

WiFi Imaging: Generating Building Maps Using Passively Obtained WiFi Signal Strengths

Shane Farrell¹, Dr. John McAuley¹ and Prof. Linda Doyle²

¹Technological University Dublin, Park House, Grangegorman, Dublin 7, Ireland

²Trinity College Dublin, College Green, Dublin 2, Ireland

Abstract

We present an algorithm for mapping a building using WiFi signal strength measurements. Our approach simultaneously localises the positions of signal measurements and maps the surrounding obstructions to the WiFi signal. While techniques for mapping buildings exist, they often require an individual to actively perform a mapping process. In contrast, we only require sets of signal strength measurements, the positions of the WiFi routers, and a bounding box for mapping. We do not require our signal strength measurements to be correlated with each other, nor do we require any additional sensors. We impose these constraints to allow mapping to be performed using passive WiFi signal strength pings performed by smartphones. We demonstrate the capability of our algorithm with two evaluations. First, we show that the algorithm can map full buildings using a synthetic dataset constructed from 500 floor plans. Second, we map the rooms of a real world building using various configurations.

Keywords

wifi, map, mapping, slam, localisation, signal, rssi, floor plan, building, trilateration

1. Introduction

In this paper we present a method of mapping the walls of a building using WiFi signal strength pings collected from mobile devices. Signal strength pings are performed periodically in the background on most modern mobile devices in order to determine which WiFi router can provide best access to the internet. We treat each signal strength ping independently, and do not need to know the location of the device during the ping. We require only a list of signal strengths associated with WiFi router MAC addresses from the measuring device. Such information can be collected without requiring any active scanning of device sensors. We combine this with the coordinates of the WiFi routers in order to map a region in space specified by the user. We do not require any additional information other than a dataset of signal strengths and the coordinates of each associated WiFi router. Existing techniques can be used to determine the coordinates of WiFi routers[1, 2]. We evaluate our approach on both a large simulated dataset and on real world data.


Unlike many other Simultaneous Localisation and Mapping (SLAM) techniques, we do not compute a path that a measuring user traverses as part of localising that user [3, 4, 5, 6, 7]. We treat each signal strength measurement separately, localising that measurement independently

IPIN 2021 WiP Proceedings, November 29 – December 2, 2021, Lloret de Mar, Spain

✉ Shane.Farrell@TUDublin.ie (S. Farrell); John.Mcauley@TUDublin.ie (J. McAuley); Linda.Doyle@tcd.ie (L. Doyle)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

of others. This broadens the scope of the data we are able to use as compared with other SLAM algorithms. We can construct architectural maps using sets of signal strength measurements taken from unknown devices.

Our contribution in this paper is as follows:

- We present an algorithm which can map an area using only WiFi signal strength measurements, WiFi router coordinates and a bounding box specifying the area to map. The signal strength measurements need not be correlated with each other, nor does the position from which they were taken need to be known.
- We evaluate the algorithm on a large simulated dataset of full building floor plans. We constructed this dataset from vectorised floor plans of Chinese houses[8] and simulated WiFi signals passing through the walls described in the floor plan using PyLayers[9].
- We evaluate this algorithm on a number of small scale real world WiFi datasets. We show that the algorithm is capable of resolving multiple walls in such datasets.

We structure this paper into five sections. In the Related Work section we discuss the state of the art techniques, and we highlight the gap our paper fills in comparison. In the Method section we explain our algorithm, first providing an intuition and then going into detail. We break our algorithm down into three steps, providing pseudocode for each step to aid implementation and understanding. In the Evaluation section we show how the algorithm performs in an experimentally controlled real world test. We also show how the algorithm performs over a large dataset of simulated WiFi signal strength maps. We then construct these maps using a dataset of building floor plans. In the Future Work section we discuss enhancements that could be made to the algorithm as well as experiments that could be run to evaluate the algorithm further. Finally, we conclude and summarise the findings of the paper.

2. Related Work

Crowdsourced WiFi signal strength map generation, as well as magnetic field map generation are a prevalent component to many modern indoor localisation techniques[10, 11, 12]. Signal strength maps can be used for localising any device based on the device’s current measured signal strengths. WiFi signal strengths are also a component of a number of modern SLAM techniques[13, 14]. The use of cameras is also prevalent in SLAM techniques [15, 16]. The work we are presenting here could be considered a SLAM technique. Hayoun et al. propose a method for using a pre-trained neural network for time efficient indoor mapping [17].

Gao et al. propose Jigsaw, a method for estimating wall positions using crowdsourced data from smartphone users[18]. Their approach uses the smartphone’s inbuilt camera, accelerometer, and gyroscope sensors to both localise the users mapping the building and to detect landmarks that they use for the mapping process itself. While their method uses crowdsourced data, the requirement for these data to contain visual recordings coupled with accelerometer data discourages broader adoption of the technology.

In contrast with Jigsaw’s crowdsourced smartphone data, Rosinol et al. propose an advanced SLAM library named Kimera[19]. They achieve very high quality mapping results on the EuRoC Micro Aerial Vehicle dataset. In their approach they combine multiple state of the art SLAM

techniques with machine learning. Their approach uses stereo images and high-rate inertial measurements as input. These high fidelity input requirements are above what is available on common smartphones, meaning the technique is limited to situations where specialised hardware can be employed.

