Noname manuscript No.
(will be inserted by the editor)

Augmenting Visual SLAM with Wi-Fi Sensing For Indoor Applications

Zakieh S. Hashemifar* · **Charuvahan Adhivarahan*** · **Anand Balakrishnan†** · **Karthik Dantu***

Received: date / Accepted: date

Abstract Recent trends have accelerated the development of spatial applications on mobile devices and robots. These include navigation, augmented reality, human-robot interaction, and others. A key enabling technology for such applications is the understanding of the device's location and the map of the surrounding environment. This generic problem, referred to as Simultaneous Localization and Mapping (SLAM), is an extensively researched topic in robotics. However, visual SLAM algorithms face several challenges including perceptual aliasing and high computational cost. These challenges affect the accuracy, efficiency, and viability of visual SLAM algorithms, especially for long-term SLAM, and their use in resource-constrained mobile devices.

A parallel trend is the ubiquity of Wi-Fi routers for quick Internet access in most urban environments. Most robots and mobile devices are equipped with a Wi-Fi radio as well. We propose a method to utilize Wi-Fi received signal strength to alleviate the challenges faced by visual SLAM algorithms. To demonstrate the utility of this idea, this work makes the following contributions: (i) We propose a generic way to integrate Wi-Fi sensing into visual SLAM algorithms, (ii) We integrate such sensing into three well-known SLAM algorithms, (iii) Using four distinct datasets, we demonstrate the performance of such augmentation in comparison to the original visual algorithms and (iv) We compare our work to Wi-Fi augmented FABMAP algorithm. Overall, we show that our approach can improve the accuracy of visual SLAM algorithms by 11% on average and reduce computation time on average by 15% to 25%.

*
Computer Science and Engineering Department, University at Buffalo, 338 Davis Hall, Buffalo, NY 14222.
Fax: +1716-645-3484
E-mail: {zakiehsa, charuvah, kdantu}@buffalo.edu

†
Computer Science Department, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089.
E-mail: anandbal@usc.edu

1 Introduction

Recent technology trends have enabled the deployment of robots and mobile devices in urban areas for applications such as telecommuting, augmented reality, service robotics, and others. Most such applications require spatial reasoning - identifying the device location as well as recognizing parts of the surrounding environment. In robotics literature, these are referred to as the coupled problems of localization and mapping - jointly called Simultaneous Localization and Mapping (SLAM). SLAM has been extensively researched in the last two decades. Recent trends in sensing have seen the use of regular and depth cameras together (with sensors such as Microsoft Kinect) for 3D mapping. RGBD SLAM (Engelhard et al., 2011), RTAB-Map (Labbé and Michaud, 2013) and ORB-SLAM (Mur-Artal and Tardós, 2017) are more recent examples. Of particular interest to this work is visual SLAM in indoor environments. Algorithms reasoning with RGBD sensors come with some challenges when performing SLAM indoors.

Some of the common problems are:

Perceptual Aliasing (Xia et al., 2016; Nowakowski et al., 2017)**:** Indoor environments tend to be symmetric and repetitive. Corridors with bland walls and repeated patterns of doors and lights could potentially cause confusion between different similar places (wrong loop clo-

sure) resulting in faulty maps and bad localization.
Computational Complexity (Cummins and Newman, 2008; Labbe and Michaud, 2014)**:** Cameras usually produce large volumes of data. For example, MS Kinect has a frame rate of 30 fps and each frame has more than 300000 points including color and depth data. This makes feature detection, matching and loop closure computationally more challenging, especially on resource constrained devices and over long runs.

In a parallel trend, Wi-Fi radio is available on most robots or mobile devices and Wi-Fi Access Points are ubiquitous in most urban environments. Wi-Fi and visual sensing are complementary to each other. While Wi-Fi sensing is less reliable than visual sensing, it is immune to perceptual aliasing. The degree of detail in Wi-Fi data is much less than visual data and therefore requires much less processing time. In this work, we present a generic workflow to incorporate Wi-Fi sensing into visual SLAM algorithms in order to alleviate perceptual aliasing and high computational complexity. The contributions of this work are as follows:

- We propose a general way to integrate Wi-Fi sensing with visual SLAM by using received signal strength as an indicator of coarse spatial locality. Unlike many other methods, our integration works in tandem with the visual SLAM operation without any requirement of prior Wi-Fi data collection phase.
- We instrument three separate open-source visual SLAM systems (RGBD-SLAM, RTAB-Map, and ORB-SLAM) using our proposed technique to show the generality of our method.
- We run our algorithm on four datasets from four different buildings to experimentally demonstrate the benefits of augmenting the three SLAM systems with Wi-Fi sensing on these four datasets.
- We compare our work with the most recent state-of-the-art in WiFi-augmented visual sensing work which is Wi-Fi augmented FABMAP (Nowakowski et al., 2017).

2 Related work

There has been a lot of research on SLAM. We will present representative work.

Visual SLAM: Prior work on SLAM has been done with multiple sensors including RGB and RGBD cameras, 2D and 3D LiDARs, 2D and 3D sonar sensors (Engelhard et al., 2011; Hess et al., 2016). A recent trend has been the use of color images with depth images. Some of more well-known visual SLAM examples include RGBD SLAM (Engelhard et al., 2011), RTAB-Map (Labbé and Michaud, 2013), and ORB-SLAM (Mur-Artal and Tardós, 2017). Since we instrument these algorithms, they will be discussed in detail in Section 5.

Wi-Fi Localization/SLAM: In robotics literature, there has been research on Wi-Fi localization (Biswas and Veloso, 2010; Ito et al., 2014) as well as Wi-Fi SLAM (Yang et al., 2014; Huang et al., 2011; Nguyen et al., 2016). Similarly, wireless localization is a hot topic in mobile computing. (Yang et al., 2012; Luo et al., 2014) use wireless fingerprinting for localization. SpotFi (Kotaru et al., 2015), Monoloco (Soltanaghaei et al., 2018) and (Karanam et al., 2018) use channel state information (CSI) to achieve decimeter-level localization. (Kumar et al., 2018) employ CSI and inertial measurements for delivering more accurate localization. In these literature, RSSI or CSI is employed either for localization of the robot or for SLAM where the landmarks are Wi-Fi Access Points (APs). We also note that most of them require a training phase for the collection of Wi-Fi data. In our work, we improve visual 3D maps rather than mapping just the APs and our method does not require a training phase.

RF-assisted positioning systems: Other RF-assisted localization and mapping systems have been proposed in literature using geomagnetism (Jung et al., 2015; Wang et al., 2016), electromagnetism (Lu et al., 2018) and visible lights (Zhang and Zhang, 2016; Kuo et al., 2014). In this work, we chose to use Wi-Fi signals from commodity wireless cards to improve visual SLAM due to its ubiquity in indoor environments.

Perceptual Aliasing: (Heshmat et al., 2013) incorporates a hardware solution to enhance feature measurements and localization by adding lateral motion to the camera. (Codd-Downey and Jenkin, 2017; García et al., 2016) combine information from multiple sensors for increasing localization and mapping accuracy. (Belter et al., 2016) utilizes spatial uncertainties caused by actual measurements and image processing for better tracking. Some approaches take into account different kinds of spatial information of image features in order to alleviate perceptual aliasing (Kejriwal et al., 2016). Recently, deep learning solutions are used for extracting better image descriptors which leads to more accurate place recognition (Xia et al., 2016).

Computational Complexity: Real-time performance is desirable in SLAM in some applications. (Labbe and Michaud, 2014) and (Labbé and Michaud, 2011) incorporate memory management techniques to isolate a small active portion of the map to perform loop-closures quicker and ensure sustained online operation. (Cummins and Newman, 2008) enables rapid multi-hypothesis labeling in appearance-only SLAM using some probabilistic bail-out condition.

