Grasping the Airwaves with a Robotic Wireless Access Point

Emerson Sie
sie2@illinois.edu

Kiran Ramnath
kiranr2@illinois.edu

Kuan-Ying Lee
kylee5@illinois.edu

1. Introduction

Imagine a robotic wireless access point (AP) that can move within the environment in order to maximize the link quality between itself and user devices. Indeed, the benefits of moving WiFi APs within indoor environments has been explored previously [11, 12] where it was found that even horizontal, centimeter-scale mobility along the ground within a 2×2 sq. foot region can result in a noticeable SNR gain of 8 dB for user devices on average across a wide range of indoor settings.

The key challenge towards creating such an AP lies in the non-intuitive and seemingly unpredictable properties of indoor wireless channels. Due to phenomena such as multipath propagation and shadowing by obstacles, it is not necessarily the case that moving closer to the AP or achieving direct line-of-sight with the AP would yield a better channel. Furthermore, due to noise and perhaps transient interferences within the environment, there is not even a guarantee that the channel at a fixed position will remain stationary. The question then becomes one of planning - how should the robotic AP navigate through such a seemingly unreliable landscape?

Inspired by the recent proliferation of low-cost high-DoF personal robots for home environments [5, 14], the success of deep end-to-end policy learning approaches from high-dimensional observations for said robots [15], hardware advances making fast inference practical on embedded devices [3], as well as software patches making high-resolution wireless channel state information (CSI) accessible on modern commodity wireless chipsets [13], we consider various data-driven approaches to solving this problem. In particular, we want to solve the problem of creating a robotic wireless AP that adaptively learns the characteristics of its environment from interacting with it so that it can serve its clients better.

2. Approach

2.1. Assumptions

Environment. We assume a mostly planar indoor environment. In particular, an indoor environment that can be adequately represented by a 2D floor plan, such as a single floor of an office building or an apartment. Our autonomous agent, a robotic AP, is fixed at some suitably chosen position in the environment and configured at a particular channel and bandwidth on some version of WiFi (for example, 802.11ac).

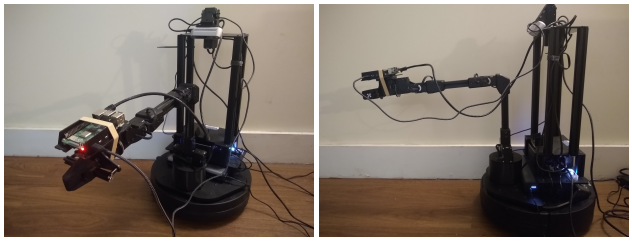

Figure 1. Robotic AP assembly. Note the Raspberry Pi 4 affixed onto the end-effector.

Clients. A set of user devices connected to the robotic AP are distributed within the environment. We can reasonably expect the spatial distribution of these devices to be sparse, as we would expect them to be at specific locations of interest (office desk, living room table, bedside table, etc) rather than uniformly distributed in the environment.

Interferences and Noise. We consider some number of latent interferers or noise sources within or adjacent to the environment which we have limited control over. These interferers can be adjacent APs within the environment (for example on different floors or nearby houses) occupying the same spectrum as our robotic AP and located within carrier sensing range, the client devices connected to the said interfering AP, moving humans within the environment, and so forth.

Mobility. We assume mobility of clients or interferers only on coarse-grained time scales and that for the majority of the time, devices can be assumed to be mostly fixed within the environment. This is a reasonable assumption, since we would expect users to only move their devices intermittently rather than continuously in indoor settings. This encompasses cases such as when users move their laptop from the living room to the dining table where it remains stationary, or when they move a wireless appliance

like a video game console from one room to another where it likewise remains fixed. For the robotic AP, we consider the regime of *tethered mobility* as defined in [11]. That is, the robotic AP is mostly confined to a small region but is free to move within it otherwise. This is so that there is enough flexibility to rectify multipath and shadowing effects, yet remain practical for crowded indoor spaces.

2.2. Problem Formulation

Environment State. We identify the relevant factors in the environment for our problem by conceptualizing the space of environment states as $\mathcal{S} = \mathcal{S}_{AP} \times \mathcal{S}_C \times \mathcal{S}_I$.

- $\mathcal{S}_{AP}$ is the robotic AP's configuration space or workspace. Due to our focus on tethered mobility this is largely dominated by the space of possible end-effector poses.
- $\mathcal{S}_C$ is the space of possible configurations of the user devices in the environment. In particular, this encompasses their possible positions and orientation within the environment as well as whether they are currently active or dormant.
- $\mathcal{S}_I$ is similar to $\mathcal{S}_D$, but with interferers and noise sources within the environment instead of user devices.

Observations. We consider the agent's observation of the environmental state at time t as $o_t = (T_t, H_t)$.