Similarly to Jigsaw, Shin et al. propose a method for mapping buildings using WiFi signal strength measurements paired with accelerometer readings[20]. Unlike Jigsaw and Kimera, Shin et al. do not use any visual SLAM. In their approach they use a post-processing algorithm to segment the map into separate rooms, producing more accurate wall estimations than we present in this paper. However, their technique relies on integrating accelerometer readings to produce a path the user traverses. A high energy cost, and the requirement for a user to traverse a significant part of the building in one pass makes the technique unsuitable for passive floor plan construction.

In their 2017 paper, Gao et al. propose Knitter, a method which allows a mostly untrained user to map a building by spending one hour walking around the building taking photographs of landmarks[21]. Their technique uses WiFi signal strengths, as well as the smartphone's camera and inbuilt gyroscope. They evaluate their method in three separate real world locations with 15 users who have spent 5 minutes practising data collection. Their technique produces encouraging results in the situation where an untrained user might want to map a building. However, the technique does still require users to perform activities they would not perform passively, limiting the scope for large passive data collection efforts.

In their 2020 paper, Zhou et al. demonstrate a method for using smartphone speakers and microphones as sonar apparatus, allowing rapid and precise mapping of buildings [22]. Their technique provides accuracy within 1-2 cm for objects closer than 4 meters from the smartphone. However, the use of phone speakers makes the technique inherently disruptive to employ. The user is also required to hold the smartphone out in front of them in a horizontal position. These factors mean the technique is less suitable for mapping public buildings. The requirement for the phone to be held in a specific position also means the technique requires active trained measurers.

3. Algorithm

In this section, we explain the algorithm we are presenting. We structure the algorithm as an optimisation problem. We simulate WiFi signals propagating from each WiFi router, giving us a signal strength map of the scene. We use these signal strength maps to localise devices that ping these WiFi routers. We then compute a cost function based on how precisely these observations are localised. The idea is that if the algorithm's simulated walls match the ground truth walls better, they will produce more precise localisations.

3.1. Scene Map Creation

We first create a mutable variable in TensorFlow. This variable is a 2D grid of floating point values. We initialise this variable with random normally distributed noise. This variable is the underlying mutable variable that we use to create our scene map. We take the sigmoid of this variable to constrain our scene map to values between zero and one.

The scene map can be thought of as a 2D image of the signal obstructions in the area. Zero means that there is no signal attenuation, and one means the signal is fully blocked. A value in between indicates that that fraction of the signal strength passing through is attenuated. So a value of 0.25 means 25% of any signal strength passing through that pixel is lost. Since walls are multiple pixels thick, the values per pixel in a wall are typically quite low.

3.2. WiFi Signal Simulation

At this point we have a 2D grid representing the scene. We preprocessed the grid to constrain its values between zero and one. The grid's values represent the amount of signal attenuation to occur at that point. In this step we simulate each WiFi router's signal passing through this scene and output a signal strength map for each router.

To create these signal maps we first unwind the scene map into polar coordinates for each WiFi router. This polar coordinate representation is another 2D grid. We will later use this polar representation to simulate WiFi signal propagation in a differentiable way. The axes on this grid are the distance from the WiFi router and the angle around the WiFi router. The dimensions of this grid do not need to match the scene's dimensions, nor do the grid spacings need to match. The higher resolution, the less able the signal is to incorrectly pass through walls unimpeded due to sampling errors. We create one polar coordinate representation of the scene for each WiFi router, with the pole of the coordinates on that router. Since the positions of the WiFi routers are both known and immutable, the sampling positions can be calculated in advance of the algorithm execution and the coordinates to be sampled themselves do not need to be differentiable. This allows for a more optimal sampling method where only the pixels actually being sampled from need to be considered. If the coordinates did need to be differentiable, each pixel's value would depend on the value of every pixel in the scene instead of just a small fixed number. We interpolate the values of pixels where the sampling lies between multiple pixels, thus each pixel in the polar coordinates depends on four pixels in the underlying scene.

Now that we have the scene map unwound into polar coordinates for each router, we create a polar coordinate signal map for each router. The signal map has the same dimensions as its corresponding polar coordinate scene map. We use polar coordinate signal maps to simulate WiFi signal propagation efficiently. We initialise the signal strength map as if there were no obstructions. This can be initialised based on the WiFi router manufacturer's specifications for how the signal varies with distance and angle. In our case we defined the attenuation devoid of obstructions using the inverse square law of electromagnetic attenuation starting from a measured value of the WiFi router's signal strength output right next to it. The inverse square law says that the signal strength decreases proportionally to the square of the distance to the WiFi router. We used omnidirectional WiFi routers in our experiments. If the WiFi router being used is not omnidirectional and the radiation pattern is known it can be inserted at this step.

In order to apply the attenuation from the scene map to the signal map, we must first smear the scene map outwards from the router's position. The signal map currently holds what the signal strength would be at each point if not attenuated by the scene map. If the scene map was multiplied directly with the signal map as is, the walls would not leave shadows, only lowering the signal strength exactly where the walls are. The values in the scene map indicate what fraction of the signal strength is lost by the signal passing through that point. As such we

perform the smear as a running product. We multiply the second row of the image by the first, and then the third by the updated second and so on. By doing this we ensure that all points behind a wall from the router's perspective are also multiplied by the wall's attenuation factor. In order to avoid floating point errors in this step we actually take the log of the scene map first, use addition instead of multiplication, and take the exponential of the result. This is however conceptually multiplication, and is mathematically equivalent.

