Gaussian distribution. More specifically, the mean μ and covariance matrix Σ are computed from the occupancy grid map data T_1 to construct a multivariate Gaussian. The number of dimensions of the Gaussian is equal to a. To determine whether a new observation $Z_i \in T_2$ is part of the first map, we evaluate the Gaussian's Probability Density Function (PDF), $\varphi_{\mu,\Sigma}(Z_i)$, and compare the results against a PDF threshold, d. We conclude that $Z_i \in T_1 \cap T_2$ when $\varphi_{\mu,\Sigma}(Z_i) > d$ and that $Z_i \notin T_1$ otherwise.

All these OCC algorithms rely on various parameters as described above and classification results are highly-dependent on them. To determine each of these parameters, we use a process that relies on the cross-validation of the parameters by using a different dataset. Our cross-validation dataset T_O covers a large residential neighborhood spanning close to 2 kilometers and encompassing more than 200 APs. The dataset is equally divided into positive and negative test cases and 50% of the positive test cases are reserved for training with the remaining 50% being allocated, along with 50% of the negative test cases, for classification. Using this data decomposition, we learn each parameter using a brute-force approach, where the parameters are sampled uniformly within their respective logical ranges. We train each OCC algorithm with a parameter sample on the training data of T_O and attempt to classify the data reserved for classification. Since we have ground truth, we can then compute the number of correct classifications. The process is repeated for all parameter samples and the one that yields the highest classification accuracy is retained. This process, which only needs to be performed once, assumes that the dataset T_O is a good representation of the datasets that will be used by OCC algorithms. We have experimentally verified this assumption, but omit the results in this paper due to space restrictions.

We compare the classification accuracy of each algorithm to determine the best one to be used for map overlap estimation function. The results presented here are based on a WiFi map acquired in the same environment used in the map merging results of Section V. The map spans about

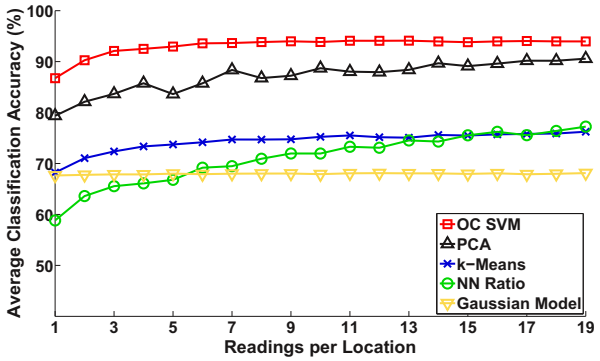


Fig. 3: Classification accuracy for each OCC algorithm, as the number of observations per location is increased. The data presented is the average of 50 experiments.

200 meters of one floor in an office building where 48 APs have been discovered. To investigate the potential effects of the number of observations per location on the algorithms' accuracy, the robot was stopped at pre-determined intervals and a total of 20 observations were recorded for each location. Similarly to the parameter estimator, the entire map is divided in two, with 50% representing positive examples and the rest representing negative examples. Out of the positive examples, the algorithms are trained on 50% of the data so that the remaining 50% is used for classification, along with 50% of the data representing negative examples (i.e., we have the same number of positive and negative examples to classify). Due to the random nature of some OCC algorithms and potential bias towards certain datasets, we randomize the training and classification datasets and run the experiments 50 times.

Figure 3 shows the results of the experiments performed on a consumer desktop with a Core 2 Duo 2.2 GHz CPU as the rest of the experiments presented in this paper. OC SVM clearly separates itself from the other algorithms, with a classification accuracy of over 90% with very few readings per location. Comparatively, OC SVM is on average 6.49%, 20.17%, 23.74%, and 26.99% more accurate than PCA, k-Means, NN Ratio, and the Gaussian Model, respectively. All of the algorithms are extremely fast, taking, in the worst case, 100ms and 20ms to train on the entire dataset and classify 100 observations, respectively.