- $T_t \in \mathbb{R}^D$ represents the robotic AP's estimation of its own pose at time t, which we assume has D degrees of freedom. This value is an estimate of the robotic AP's true pose due to sensor and actuation error within the motors of the robotic AP.
- $H_t \in \mathbb{C}^{C \times N \times M \times K}$ is a tensor representing the measured channel from each client device at the robotic AP's receiver. Following the convention in [16], C represents the number of user devices that the robotic AP serves, N is the number of antennas at the robotic AP's receiver, M is the number of antennas used at the transmitter, and K is the number of subcarriers in the wireless channel which depends on the configured bandwidth of the AP. The observed channel from each user device features ambient noise and various interferences from the environment, and thus can be viewed as an estimation of the true underlying channel.

Channel Quality. The goal of the robotic AP should be threefold. First, it should maximize the channel quality to clients as measured at its own receiver. Secondly, it should minimize the effect of the latent interferers and noise sources within the environment. Thirdly, it should achieve some notion of fairness towards each client. Hence, we require a metric $Q : \mathbb{C}^{C \times N \times M \times K} \to \mathbb{R}$ to rate the quality of an observed channel H_t.

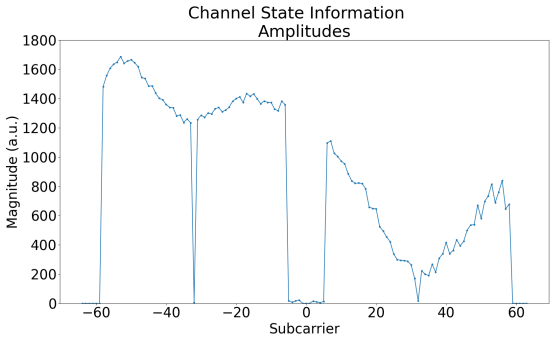

Figure 2. Example 40MHz CSI vector obtained from a single frame, where the user device was placed adjacent to the AP. Note that the channel experiences deep fading at around subcarrier 32.

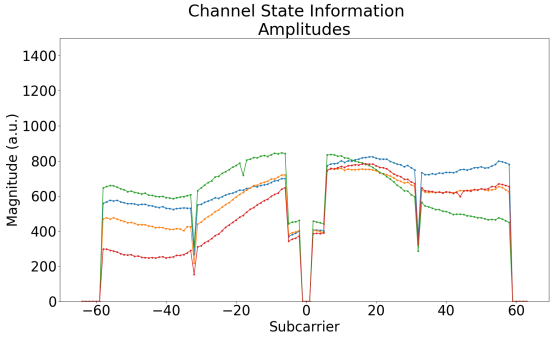

Figure 3. Variation of the 40MHz CSI to one client device for 4 poses obtained by varying the end-effector roll from $-\pi/2$ to $\pi/2$.

Since OFDM coded frames share the same modulation scheme across all subcarriers, it can be argued that when significant SNR variation occurs across subcarriers in a wideband OFDM channel due to frequency-selective fading as in Figure 2, the effective SNR is bottlenecked by the lower SNR subcarriers [10]. This suggests that subcarriers with lower SNR should have a larger contribution to the channel quality. We thus consider two metrics - the first is obtained by simply summing the magnitude of the channel coefficient at each subcarrier, whereas the other is obtained by taking the minimum.

To mediate the effects of noise within the environment, we can use the standard additive white Gaussian assumption which asserts the noise is zero-mean and stationary. This means we can reasonably expect to get a good estimate of the true underlying channel by time averaging across many samples.

As far as fairness is considered, we can safely assume that each user device should be served equally, as is standard practice in system design.

Bringing the above discussion together, and using a

SISO-only system ($N = M = 1$) as an example so that $H_t \in \mathbb{C}^{C \times K}$ for simplicity, this leads us to the following instantaneous quality metrics before time-averaging.

$$Q(H_t) = \frac{1}{CK} \sum_{c=1}^{C} \sum_{k=1}^{K} \|H_{t,c,k}\| \tag{1}$$

$$Q_{\min}(H_t) = \frac{1}{C} \sum_{c=1}^{C} \min_{k=1}^{K} \|H_{t,c,k}\| \tag{2}$$

Task Setup. We consider our robotic AP as an agent in an episodic environment. The beginning of each episode is triggered when the observed channel at the robotic AP changes significantly as a result of a change in environmental state that is extrinsic to the robotic AP. For example, this state change can be caused by either the user devices shifting, devices turning on or off, intereferers moving away from the range of the robotic AP, and so on.

At each time t within the episode, the robotic AP receives observations of the environmental state o_t, and must choose some action a_t. The objective is to learn a policy $\pi(a_t|o_t)$ that guides the robotic AP to some optimal pose, where it terminates the episode and remains fixed until the beginning of the next episode. Note that it would not be necessary to run the policy perpetually given our assumption that the environmental state is not constantly changing, but only changing at coarse-grained time scales.