Once we have the smeared scene map for each WiFi router, we can multiply the result with the signal map that does not account for obstructions. Since WiFi signal strengths are measured in decibels, this operation that can be thought of as multiplication is also performed as addition. Ten multiplied by the log of the value of the smeared scene map is added to the signal strength map to calculate the resultant decibel value. Since all the values on this smear map are between zero and one, the value added is negative and exclusively acts to lower the signal strength.

After the polar coordinate signal maps are created, we convert them back to Cartesian coordinates. This is performed by taking the inverse of the original sampling. The Cartesian map is sampled from the polar coordinate map. In the case where the value falls between pixels interpolation is used.

Data: The positions of all involved routers

Data: The scene map of the building being simulated

Result: A simulated signal map for each router

```

for each router do
    create a transform of the scene map in polar form centred on this router;
    for each row on the polar form scene map do
        update the values on this row by multiplying by the corresponding values on the
        row above;
    end
    for each row on the polar form scene map do
        calculate how physically far away this row is from the router;
        calculate the signal strength of the router based on this distance assuming no
        obstructions;
        multiply the values on this row by this signal strength;
    end
    transform the signal map back into Cartesian coordinates centred on the router;
end

```

Algorithm 1: Signal Simulation step

3.3. Position Estimation

At this point we have a signal strength map simulated for each WiFi router. This signal strength map is what the signal strength is at each point after being simulated through the scene map. We have maintained differentiability through the simulation process so that gradient descent can be used to optimise the scene map used to create these signal maps. The signal maps have the same dimensions and positional bounding box as the scene map. In this step we estimate the positions

of observations within the scene. An observation is a WiFi signal strength measurement taken from an unknown location within the scene. If an observation is taken outside the bounding box of the scene it will contribute to error in the mapping of the scene as the optimiser fails to localise it.

We load observations in batches to speed up convergence. An observation dataset consists of a 2D array of floating point values. Each column corresponds to a different WiFi router in the scene and each row corresponds to a separate observation. Each cell encodes the signal strength measured for that WiFi router as part of that observation. If an observation does not contain an entry for a WiFi router, we use a special value to signify this. In case of this value we exclude the WiFi router's contribution to the positioning step for this observation by multiplying any error it contributes by zero.

After loading a batch of observations, we create a square difference map for each router for each observation in the batch. These square difference maps encode the square difference between the observation's measured value for a given AP, and the signal map for that AP. Thus if the measured value was -50 dB for a given router, the decibel value in each cell of the signal map will be subtracted from this value and the result will be squared.

We then take the cell-wise mean of all the square difference maps for each observation. We call this mean of square difference maps the position map for the observation. The lower the value in each cell of the position map, the more likely the algorithm thinks it is for the observation to have been taken from that cell's position. Low values in the position map mean that at that point in space the measured signal strengths more closely match the simulated set of signal strengths at that point.

Data: The signal map of each router

Data: Measurements of the signal strength of each router, taken at various unknown points

Result: A sum of squared differences map for each measurement

for *each measurement* **do**

for *each router* **do**

 get the measured signal strength for this router;

 get the signal map for this router;

 calculate the square difference between each pixel of the signal map and the measured signal strength;

end

 calculate the mean of the square difference maps from each router for this measurement;

end

Algorithm 2: Position Estimation step

3.4. Cost Functions

At this point, we have created a position map for each observation in the current batch. These position maps consist of the mean squared error between the signal map for each router and

the observation's measured value for that router. Finally we must implement a cost function to condense this information into a single value for the optimiser to perform gradient descent. We must also implement a regularisation cost function to prevent the optimiser from falling into some problematic local minima.

We calculate the primary cost function by first taking the minimum value from each position map, and then taking the mean of those minimum values. The minimum value of a position map encodes the degree to which the WiFi routers cannot agree on a position to localise the observation. Taking the mean of many of these values means that the cost is how much the WiFi routers disagree with where the observations were taken from on average over the batch. Minimising this means that a scene is found where the routers agree on the positions of observations more. The scene that theoretically causes the WiFi routers to agree the most is the scene that the observations were measured in, which is the ground truth.

We calculate the regularisation cost function as the ratio between the surface area of the walls in the scene and the volume of said walls. The cost is high when there is a lot of surface area for the amount of volume there is. Thus the algorithm does not get penalised for making lots of walls, it gets penalised for making those walls not contiguous or containing too much fine detail or creating specks of wall in the middle of an empty room. The surface area to volume ratio is calculated as the square of the divergence of the gradient of the scene divided by the square of the scene. The divergence of the scene is like an omnidirectional first derivative. Thus the square of the divergence is positive when the cell is a border cell between high and low values, as that is where the square of the first derivative produces high values. The divergence is calculated by creating a 3x3 Gaussian blur of the scene and subtracting the scene from it. The square is then taken to get the square divergence. This value is then divided by the square of the scene.

The final cost function we give to the optimiser to perform gradient descent is the sum of the two costs. We cap the regularisation cost so that it cannot fall below a value of 1. This means that when the algorithm has it on 1 it considers that to be an acceptable amount of regularisation error as there is nothing it can do to reduce it. This ensures the algorithm does not hyper focus on minimising the surface area to volume ratio to the detriment of the results, since real world scenes do need to contain some level of detail.

