Bayesian Filters State Estimation on Directed Graphs: On The Toronto Subway System

Robert (Rupert) Wu
Department of Computer Science, University of Toronto
rupert@cs.toronto.edu

Roland Gao
Department of Computer Science, University of Toronto
roland.gao@mail.utoronto.ca

Abstract

Despite its limited expanse, Toronto's Subway System is complicated and often controversial. It is quite useful for the average resident to get around, but it is large enough that you might get lost or just curious. If you do get lost, you might only make some observations about the stations you visit. In this paper, we explore three approaches in the problem of State Estimation using Bayesian Filters. In addition to the traditional approach, we also implemented Bayesian Filters using (un)directed graphs and the Probabilistic Adjacency Matrix (PAM) to streamline predictions. We also have a public repository and results from some of our interesting experiments.

Note to the Reviewer: We previously had two other group-mates who dropped CSC412/2506.

Note to the Reviewer: We did not strictly follow the order of the rubric sections, but have covered all expectations therein.

1 Background

1.1 Bayesian Filters

Data can be defined as $\{(u_i, z_i)$ for action u_i and observation $z_i\}$. The prior distribution $P(x)$ gives the probability of being in state x before receiving any data. The posterior distribution (aka the belief) of the state at iteration t is defined as the $Bel(x_t) = P(x_t|u_1, \ldots, u_t, z_1, \ldots, z_t)$.

Similar to the Hidden Markov Model, we have the Markov Assumption, which means that the probability of seeing the observation z_t given everything we know is equal to the probability of it given the current state x_t, and, similarly, the probability of the state x_t given everything we know is equal to the probability of it given the previous state x_t and the action u_t. More specifically, the Markov Assumption in our case is the following two equations.

$$P(z_t|x_1, \ldots, x_t, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) = P(z_t|x_t) \tag{1}$$
$$P(x_t|x_1, \ldots, x_{t-1}, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) = P(x_t|x_{t-1}, u_t) \tag{2}$$

At each iteration, we update the belief $Bel(x_t)$ based the previous belief $Bel(x_{t-1})$, the action u_t taken, and the observation z_t. Motivated by the Markov Assumption, we have a sensor model $P(z|x)$ that gives the probability of seeing the observation given the state that you are in, and an action model $P(x|u, x')$ that gives the probability of arriving at state x when taking action u with state x'.

The update procedure for one iteration can be summarized by the following formula, where λ is a scalar that normalizes the total probability of $Bel(x_t)$ to 1. The derivation of the following equation is given in the Appendix.

$$Bel(x_t) = \lambda P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \tag{3}$$

1.2 Related Works

- Chen, Zhe (2003) surveys the recent progress of Bayesian filtering, such as nonlinear filters, Kalman filters, and particle filters. He discusses many implementation details and the theoretical and practical importance of Bayesian filtering.
- Gordon et al. (1992) proposes a new algorithm to efficiently implement Bayesian filtering, while not requiring the transition function to be linear.
- The Fox et al. (2003) briefly survey Bayesian filtering implementations and show their application to real-world location-estimation tasks common in pervasive computing using multisensor fusion and identity estimation.

2 Our Dataset

State Estimation has many predictive and modelling applications, which can be discrete (such as a finite state machine) or continuous (predicting a drone's location across the continental United States based on elevation). To create a manageable problem where we can meaningfully adjust hyperparameters to run experiments, we chose to employ a discrete problem domain that can be reasonably represented by a *Graph*.

2.1 The Toronto Subway System

We chose the Toronto Subway System as our dataset, not only because it is where the University of Toronto is based, but because it is large and varied enough to provide an interesting modelling problem. Toronto has a rather diverse landscaping and infrastructure layout as the Downtown Core built much more densely than the rest of the city, and there are large swaths of parks and special zones scattered all around the suburban Greater Toronto Area (GTA). The diverse population also produces varied transit patterns as people commute to work, school, and elsewhere.

2.1.1 Domains of Data

The Toronto Transit Commission (TTC) operates a sprawling network of subway stations supported by the bus network. From official data sheets to video-taped subway rides to actually surveying the sites, there is a lot of data available from the subway stations. The possible data that could be collected might include but is not limited to the following.

- Travel time. How long/far did the train drive?
- Track observations: length/curvature of track between stations. Was it above or underground? What direction did the train travel.
- Dimensions of the stations. How long and wide are the platform? How tall is the ceiling?
- Ridership. How many people were waiting on the platform? How many people were on the train? How many alighted?
- Idle time. How long is the train holding at the station?
- Platform orientation. Side or island? Anything special?
- Lighting level/shade on the platform. Is there natural light?
- How many cars are on the train? Is there a train on the other track(s)?