B. Probability Density Function Estimation

Once the map overlap is estimated, the labels that belong to the overlapping regions of the maps are considered for merging. These identified labels are then matched using the WiFi readings associated to each label. Building upon the discovery that it is possible to differentiate between different locations by only considering WSS readings received from APs [1], we cast the problem of label matching as a classification problem where we compute a function $f : Z_i \in T_2 \rightarrow L_j^1$ from the data in the WiFi map T_1. Function f takes an observation $Z_i = [z_i^1, \ldots, z_i^a]$ tied to L_i^2 in the appearance-based map and returns the matching label in the grid map, L_j^1, from which we can look up

the Cartesian label C_j. Computing f from the training data T_1 can be achieved using a Random Forest (RF), which is shown to provide the best classification results for WSS data [1]. An RF is an ensemble of decision trees built to reduce over fitting behaviors often observed in single decision trees where each decision tree is constructed as a binary tree from a random portion of the training data. In these tree structures, each node corresponds to a decision made on one of the input parameters z^k and leaves correspond to class labels. In order to classify a new observation Z in a decision tree, the parameter z^k is compared at each node, the decision of which dictates which branch is taken - a step that is repeated until a leaf is reached. This procedure is repeated in all trees in the forest resulting in a collection of votes. The label with the most votes can be chosen as the matching label. The number of trees in the RF creates a tradeoff between speed and accuracy. We trained a RF with 250 trees using T_1 in order to accurately classify WSS readings of all overlapping images in the appearance-based map, $Z_i \in T_2$, and determine their matching labels L_j^1 in the grid map.

Algorithm 1 Construct-GMM($Z_i \in T_2$)

1: **for** $j \leftarrow 1$ to c_1 **do**
2: $\mu(j) \leftarrow C_j$
3: $\Sigma(j) \leftarrow \sigma^2 I$ // I: Identity matrix
4: $\phi(j) \leftarrow P(L_j^1|Z_i) \leftarrow$Random-Forest-Predict($Z_i$)
5: **return** $g_i \leftarrow$Build-GMM(μ, Σ, ϕ)

The RF classification not only determines the label with the most votes but, thanks to its voting scheme, also provides the distribution of votes, which can be interpreted as a probability distribution over the set of labels $P(L_j^1 = L_i^2|Z_i)$ where $Z_i \in T_2$. We model this probability distribution function using a Gaussian Mixture Model (GMM) described in Algorithm 1. The GMM is introduced to non-linearly propagate in the two-dimensional Cartesian space the results acquired from the RF. More specifically, we build a GMM comprised of c_1 mixture components (line 1). Each mixture component is constructed from a Gaussian distribution with a mean $\mu(j)$ (line 2), corresponding to the Cartesian coordinates of the cluster center labeled as L_j^1 in the occupancy grid map, covariance $\Sigma(j)$ (line 3), and mixture weights $\phi(j)$ that are acquired directly from the RF's voting scheme (line 4). Line 4 highlights the key difference between the classification and regression methodologies, which arises from the fact that taking the mode of the RF's results, as is done in classification, discards valuable information that is instead exploited in by the GMM. The mixture weights are proportional to the RF's belief of being at location C_j given the observation $Z_i \in T_2$ (i.e., $\phi(j) = P(L_j^1|Z_i)$). The algorithm requires one parameter to be set, σ which dictates how much the Gaussian components influence each other and should be approximately set to the distance between the clusters' centers in the training data, T_1. Given an observation Z_i, the GMM maps a three-dimensional surface to the X-Y Cartesian space represented by the occupancy grid map, where the higher the surface's Z-value the more probable the X-Y location.

C. Edge-Based Refinement

For each image in the appearance-based map, a GMM is generated using RF classification, from which a Cartesian coordinate can be computed. This single-shot localization approach only focuses on the WiFi readings associated with vertices. It can, however, be greatly improved by including the extra information inherently associated with the edges. As described in Section III, edges of appearance-based maps symbolize similarity between connected images. Based on the idea that similar images should be captured from the same area in the environment, the position estimate of an image can be fine-tuned using its neighbors. The power of this local consistency approach is evident when one image is misclassified while the position of all its neighbors are correctly estimated. To address this and similar issues, we create, for each image I_i, an aggregated GMM $\tilde{g}_i$ that combines each of i's 1-hop neighbor's GMMs. The contribution of each neighbor is weighted with the edge's weight that encodes their image similarity. In other words, the more similar the neighbor, the higher its contribution to the aggregated probability distribution. The neighbor's GMM is combined with the GMM of image i and the resulting distribution is normalized.