2.3. WiFi Regrasping

Our key insight is that given a fixed position of a client, the agent can observe whether a given action leads to a better or worse channel quality. We can use this self-supervision, combined with the idea of war-driving (sampling the channel quality at certain carefully selected points within in the environment) to generate a dataset to train an action-conditional model of the environment. We can then use this model as a basis for the policy for that environment. Thus, we consider using the approach as outlined in [8, 17]. That is, we treat the problem as some form of a model-based WiFi regrasping task.

Dataset Generation. To generate the dataset, we identify several broad regions of interest within the environment (for example, desk, living room table, bedroom, etc). For each region of interest, we sample some number of locations within the region of interest, each of which forms a data point in our dataset. Each of these regions of interest can be viewed somewhat analogously to a distinct object in a grasping task. Extending the same analogy, this means that locations within the same region of interest can be viewed somewhat analogously to the orientations or views of the same object when it is presented to the grasping agent.

For each of these locations, we collect tuples of experience of the form (o_t, a_t, o_{t+1}), where a_t is a randomly sampled action from a standard Gaussian distribution, scaled by joint-specific multipliers. We ensure that the action is not too large and does not drive the robot to an invalid configuration. We compare the difference between the channel quality before and after executing the random action in order to assign the reward of executing the action, r_t. A positive reward implies the grasp was successful.

Model Architecture. We train a model f_{REGRASP} to predict whether a given action at a certain state leads to a better outcome for a single client. Formally $f_{\text{REGRASP}}(T_t, H_{t,c}, a_t) \in \{0, 1\}$ represents the probability whether there will be an increase in channel quality towards a client c given that the current pose is T_t, the current observed channel quality to said client is $H_{t,c}$, and we take the action a_t. We parameterize the model as a deep neural network capped with a sigmoid unit, whose architecture is shown in Figure 4.

Regrasping with Stochastic Search. To evaluate the learned model f_{REGRASP}, we first place clients at a subset of the test locations. To mark the beginning of an episode, we first reset the robot arm to go to the home position. Running the policy for each step within the episode entails choosing the action that maximizes the expected increase in channel quality to all clients. In other words,

$$a_t = \underset{a}{\operatorname{argmax}} \sum_{c=1}^{C} f_{\text{REGRASP}}(T_t, H_{t,c}, a) \tag{3}$$

In practice, in order to choose a_t we sample a set of randomized actions (using the same distribution and constraints that were used within the training phase) rather than exhaustively searching through the entire action space. The episode is terminated when none of the sampled actions lead to an improved quality channel according to the model (see Algorithm 1).

3. Results

3.1. Experimental Setup

In our experiments we use the LoCoBot [5] with either a Raspberry Pi 4 [6] or Nexus 6P phone fixed to its end-effector to emulate the robotic AP.

The LoCoBot features a mobile Kobuki base [4] and a 5-DoF WidowX 200 Mobile Robot Arm [7] with a maximum payload of 200g. To manipulate the robot, we use the PyRobot [18] library.

Both the Nexus 6P and the Raspberry Pi 4 features Broadcom wireless cards. In particular, they feature the

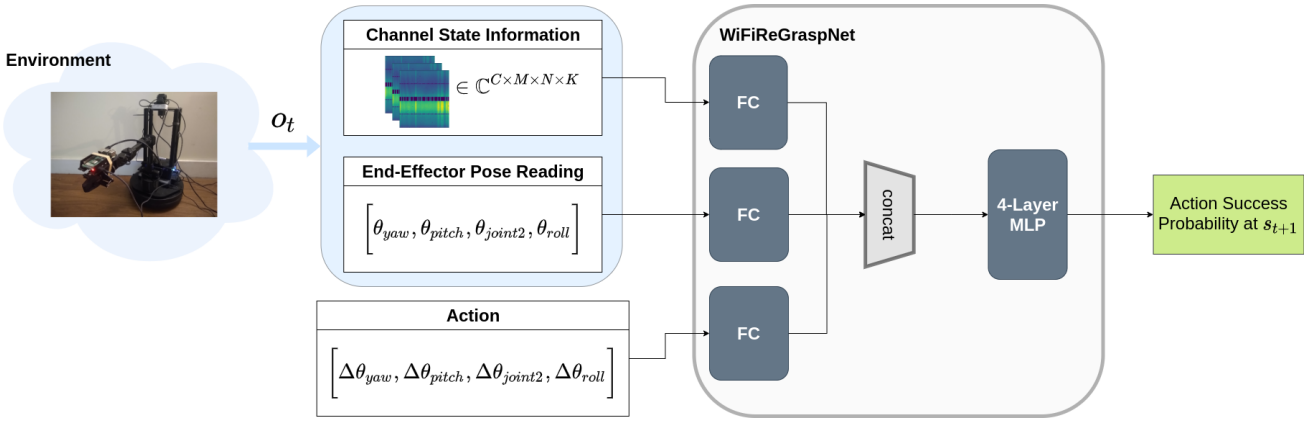

Figure 4. The action-conditioned WiFi grasper model architecture.