Visual and Wi-Fi Integration: Recent research has gravitated towards fusing alternate sensors, especially Wi-Fi, with visual sensing for improvements in localization accuracy. In (Ito et al., 2014), they model WiFi signal strength using a Gaussian process and use it for finding an initial seed estimate of the robot's location which is then refined with RGBD data. (Quigley et al., 2010) utilizes a training phase for Wi-Fi modeling and then applies particle filters for fusing different sensors. (Nowicki, 2014; Dong et al., 2015) employ a mapping phase for collecting Wi-Fi signatures and visual images and utilize Wi-Fi data in localization phase for more accurate place recognition. (Clark et al., 2016) uses Wi-Fi sensing along with other sensing modalities for more accurate localization estimates in less time. They incorporate sensing information directly in the map and use Monte Carlo estimator policy to allow point-matching only in nodes which is more probable to succeed. While these research works take advantage of Wi-Fi data along with visual sensing for more accurate *localization*, none of them do SLAM and all of them employ an initial phase of Wi-Fi data collection or training which is later utilized for localization.

Closest to our work are (Berkvens et al., 2014; Jacobson et al., 2015; Nowakowski et al., 2017). In (Berkvens et al., 2014), authors incorporate Wi-Fi sensing only when visual data is no longer applicable. Our proposal is different in that it actively uses Wi-Fi data along with visual data and is generic making it applicable any visual SLAM algorithm. In (Jacobson et al., 2015), the authors propose a voting based metric to fuse sensor information to find loop closures whereas our approach is different in that the visual matches are allowed only if Wi-Fi based matches are made to improve runtime efficiency. FABMAP is augmented in (Nowakowski et al., 2017) by tagging images with Wi-Fi vectors indicating the presence of APs. Apart from FABMAP being a topological SLAM, we use vectors of RSSI values instead of binary vectors. In section 6, we compare our approach with Wi-Fi augmented FABMAP in more detail.

Overall, we'd also like to note that none of these works provide a general way to integrate Wi-Fi sensing with SLAM as our work does. This allows researchers to integrate wireless sensing into future SLAM algorithms as well as current ones.

3 Wi-Fi sensing

Wi-Fi has become common in our lives and has enabled the mobility of our computing. Wireless sensing is a popular topic of research in mobile and sensor systems communities as well. Typical sensing includes received signal strength (RSSI). Some modern Wi-Fi cards have the capability to calculate the wireless channel properties such as Channel State Information (CSI). We chose to use Received Signal Strength (RSS) rather than CSI due to the following reasons: (a) CSI values were not available with APs at all locations that we measured, (b) Our preliminary CSI measurements did not demonstrate the accuracy reported by works such as SpotFI (Kotaru et al., 2015), and (c) CSI empirically seems much more sensitive to small dynamics in the environment which affects robustness in our methodology.

3.1 Wi-Fi Similarity using Received Signal Strength

As per the IEEE 802.11 standard, all Access Points (APs) constantly broadcast beacon frames that advertise their existence for potential clients. Additionally, all clients calculate the Received Signal Strength Indicator (RSSI) value for each AP that is visible to the client. RSSI is affected by many factors including distance, obstacles, and interference. Furthermore, each AP has an identifier called the Basic Service Set Identifier (BSSID), a value that must be unique to it for the functioning of any Wi-Fi network. BSSIDs can be read along with the RSSI values.

In modern urban environments, it is not uncommon to see fifteen to thirty BSSIDs at any given indoor location. This large number is due to different APs catering to different populations and providing access control to the network. On the robot/mobile device, RSSI values from multiple APs could be collected and used to construct a vector of RSSI values to form a *Wi-Fi signature*. Henceforth, in this paper, we use the term *Wi-Fi signature* and *signature* interchangeably. Such a vector is typically different at different locations that are sufficiently apart due to the fact that APs are usually spread out in space to maximize efficiency and connectivity. While such Wi-Fi signatures have been used previously either as a lone sensor or combined with other sensing modalities for localization and/or mapping, our intent is to use them in an online fashion to augment existing SLAM algorithms for improved localization/mapping accuracy.

To understand whether the similarity values are different at different locations, we collected Wi-Fi signatures at different points on a trajectory that starts to move away and return to the originating point. Specifically, we use Cosine Similarity between vectors as a measure of Wi-Fi similarity. Shown in Figure 1 are aggregates of similarities in Wi-Fi vector space compared to the physical distance between locations from a building at our university. The trend clearly shows a coarse

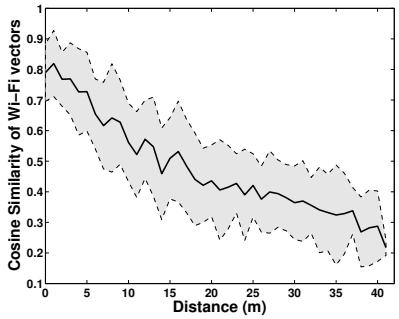


Fig. 1: Aggregate behavior of Wi-Fi Cosine Similarity against spatial distance as measured from various APs in B Hall. Results from other buildings were comparable and showed a similar trend

inverse correlation between physical distance and Wi-Fi similarity. This is expected since several models, including the ITU indoor propagation model, use this relation to find distance from RSSI values.

3.2 Wi-Fi data processing

Wi-Fi BSSID Dynamics: An observation in RSSI aggregation is that each unique Access Point(AP) advertises different BSSIDs for different frequencies, 2.4GHz and 5GHz, and different access control mechanisms. These BSSIDs are only different in the last nibble of their MAC addresses. We aggregate all measurements from an AP by averaging the signal strength measured for all BSSIDs from a single AP.

Wi-Fi Data Collection: The nature of Wi-Fi cards requires us to be stationary at a given location to collect steady signal strength readings. Therefore, during our measurements, our robot pauses for about ten seconds every few meters to collect Wi-Fi signals.

Wi-Fi Similarity Metric: The similarity between two different RSSI vectors v and w equals $Similarity = \frac{v \cdot w}{|v||w|}$. We use the cosine similarity measure in this work because it is invariant to scale. It is less sensitive to RSSI fluctuations due to configuration changes. If some APs are unavailable at some respective position, we set the corresponding values to zero.

4 General approach

In Section 3, we described Wi-Fi similarity as a measure that correlates with coarse spatial locality. Our intent is to use this measure to improve visual SLAM. Figure 2 shows the block diagram of our general proposed approach to incorporate Wi-Fi sensing into any

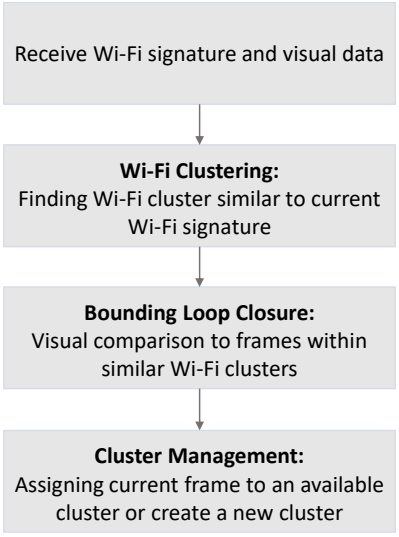


Fig. 2: General approach to incorporate Wi-Fi into any visual SLAM algorithm.

visual SLAM algorithm. Following, we discuss each module in detail.

Wi-Fi and Visual Association: The first step is to associate a visual frame (image) with a corresponding Wi-Fi signature. For every 3 or 4 meters, the robot or mobile device pauses for about 10 seconds for Wi-Fi signature aggregation. Then, it associates any visual frame that follows with this Wi-Fi signature until the next pause.

Wi-Fi Clustering: Each Wi-Fi cluster represents a spatially separate region in the environment and has a representative Wi-Fi signature. It includes those frames which their signatures are similar to the representative signature and have at least one visual edge to another frame within the same cluster. A visual edge represents a visual transformation between correspondent visual frames calculated through alignment of their keypoints. In this module, we compute the cosine similarity between the Wi-Fi signature of the current frame and the representative signatures of all available Wi-Fi clusters within the database to find similar clusters. Any cluster within a threshold level of similarity is considered similar. These similar clusters represent spatial proximity to the current frame.