The subway system has been a contentious issue in civic and political discussion, and an important topic therein is the usefulness and equality of benefit of transit infrastructure investments. One

intuitive way to measure these points of interest is by considering ridership, which provides an inconsistent and varied data domain across the entire subway network.

We also wanted a point of data that introduced more entropy and idle time would be perfect for that since on an average trip, the idle time of a train at a station is fairly consistent, with a slightly correlation with the ridership. One detail to note is that stations with especially high ridership or terminal stations will experience longer-than-normal idle times.

Moving forward, we assume that the modelling will take place during an average workday rush hour. We also suppose that the two data points are on a Guassian/Normal distribution.

2.1.2 System Specification

As of 2018, the following rapid transit lines are operated by the Toronto Transit Commission (TTC) on a total of 75 subway stations, including 5 served by two lines.

Table 1: The Toronto Subway System (2018) **??**

Line	Name	Stations	Terminus	Terminus
1	Yonge-University(-Spadina)	38	Vaughan Metro Centre	Finch
2	Bloor-Danforth	31	Kipling	Kennedy
3	Scarborough RT	6	Kennedy	McCowan
4	Sheppard	5	Sheppard-Yonge	Don Mills

As of the writing of this paper, *Line 5 Eglinton* is excluded because it is still under construction and there exist no reliable statistics. We included the *Toronto-York Spadina Subway Extension* but it had only recently opened in 2018, so ridership numbers may not be consistent with 2021.

2.2 Data Collection & Processing

We collected our ridership statistics from the *Toronto Transit Commission Subway ridership – 2018* **??**; these statistics may be slightly dated in 2021 but they were the most recent we could retrieve. The ridership numbers r_D listed in this data sheet are the "typical number customers travelling to and from each station platform on an average weekday". For our observations, we want to include instant ridership r, which is the number of people using the station at a given time when a train arrives in a station. Because during rush hour, trains are around 2-3 minutes apart but are less frequent on off-hours, we assume for simplicity that all stations operate for 18 hours per day at an average of 5 minutes between trains. As such, any ridership number r would be adjusted with

$$r := \frac{r_D}{N_T} = \frac{r_D}{18 * 12}$$

where $N_T = 18(12)$ is the number of trains on the average workday because 5 minutes between trains means 12 trains per hour. This will give us the approximate ridership per observation.

Rupert has lived in Toronto for 21 years and used his memory and some YouTube videos as references to set the average idle times. In general there is a slight correlation between idle time and ridership, and obviously termini stations will have extremely long idle times; Kennedy on Line 3 is an exception to this rule because it only has a single platform.

2.2.1 Standard Deviation

We were unable to obtain large datasets to produce the statistics themselves, so we estimated the standard deviation values for both metrics to be the square-root of the mean. The reasoning behind this decision may not be scientifically sound, but the samples generated with these parameters satisfied us qualitatively.

3 Model

We explore two main approaches to the problem of State Estimation with Bayesian Filters.

3.1 A Traditional Approach to Bayesian Filters

To estimate the probability of being in a given state given the actions and observations, we use the Bayesian Filters described in Section 1.1.

There are 75 states, 1 state for each subway station. The observation $z = (z^1, z^2)$ contains the idle time and the number of passengers. The sensor model $P(z|x)$ is a factorized Gaussian distribution, where the mean and standard deviation of the idle time and the number of passengers for each state are provided by the data, as described in Section 2. The action model $P(x|u, x')$ is implemented as a conditional probability table (CPT). The three possible actions for u are $1, 0, -1$, which stands for moving forward, staying still, and going backwards, respectively. Given the previous station x', forward(x') denotes the next station, identity(x') denotes the same station, and backward(x') denotes the previous station. All other stations have a probability of 0 given x' because they are unreachable given a single time step. For simplicity, we show the case where x' is not a terminal station.

Table 2: CPT of $P(x|u, x')$

$x \setminus u$	1	0	-1
forward(x')	0.85	0.1	0.05
identity(x')	0.05	0.9	0.05
backward(x')	0.05	0.1	0.85
other	0	0	0

We then update the belief $Bel(x_t)$ using the equation (3).

See Appendix 6.3 for the commit hash of our previous code for the traditional implementation of Bayesian Filters. We referenced our former group-mate Zizhao Chen's Colab notebook to aid us in implementing.