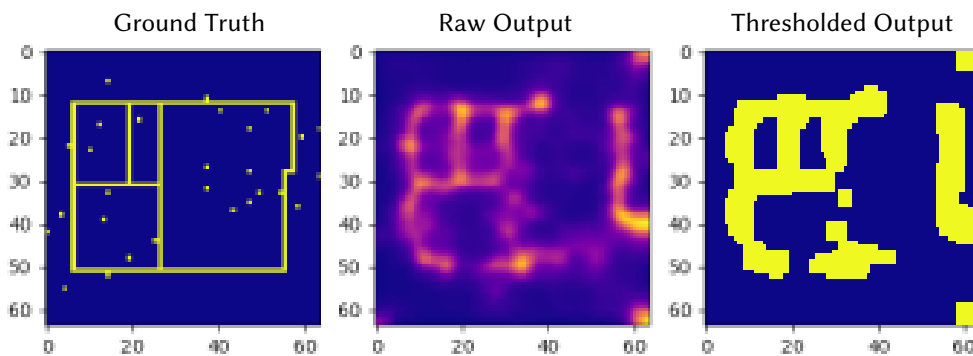


Figure 1: Wall estimation on a simulated building layout.

Data: The positions of all involved routers
Data: Measurements of the signal strength of each router, taken at various unknown points
Result: An estimation of the signal strength attenuation of the scene at each point
initialise scene map as random noise;
while *not done optimising* **do**
 perform the Signal Simulation step with the current scene map;
 perform the Position Estimation step with the generated signal maps;
 for *each measurement* **do**
 calculate the minimum value of this measurement’s sum of squared differences
 map;
 end
 average these minimum values over each measurement;
 feed this averaged value to the optimiser as the cost function, along with the
 regularisation cost;
end

Algorithm 3: Cost function and optimisation loop

4. Evaluation

In this section, we provide details on the two experiments we conducted to evaluate the effectiveness of our algorithm. In the first experiment we evaluate the algorithm’s ability to identify walls in a simulated dataset based on complete architectural drawings. We evaluate the algorithm on 500 simulated building layouts created from floor plans. In doing so we test the algorithm’s ability to map full building layouts. A simulated experiment gives us greater control over environmental variables, and allows us to average the result over our dataset, making the metric less dependent on the specific building shape being evaluated. In the second experiment we evaluate the algorithm in 4 real world configurations. In doing so we test the algorithm’s ability to generalise its predictive power to real world WiFi signals.

Table 1
Evaluation Results

Experiment Type	Configuration	Mean Wall Displacement Error
Simulated Configurations	500 Full Building Floor Plans	0.253 meters
Real World Configurations	Single Wall	0.477 meters
	Two Walls	0.294 meters
	Corner Wall	0.714 meters
	Two Walls Distributed	0.689 meters

We evaluate our experiments using a metric we call displacement error. Displacement error is the mean distance from a predicted wall pixel to the nearest ground truth pixel. To calculate the displacement error we perform a distance transform on the ground truth image. This produces

an image where each pixel value is the distance in meters to the nearest ground truth pixel. We then threshold the output of our algorithm using the Otsu threshold value. The Otsu threshold value is a dynamically calculated threshold value common in image processing which is used to separate foreground and background pixels. We multiply the distance transform by this thresholded image so that only predicted wall pixels are not zero. We then sum the resulting pixel values up and divide by the total number of predicted pixels. This gives us the mean distance of each pixel in meters to the nearest ground truth pixel.

4.1. Simulation

In our first experiment, we evaluated the algorithm on simulated data. This allowed us to fully constrain environmental variables, and to evaluate the algorithm on a much larger set of buildings than is practical in a real world experiment. We used these features of simulated data to conduct an experiment over a dataset modelling 500 different floor plans. We then calculated the average wall displacement error for the entire dataset. By doing so we provide a wall displacement error that is less influenced by any particular building shape, while also uninfluenced by unforeseen interference or environmental factors.

We created our dataset of WiFi signal simulations using PyLayers[9]. PyLayers is a ray tracing based WiFi signal simulator which allows users to specify the locations of walls and WiFi routers. PyLayers generates signal maps for the specified WiFi routers through the user provided environment. We provided PyLayers each wall in the form of pairs of coordinates specifying the two endpoints of the wall. PyLayers treated these walls as infinitely thin lines that attenuated signal strengths by an amount equivalent to a 6 inch thick brick wall. We defined a bounding box for the floor plan using the extreme values of the coordinates of each wall endpoint. We then expanded this bounding box to provide 2 meters of padding around the edges of the building. We placed 30 WiFi routers at uniformly distributed random coordinates within the bounding box. Figure 1 shows an example of a simulated setup and the predicted wall locations. In the figure's ground truth image the lines indicate the infinitely thin vector walls we provide to PyLayers, and the pixels indicate the positions of the 30 WiFi routers. We provided the algorithm with the coordinates of these WiFi routers, but not the locations of the walls, which were the target feature. To simplify the execution of the algorithm, we added additional padding around each bounding box so that the bounding box would fit in a square. This enabled us to treat each floor plan identically for the execution of the algorithm. We constrained this additional padding to be zero everywhere, preventing the algorithm's optimiser from modifying it.

We executed the algorithm on a dataset of 500 floor plans. Each floor plan was taken from the output dataset of Raster-to-Vector[8]. Raster-to-Vector converts images of floor plans into vector representations. In order to ease compatibility with PyLayers, we removed any information about windows and doors from the Raster-to-Vector dataset, instead leaving them as solid walls. Additionally, the floor plans did not feature any scale information, we thus scaled the buildings such that 0.8 meters, based on the average door width in Chinese houses[23]. Each floor in the Raster-to-Vector dataset was of a Chinese house.