Bounding Loop Closure Search: A major challenge in SLAM is the problem of identifying a previously visited place. For example, if we go in a loop along the corridors of a building, we need to be able to recognize that we are back at the starting point once we complete the loop. This problem is called *Loop Closure*. As the map grows, SLAM algorithms accumulate many frames and it becomes computationally intensive to check for loop closures. Reducing the search space greatly

benefits the timely working of a SLAM system. In this module, we reduce the search space by comparing the current frame only to frames within similar clusters. We do this to emulate comparison to frames from close-by regions.

Cluster Management: After permissible visual comparisons, the next step is to assign the current frame to the "correct" cluster. If any visual edge is added between the current frame and any frame within similar clusters, the current frame is assigned to that cluster. If there are multiple such clusters, the one with the highest cosine similarity is chosen. If no valid visual edges or similar clusters are found, a new Wi-Fi cluster is created and the Wi-Fi signature of the current frame is assigned as the representative signature of that cluster.

In the beginning and upon receiving the first visual frame and Wi-Fi signature, we create the first cluster and assign the Wi-Fi data as its representative Wi-Fi signature. When a new visual frame along with its correspondent Wi-Fi data is received, we compute the cosine similarity between its Wi-Fi signature and the representative Wi-Fi signatures of all available clusters in the *Wi-Fi Clustering module to find similar clusters. Similar clusters are the one which their cosine similarity is higher than a certain threshold. In the next step in Bounding Loop Closure Search, the new visual frame is compared against the visual frames within similar clusters to find acceptable visual transformations. If a visual transformation is accepted between the current frame and any frame within one of similar clusters, the current frame will be assigned to that cluster. Upon having more than one such clusters, the one with highest Wi-Fi similarity is selected. If no such visual transformation is available, a new cluster will be created and the current Wi-Fi signature would be assigned as its representative Wi-Fi signature. This process is executed for all frames.*

5 Augmenting SLAM with Wi-Fi Sensing

To demonstrate the utility of augmenting visual SLAM with Wi-Fi, we instrument three well-known SLAM algorithms with Wi-Fi sensing. For this, we chose RGBD SLAM (Engelhard et al., 2011), RTAB-Map (Labbé and Michaud, 2013) and ORB-SLAM (Mur-Artal and Tardós, 2017). All of them have open-source implementations making it convenient to modify them. In RGBD SLAM and ORB-SLAM, no odometry information is used which makes them easy to use on wearable devices as well as robots. We now describe each instrumentation in detail. For this, we first describe the original SLAM system followed by a description of our augmentation.

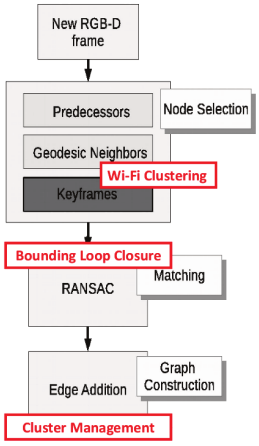


Fig. 3: Control Flow of RGBD SLAM for each new RGBD frame. Our augmentation of Wi-Fi sensing is shown in red

5.1 RGBD SLAM

5.1.1 ***Background***

RGBD SLAM (Engelhard et al., 2011) is a graph-based visual SLAM where nodes correspond to RGBD frames and edges correspond to 3D visual transformations between them. Also, any frame with unique visual features constitutes a keyframe. RGBD SLAM represents an early SLAM system built for RGB-D sensors. It is somewhat brittle and computationally heavy as shown in our results.

Figure 3 shows a block diagram of RGBD SLAM. We also show how Wi-Fi sensing is augmented in red using the described modules in general proposed approach. In RGBD SLAM, each new frame is compared to a subset of previous frames for motion estimation and place recognition (Node Selection in Figure 3). (a) *Predecessor Nodes*: A fixed number of nodes prior to the current node, (b) *Geodesic Neighbors*: A fixed depth of the graph before the current node, and (c) *Keyframes*: A randomized sub-set of previous keyframes. Intuitively, the predecessor nodes and geodesic nodes are used for motion estimation and the random sub-set of keyframes are for identifying long-term loop closures. Ideally, each frame should be compared with all relevant keyframes for best results. However, this is not tractable since the number of keyframes increases as the map grows. Thus, the keyframe selection is limited to a constant random number to reduce the computational complexity of this step as the map grows. This results in lower probability of finding correct long-term loop closures over time as the size of the graph increases.

5.1.2 *Wi-Fi Augmentation*

We intend to improve long-term loop closure (avoiding perceptual aliasing) by incorporating Wi-Fi sensing. As discussed earlier, comparing all keyframes to the current frame results in huge computational overhead. It would be ideal to select a subgraph of the existing map that corresponds to the places close (distance-wise) to current frame for loop closure. This should improve the accuracy of loop closure detection along with runtime reduction.

To improve the selection of related keyframes, we apply our proposed approach.

- **Wi-Fi Clustering:** We divide the RGBD keyframes into different clusters based on their Wi-Fi signature. For each new RGBD frame, we compute the cosine similarity between its signature and the representative signatures of all clusters to find similar clusters.
- **Bounding Loop Closure Search:** We compare the current frame only to RGBD keyframes within similar clusters for visual transformation calculation. This approach limits the number of keyframes to be compared against for loop closure detection.
- **Cluster Management:** If the current frame is identified as an RGBD keyframe, this module is activated for assigning it to the right cluster as discussed in 4.

Keyframe clustering based on the similarity between Wi-Fi signatures allows us to select a subset of RGBD keyframes that correspond to the similar location range as the current frame for loop closure detection. Also, due to low computation overhead of determining Wi-Fi similarity, we can compute similarities between the current Wi-Fi signature and *all* Wi-Fi clusters quickly, which is very beneficial in identifying keyframes from close-by regions from the whole map instead of picking random keyframes like the original method. This allows us to not only improve the loop closure accuracy but also decrease computational complexity.

5.2 RTAB-Map

5.2.1 *Background*

RTAB-Map (Labbé and Michaud, 2013) is a real-time graph-based method similar to RGBD SLAM in structure, but it uses odometry. This makes it particularly suitable for robots or wearable devices with inertial sensing and low mobility. The main difference between RTAB-Map and RGBD SLAM is the way memory is managed and loop closure detection is handled.

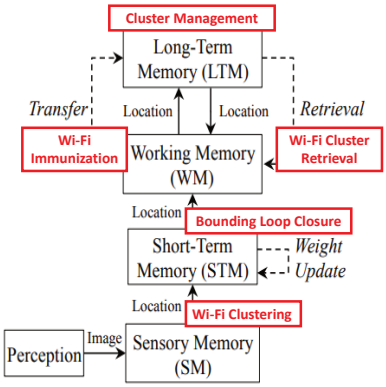


Fig. 4: Block diagram of the workflow of RTAB-Map for each new RGBD frame (Labbé and Michaud, 2013) along with our Wi-Fi sensing augmentation in red.

In this algorithm, three types of memories are defined; Short Term Memory(STM), Working Memory(WM) and Long-Term Memory. (a) STM is for a fixed number of the most recently added nodes , (b) WM includes the nodes which are candidates for comparison for loop closure detection. Every node is transferred from STM to WM after a while, and (c) LTM is for the nodes which will not be used for any purpose anytime soon. Figure 4 shows the different memories and their interactions along with segments with our augmentation in red. We have built two additional modules called *Wi-Fi Immunization* and *Wi-Fi Cluster Retrieval* which are specific to this algorithm apart from those in our general approach.

One of the parameters that a user can control in the RTAB-Map system is `real_time threshold`. This corresponds to the time that is considered acceptable for the processing of a new frame. During the execution, whenever the processing time of the current node exceeds the specified `real_time threshold`, some nodes are transferred from WM to LTM. Selection of nodes for transfer is done based on many criteria such as nodes that are not graph-wise close (i) to the current node and (ii) to nodes which have high visual similarity with the current node. While these conditions are reasonable, there is still a fair chance of losing relevant candidates, especially for long-term loop closure as discussed in the original paper (Labbé and Michaud, 2013).

