1 Introduction

1.1 Motivation and Related Work

Can we turn our smartphones into multi-purpose scanning devices, that can detect tumors, locate objects and ‘see’ beyond walls? Radio waves get reflected by the environment arrive back at the transmitter, where the nature of each reflected wave depends on the reflecting surface. This suggests the possibility of learning the physical environment by detecting properties of backscatters. For example, the time-delay and angle of arrival can be used to locate the reflecting object, and the difference between reflection coefficients can be used to distinguish between materials.

Realization of a scanning device based on this principle involves two main challenges:

1. Detecting and measuring backscatter itself: the received signal is prone to a large amount of self-interference caused both by leakage of the transmitted signal in the receiver, and by the different components in the backscatter itself.

2. Estimating the number of reflectors and the parameters of each reflection: amplitude, delay, angle of arrival and phase-shift.
The Networked Systems Group at Stanford University (Bharadia, Joshi and Katti) has recently reported initial progress related to these tasks [3]. At its current stage the project makes use of a Matlab model of a backscatter channel. The simulation produces the signal expected at the receiver given a number of reflective surfaces. It is possible to set the number of reflectors and tune the following parameters per each one of them: amplitude, delay, phase, angle of arrival (AoA). Other related works are the Wi-Vi [1] and Wi-See [9] projects. Wi-Vi uses a wide-band transmitted signal and iterative cancellation of reflections from the static environment to track moving objects. Wi-See relies on the Doppler effect in order to identify movement.

The current algorithm for backscatter channel modeling implemented by SNSG assumes knowledge of the transmitted pulse. As a first step, it estimates the composite channel for each antenna in a MIMO array (the superposition of all the reflections). This is a simple deconvolution of the received signal with the known transmitted signal. Later on the composite channel is modeled as a combination of reflected pulses. The subchannel phases, delays, gains and angles are estimated by solving a multi-dimensional optimization problem. An assumption is made regarding the number of reflectors to construct the model of a new environment. If for a certain assumed number of reflectors the algorithm doesn’t converge, another attempt is made with a different number. We can therefore significantly speed-up performance if we had a way to approximate the number of reflectors prior to fitting the model parameters. According to feedback provided by Kiran Joshi (leading the project in SNSG), approximation of the number of dominant reflectors (even a not very accurate one) could speed up the channel modeling algorithm convergence.

We aim at extracting useful information using simpler techniques, without complicated channel modeling and working with relatively raw data. Our premise is that while the received signal contains a lot of information, whenever a single parameter is changed over multiple measurements, the intrinsic structure may be revealed by means of dimensionality reduction. One particular advantage of applying dimensionality reduction to high-dimensional data is enabling data visualization. Examining a 2-dimensional or 3-dimensional rep-
Figure 1.2: Example of a non-linear manifold actually having only two dimensions.

representation of the data can tell us whether we are in the right direction in our feature selection and reinforce or disprove a decision to take a certain path. We are focusing on two main tasks:

1. Blind detection of the number of dominant reflective surfaces, alleviating the need to test multiple possibilities.

2. Simplifying gesture detection by finding a low-dimensional manifold or curve representation of a signal reflected from different angles.

1.2 Dimensionality reduction via manifold learning

Dimensionality reduction refers to the task of extracting information from a set of data points with a large environmental dimension. The relevant information can be, for example, a varying parameter in the model that uses to create the data, which in our case is the number of reflectors in the channel. Manifold learning approach to dimensionality reduction assumes the data points lie on a low dimensional manifold in some high dimensional space. “Learning” the manifold refers to successfully mapping it into a space with the right number of dimension, in a way that preserves its topological structure (homeomorphism). As an example, consider the “swiss-roll” in Figure 1.2. Given a set of data points taken from the roll shaped manifold on the left, learning the manifold means finding a mapping $\mathbb{R}^3 \rightarrow \mathbb{R}^2$ that preserves the local manifold structure, i.e. that “unrolls” the roll. Note that any linear map into a two dimensional plane fails to do that: the image of such linear map necessarily contains two adjacent point which are far away in the original space. It follows that performing coarse clustering on such a linear image of the points would group together samples that are actually located on the opposite ends of the strip, whereas performing clustering on its embedding yields correct classification of the points (as can be seen from the coloring).