For each floor plan we produced a raw output image like in figure 1. The displacement error of the simulated floor plans had a mean result of 0.253 meters over the dataset, with a standard

deviation of 0.054 meters.

4.2. Real World Experiment

Our previous experiment showed that the algorithm can map full building layouts in a simulated context. We used the large number of simulated architectural layouts to evaluate the algorithm’s generalisability. In this experiment we evaluate whether the algorithm generalises to real world data. We configure the experiment to evaluate the algorithm’s ability to converge on real world data in an old Irish Victorian house, a contrast to the previous experiment. We selected several walls within the building of varying degrees of thickness. We began with the simple case of a single thick wall and increased the level of complexity in later iterations. We ran four experimental configurations in total, evaluating the increasing complexity in each case. We describe each experiment under the proceeding work: single wall configuration, two wall configuration, corner wall configuration, and distributed routers configuration. In each

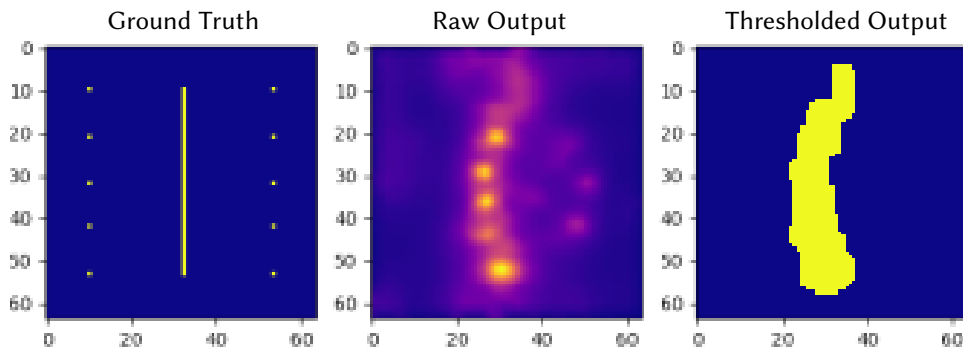


Figure 2: Real World Single Wall Configuration.

experiment we used ten Ubiquiti UniFi nanoHD WiFi routers. Other WiFi routers existed within the building and surrounding buildings, however we excluded these signals from the dataset. For each of the ten WiFi routers we created two WiFi networks, one at 2.4 gigahertz, and one at 5 gigahertz. This totalled 20 WiFi networks with known router positions, though only originating from 10 different locations. We measured each WiFi router’s coordinates and considered the centre of the router as its coordinate. We measured the WiFi signal strengths using a Samsung SM-A750GN/DS running Android 10 with a custom Android app. WiFi measurements were taken by walking around randomly within the scene with the phone held at chest height. We took each measurement by pressing the app’s button to take a measurement and standing in place until the measurement was completed. We then walked to a different location and took a measurement. A measurement consisted of the App requesting the Android operating system to actively ping the surrounding WiFi routers and then retrieving that data. In advance of running the experiment we measured the signal strength of one of the routers at various distances in order to calibrate our algorithm’s internal representation of the router’s signal strengths to the real world equivalent. Instead of inputting these measurements directly, we used the measurements to calibrate the signal falloff curve due to the inverse square law. Thus the falloff curve approximately matched what we found to be the case in the real world, but it was not an

exact encoding of the router's radiation pattern.

Some of the walls involved in the experiment are unusually thick compared with more modern developments, this means the attenuation effect of those walls is greater and thus represent an easier experimental configuration for the algorithm. We include a thickness measurement for each wall involved in the experiment. We performed each experimental setup individually on a single floor, placing WiFi routers on the ground in each case other than the final configuration. All walls and areas were indoors.

4.2.1. Single Wall Configuration

We first evaluated the algorithm's ability to identify a single wall. The wall was 0.5 meters thick and contained a large bookcase along its length, which was also 0.5 meters thick. In this experiment we placed 5 WiFi routers on each side of the wall. We placed each WiFi router 2 meters from the corresponding surface of the wall, ignoring the book shelf. We placed each WiFi router 1 meter apart from each other parallel to the wall in a line, the line thus spanned 4 meters. We ensured that each WiFi router was placed opposite to the router on the opposite side of the wall, thus making the routers form a grid. We placed each WiFi router on the ground, facing towards the wall. We set the simulation bounding box as the rectangle containing all the routers with one meter of additional padding. Thus in the scene there were 5 WiFi routers 1 meter from the top of the image, and 5 routers 1 meter from the bottom, with the wall in the centre of the image.

When we executed the algorithm on this configuration, we produced the image seen in figure 2. Evaluating the displacement error we received a result of 0.477 meters. The algorithm converged correctly on the simple case of one wall.

4.2.2. Two Wall Configuration

Our objective with this configuration was to evaluate whether the algorithm could correctly identify two individual and separate walls, as the algorithm correctly converged on one wall in the first setup. However, with the output being a single line, the setup was simple enough that the convergence may have been coincidental. In this configuration, we extended the configuration of the single wall configuration. We included a second 0.9 meter thick brick wall. This second wall featured a fireplace also. This setup consisted of a large hallway with a large room on each side, a dining room and an office. This office featured a desk placed against the wall between it and the hallway. The fireplace on that wall was built into it from the side of the hallway. We placed all WiFi routers on the floor facing the same direction, facing from the hallway towards the office. The three rooms shared a common back wall, which was the back of the house. This wall was used in determining the placement of the WiFi routers in each room.