Algorithm 1: Stochastic Greedy Search using WiFi Regrasper

for $1 : n$ **do**
  Initialize candidate-action set $S = \{\}$;
  while $|S| < m$ **do**
    a = SAMPLE();
    if *VALID(a)* **then**
      $S \leftarrow S \cup a$;
    end
  end
  Obtain grasp success probability $p(a) = f_{\text{REGRASP}}(o_t, a)\ \forall a \in S$;
  if $\nexists\, p(a) \geq 0.5$ **then**
    EXIT;
  else
    $b = \text{argmax}_{a\in S}\, p(a)$;
    Command action b on end-effector;
  end
end

BCM4358 [2] and BCM43455c0 [1] respectively. The drivers of these devices can be patched using the Nexmon project [19, 13] in order to extract CSI from its wireless interface on a per-frame basis. This tool reports the real and complex components of the CSI values with up to 14 bits of resolution, but the unit of normalization is undocumented.

Since CSI extraction requires the wireless interface to function in monitor mode, it cannot function as a true wireless AP since `hostapd` requires the interface to be in AP mode. To mitigate this shortcoming we use a separate wireless router, an ASUS RT-AC86U, to act as the 802.11ac AP. We place this router adjacent to the LoCoBot. The firmware on the RT-AC86U allows us to set the channel and bandwidth of the wireless network. In our experiments, we use channel 100 and a bandwidth of 40MHz unless otherwise noted.

The client devices used are Raspberry Pis scattered throughout the indoor environment. These Raspberry Pis are connected to the wireless network of the RT-AC86U and periodically generate frames at a fixed rate using the `ping` utility. The Raspberry Pi or Nexus phone at the LoCoBot can be configured to sniff frames originating from up to a maximum of four clients simultaneously and dump the CSI corresponding to each sniffed frame. Since both the client devices are Raspberry Pis which only feature a single antenna, we only consider SISO/SIMO channels.

3.2. Data Collection

All data was collected within a 20m $\times$ 10m apartment. The three regions of interest chosen are `kitchen`, `bed`, and `desk`, all of which are roughly equidistant (around 5m away) from where the robotic AP was placed. The `bed` locations feature a direct line-of-sight (LoS) path to the robotic AP, whereas both the `desk` and `kitchen` locations are occluded due to being in a different room. Within each region, we sample 4 uniformly spaced client locations roughly a meter apart. For each client location, we collect 1000 agent-environment interactions. When sampling the channel, we time-average over a 100 frames, which the clients send at a rate of 20 Hz.

There are some caveats to keep in mind. First, due to accumulated actuation error from the weight of the monitor device placed on the end effector of the LoCoBot arm, it was necessary to reset the arm to the home position around every 10 interactions. Secondly, due to the tendency of the end effector to collide with the ground when the first joint of the arm is enabled, we do not use the first joint of the arm. That is, we restrict the range of motion of the arm to that of a 4-DoF arm. Thirdly, due to the presence and movement of human occupants in the environment, the data is expected to

Table 1. Regrasping model accuracy (mean $\pm$ stdev for $K = 3$)

Setting	Training Accuracy	Validation Accuracy
All combined	73.18 ± 0.002	72.85 ± 0.005
Bed	73.18 ± 0.007	71.84 ± 0.014
Desk	72.53 ± 0.011	70.30 ± 0.017
Kitchen	70.64 ± 0.008	68.06 ± 0.016

be somewhat noisy.

In total, the dataset contains around 12000 interactions and takes up around 1.3 GB of space. Since each agent-environment interaction takes about 4 seconds for the LoCoBot to enact, the total time taken to collect the dataset was around 13 hours.

3.3. Mapping the Channel Landscape

Similar to [11], we first figure out how the channel varies across the configuration space of the robot given a fixed client. For each of the sets of 1000 sampled poses in the dataset corresponding to a single client position, we compute the channel quality metric at each pose and use this to create the histograms in Figure 6 and Table 2.

Next, we keep the robotic AP static and instead vary the position of the client device. We consider three cases. First, we verify that the CSI remains stationary when the client is not moving. Secondly, we consider how the CSI changes when the client is moved from an initial spot to a nearby adjacent spot about 10cm away. Third, we consider the effect of various transient interferers and noise sources in the environment. In our case, we consider interference from moving humans, which is reasonable for indoor environments. The results are shown in Figure 5.

3.4. Regrasping the Airwaves

Training. We train the model described in Fig. 4 using K-fold cross validation ($K = 3$). Each fold splits the dataset into train and test sets in the ratio $3 : 1$. Fully-connected layers use ReLU activation. The inputs are fused via late-fusion. We see that this stabilizes the training variance although it does not improve the model performance itself. Adam optimizer with learning rate $1e-3$ trains each network for 500 epochs. This takes less than 5 minutes on a CPU for all three folds combined. The accuracy of the model can be seen in Table 1. The accuracy gap between training and validation is small, suggesting scope for using more capacity. However, attempts at doing so in the form of deeper networks and skip-connections or using a learning-rate schedule did not provide any benefits.