If the raw samples are not naturally mapped into a metric space, an intermediate step is necessary. We need to map the raw samples into a high
dimensional metric space in which each coordinate can be a feature, or a property, of the sample. This mapping is called feature extraction and we denote it by \( f \). In our case, where each sample is an output of a backscatter channel, possible features can be: the variance of the sample, its \( r \) bins histogram and the \( l \) bins histogram of its Fourier transform. The image of a sample \( y_i, i = 1,\ldots,N \), under this mapping is called the feature vector, which represents a point \( f(y_i) \) in the feature space. The feature extraction step should preserve enough information on the data so that classification is still possible. Therefore, there should be a relatively large number of features, making the features space a high dimensional space. This is the space where the actual learning is carried out.

Manifold learning approach to our detection problem has the following outline:

1. Map each sample \( y_i, i = 1,\ldots,N \) to its corresponding features vector \( z_i = f(y_i) \) in the feature space of dimension \( n \).

2. Embed the points in the feature space \( z_1,\ldots,z_N \) onto a low dimensional Euclidean space \( \mathbb{R}^k, k \ll n \), in a way that preserves the local manifold structure which \( z_1,\ldots,z_N \) are assumed to be part of.

The problem is solved once we manage to obtain a satisfactory spacial separation between the embedded points that corresponds to different number of reflectors. In this case, any point in \( \mathbb{R}^k \) is associated with some number of reflectors, and new data points can be classified according to their position.

1.3 Diffusion Maps

In this work we use diffusion maps as the manifold learning technique. Diffusion maps is a relatively new manifold learning method developed by Coifman and Lafon [4]. It is closely related to Laplacian Eigenmaps [2] and belongs to a larger family of manifold learning and dimensionality reduction methods that also include algorithms such as ISOMAP, LLE (Locally Linear Embedding), Hessian Eigenmaps [5] and MDS (Multi-dimensional Scaling). A very inspiring and interesting application of Diffusion Maps was presented by Lafon et al. in [8].

The common approach when using diffusion maps is to define an affinity metric \( k(y_i, y_j) \) between pairs of channel outputs based on the following Gaussian kernel defined on the feature space

\[
k(y_i, y_j) = \exp \left\{ -\frac{1}{2} (z_i - z_j)^* \epsilon^{-1} (z_i - z_j) \right\}, \quad i, j = 1,\ldots,N \tag{1.1}\]

where \( \epsilon = \text{diag}(\epsilon_1, \ldots, \epsilon_n) \) is feature-wise scaling factor and \( n \) is the number of features. Using a different scale for each feature takes into account its range of values which might differ from that of other features. The scale \( \epsilon_i \) is chosen as the variance of the \( i \)-th feature across all samples, multiplied by some constant \( C_i \), which we refer to as the scale control parameter. The scale control parameter
could be chosen empirically according to an optimization performed during the training stage.

The vectors $z_1, \ldots, z_N$ can be viewed as nodes of an undirected symmetric graph. Two nodes $z_i$ and $z_j$ are connected by an edge with weight $k(y_i, y_j)$, that corresponds to the affinity between $y_i$ and $y_j$. Note that the Gaussian kernel (1.1) emphasizes the connection between near points, since an edge connects two relatively distant points is given very low weight. Next, consider a random walk on this graph defined by the following transition probability from node $y_i$ to node $y_j$:

$$p(y_i, y_j) = \frac{k(y_i, y_j)}{d(y_i)},$$

where $d(y_i) = \sum_{j=1}^{N} k(x_i, x_j)$, i.e. the corresponding stochastic matrix $P$ is given by