We placed 3 WiFi routers in the dining room. We placed each router 1.4 meters from the dining room's hall wall. This distance excludes the bookshelf. The bookshelf was 0.5 meters thick, full of books, and placed on the dining room side of the dining room wall. The dining room wall (brick) was an additional 0.5 meters thick. We placed the first WiFi router 1 meter from the common back wall, the second router 2 meters, and the third 3 meters. We placed 4 WiFi routers in the hall, placing each router 3.75 meters from the line of routers in the dining

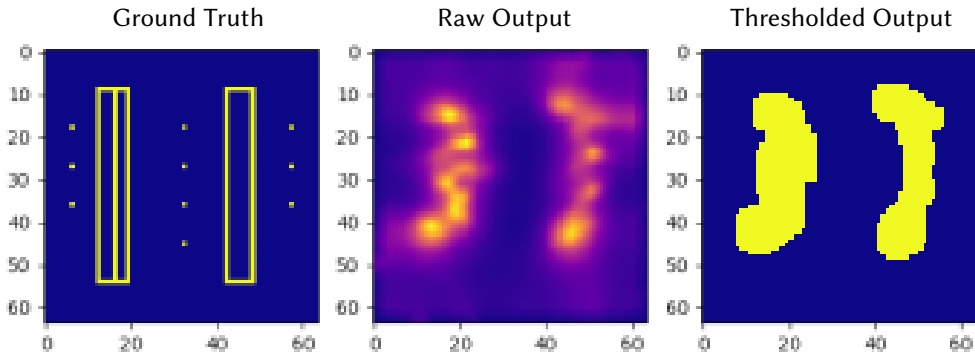


Figure 3: Real World Two Wall Configuration.

room. The first of these routers was 1 meter from the common back wall, the second was 2 meters, the third was 3 meters, and the fourth was 4 meters.

We placed 3 WiFi routers in the office, placing each router 3.75 meters from the line of routers in the hall. We placed these routers at 1, 2 and 3 meters from the back wall.

We executed the algorithm on the data collected from this configuration and produced the image seen in figure 3. Evaluating the displacement error we received a result of 0.294 meters. The algorithm converged correctly on two separate walls.

4.2.3. Corner Wall Configuration

We evaluated the algorithm on a setup consisting of two walls forming the corner of a room. With this setup our objective was to evaluate whether the algorithm could handle the unique signal reflection issues caused by a corner. We also wanted to test whether the degree to which the algorithm would create a clearly defined corner, or whether it would round the corner off and join the two walls into one curved wall. This configuration was also performed in a different area of the building and featured thinner walls. The walls in this setup were brick walls of a more standard thickness of 0.15 meters. Thus we were also checking whether the algorithm could converge correctly with walls that produced substantially less attenuation.

This setup was performed between two rooms and a hallway. The hallway ran along the side of a storage room and into a bedroom. The bedroom was wider than the storage room and thus its width spanned the entire wall of the storage room and the width of the attached hallway, the two walls forming the corner to be simulated were the storage room's bedroom wall and the storage room's hall wall. We placed the WiFi routers such that the door from the hall into the storage room was not included in the measured area, though it was nearby. In all cases we placed the WiFi routers on the ground.

We placed 3 WiFi routers in the hallway, although due to the hallway being small one of these routers is placed just inside the bedroom door. Each hallway router is placed 1 meter from the surface of the hall's storage room wall. The first router was placed 1.5 meters from the extended plane of the storage room's bedroom wall's surface. The second router was placed 0.75 meters from this surface. The third router was placed 1 meter behind this surface, placing it just inside the bedroom. We placed each hall router facing directly away from the hall's storage room wall.

We placed 3 WiFi routers in the storage room. Due to the hall wall being substantially shorter

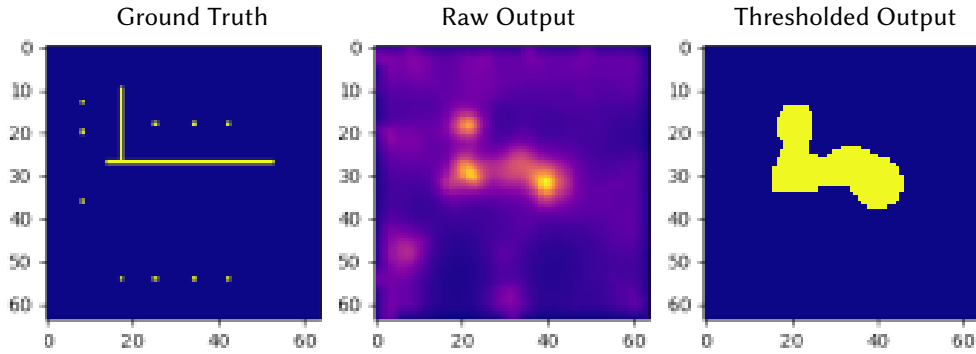


Figure 4: Real World Corner Wall Configuration.

than the bedroom wall, we placed the storage room routers in a line parallel to the bedroom wall. We placed each of these routers 1 meter from the storage room's bedroom wall's surface. We placed the first of these routers 1 meter from the storage room's hall wall's surface. The second router was 2 meters from said surface, and the third router was 3 meters from it. We placed all 3 routers facing towards the bedroom wall. This room featured a metal weights set along the wall between the storage room and the bedroom.