5.2.2 *Wi-Fi Augmentation*

RTAB-Map trades off localization and mapping accuracy for computational efficiency by moving frames from working memory (WM) to long-term memory (LTM). This process increases the probability of missing a loop closure due to unavailability of related frames in WM.

We intend to improve the choice of frames to transfer to LTM using Wi-Fi sensing.

- **Wi-Fi Clustering:** Upon arrival of a new frame, we compute the cosine similarity between the new Wi-Fi signature and all Wi-Fi clusters within memory in order to find similar clusters.
- **Wi-Fi Immunization:** If the RGBD frames of similar clusters are in WM, they are marked not to be moved to LTM.
- **Wi-Fi Cluster Retrieval:** If the RGBD frames of similar clusters are in LTM, they are retrieved back to WM and marked not to be moved to LTM.
- **Bounding Loop Closure Search:** Visual transformation calculation happens between the current frame and the frames within similar clusters.
- **Cluster Management:** The current frame is assigned to the right Wi-Fi cluster as discussed in 4.

5.3 ORB-SLAM

5.3.1 *Background*

ORB-SLAM is a recent graph-based visual SLAM algorithm similar to RGBD SLAM in structure. A primary contribution of this algorithm is the construction of a dictionary relating visual words to keyframes which have observed them. This dictionary helps in quick lookup of similar keyframes (ones with similar visual words) for comparison to current keyframe for long-term loop closure. In this manner, each new keyframe is compared to the keyframes which have at least one visual word in common with it. Another contribution is the definition of co-visible keyframes which identifies keyframes sharing map points. Figure 5 shows the block diagram of ORB-SLAM along with segments where Wi-Fi modules are incorporated in red. Although the dictionary lookup approach increases the probability of accurate detection of positive loop closures, it still suffers from perceptual aliasing in symmetric environments.

5.3.2 *Wi-Fi Augmentation*

In ORB-SLAM, there is a visual word dictionary which maps visual words to RGBD keyframes which have observed them. For loop closure detection, each RGBD keyframe is compared to any keyframe which has at least one common visual word with it. While this approach probably would be able to find any correct loop closure, it is not immune to perceptual aliasing. Incorporating Wi-Fi sensing could alleviate this problem. Here is how we augment ORB SLAM with Wi-Fi sensing.

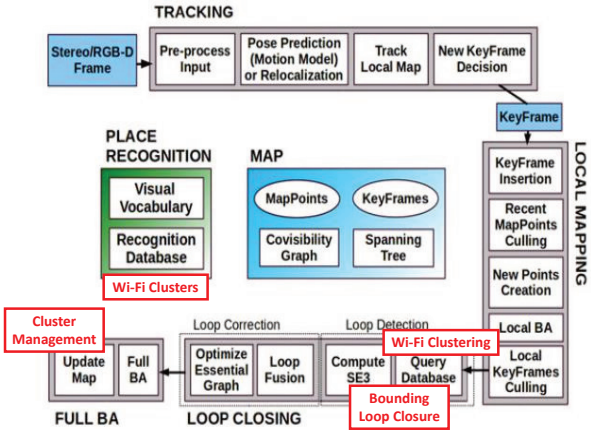


Fig. 5: Control flow of ORB-SLAM for each new RGBD frame (Mur-Artal and Tardós, 2017) along with our incorporation of Wi-Fi sensing in red.

- **Wi-Fi Clustering:** We compute the cosine similarity between the Wi-Fi signature of new RGBD keyframe to all available Wi-Fi clusters to find similar clusters.
- **Bounding Loop Closure Search:** We only use the visual word dictionaries of similar clusters for finding RGBD keyframes which have visual words in common with the current frame. This would significantly lower the number of candidates for comparison and reduce the chances of perceptual aliasing.
- **Cluster Management:** If a valid visual edge is constructed or there are any co-visible keyframes within similar clusters, the current frame is assigned to the corresponding cluster. Otherwise, a new Wi-Fi cluster with a separate visual word dictionary is created.

We intend to open-source all implementations that we have augmented with Wi-Fi sensing on the publication of this work. We hope that this will help the community refine integrating Wi-Fi sensing and visual sensing. We will now evaluate the performance of each of our Wi-Fi augmented SLAM systems and compare them to the original approaches.

6 Evaluation

We now evaluate the computational complexity, localization and mapping accuracy of our Wi-Fi augmented SLAM algorithms. Section 6.1 describes the metrics used for evaluation in detail. In section 6.2, we provide information about our setup for data collection and the collected datasets. Finally, we describe our results for all the datasets.

6.1 Metrics to evaluate SLAM performance

Here are some metrics that indicate SLAM performance:
False Positive/Negative in Computing Loop Closure: Identifying incorrect loop closures (false positive) and missing correct ones (false negative), especially in long-term, could have a huge effect on the accuracy of the constructed map. Based on ground truth knowledge of our datasets, we count the false positive and false negative loop closures for each SLAM algorithm. For this purpose, we find all loop closures that any well-constructed map has detected. Then any extra loop closure is counted as false positive and any missing one is counted as false negative.
Error in Estimated Trajectory: As described below, we record ground truth for our datasets. To measure error, we use the Kabsch algorithm (Kabsch, 1976) to align the trajectory generated by the SLAM algorithm with the ground truth as they are not in the same coordinate frames. Then we calculate the RMS error between corresponding poses of ground truth trajectory and estimated trajectories using the SLAM algorithm.
Computation Time: A principal challenge for SLAM approaches is the amount of time required to process the data. For all three algorithms, we have measured the difference in computation time between the original algorithm and our Wi-Fi augmented version. More specifically, we micro-benchmark the difference in computation times for individual steps from our proposed approach. This includes the reduction in computation time due to *Bounding Loop Closure* and the overhead resulting from *Wi-Fi Clustering* and *Cluster Management*.

6.2 Datasets

Datasets with Wi-Fi measurements alongside RGBD measurements are not readily available. Hence, we collected four different datasets from four different buildings at our university.

For data collection, we used a Turtlebot[1] mounted with a Kinect 360 and a Velodyne VLP-16 LiDAR [2]. The Kinect 360 provides RGB-D data with RGB images of 640X480 resolution at 30 frames per second and depth images of 320X240 resolution at 30 frames per second. Its depth range goes from 1m to 4m approximately. The LiDAR provides 300,000 points per second with a 360°horizontal field of view and ± 15°vertical field of view. It has a depth range of over 100m. In our datasets, the LiDAR is used for ground truth trajectory estimation. For this purpose, we used *Google Cartographer* (Hess et al., 2016) for 2D ground truth trajectory estimation at a 5 cm resolution. All data is collected using ROS[3] and a laptop on the Turtlebot during execution. We use Intel 7265 wireless card for our Wi-Fi measurements.

We collected four separate datasets from four buildings on campus.
C Hall: This dataset was collected during traversal of a medium-sized square loop with corridors that are 20 meters long. There are 24000 images, less than 40 APs and a total of 7 Wi-Fi clusters in this dataset.
B Hall: This dataset includes one long and one short loop that links together to look like the number 8. There are 28000 images, about 40 APs and 13 Wi-Fi clusters in this dataset.
J Hall: This dataset includes one long loop and an adjoining trajectory which together looks like the number 9. There are a total of 19 Wi-Fi clusters, around 70 APs and 33000 images in this dataset.
A Hall: This dataset is one loop of a long jogging track with sparse visual features. There are not many blocking walls between different places on the trajectory. The number of frames, APs and Wi-Fi clusters are 50000, 45 and 8 respectively.

6.3 RGBD SLAM performance

As we ran the RGBD SLAM with default parameter settings on *C Hall* dataset, we got very inaccurate maps and trajectories. So, we decided to perform parameter tuning and find the best possible outcomes of vanilla RGBD SLAM based on our metrics defined above. The first parameter is the minimum number of matched features required for accepting a transformation which is called *min-matches*. The second parameter is the maximum distance allowed for inlier points when using RANSAC for transformation estimation and is called *inlier-distance*.

We pick three different reasonable values for inlier-distance and four values for min-matches and run vanilla RGBD SLAM on each set of parameters for *C Hall Dataset*.