$$P = D^{-1}K,$$

where $(K)_{i,j} = k(y_i, y_j)$ and $D = \text{diag}(d(y_1), \ldots, d(y_N))$. We also use $p_t(y_i, y_j) = (P^t)_{i,j}$ to denote the probability of transition in $t$ steps from node $y_i$ to node $y_j$. Denote by $\{\phi_j, \psi_j\}^{N}_{j=1}$ the right and left eigenvectors of $P$ that correspond to the eigenvalues $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$. For any integer $t$, the diffusion distance is defined by

$$D^2_t(y_i, y_j) = \sum_k (p_t(y_i, y_k) - p_t(y_j, y_k))^2 / (\phi_1)_k^2.$$

This metric relates to the evolution of the transition probability distribution $p_t$. It enables to describe the relationship between pairs of samples in terms of their graph connectivity, and integrates local affinities into a global metric. For $t > 0$ and an integer $k$ typically much smaller than $n$, define a map from the sample space into $\mathbb{R}^k$ by

$$\Psi_t(y_i) = [(\psi_1)_1, \lambda_1^t, (\psi_2)_2, \lambda_2^t, \ldots, (\psi_k)_k, \lambda_k^t]^T.$$

This family of maps is termed diffusion maps, and we have the following relation [4]

$$D^2_t(y_i, y_j) = \|\Psi_t(y_i) - \Psi_t(y_j)\|^2$$

where the last norm is the Euclidean norm in $\mathbb{R}^k$. It means that under the diffusion map allows us to study the relation between the data points presented by the diffusion matrix, using the Euclidean norm. In addition, dimensionality reduction is obtained by a fast decay of the sequence of eigenvalues, as coordinates in (1.2) become negligible for large $k$. This decay depends on the diffusion depth $t$ which is a tunable parameter in the algorithm.

Another possible normalization of the kernel (1.1) is the symmetric normalization that approximates the Fokker-Planck operator [4]

$$p_{\text{sym}}(y_i, y_j) = \frac{k(y_i, y_j)}{\sqrt{d(y_i)} \sqrt{d(y_j)}}$$

which can be represented by the symmetric affinity matrix

$$P_{\text{sym}} = D^{-1/2}K D^{-1/2}.$$
This normalization is less sensitive to the distribution of samples and therefore more suitable in representing non-uniformly sampled data.

1.4 Classification

Given a training set \(y_1, \ldots, y_N\), the embedding of the samples into \(\mathbb{R}^k\) is given by (1.2). In order to find the coordinates in \(\mathbb{R}^k\) of a new sample \(y\) which is not part of the training set, the following mapping was proposed in [7]:

\[
\tilde{\Psi}(y) = \left[\tilde{\psi}_1(y), \ldots, \tilde{\psi}_1(y)\right],
\]

where

\[
\tilde{\psi}_j(y) = \frac{1}{\lambda_j} \sum_{i=1}^N p(y, y_i) (\psi_j)_i, \quad j = 1, \ldots, k.
\]

It is can be understood as the approximation of the embedding coordinates for a new sample using the eigenvectors of the transition matrix obtained for the training set. We note that a for point \(y_i\) in the training set, we have \(\tilde{\Psi}(y_i) = \Psi(y_i)\).

2 Methods

The outline of our method for detecting the number of backscatters appears in Figure 2.1. The different steps are explained bellow.
Measurement Generation