We placed 4 WiFi routers in the bedroom, as this was the largest room. Additionally, one of the hall routers was just within the bedroom. We placed each of these routers 3 meters from the bedroom's storage room wall. This large gap was due to the presence of a bed with its back on the bedroom's storage room wall. We placed the first bedroom router in line with the extended plane of the surface of the storage room's hall wall. We placed the second router 1 meter from this surface, we placed the third router 2 meters, and the fourth 3 meters. This means the second, third and fourth routers in the bedroom were directly opposite the three routers within the storage room. We placed all of the bedroom routers facing towards the storage room wall.

When executing the algorithm on the data collected on this experiment, we produced the image in figure 4. Evaluating displacement error we received a result of 0.714 meters. The algorithm correctly produced a sharp corner. Nothing in the algorithm specifically prefers right angles or corners over curves. Thus the algorithm correctly inferred the structure of a corner correctly from real world data.

4.2.4. Distributed Routers Configuration

In the final configuration we repeated the two wall setup with the WiFi routers in more realistic positions. Instead of placing the routers in lines, we placed them in a more realistic configuration, with routers scattered around the experimental configuration. With this configuration our objective was to evaluate the effect of placing the WiFi routers right next to walls and off ground level. We also left all furniture in situ and attempted to create as realistic a configuration as possible. As such this setup represents a much more realistic environment. The distribution of the WiFi routers relative to the walls can be seen in figure 5, where each dot in the ground truth image represents the coordinates of a router.

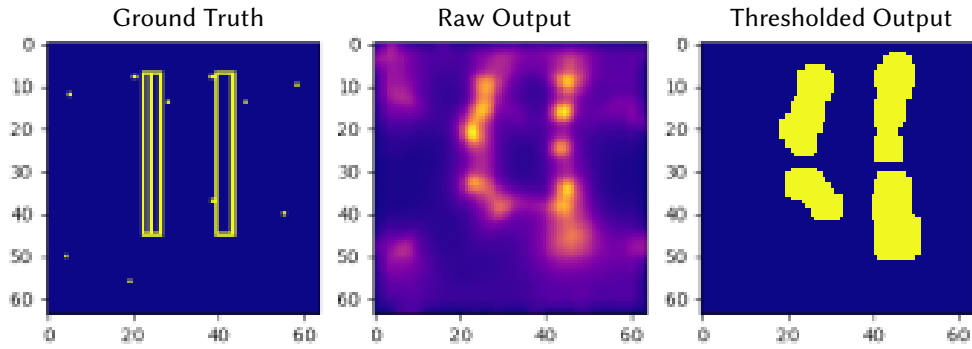


Figure 5: Real World Distributed Routers Configuration.

When we executed the algorithm on the collected data, we produced the image in figure 5. Evaluating the displacement error we received a result of 0.689 meters. A more challenging configuration did not impede the ability of the algorithm to converge and correctly reflect the experiment configuration.

5. Future Work

In the future we will extend our evaluation to cover a full realistic real world setup, using existing WiFi routers in a public building. We have shown the algorithm to work on full architectural plans in simulations, and on building sections in real world configurations. However, a full scale public real world environment poses additional challenges, such as having a lower number of WiFi routers per unit area than in our current experimental setups. To this end we will work on making the algorithm more robust to setups with less routers to draw information from.

6. Conclusion

In this paper we presented an algorithm for passively mapping buildings using WiFi signal strengths. We demonstrated that the algorithm functions on a dataset of 500 simulated buildings. We also demonstrated that the algorithm can resolve walls in four real world experimental configurations. Our algorithm requires the coordinates of the WiFi routers to be known, as well as a set of measurements consisting of the signal strength of those WiFi routers. We do not require to know the coordinates from which the WiFi signal strengths are measured, nor do we require the measurements to be correlated with each other in any way. Each measurement is treated independently.

Acknowledgments

This work was funded by the Science Foundation Ireland Connect Centre.