During deployment, we set the stochastic greedy search-depth $n = 10$ and size of candidate action set $m = 10$.

Evaluation. For simplicity during the evaluation, we only consider the case of a single client as it would be trivial but tedious to generalize to multiple clients. For each test case, we choose a random location in a chosen region of interest (either `bed`, `desk`, or `kitchen`) that does not appear in the dataset and run the stochastic search policy induced by our trained model (Algorithm 1).

For each test case, we compare the quality of the final channel at the end of each episode with the initial channel quality at the robotic AP's home position. We summarize the results in Table 3. Additionally, we compare the terminal channel qualities at the end of each episode with the distribution of attainable channel qualities for each region as shown in Figure 6 and Table 2. The results of this are shown in Table 4.

4. Discussion

4.1. How Useful are Channel-Derived Features?

Channel Sensing. One of the striking features of our agent is that it does not use any features other than the channel characteristics to each client to infer what actions should be performed at any given moment. That is, the agent does not know or even attempt to estimate the states of the clients or of the interferers within the environment. In order for this to be feasible, it is necessary that the channel contains meaningful semantic information about the environment that the agent can use to achieve its goal.

Unlike visual features commonly used in computer vision which are based on the visible part of the electromagnetic spectrum, humans do not have an intuitive understanding of the propagation characteristics of wireless signals in the radio spectrum. Adding to the lack of understanding of these wireless signals is the fact that credible, high-quality simulators of radiowave propagation, especially for indoor environments, are sparse to non-existent. Yet, it can be argued that with the advent and rollout of 5G networks across the world, such signals will become more and more ubiquitous, both within indoor and urban environments.

Thus, it is beneficial to understand the characteristics of such signals so that agents can leverage them as a sensing modality beyond vision. Just as the success of convolutional neural networks in computer vision was informed by an understanding of the workings of human perception (spatial locality, translational invariance, and so on), it is conceivable that a similar level of understanding of the idiosyncrasies of wireless signals will lead to novel and interesting model architectures for such signals.

Sensitivity to Orientation. As is shown in Figure 5, the measured channel is sensitive even to the mere turning of the end-effector. This confirms the notion that the channel state in indoor environments is highly sensitive to not