From Figure 6, increasing min-matches and decreasing inlier-distance decreases the number of false positives as expected. The computation time decreases as the number of min-matches increase and inlier-distance decreases. This is a direct result of the reduction in permissible transformations between frames. There is a lack of structure in the way trajectory error varies with these parameters. Our conjecture is that this is because of the randomness in node selection as well

[1] `https://www.turtlebot.com/turtlebot2/`
[2] `http://velodynelidar.com/vlp-16.html`
[3] `http://wiki.ros.org/`

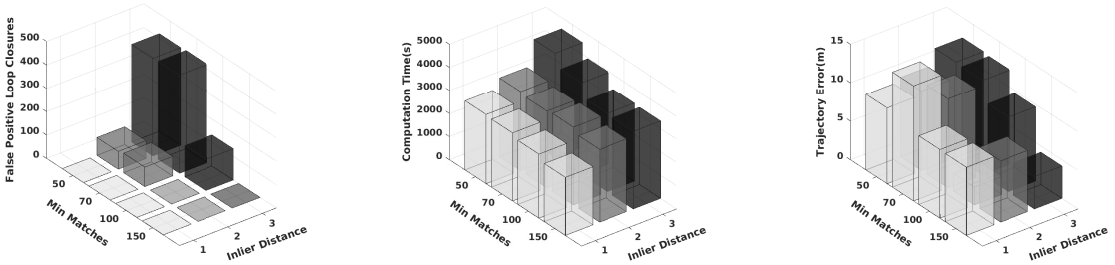


Fig. 6: False positive loop closures (left), computation time (center) and trajectory error (right) when the parameters *min-matches* and *inlier-distance* are varied.

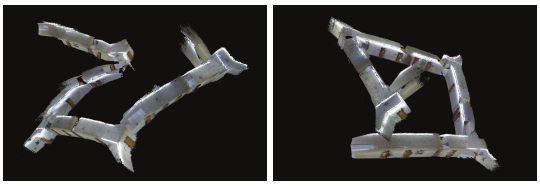


Fig. 7: Constructed maps of vanilla RGBD SLAM with *min-match*=100/150 and *inlier-distance*=2. Both suffer from rotational deviations due to low number of visual transformations

as the effect of associated false positive/negative loop closures.

Setting min-match to very high values and inlier-distance to very low values cause the graph to have a very low number of permissible visual transformations. Potentially too low to even create a reasonable map. Figure 7 shows the constructed maps with min-match = 100/150 and inlier-distance=2 which suffer from rotational deviations due to a very low number of visual transformations. Therefore, we have picked lower values for min-match and higher values for inlier-distance to provide parameters that give reasonable performance with respect to the number of visual transformations. Because Wi-Fi sensing is not able to force a visual transformation if it is rejected due to a very high value of min-match or very low value of inlier-distance.

Figure 8 compares the original method and our proposed method for the C Hall dataset. Results show that our proposed method gets rid of all false loop closures for both settings.We believe this is because Wi-Fi similarity allows us to identify a good sub-set of keyframes to compare against for accurate loop closure as opposed to the fixed set of random keyframes selected by vanilla RGBD SLAM. Similarly, we reduce the computation time of affected modules by bounding loop closure by more than 50% for min-match=50 and more than 30% for min-match=70. This is because Wi-Fi similarity allows us to eliminate unnecessary comparisons. Moreover, having a smoother map reduces the optimization time as well. Finally, we also observe a reduction in RMS error of 80% for min-matches $\in (50, 70)$. This is also a result of identifying a good sub-set of keyframes for loop closure comparison and not allowing visual transformations between frames having distant Wi-Fi signatures.

Based on the shown result, it seems that the performance of RGBD SLAM in symmetric environments with a low number of features is very poor. So we decided not to continue running this algorithm on longer and harder datasets in order to avoid unreasonable comparisons.

6.4 RTAB-Map performance

RTAB-Map is another state-of-the-art SLAM system. A key parameter in RTAB-Map is the ability to control the running time of the algorithm by setting real-time threshold parameter. The algorithm tries to keep the processing time of each node under real-time threshold by moving unused frames to LTM only to retrieving them back when necessary. In our experiments, we set different real-time threshold values and compare the behavior of vanilla RTAB-Map with our proposed approach for different datasets. We note that real-time threshold=∞ means no threshold is set and no frame is moved to LTM.

Table 1 represents the trajectory error of both original and Wi-Fi augmented RTAB-Map for different datasets and real-time thresholds. For real-time thresholds of ∞ and 200, the trajectory error of both approaches is on the same order. This is because of the number of frames moved to LTM. For a value of 0, no frame is moved to LTM. For a value of 200, the number of transferred frames is too low, because the processing time of most of the frames is less than this threshold. Some of the error differences for these two real-time threshold values are due to the randomness of matched frames.

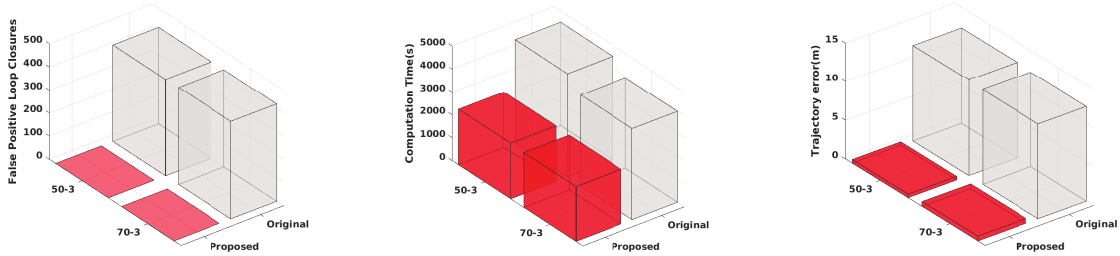


Fig. 8: Accuracy comparison between vanilla RGBD SLAM and our proposed method.

Table 1: **RTAB-Map:** Trajectory error (m) for different datasets and different *real-time thresholds*

Real-time Threshold (ms)								
	∞		70		100		200	
Dataset	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi
A Hall	0.23	0.22	0.48	0.25	0.48	0.28	0.23	0.20
B Hall	0.24	0.22	0.12	0.22	0.12	0.22	0.19	0.22
C Hall	0.12	0.11	0.11	0.10	0.11	0.10	0.11	0.14
J Hall	0.30	0.21	1.34	0.19	1.34	0.19	0.23	0.21

Table 2: **RTAB-Map:** False negative loop closures for different datasets and different *real-time thresholds*

Real-time Threshold (ms)								
	∞		70		100		200	
Dataset	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi
A Hall	0.00	4.44	100.00	17.77	100.00	31.11	11.11	13.33
B Hall	8.00	16.00	100.00	0.00	100.00	0.00	6.00	12.00
C Hall	6.06	0.00	100.00	9.09	100.00	9.09	27.27	3.03
J Hall	9.09	9.09	100.00	13.63	100.00	40.90	0.00	9.09

Fig. 9: Constructed maps of J Hall (top row) using original RTAB-Map (left) and Wi-Fi augmented RTAB-Map (right). They show the difference in constructed map without loop closure in original approach (left) and with correct loop closures in Wi-Fi RTAB-Map (right)

Among a set of consecutive frames, selecting either one for loop closure may result in a slightly different visual transformation. For real-time thresholds of 70 and 100, the original approach is not able to detect any loop closure, because all the related nodes are transferred to LTM. But in our approach, we are able to retrieve back the transferred nodes to WM using Wi-Fi sensing. So we are able to make correct loop closures and experience a much smaller trajectory error, especially in A Hall and J Hall. These two datasets are very large and odometry measurements are subject to noise accumulation. Therefore, a correct loop closure benefits the localization accuracy to a high extent. Figure 9 presents the constructed maps of both approaches for J Hall in real-time threshold=70. For B Hall, the results show a lower error for vanilla RTAB-Map. We believe this stems from the small size of the dataset. Because in vanilla RTAB-Map, the trajectory error of B hall with no loop closure in real-time threshold of 70 and 100 is less than the case with correct detected loop closures with real-time threshold=∞ when no frame is trans-

ferred to LTM. In such an environment, the odometry may be able to provide a more accurate trajectory than visual estimations.