Algorithm 2 Regression($\tilde{g}_i$, T_1, $Z_i \in T_2$)

1: $nn \leftarrow$k-NN(T_1, Z_i, k)
2: $\hat{C}_i \leftarrow \frac{\sum_{i=1}^{k} nn_i \times \text{PDF}(\tilde{g}_i, nn_i)}{\sum_{i=1}^{k} \text{PDF}(\tilde{g}_i, nn_i)}$
3: **return** $\hat{C}_i$

Algorithm 2 provides pseudo-code for the rest of the regression algorithm. Once the GMM is built (Algorithm 1) and updated using the inputs from the neighbors as described above, a couple of approaches are available, the most popular of which consists in taking the weighted mean of the model or drawing samples from the GMM with probability $\phi(j)$ and computing the samples' weighted mean. However, applying regression directly on the model puts too much emphasis on the RF classification results and does not provide a fail-safe mechanism for misclassifications. Furthermore, our initial assumption that similar images should be localized nearby may not hold for all image pairs, especially when two images are connected together due to perceptual aliasing, which occurs when two different places are visually similar due to shared objects or structural properties of the environment such as repetitive hallways or rooms decorated with the same type of furniture. In such special cases, edge-based GMM updates introduce high probabilities at false locations and the GMM's direct usage causes higher estimation errors. Consequently, instead of sampling from the GMM, our regression algorithm uses a k Nearest Neighbor Search (line 1) to provide k samples that are not only dependent on the observation Z_i, but also come from a different model

than the RF. Given an observation Z_i, the Nearest Neighbor Search returns the Cartesian coordinates of the k most similar observations in T_1. This is a crucial step that adds robustness to the algorithm by combining two different classification algorithms. Where and when one algorithm might fail, the other might succeed. The returned Cartesian coordinate (line 3) is finally calculated as the weighted mean of the k nearest neighbors, where the weight of each nearest neighbor is set from the PDF of the updated GMM (line 2).

While it is evident that including information from correctly localized neighbors improves the accuracy, the contribution of a misclassified neighbor to the aggregated GMM still creates a negative impact on regression results. With the same motivation that each image should not be localized far away from its similar neighbors, we introduce an iterative process that enforces local consistency among the images in the appearance-based map. The refinement process described in Algorithm 3 is based on the idea that any two correctly localized neighbors should lie within a maximum distance r_{max} of each other. To estimate this variable we compute the distances between every pair of images using initial estimations from the regression algorithm and set $r_{max} = \mu + 3\sigma$ where μ and σ are the mean and the standard deviation of the resulting distance distribution, respectively. The iterative algorithm is bootstrapped with the initial positions estimated by applying regression for each image I_i (line 2-4). The GMM $\tilde{g}_i$ associated with each image i in the appearance-based map (line 5) takes into account all of its neighbors (line 6). If any Gaussian mixture component's mean (line 7) is located at a distance of more than r_{max} away from i's estimated position, the mixture component of $\tilde{g}_i$ is removed (line 8). Once all GMMs are pruned, a new set of locations is computed by running the regression algorithm on the updated GMMs (line 2-4). This iterative refinement process continues until convergence (i.e., the mean difference between previous and new location estimations drops below a threshold ε).

Algorithm 3 Refinement

1: **repeat**
2: **for** $i \leftarrow 1$ to c_2 **do**
3: $\hat{C}_i \leftarrow$Regression($\tilde{\mathbf{g}}_\mathbf{i}$, T_1, $Z_i \in T_2$)
4: **for** $i \leftarrow 1$ to c_2 **do**
5: **for all** $I_n \in$ Neighbors(I_i) **do**
6: **for** $j \leftarrow 1$ to c_1 **do**
7: **if** dist($\hat{C}_j$,$\hat{C}_n$) $\geq r_{max}$ **then** $\tilde{\mathbf{g}}_\mathbf{i}(\phi_j) = 0$
8: **until** convergence
9: **return** $\hat{C}$