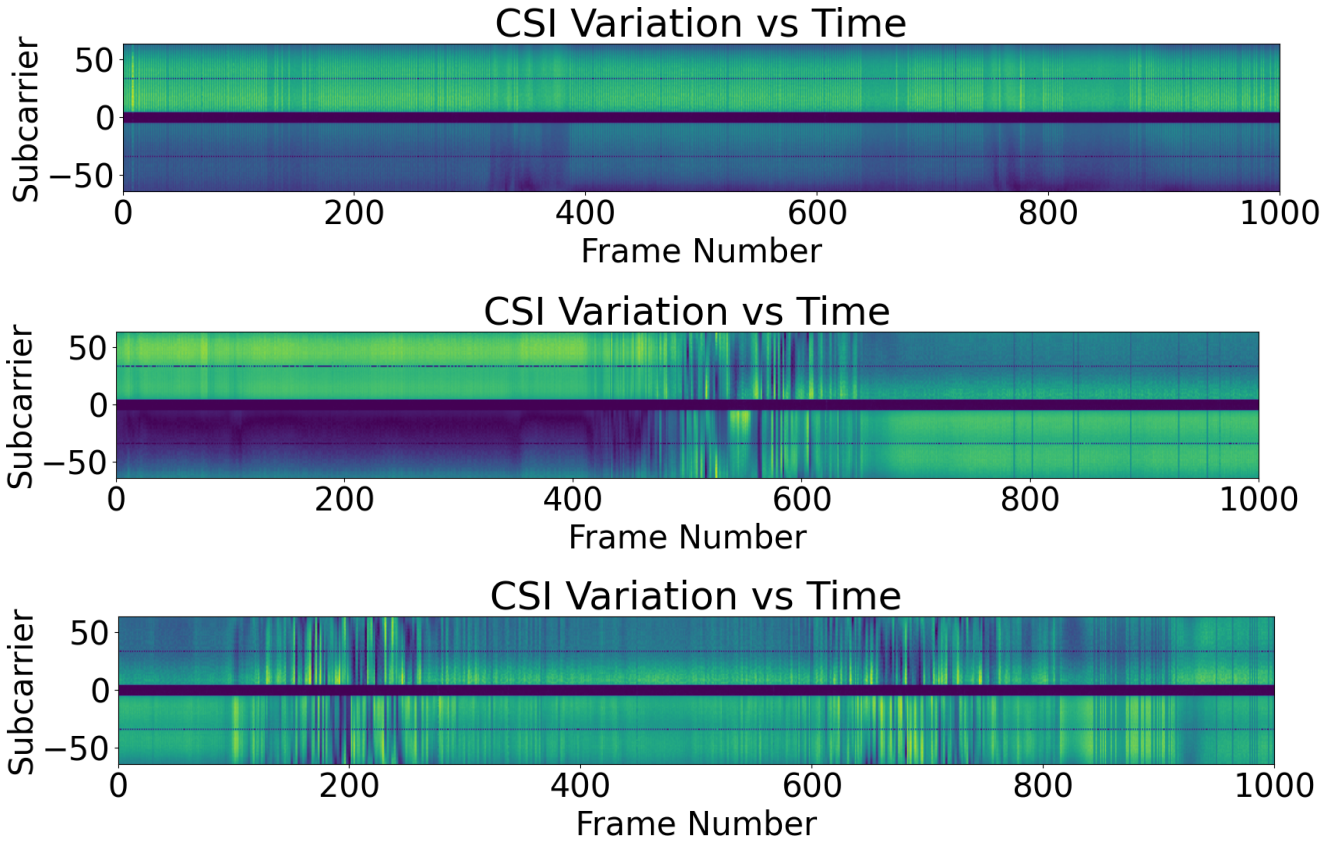

Figure 5. Variation of the channel state vs. time under three different scenarios. (Top) When the client device is completely stationary, the CSI remains stable over time. (Middle) When the client is moved to an adjacent spot, the channel is momentarily perturbed before again becoming stationary. (Bottom) In this trial, a human walks forwards and backwards once between the client device and the robotic AP while both are held static. The forward and backward pass is clearly visible in the variation of the CSI.

Table 2. Distribution of channel quality across the configuration space for each fixed client position.

Metric		Bed				Desk				Kitchen			
		0	1	2	3	0	1	2	3	0	1	2	3
Q	Mean	590.67	576.94	580.34	566.75	565.42	602.87	623.59	595.00	573.09	585.81	575.93	579.48
	Stdev	55.60	54.51	56.00	58.10	61.87	54.07	56.96	57.24	54.50	54.23	58.62	56.52
$Q_{\min}$	Mean	234.23	184.50	207.95	154.20	174.78	219.28	246.45	213.45	97.23	132.59	137.50	102.85
	Stdev	74.37	90.39	76.72	100.39	84.97	70.97	66.29	70.56	95.75	100.70	102.83	98.77

just centimeter scale lateral motion, but rotational motion as well. In our context, this means that an agent can be reasonably assured that a significant change in the channel implies that either motion of the client or the agent itself has taken place.

Stability of Channel Features. In order to utilize the channel for inference, it is necessary for it to have stable features that do not vary inexplicably.

As is shown in the top part of Figure 5, when neither the robotic AP nor the client device is moving, the channel remains stationary. This is crucial since an agent that builds its state representation of the environment based on the channel can develop a sense of object permanence. In particular, if a client is fixed in the environment and the agent observes the channel to the client, if the agent returns to the same configuration previously, it can expect to detect the same channel.

In the middle part of the figure, we consider how the channel varies when motion is in play. In particular, we are interested in whether after either the client or agent settles after a brief period of motion, we can expect the channel to resume its stationary properties as discussed previously. The figure confirms this, as the period of motion can be