Table 2 shows the percentage of false negative loop closures of each approach for different real-time threshold values. For real-time threshold=200 and real-time threshold=∞, the percentage of false negatives of both approaches is less than 20. This is due to infrequent or no frame transfers to LTM. Percentage values higher than zero in real-time threshold=∞, where no frames are transferred to LTM, show that several loop closures are missed due to the random absence of some visual features in multiple frames. As shown, all the possible correct loop closures are missing in the original approach for real-time thresholds of 70 and 100 due to unavailability of corresponding frames in WM. Two conditions lead to non-zero percentage values in our approach: 1- Random absence of visual features in some frames, 2- Availability of frames in WM; different pools of frames in WM generate different loop closure candidates. But based on results in Table 1, low percentage of false negatives doesn't affect the trajectory accuracy. We don't see any specific reason for higher percentage values of A and J Hall datasets for real-time threshold of 100 except the above conditions. Since there are no false positive loop closures detected in either approach in RTAB-Map, no results are shown.

In Table 3, we show the computation time of loop closure detection for both approaches and all datasets. Bounding loop closure is supposed to save computation time in this process by restricting the number of comparisons. The results confirm that almost for all datasets and *real-time thresholds*, we are spending much less time for loop closure detection. The only different cases are for very long A dataset for *real-time thresholds* of 70 and 100. This is due to the size and shape of the environment. The A Hall dataset is very large and there are not many blocking walls along the trajectory which causes less RSSI attenuation between different places. This leads to less number of Wi-Fi clusters. As previously mentioned, there are only 8 Wi-Fi clusters created in this very large environment while the same number for smaller J and B datasets are 19 and 13. A low number of Wi-Fi clusters in a large environment lead to a very high number of keyframes in each cluster. In this situation, visual comparison to even one single similar cluster takes more time while the original approach is transferring many nodes to LTM due to the low *real-time threshold* and does a very low number of visual comparisons.

It should be noted that if we increase the Wi-Fi similarity threshold, the number of clusters would increase which may help in reducing the loop closure computation time. But it would also decrease the number of detected similar clusters for each frame which may lead to loosing some correct loop closures.

Table 4 represents the computation overhead of our approach for all datasets caused by *Wi-Fi Clustering* and *Cluster Management* modules. As shown, these overheads are very small and could be ignored with respect to saved computation time shown in Table 3. Based on the shown result, the amount of computation overhead is dependent on both the number of Wi-Fi clusters and the size of the dataset (number of frames). Higher number of clusters and higher number of frames lead to more comparisons and more computation overhead as expected. So although B dataset has a higher number of clusters than A dataset, it has a lower computation overhead due to less number of frames (748 compared to 1270).

6.5 ORB-SLAM performance

For ORB-SLAM, the tunable parameter is *min-matches* which is the number of inliers required in matching frames to accept a visual transformation. **NOTE**: We were not able to get any result for A Hall in ORB-SLAM due to it having very low number of features.

Table 5 shows the error of estimated trajectories for different *min-matches*. B Hall and C Hall experience false positive loop closures for *min-matches* of 10 and 15. So the estimated trajectories are inaccurate. But we are able to avoid any false positive loop closure even with low values for *min-matches*. This shows that our proposed approach is applicable in symmetric environments having similar looking scenes. For higher values of *min-matches*, the trajectory error of both approaches is similar to each other. In *min-matches*=100, the trajectory error is higher than in other settings. This is due to having a lower number of permissible visual transformations and more false negatives.

In Table 6, we show the processing time of loop closure detection in the original approach and in our approach. Results show that for a similar level of accuracy in Table 5, we spend less time (15%-25% on average) for loop closure detection resulting in faster execution. The difference in computation time grows larger as the dataset gets bigger(J Hall compared to B and C Hall). The reason is that the original approach always searches through all keyframes, but our approach is able to bound the searching process to keyframes of regions spatially close to current frame using Wi-Fi sensing. Further, the false positive loop closures increase the computation time by running global bundle adjustment wrongly. This shows that our Wi-Fi sens-

Table 3: **RTAB-Map:** Loop closure compute time (s) for different datasets with different *real-time thresholds*

Real-time Threshold (ms)								
	∞		70		100		200	
Dataset	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi
A Hall	7.11	2.35	3.57	5.05	3.74	5.02	6.95	3.99
B Hall	2.36	0.63	4.67	1.85	4.93	1.88	2.61	1.69
C Hall	1.69	0.62	4.90	1.66	4.71	1.67	3.50	1.38
J Hall	4.72	0.80	7.57	2.60	6.75	2.64	4.49	2.81

Table 4: **RTAB-Map:** Compute overheadi (s) of Wi-Fi clustering and cluster management for different datasets and different *real-time thresholds*

Real-time Threshold (ms)				
	∞	70	100	200
Dataset	WiFi	WiFi	WiFi	WiFi
A Hall	0.32	0.34	0.34	0.33
B Hall	0.21	0.22	0.22	0.21
C Hall	0.17	0.17	0.18	0.16
J Hall	0.56	0.63	0.62	0.62

ing is able to save time in the optimization process by avoiding false loop closures as well.

Table 7 represents the computation time overhead caused by *Wi-Fi Clustering* and *Cluster Management* modules in our approach. As shown in Table 6, the computation time savings far outweigh these overhead values. These overhead values are dependent on the number of keyframes and Wi-Fi clusters. J dataset has a little more overhead due to more number of keyframes(about 2800) and 19 Wi-Fi clusters. B and C datasets are analogous to each other due to some randomness in their number of keyframes for different values of *min-matches*. For example for *min-matches*=50, the respective keyframes of B and C datasets are 1200 and 1500. So although C dataset has a lower number of Wi-Fi clusters, it has more computation overhead.

In Table 8, we show the percentage of false positive loop closures with respect to all detected ones in each parameter setting. As discussed earlier, B and C dataset suffer from some false positives for *min-matches* $\in (10, 15)$. This shows the vulnerability of the original approach to symmetric environments and demonstrates the ability of our version of ORB-SLAM to work well even when the *min-matches* value is low. Our approach using Wi-Fi does not incur any false positives for loop closure detection and is therefore not shown in the table. Also, there are no false negatives observed in any settings except for *min-match*=100 which is equal for both approaches.

6.6 Comparison to Wi-Fi augmented FABMAP

To situate our work wrt other visual SLAM algorithms that integrate Wi-Fi, we chose Wi-Fi FABMAP (Nowakowski et al., 2017), the most recent work that we came across that integrates wireless sensing with a specific visual SLAM algorithm.

Wi-Fi augmented FABMAP (Nowakowski et al., 2017) is a topological localization algorithm which introduces a new approach for early fusion of visual and Wi-Fi information. It is executed in two phases: the mapping phase and the localization phase. In the mapping phase, the robot is driven through the target area to collect Wi-Fi AP MAC addresses and images of the environment. In this phase, each image is assumed to be from a different location and is associated with the spatially closest collected Wi-Fi vector, which is a binary vector indicating the presence of APs. In the localization phase, the feature vector extracted from the query image is concatenated with the Wi-Fi vector collected at the location and fed into the FABMAP algorithm to find the best match among the images collected during the mapping phase.

Note: (i) it is a two-phased method that requires war-driving. (ii) Wi-Fi FABMAP does not use signal strength values. Instead, it simply creates the Wi-Fi vector as a vector of binary values indicating the presence or absence of APs. (iii) It uses the Wi-Fi information only in the localization phase as opposed to integrating it into the SLAM process.