V. Results

To evaluate the accuracy of proposed map merging algorithm two P3AT robots were used. One was equipped with a LMS200 laser range finder (LRF) and other with a webcam as well as the LRF. The LRF on the second robot, however, is only used to extract ground truth locations of images in the appearance-based map to compute presented localization accuracies. Both explored the first floor of the Engineering building at UC Merced in 3 independent runs covering a total distance of over 1 km. Each robot generated either an appearance-based map using the map building algorithm presented in [4] or an occupancy grid map by employing the SLAM algorithm [8]. Both robots collected WiFi observations Z_i from a total of 75 unique APs ($a = 75$) using their WiFi cards. Appearance-based maps are composed of 234 images in average where a single WiFi reading is attached to each image. In the occupancy grid maps over 1000 WiFi readings per map are partitioned into an average of 229 clusters whose centers' Cartesian coordinates are used as WiFi map labels. Each appearance-based map is merged with each occupancy grid map after the map overlap is computed using OC SVM algorithm with an accuracy of over 99% in average. For the images that are classified as a part of the occupancy grid map, a Cartesian coordinate is assigned as described in Section IV. An illustrative example of merged map with full overlap where all the images are correctly assigned to a position estimate is shown in Figure 4. To account for the randomness associated with RF training process, we merged each map pair 10 times. The average error of all 90 trials is 1.21 meters with $\sigma = 1.09$. The overall heterogeneous map merging algorithm including the map overlap computations takes in average 129 seconds to merge an appearance-based map and an occupancy grid map covering an area of over 300 m^2 with around 250 vertices and clusters, respectively.

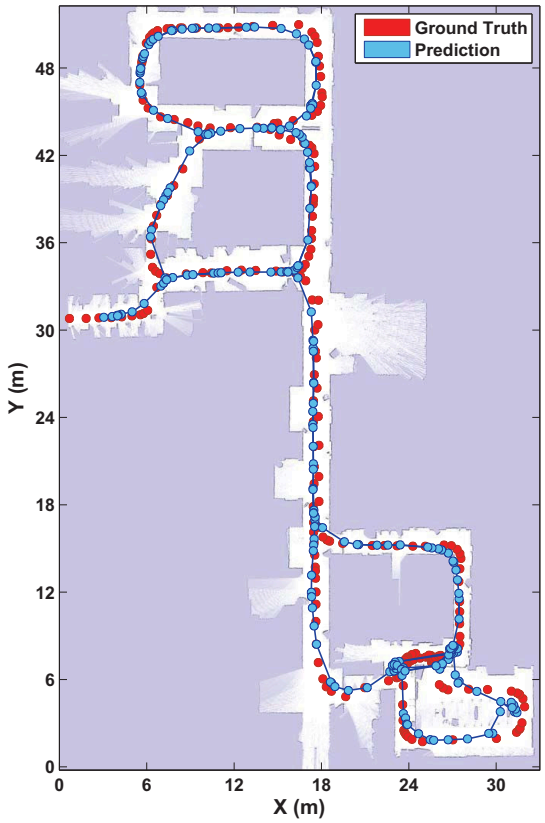


Fig. 4: An appearance-based map is merged with an occupancy grid map after OCC algorithm correctly estimated 100% overlap between two maps.

To demonstrate the true potential of the proposed merging algorithm, we fused an appearance-based map M_A with two smaller non-overlapping occupancy grid maps M_O^1 and M_O^2 created in addition to the ones described above. Two grid maps with an unknown transformation between their coordinate frames provide no additional information when used together. For instance, a robot using these maps cannot navigate from one to another. However, when the appearance-based map is merged with these maps, it creates the needed link between them so that the merged map carries more value either to the operator or the robot utilizing it. This representative example is illustrated in Figure 5 where on the left the appearance-based map is drawn using the graphviz software and grid maps created by navigating in the top and bottom portions of the environment depicted in Figure 4 are presented on the right. Note that with no information on relative transformations between their local coordinate frames, the maps are placed at an arbitrary position while the orientations are set by their corresponding drawing software, gmapping and graphviz. To determine the overlap between maps, OC SVM is trained for each occupancy grid map and each image in M_A is classified as either in-M_O^1, in-M_O^2, or non-matching. The correct classification accuracies for M_O^1 and M_O^2 are 97.54% and 99.01%, respectively.The Cartesian coordinates of images are then computed and they are placed to their estimated locations in their matching grid map while still being connected to their appearance-based map neighbors through their edges. The vertices of M_A shown in green in Figure 5 correspond to images captured in the hallway connecting two grid maps which is not visible in any of them. These vertices are successfully identified as non-matching and constitute a critical part of the merged map by holding all three individual maps connected. The accuracy of localized images is 1.06 meters in average and merging process takes less than 45 seconds. Interested readers should also visit our website[1], where they will have access to the raw data, trained classifiers, generated maps, and code presented in this paper. Moreover, the companion video associated with this paper shows how the described merging process evolves in time.