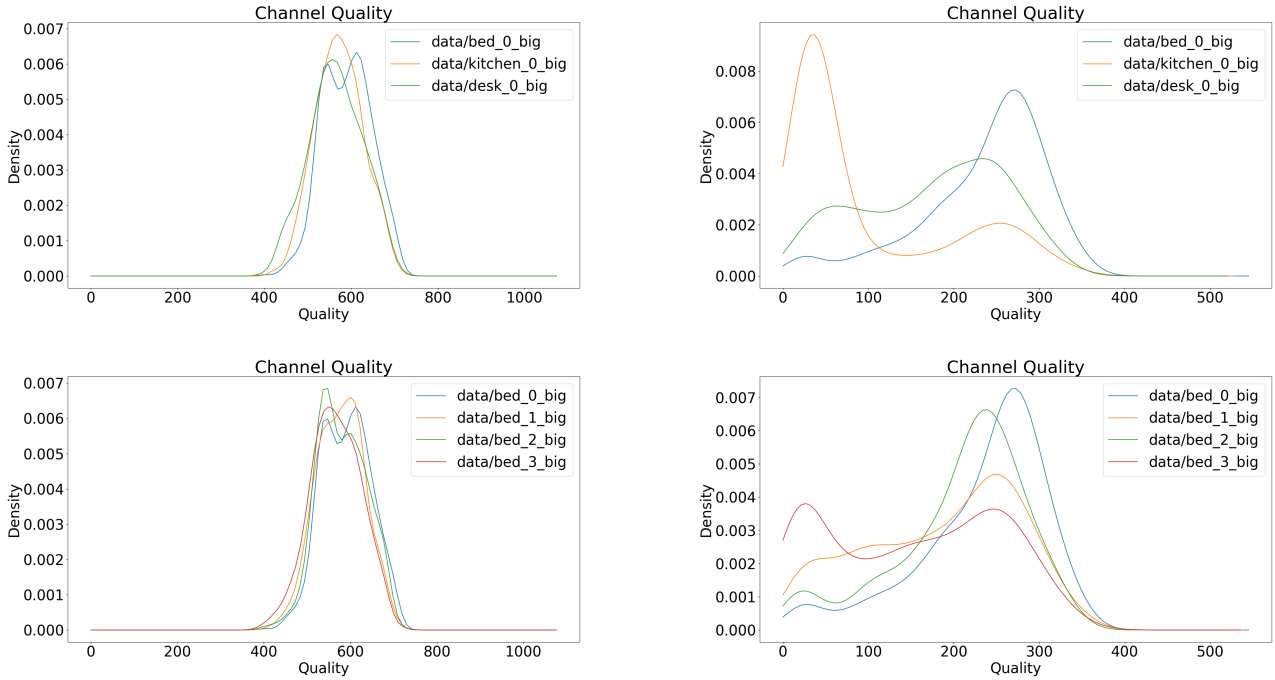

Figure 6. The distribution of channel quality over randomly sampled poses in the robotic AP's configuration space. (Top) Inter-regional distribution. (Bottom) Intra-regional distribution for a chosen region (`bed`). (Left) Using metric Q. (Right) Using metric $Q_{\min}$.

Table 3. Regrasping trials for 5 random locations in each region that are not inside the dataset.

Test Case		Q_{init}	Q_{term}	ΔQ	# Iterations
Bed	1	539.71	665.50	+125.79	1
	2	529.89	633.17	+103.18	2
	3	526.66	618.84	+92.18	1
	4	548.56	625.31	+76.75	3
	5	525.34	653.60	+128.26	1
Average				+105.232	1.6
Desk	1	542.02	606.89	+64.87	2
	2	534.97	676.39	+141.42	1
	3	520.97	661.84	+140.87	1
	4	549.32	660.49	+111.17	1
	5	569.19	694.40	+125.21	3
Average				+116.71	1.6
Kitchen	1	525.92	651.35	+125.43	3
	2	550.06	669.20	+119.14	2
	3	505.29	635.98	+130.69	4
	4	464.89	621.54	+156.65	1
	5	571.56	624.97	+53.41	2
Average				+111.41	2.2

clearly discerned to be in the middle of the time interval.

Lastly, we consider how the channel is affected by transient interferences or noise sources within the environment. We consider a human as one such source since that is a reasonable assumption for liveable spaces. As the figure shows, when a human moves in between the LOS path between the client and the agent, the channel is perturbed momentarily before coming back to rest.

Table 4. Optimality of the pose at episode termination.

	Bed	Desk	Kitchen
Mean Q_{term} Percentile	80.25%	86.4334%	85.79%

How Predictable is the Channel? Perhaps the most favorable structure that we would like the channel to have is continuity. That is, it varies continuously with regards to perturbations in the environmental state, i.e. small charges in the environmental state lead to similarly small changes in the observed channel. This would allow for topological approaches to planning over the channel space, such as [9].

Clearly, this does not always hold due to phenomena such as wireless shadowing, which can cause an abrupt dropoff in channel quality due to environmental obstructions, but it is worth noting that in many cases, an assumption of continuity can prove useful.

Perhaps the most compelling piece of evidence of this assumption being useful in a local sense is illustrated in

Figure 6. Notice that in the right side of the figure, the $Q_{\min}$ profiles across different regions differ, but the $Q_{\min}$ profiles for locations in the same region have the same shape. This observation is consistent with [20], which is an indoor localization scheme which leverages the fact that the multipath propagation characteristics towards adjacent locations are somewhat similar and can be correlated using Dynamic Time Warping (DTW). This also helps to explain why the regrasping model trained on sampled locations within each region is effective for test locations in the same region.

4.2. How does the Regrasper Stack Up?

Quite remarkably, a model trained on CSI data from diverse locations does better than a model trained on data from only one location (see Table 1). This suggests that the function $f_{\texttt{REGRASP}}$ learns to generalize the behaviour of wireless signals in an indoor environment.

On deploying it in a search-and-rank fashion, the search always terminates with a positive change to the channel quality within a reasonable number of steps for each test-case (see Table 3). Given that each client interaction takes roughly 4 seconds, this improvement in quality happens almost instantly. In contrast, using exhaustive search would require commanding hundreds of actions in a naive fashion, each accompanied with a delay in recording the channel state and possible accumulation of actuation errors, before finally making a decision on the best outcome. Furthermore, the speed of the regrasping approach does not compromise on the final optimality. In each client-location, the final link quality ranks high when compared to the distribution observed at that location (see Table 4 and Figure 6).

4.3. What's Missing?

DDPG. We implemented a Deep Deterministic Policy Gradient (DDPG) network using codes provided at `https://github.com/ghliu/pytorch-ddpg`. It takes the current pose along with the current CSI as input (113 dimension) and its goal is to predict the action (5 dimension) for controlling the robot. It is composed of two fully connected layers of size 400 and 300. We attempted normalization on the same dataset as Regrasper Network as we found the values of CSI have a wide range in different settings.