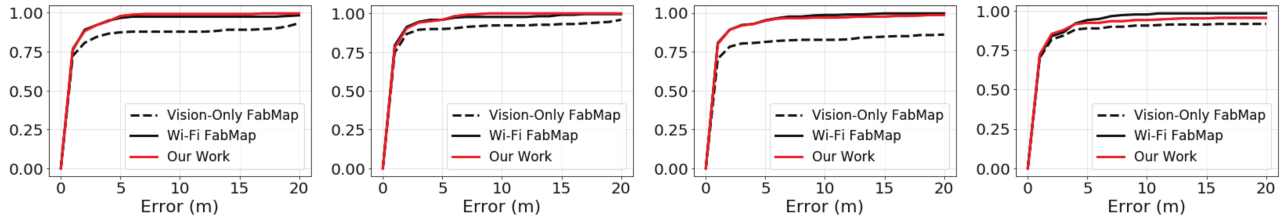
We faithfully re-implemented their algorithm by adapting the available open source code-base for FABMAP (Glover et al., 2012). We collected data for input as per their paper by acquiring images every 2s and Wi-Fi data every 10s while the robot is in motion. The collected data is split into two sets: around 40% for the mapping phase and 60% for the localization phase. We also associated the Wi-Fi data with the images as per the procedure

Table 5: **ORB-SLAM:** Trajectory error (m) of different datasets and different *min-matches* values

	Min-Matches									
	10		15		20		50		100	
Dataset	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi
B Hall	10.96	0.21	10.09	0.23	0.25	0.23	0.21	0.20	0.41	0.45
C Hall	12.22	0.16	13.50	0.15	0.11	0.09	0.21	0.20	0.33	0.38
J Hall	0.86	0.86	0.82	0.73	0.74	0.66	0.85	0.77	2.72	2.84

Table 6: **ORB-SLAM:** Loop closure compute time (s) for different datasets and different *min-matches* values. Our method reduces this time by 15%-25% on average.

	Min-Matches									
	10		15		20		50		100	
Dataset	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi	Vanilla	WiFi
B Hall	10.09	7.04	9.68	7.29	9.10	6.50	8.91	7.11	9.23	7.69
C Hall	19.03	16.49	19.67	15.90	21.17	18.84	18.27	15.78	18.50	15.58
J Hall	25.28	19.63	25.37	20.08	24.96	19.99	25.32	20.32	24.68	19.79



(a) C Hall (b) B Hall (c) J Hall (d) A Hall

Fig. 10: CDF of Error in Wi-Fi augmented FABMAP compared to our approach

Table 7: **ORB-SLAM:**Compute overhead (s) of *Wi-Fi Clustering* and *Cluster Management* modules for different datasets and *min-matches* values. It is about 3%-8% on average.

	Min-Matches				
	10	15	20	50	100
Dataset	WiFi	WiFi	WiFi	WiFi	WiFi
B Hall	0.42	0.46	0.41	0.45	0.53
C Hall	0.49	0.39	0.44	0.65	0.46
J Hall	1.75	1.65	1.59	1.70	1.77

Table 8: **ORB-SLAM:** Percentage false positive loop closures for different datasets and different *min-matches* values. Our method has zero false positives and is therefore not shown in the table.

	Min-Matches				
	10	15	20	50	100
Dataset	Vanilla	Vanilla	Vanilla	Vanilla	Vanilla
B Hall	40.00	33.30	0.00	0.00	0.00
C Hall	50.00	50.00	0.00	0.00	0.00
J Hall	0.00	0.00	0.00	0.00	0.00

mentioned separately for the two phases. The visual vocabulary was learned from 1500 images of corridor scenes from (Yang et al., 2016; Quattoni and Torralba, 2009) and self-collected images in our university.

FABMAP is a topological SLAM algorithm and not a metric SLAM algorithm such as RTAB-Map, and ORB-SLAM. Therefore, FABMAP and its derivatives would perform poorly in direct localization or mapping error comparison. For a fairer comparison, *we chose the exact same metrics* as used in (Nowakowski et al., 2017). This is the Cumulative Distance Function (CDF) of distances between the estimated location and ground-truth

averaged over the query images. To enable this comparison, we also provide localization results from our method over the query images rather than produce a trajectory. In our approach, we use Wi-Fi signatures, which is a vector of RSSI values, rather than a binary vector used by Wi-Fi augmented FABMAP. We first find a representative Wi-Fi signature for each Wi-Fi cluster and the time stamp associated with it. Then we associated every map image to a specific cluster based on their acquisition time stamps. For localization, the query image is also assigned a Wi-Fi signature. This is the signature recorded at the last pause prior to its acquisition. To get the best map image matching the query image, we first select clusters that their representative Wi-Fi signatures have high cosine similarity with the Wi-Fi signature of the query image named *similar clusters*. Then among the map images associated with similar clusters, we select the map image with maximum visual likelihood with the query image.

Figure 10 shows the CDF of error between estimated localization and ground-truth of our Wi-Fi clustering method, Wi-Fi augmented FABMAP and visual FABMAP for all four datasets. We perform better for Halls B and C because:

- In some instances, the high similarity in visual features from physically different locations causes the algorithm to make the wrong matches even with the Wi-Fi data.
- The constructed Wi-Fi Chow Liu tree could be inaccurate in capturing relations due to using raw Wi-Fi data. The randomness associated in the detection of APs over time could cause the Chow Liu tree to have dependencies that don't necessarily hold. Using our approach and aggregating the BSSIDs differing only in the last nibble could reduce the problem.

Our performance is similar to (Nowakowski et al., 2017) for J Hall. For Hall A, Wi-Fi augmented FABMAP performs better. The reason for this is that A Hall is a wide area with fewer features that doesn't include many blocking objects like walls along the trajectory. This causes less RSSI attenuation and therefore lesser number of Wi-Fi cluster. Since there are fewer clusters, there are more images per cluster and this increases the chances for perceptual aliasing. But we do note that even under these circumstances, more than 90% of the query images are within 4 meters accuracy which is similar to Wi-Fi augmented FABMAP.

In summary, the demonstrated results show our performance improvement, in terms of localization/mapping accuracy and computational complexity, in visual SLAM algorithms. The low dimensionality of Wi-Fi data and its immunity to perceptual aliasing are the key elements of the performance gain. Further, the comparison with the state-of-the-art Wi-Fi FABMAP, shows a similar if not better performance. This shows the generality of our approach unlike Wi-Fi augmented FABMAP which is specifically designed for visual FABMAP.

7 Discussion

We incorporate Wi-Fi sensing into visual SLAM to combat two specific challenges of SLAM indoors — perceptual aliasing and computational overhead. We discuss the relevance of our work to this community and the implications of some of our choices here.

Extent of Accuracy Improvement: Wi-Fi clustering improves the accuracy of visual SLAM algorithms through limiting the number of frames which a visual frame is compared against and leads to decreasing the number of false positive and false negative loop closures. Computing more accurate visual transformations between frames in not in the context of this work.

Relevance to Sensor Systems: Our work is useful for robots as well as mobile devices. With the advent of RGB-D cameras for mobile devices such as Intel RealSense and augmented reality/mixed reality devices such as the MS Hololens and MagicLeap One, we expect an increase in spatial applications that will use visual SLAM. Therefore, we believe that this topic is of relevance to both robotics and the sensor systems communities.

Environmental Dynamics: Wi-Fi RSSI values can vary with environmental dynamics and background communications such as the number of people in the area and the number of devices connected. Our solution is pausing, collecting and averaging RSSI information over a few seconds and our empirical observation from many repeated data collection trials is that this helps in having more stable Wi-Fi signatures at various positions. Also, we could adjust the frequency of pauses for Wi-Fi RSSI collection or the time length of the pauses for averaging the data to compensate for some of the dynamics. This will get affected by higher degree of dynamics or pause-free continuous movements. Depending on the degree of environmental dynamics and the number of wireless clients in a given area, it might or might not limit the applicability of the approach.

Wi-Fi Similarity Tuning: This parameter is analogous to the *min-matches* parameter in visual slam algorithms. We tried many values to find the optimal initialization which seems dependent on the size and the degree of dynamics of the environment. Spaces including more dynamics like A Hall dataset with many

people in motion require lower values in order to compensate for fluctuations. In general, very high values would increase the number of false negative loop closures and very low values would make it inapplicable for getting rid of perceptual aliases.