Our backscatter channel modeling is based in general on the model used by SNSG (Stanford Networked Systems Group, Joshi et al.). Instead of raytracing a transmitted signal and simulating reflections it simply generates a number of signals from sources with different gain, phase, delay and angle parameters. Formally, $L$ is the number of reflectors set for a simulation. For $i = 1, \ldots, L$ we have delays (in seconds) $\tau_i$, phases $\nu_i$, gains $\alpha_i$ and AoA-s $\theta_i$. The signal is received by an array of $K$ antennas designated by $k = 1, \ldots, K$, and spaced $d$ meters one from another. The reflections are modeled as pulses in frequency, of width $B$. We then have

$$\beta_{ik} = \alpha_i \cdot e^{-j2\pi f_c \tau_i + \nu_i} \cdot e^{j \frac{2\pi}{\lambda} (k-1)d \sin(\theta_i)}$$

$$h_{ik}(t) = \beta_{ik} \cdot \text{sinc}(B(t - \tau_i))$$

$$h_k(t) = \sum_{i=1}^{L} h_{ik}(t)$$

where $h_k$ is the composite backscatter channel as seen by the $k$-th antenna. Transmitting some known sequence $x$ we then have

$$y^k(t) = (h \ast x)(t)$$

where $y^k$ is the signal sensed at the $k$-th antenna due to the transmitted signal $x$. The corresponding discrete time channels and received signals are obtained by substituting $t = nT$ for some sampling period $T$. The sampling period $T$ has to match our assumption of a narrowband regime.

We generate measurements $y_1, \ldots, y_N$ using the channel simulator. Each measurement is the output of a different channel snapshot at $K$ antennas over time (i.e. $y_i$ is of size $K \times M$, where $M$ is the length of the received signal at each antenna). At each channel snapshot the number of reflectors $L$ can take any integer value from 1 to $L_{\text{max}}$, where $L_{\text{max}}$ is much smaller than the number of measurements $N$. The other channel parameters take random values which are different for each sample.

Features Extraction

We map each sample $y_i$ into $z_i = f(y_i)$ in $\mathbb{R}^n$, where each coordinate corresponds to a different feature. The set of features is chosen according to the particular experiment.

Diffusion Maps Embedding

We use the diffusion maps technique presented in Section 1.3 to embed the points $z_1, \ldots, z_N$ into $\mathbb{R}^k$. The tunable parameters in this step are the diffusion depth $t$, the number of dimensions $k$ and the scale control parameter $C$. 

7
Classification

Given a new validation measurement \( y \), generated by the channel simulator, we use the extended map \( \Psi \) presented in 1.3 to embed \( y \) into \( \mathbb{R}^k \). We then apply \( K \)-Nearest Neighbors (K-NN) classification to associate \( y \) with a certain cluster (the different clusters representing the number of reflectors in the first experiment, and the angle in the second).

3 Results

3.1 Discovering the number of reflectors

We applied the proposed method of Section 2 using two approaches of measurements generation: simulating transmission of the same channel input sequence \( x \) for each channel snapshot, and generating a random sequence \( x \) for each snapshot. The constant transmitted signal approach makes sense because in the realistic scenario we transmit a known signal and measure backscatters. The second approach is of interest since if we show that even for an unknown transmitted signal we can differentiate between different numbers of reflectors, we have a totally blind measurement technique, and can also apply it for cases when we receive reflections that do not originate in a signal we have transmitted (from another source). In the experiments below we used \( L_{\text{max}} = 4 \) reflectors and a high SNR of 60 dB. The features we used for each snapshot are simply the mean, variance, maximum and minimum of the signal sequences received by each one of the antennas, concatenated to a single vector.

In Fig. 3.1 we show the embedding of 4000 snapshots of a randomized channel where the transmitted signal is constant and is the same for all snapshots. We can see that the points for the case of a single reflector are tightly grouped (green). The snapshots for two reflectors seem mixed with the cases of 3 and 4 reflectors. For 3 and 4 reflectors, while there is a significant mixed region, they still have major well separated regions.

In Fig. 3.2 we show the results of embedding 4000 snapshots of a randomized channel. In this case we generated a new transmitted signal for every snapshot. We can see that while the points for 1 and 2 reflectors are all gathered in a small region there is reasonably good separation between the regions corresponding to 3 and 4 reflectors.