References

- [1] J. Choi, Y.-S. Choi, S. Talwar, Unsupervised Learning Technique to Obtain the Coordinates of Wi-Fi Access Points, in: 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2019, pp. 1–6. doi:10.1109/IPIN.2019.8911824, iSSN: 2162-7347.
- [2] WiGLE: Wireless Network Mapping, 2001. URL: <https://wigo.net/>.
- [3] G. Jiang, L. Yin, S. Jin, C. Tian, X. Ma, Y. Ou, A Simultaneous Localization and Mapping (SLAM) Framework for 2.5D Map Building Based on Low-Cost LiDAR and Vision Fusion, *Applied Sciences* 9 (2019) 2105. URL: <https://www.mdpi.com/2076-3417/9/10/2105>. doi:10.3390/app9102105.
- [4] R. Liu, S. H. Marakkalage, M. Padmal, T. Shaganan, C. Yuen, Y. L. Guan, U. Tan, Collaborative SLAM Based on WiFi Fingerprint Similarity and Motion Information, *IEEE Internet of Things Journal* 7 (2020) 1826–1840. doi:10.1109/JIOT.2019.2957293, conference Name: IEEE Internet of Things Journal.
- [5] Q. Liang, L. Wang, Y. Li, M. Liu, Indoor Mapping and Localization for Pedestrians using Opportunistic Sensing with Smartphones, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1649–1656. doi:10.1109/IROS.2018.8594254, iSSN: 2153-0866.
- [6] A. J. Ben Ali, Z. S. Hashemifar, K. Dantu, Edge-SLAM: edge-assisted visual simultaneous localization and mapping, in: Proceedings of the 18th International Conference on Mobile Systems, Applications, and Services, MobiSys '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 325–337. URL: <https://doi.org/10.1145/3386901.3389033>. doi:10.1145/3386901.3389033.
- [7] J. Liang, Y. He, Y. Liu, SenseWit: Pervasive Floorplan Generation Based on Only Inertial Sensing, in: 2016 International Conference on Distributed Computing in Sensor Systems (DCOSS), 2016, pp. 1–8. doi:10.1109/DCOSS.2016.25.
- [8] C. Liu, J. Wu, P. Kohli, Y. Furukawa, Raster-to-Vector: Revisiting Floorplan Transformation, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2214–2222. doi:10.1109/ICCV.2017.241.
- [9] N. Amiot, M. Laaraiedh, B. Uguen, PyLayers: An open source dynamic simulator for indoor propagation and localization, in: 2013 IEEE International Conference on Communications Workshops (ICC), 2013, pp. 84–88. doi:10.1109/ICCW.2013.6649206.
- [10] A. Ayanoglu, D. M. Schneider, B. Eitel, Crowdsourcing-Based Magnetic Map Generation for Indoor Localization, in: 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2018, pp. 1–8. doi:10.1109/IPIN.2018.8533832, iSSN: 2162-7347.
- [11] J. Bi, Y. Wang, H. Cao, H. Qi, K. Liu, S. Xu, A Method of Radio Map Construction Based on Crowdsourcing and Interpolation for Wi-Fi Positioning System, in: 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2018, pp. 1–6. doi:10.1109/IPIN.2018.8533749, iSSN: 2162-7347.
- [12] G. Pipelidis, N. Tsiamitros, E. Ustaoglu, R. Kienzler, P. Nurmi, H. Flores, C. Prehofer, Cross-Device Radio Map Generation via Crowdsourcing, in: 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2019, pp. 1–8. doi:10.1109/IPIN.2019.8911766, iSSN: 2162-7347.
- [13] Y. Zhao, Z. Zhang, T. Feng, W. Wong, H. K. Garg, GraphIPS: Calibration-free and Map-

- free Indoor Positioning using Smartphone Crowdsourced Data, *IEEE Internet of Things Journal* (2020) 1–1. doi:10.1109/JIOT.2020.3004703, conference Name: IEEE Internet of Things Journal.
- [14] C. A. Schroth, M. Leng, N. Saba, S. G. Razul, Indoor Human Localization and Interior Structure Mapping using WiFi Signals, in: 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), 2018, pp. 1–5. doi:10.1109/ICDSP.2018.8631675, iSSN: 2165-3577.
- [15] S. Prophet, J. Bao, G. F. Trommer, Comprehensive Floor Plan Mapping for Indoor Exploration and Guidance of Aerial Vehicles, in: 2019 International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2019, pp. 1–8. doi:10.1109/IPIN.2019.8911807, iSSN: 2162-7347.
- [16] S. Chen, M. Li, K. Ren, C. Qiao, Crowd Map: Accurate Reconstruction of Indoor Floor Plans from Crowdsourced Sensor-Rich Videos, in: 2015 IEEE 35th International Conference on Distributed Computing Systems, 2015, pp. 1–10. doi:10.1109/ICDCS.2015.9, iSSN: 1063-6927.
- [17] S. Y. Hayoun, E. Zwecher, E. Iceland, A. Revivo, S. R. Levy, A. Barel, Integrating Deep-Learning-Based Image Completion and Motion Planning to Expedite Indoor Mapping, arXiv:2011.02043 [cs] (2020). URL: <http://arxiv.org/abs/2011.02043>, arXiv: 2011.02043.
- [18] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, X. Li, Jigsaw: Indoor Floor Plan Reconstruction via Mobile Crowdsensing, in: Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, MobiCom '14, ACM, New York, NY, USA, 2014, pp. 249–260. URL: <http://doi.acm.org/10.1145/2639108.2639134>. doi:10.1145/2639108.2639134.
- [19] A. Rosinol, M. Abate, Y. Chang, L. Carlone, Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping, in: 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 1689–1696. doi:10.1109/ICRA40945.2020.9196885, iSSN: 2577-087X.
- [20] H. Shin, Y. Chon, H. Cha, Unsupervised Construction of an Indoor Floor Plan Using a Smartphone, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42 (2012) 889–898. doi:10.1109/TSMCC.2011.2169403, conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
- [21] R. Gao, B. Zhou, F. Ye, Y. Wang, Knitter: Fast, resilient single-user indoor floor plan construction, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9. doi:10.1109/INFOCOM.2017.8057186.
- [22] B. Zhou, M. Elbadry, R. Gao, F. Ye, Towards Scalable Indoor Map Construction and Refinement using Acoustics on Smartphones, *IEEE Transactions on Mobile Computing* 19 (2020) 217–230. doi:10.1109/TMC.2019.2892091, conference Name: IEEE Transactions on Mobile Computing.
- [23] China Building Code, 2000. URL: <http://web.mit.edu/cron/group/chinahousing/english/resources/BuildingCode.html>.