VI. Conclusions

In this paper we have presented a system that, to the best of our knowledge, is the the first that can successfully merge together heterogeneous spatial models, i.e., occupancy grid maps and appearance based maps. To remedy the lack of a common representation, we use WiFi signals as a common substrate. Our method uses contemporary machine learning algorithms to determine overlap between maps, establish correspondences, and refine the result. Our proposed system has been experimentally evaluated with various maps we produced with our and third party mapping algorithms. In the future, we plan to extend the system to include also topological and feature based maps.

[1]https://robotics.ucmerced.edu/Robotics/IROS2013/MapMerging

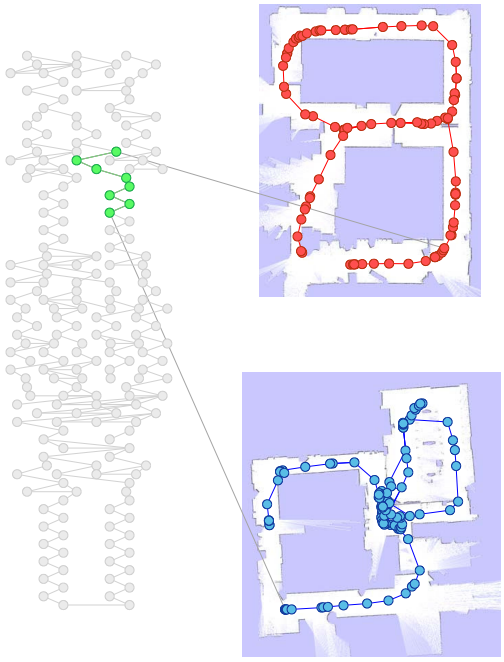


Fig. 5: A representative example showing an appearance-based map (left) merged with two occupancy grid maps (right) is presented.

References

[1] B. Balaguer, G. Erinc, and S. Carpin. Combining classification and regression for WiFi localization of heterogeneous robot teams in unknown environments. In *IROS*, pages 3496–3503, 2012.

[2] A. Birk and S. Carpin. Merging occupancy grids from multiple robots. *Proceedings of the IEEE*, 94(7):1384–1397, 2006.

[3] S. Carpin. Fast and accurate map merging for multi-robot systems. *Autonomous Robots*, 25(3):305–316, 2008.

[4] G. Erinc and S. Carpin. Anytime merging of appearance based maps. In *ICRA*, pages 1656–1662, 2012.

[5] B. Ferris, D. Fox, and N. Lawrence. WiFi-SLAM using Gaussian process latent variable models. In *IJCAI*, pages 2480–2485, 2007.

[6] S. Saeedi Gharahbolagh, L. Paull, M. Trentini, M. Seto, and H. Li. Efficient map merging using a probabilistic generalized voronoi diagram. In *IROS*, pages 4419–4424, 2012.

[7] S. Saeedi Gharahbolagh, L. Paull, M. Trentini, M. Seto, and H. Li. Map merging using Hough peak matching. In *IROS*, pages 4683–4688, 2012.

[8] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2006.

[9] A. Howard, S. Siddiqi, and G. Sukhatme. An experimental study of localization using wireless ethernet. In *International Conference on Field and Service Robotics*, 2003.

[10] W.H. Huang and K.R. Beevers. Topological map merging. *International Journal of Robotics Research*, 24(8):601–613, 2005.

[11] A. Ladd, K. Bekris, A. Rudys, D. Wallach, and L. Kavraki. On the feasibility of using wireless ethernet for indoor localization. *IEEE Transactions on Robotics and Automation*, 20(3):555–559, 2004.

[12] J. Nieto, J.E. Guivant, and E.M. Nebot. The HYbrid Metric Maps (HYMMs): a novel map representation for DenseSLAM. In *ICRA*, pages 391–396, 2004.

[13] L. Schwardt and J. du Preez. Clustering. Technical Report PR414 / PR813 - 2005, University of Stellenbosch, 2003.

[14] D. Tax. *One-class classification; Concept-learning in the absence of counter-examples*. PhD thesis, Delft University of Technology, 2001.

[15] D. Tran and T. Nguyen. Localization in wireless sensor networks based on support vector machines. *IEEE Transactions on Parallel and Distributed Systems*, 19(7):981–994, 2008.

[16] K.W. Wurm, C. Stachniss, and G. Grisetti. Bridging the gap between feature- and grid-based SLAM. *Robotics and Autonomous Systems*, 58(2):140–148, 2010.