This result is interesting considering we used the simplest statistics of the raw received signal. It is also worth to notice that even when the transmitted signal is unknown (i.e. a random transmitted signal is generated for each snapshot) we are able achieve a similar clustering and classification of samples as in the case of a constant transmitted signal. This is particularly interesting for cases in which we have no control over the source of transmission, and need to blindly estimate the channel.

We also tested performance of the classification of new samples using the precalculated diffusion embedding. The testing was performed using 10-fold cross-validation. While the correct classification rate was not very high (around
Figure 3.1: Embedding of 4000 snapshots of randomized channel for the same (known) transmitted signal. $C = 1000$, $t = 3$, $k = 3$.

Figure 3.2: Embedding of 4000 snapshots of randomized channel and random transmission (the two views present the same embedding from different angles).
50%) it is still much better than a random guess. Perhaps a better tuning of the diffusion maps parameter can improve the separation. Another thing we have noticed is that our misclassification rate is in fact not much better than that of K-NN in the original feature space (which is around 70%). So while we get a nice visualization that can be helpful in analyzing the data and reach conclusions, we do not gain much in successful classification rate.

3.2 Object tracking

For the task of object tracking, or gesture recognition, it would be beneficial to have a coarse estimate of the angle of arrival (AoA) of a backscatter from a single reflector. For this evaluation we set $L = 1$ (single source), and generate multiple snapshots with angle $\theta$ in the range $[0, \pi]$. We simulate reflection of a known transmitted signal which is the same for each snapshot. We tried to use different features based on cross correlation between the signal vectors received by the 4 antennas in addition to the raw signal statistics. We assumed that the cross correlation would capture the lag between the signal arrivals at different antennas (similar to known AoA estimation methods). For 4 antennas we have 10 cross correlation vectors in total (including the auto-correlations). The primary features we tried were:

- The maximum of each cross correlation and its offset (lag) in the cross correlation vector.
- Cross correlation maximum, minimum, mean and variance.
- Taking the cross correlation vectors as is, and concatenating them to obtain a feature vector.

Since this application reminds in some ways the example in [6], and only one parameter changes during the simulation, we had high hopes for obtaining a neat 2-dimensional visualization as well as good classification. Unfortunately, for this simulation we were unable to obtain a plausible embedding and the snapshots from different angles appear mixed up.

In Fig. 3.3 we show the embedding of snapshots generated for this scenario. We divide the samples into two groups, one obtained for $\theta < 90^0$ and the other for $\theta > 90^0$. We see that the samples from the two groups are mixed. We suspect there might be a problem arising from the symmetry of our features (since the order of the features in the feature vector does not necessarily affect the distance between two samples leading to similar affinities with samples from the two different directions) which might be fixed by incorporating a metric that depends on the index of each antenna. This is only a speculation at this point and has to be tested. Despite our disappointment with this particular result, we think that putting more efforts into finding better features would eventually yield a satisfactory embedding that could be useful for coarse classification of AoA.
4 Conclusions

We proposed a manifold learning approach for detecting the number of reflectors in a random channel and for gesture detection. This was tested using a backscatter channel simulator and diffusion maps. We achieved a classification success rate of about 50% for detecting the number of multipaths. While much better than a random guess, it does not exhibit improvement over classification in the feature space. Object tracking was not successful.

We propose the following improvements over our method:

- Better tuning of the diffusion maps parameters: $C$, $t$, and $k$.

- The number of features used seems to be small compared to the actual dimension of the problem, and it is not clear whether the feature vector actually preserves the relevant information need for such separation. We expect that significant improvement on the current results can be achieved by using different and possibly more complicated features. Such features can include for example the output or intermediate calculation of state of the art algorithms designed for the task of estimating the number of multi-paths.

- The parameters of the channel were chosen for each snapshot uniformly and independently over the space of parameter (for example, AoA of each reflector is uniformly distributed over $[0, 2\pi]$, and it is independent of the gain of that multi-path component). In practice, a more realistic channel model that exhibits correlation between the parameters should be assumed.

- Object tracking definitely requires finding more suitable features.
References