Number of Access Points: Depending on the placement and number of visible access points, the effectiveness of Wi-Fi sensing might vary. The performance gain would increase with higher number of APs especially if they are scattered and not co-linear. Based on our datasets, which reasonably represent modern urban settings, our approach works well with as low as 40 APs scattered around a square shaped environment.

8 Conclusion

In this work, we proposed a general approach to incorporate Wi-Fi sensing into visual SLAM algorithms. To demonstrate, we augment three recent SLAM algorithms to show improved mapping/localization accuracy as well as speed-up in operation. We demonstrated this functionality on data collected from four university buildings, which are representative spaces of modern urban environments.

We also compared our proposed approach to recently proposed Wi-Fi augmented FABMAP and showed a comparable if not better performance. This comparison also confirms the generality of our approach unlike Wi-Fi augmented FABMAP which is only designed for visual FABMAP.

In the future, we hope to demonstrate the utility of Wi-Fi sensing for sustained long-term use in an urban space. While Wi-Fi signal strength used for this work is a useful measure, novel wireless sensing technologies such as 60 GHz sensing can be used to further improve the overall accuracy as well as the computational complexity of SLAM algorithms in the future as well.

References

Belter, D., Nowicki, M., and Skrzypczyński, P. (2016). Improving accuracy of feature-based rgb-d slam by modeling spatial uncertainty of point features. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1279–1284. IEEE.

Berkvens, R., Jacobson, A., Milford, M., Peremans, H., and Weyn, M. (2014). Biologically inspired slam using wi-fi. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1804–1811.

Biswas, J. and Veloso, M. (2010). Wifi localization and navigation for autonomous indoor mobile robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 4379–4384.

Clark, R., Wang, S., Wen, H., Trigoni, N., and Markham, A. (2016). Increasing the efficiency of 6-dof visual localization using multi-modal sensory data. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 973–980. IEEE.

Codd-Downey, R. and Jenkin, M. (2017). On the utility of additional sensors in aquatic simultaneous localization and mapping. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5163–5168.

Cummins, M. and Newman, P. (2008). Accelerated appearance-only slam. In *2008 IEEE International Conference on Robotics and Automation*, pages 1828–1833.

Dong, J., Xiao, Y., Noreikis, M., Ou, Z., and Ylä-Jääski, A. (2015). imoon: Using smartphones for image-based indoor navigation. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, pages 85–97. ACM.

Engelhard, N., Endres, F., Hess, J., Sturm, J., and Burgard, W. (2011). Real-time 3D visual SLAM with a hand-held camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden.

García, S., López, M. E., Barea, R., Bergasa, L. M., Gómez, A., and Molinos, E. J. (2016). Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 205–210.

Glover, A., Maddern, W., Warren, M., Reid, S., Milford, M., and Wyeth, G. (2012). Openfabmap: An open source toolbox for appearance-based loop closure detection. In *Robotics and automation (ICRA), 2012 IEEE international conference on*, pages 4730–4735. IEEE.

Heshmat, M., Abdellatif, M., and Abbas, H. (2013). Improving visual slam accuracy through deliberate camera oscillations. In *2013 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 154–159.

Hess, W., Kohler, D., Rapp, H., and Andor, D. (2016). Real-time loop closure in 2d lidar slam. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 1271–1278. IEEE.

Huang, J., Millman, D., Quigley, M., Stavens, D., Thrun, S., and Aggarwal, A. (2011). Efficient, generalized indoor wifi graphslam. In *2011 IEEE In-*

ternational Conference on Robotics and Automation, pages 1038–1043.

Ito, S., Endres, F., Kuderer, M., Tipaldi, G. D., Stachniss, C., and Burgard, W. (2014). W-rgb-d: Floor-plan-based indoor global localization using a depth camera and wifi. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 417–422.

Jacobson, A., Chen, Z., Rallabandi, V. R., and Milford, M. (2015). Multi-scale place recognition with multi-scale sensing. In *Australasian Conference on Robotics and Automation (ACRA 2015)*, Canberra, A.C.T. Australasian Robotics and Automation Association.

Jung, J., Oh, T., and Myung, H. (2015). Magnetic field constraints and sequence-based matching for indoor pose graph slam. *Robotics and Autonomous Systems*, 70:92 – 105.

Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923.

Karanam, C. R., Korany, B., and Mostofi, Y. (2018). Magnitude-based angle-of-arrival estimation, localization, and target tracking. In *Proceedings of the 17th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 254–265. IEEE Press.

Kejriwal, N., Kumar, S., and Shibata, T. (2016). High performance loop closure detection using bag of word pairs. *Robotics and Autonomous Systems*, 77:55–65.

Kotaru, M., Joshi, K., Bharadia, D., and Katti, S. (2015). Spotfi: Decimeter level localization using wifi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 269–282, New York, NY, USA. ACM.

Kumar, S. S., Gil, S., Katabi, D., and Rus, D. (2018). Indoor localization of a multi-antenna receiver. US Patent 9,885,774.

Kuo, Y.-S., Pannuto, P., Hsiao, K.-J., and Dutta, P. (2014). Luxapose: Indoor positioning with mobile phones and visible light. In *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, MobiCom '14, pages 447–458, New York, NY, USA. ACM.

Labbe, M. and Michaud, F. (2014). Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666. IEEE.

Labbé, M. and Michaud, F. (2011). Memory management for real-time appearance-based loop closure detection. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1271–1276.

Labbé, M. and Michaud, F. (2013). Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics*, 29(3):734–745.

Lu, C. X., Li, Y., Zhao, P., Chen, C., Xie, L., Wen, H., Tan, R., and Trigoni, N. (2018). Simultaneous localization and mapping with power network electromagnetic field. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, MobiCom '18, pages 607–622, New York, NY, USA. ACM.

Luo, C., Hong, H., and Chan, M. C. (2014). Piloc: A self-calibrating participatory indoor localization system. In *Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, IPSN '14, pages 143–154, Piscataway, NJ, USA. IEEE Press.

Mur-Artal, R. and Tardós, J. D. (2017). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

Nguyen, D. V., Recalde, M. E. V., and Nashashibi, F. (2016). Low speed vehicle localization using wifi fingerprinting. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–5.

Nowakowski, M., Joly, C., Dalibard, S., Garcia, N., and Moutarde, F. (2017). Topological localization using wi-fi and vision merged into fabmap framework. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3339–3344.

Nowicki, M. (2014). Wifi-guided visual loop closure for indoor navigation using mobile devices. *Journal of Automation Mobile Robotics and Intelligent Systems*, 8(3):10–18.

Quattoni, A. and Torralba, A. (2009). Recognizing indoor scenes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 413–420. IEEE.

Quigley, M., Stavens, D., Coates, A., and Thrun, S. (2010). Sub-meter indoor localization in unmodified environments with inexpensive sensors. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2039–2046.

Soltanaghaei, E., Kalyanaraman, A., and Whitehouse, K. (2018). Multipath triangulation: Decimeter-level wifi localization and orientation with a single unaided receiver. In *Proceedings of the 16th Annual International Conference on Mobile Systems,*

Applications, and Services, MobiSys '18, pages 376–388, New York, NY, USA. ACM.

Wang, S., Wen, H., Clark, R., and Trigoni, N. (2016). Keyframe based large-scale indoor localisation using geomagnetic field and motion pattern. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1910–1917.

Xia, Y., Li, J., Qi, L., and Fan, H. (2016). Loop closure detection for visual slam using pcanet features. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2274–2281.

Yang, S., Maturana, D., and Scherer, S. (2016). Real-time 3d scene layout from a single image using convolutional neural networks. In *Robotics and automation (ICRA), 2016 IEEE international conference on*. IEEE.

Yang, S. W., Yang, S. X., and Yang, L. (2014). Method of improving wifi slam based on spatial and temporal coherence. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1991–1996.

Yang, Z., Wu, C., and Liu, Y. (2012). Locating in fingerprint space: Wireless indoor localization with little human intervention. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 269–280, New York, NY, USA. ACM.

Zhang, C. and Zhang, X. (2016). Litell: Robust indoor localization using unmodified light fixtures. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking*, MobiCom '16, pages 230–242, New York, NY, USA. ACM.