While we saw the training loss decreased as number of steps increased, the model failed to predict reasonable actions in real robot testing, outputting the same action regardless of the current CSI and robot position.[1]

MIMO. Although we had a Nexus 6P which featured 2 antennas and can thus derive a richer state representation of the environment, its increased weight (178g) compared to the Raspberry Pi (50g) meant that the scale of the actuation errors in the robot arm increased. Out of fear of doing damage to the robotic arm due to it being near the maximum load of the arm (200g), it was decided that the Raspberry Pi would be used instead.

[1] I guess that's why they call it deterministic.

5. Conclusion

This work studies the use of a model-based deep network to the problem of improving WiFi link quality to clients in an indoor environment. Through a rigorous characterization of the Channel State Information (CSI), we identify and leverage this as a reliable source of state information for approaching this problem. An action-conditioned binary classifier deployed in a stochastic greedy search is shown to consistently improve link quality to a client across diverse locations. Further avenues for improvement were identified (Section 4.3). We foresee this approach generalizing well to other problems such as indoor localization. Through this work, we hope to motivate research towards more application-driven robot-learning use-cases.

6. Statement of Individual Contributions

Emerson:

1. Acquiring and deploying all the hardware.
2. Writing software (gym interface) to interface with low-level hardware.
3. Wrote scripts that visualize the channel space and generate figures.
4. Wrote introduction, problem formulation, experimental setup, data collection, discussion.

Kiran:

1. Scripts for data gathering and massaging.
2. Designing and training WiFi Regrasper Network.
3. Deploying network with stochastic greedy search on the LoCoBot.
4. Parts of final report to do with the model.

Kuan-Ying:

1. Implemented DDPG and bridged it to the framework.

References

[1] Broadcom BCM43455c0. `https://www.cypress.com/file/298786/download`.

[2] Broadcom BCM4358. `https://www.broadcom.com/products/wireless/wireless-lan-bluetooth/bcm4356`.

[3] Jetson TX-2. https://developer.nvidia.com/embedded/jetson-tx2.

[4] Kobuki Base. http://kobuki.yujinrobot.com/about2/.

[5] LoCoBot - An Open Source Low Cost Robot. http://locobot.org.

[6] Raspberry Pi 4. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.

[7] WidowX 200 Mobile Robot Arm. https://www.trossenrobotics.com/widowx-200-robot-arm-mobile-base.aspx.

[8] Roberto Calandra, Andrew Owens, Dinesh Jayaraman, Justin Lin, Wenzhen Yuan, Jitendra Malik, Edward H. Adelson, and Sergey Levine. More than a feeling: Learning to grasp and regrasp using vision and touch. *CoRR*, abs/1805.11085, 2018.

[9] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *CVPR*, 2020.

[10] Lara Deek, Eduard Garcia-Villegas, Elizabeth Belding, Sung-Ju Lee, and Kevin Almeroth. A practical framework for 802.11 mimo rate adaptation. *Computer Networks*, 83:332 – 348, 2015.

[11] Mahanth Gowda, Ashutosh Dhekne, and Romit Roy Choudhury. The case for robotic wireless networks. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 1317–1327, Republic and Canton of Geneva, CHE, 2016. International World Wide Web Conferences Steering Committee.

[12] Mahanth Gowda, Nirupam Roy, and Romit Roy Choudhury. Infrastructure mobility: A what-if analysis. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, HotNets-XIII, page 1–7, New York, NY, USA, 2014. Association for Computing Machinery.

[13] Francesco Gringoli, Matthias Schulz, Jakob Link, and Matthias Hollick. Free your csi: A channel state information extraction platform for modern wi-fi chipsets. In *Proceedings of the 13th International Workshop on Wireless Network Testbeds, Experimental Evaluation & Characterization*, WiNTECH '19, page 21–28, 2019.

[14] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *CoRR*, abs/1807.07049, 2018.

[15] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

[16] Yongsen Ma, Gang Zhou, and Shuangquan Wang. Wifi sensing with channel state information: A survey. *ACM Comput. Surv.*, 52(3), June 2019.

[17] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *CoRR*, abs/1703.09312, 2017.

[18] Adithyavairavan Murali, Tao Chen, Kalyan Vasudev Alwala, Dhiraj Gandhi, Lerrel Pinto, Saurabh Gupta, and Abhinav Gupta. Pyrobot: An open-source robotics framework for research and benchmarking. *arXiv preprint arXiv:1906.08236*, 2019.

[19] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. Nexmon: The c-based firmware patching framework, 2017.

[20] Jue Wang and Dina Katabi. Dude, where's my card? rfid positioning that works with multipath and non-line of sight. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, page 51–62, New York, NY, USA, 2013. Association for Computing Machinery.