3.2 A Graph-Oriented Approach

Another approach we considered was Graph-Oriented. Because the Toronto Subway System could be represented as a data object on its own, we create one Station node for every station on each line, with two for the interchanges. This allowed us to apply graph theory and draw edges between adjacent stations and any transfer nodes on an interchange. Originally, this graph representation was to aid the prediction model of the traditional approach to Bayesian Filters, but we were able to simplify the problem by augmenting the concept of the *Adjacency Matrix*.

3.2.1 Probabilistic Adjacency Matrix (PAM)

Because we were storing the state likelihood estimation in a vector $v_{ll} = [v_0, v_1, \ldots, v_n]$ for n stations such that v_i represents the normalized likelihood of a given station being the current position, we could use matrix algebra to apply these likelihoods for every iteration. Our application of the traditional approach used linear functions, so we can employ matrix algebra to abstract the transition functions across all nodes of a graph. To begin with an undirected graph, for the general Station s_i, we implemented a weighting scheme to express how likely it is that each other Station s_j might be the next state. We originally assigned a weight of $w_s = 1$ to s_i itself meaning that one unit of the total weight W would be assigned to s_i itself, $w_c = h$ to those s_j which share an edge with s_i, and 0 otherwise.

$$w_{i,j} = \begin{cases} 1 & s_i = s_j \\ h & s_j \in \text{connections}(s_i) \\ 0 & \text{otherwise} \end{cases}$$

These weights are normalized per-Station to express probabilistic values; so for example, the probability that s_i would follow itself in the simulation is $\frac{1}{W}$. Then the weights are arranged in a Probabilistic Adjacency Matrix (PAM) M such that $M_{i,j}$ represents the normalized probability that s_j follows s_i. For example, see the PAM for Line 4 in 6.5.

3.2.2 Using a Directed Graph

However, to make things more interesting and to be more accurate, we consider that not all connections are equal. At each iteration, staying at the same station is highly unlikely. Backtracking is slightly more but still not very likely; and while transfers are more likely, they are still not as likely as simply progressing forward. Therefore, it would make sense to introduce some sense of direction and history.

To this end, we doubled up every interior (meaning non-terminus) node to represent the direction the train came from; these pairs refer to each other as opposite nodes of the same station and are sub-indexed 0 or 1 to mean travelling in the obverse or reverse directions. Terminals only have one node because there is only one direction the train could have come from. As described in Appendix 6.2, forward edges are drawn from each node to its successor in the same direction, backward edges are drawn from each node to the successor of its opposite node. Then, the edges are assigned weights w_f, w_b, w_t respectively and included in the PAM and normalized per-row.

For example, see the PAM for Line 4 in 6.6.

4 Experiments

As a baseline, we ran simulations on entire lines, and in general, they remain pretty accurate save for a few incorrect backtracking predictions, where the model occasionally thinks it is possible that the path backtracked. This is to be expected since most stations on the longer lines have riderships that do not vary too much.

We also ran the following simulation themes.

- Entire lines excluding the termini.
- Cycles.
- Around Interchanges.
- Short paths.

4.1 Analysis & Patterns

In general, our PAM was effective at keeping predictions at or near the true position. In fact, the most inaccurate predictions were near the beginning of simulations where the true position was confused with a station that has a similar ridership; the rest of the iterations had decently accurate predictions. However, one pattern we noticed was that longer paths (like more than six nodes) do not necessarily improve accuracy. Uncertainty introduced at each iterations can lead to predictions that are very off, such as the backtracking mentioned above. This behaviour happens randomly and is likely due to the small number of features in the domain and the random samples therefrom.

4.1.1 Termini with Long Idle Times

Naturally, terminal stations will have above-average ridership and exceptionally high idle times; for example, trains on average idle for over 5 minutes on either end Line 2 while the average station has trains idling between the 10 and 25 seconds. Idle time samples in observations at the termini will therefore be near "dead give-aways" regarding the true location of the observation. Therefore, any interesting observations will exclude termini stations.

However, one exception to this rule was *Kennedy (3)* on Line 3, probably because it has a single platform and an average idle time of 25 seconds, which is in-line with the busier interior stations in the system. Despite having above-average idle-time, the our model had difficulty estimating the true positions around *Kennedy (3)* until an observation from *Ellesmere*, whose very low ridership was a dead give-away. See Appendix 6.7 for our experiment around this.

4.1.2 Low/High Ridership Outliers

As a general rule, anomalous stations will generate There are several stations with exceptionally low ridership, like *Old Mill*, *Highway 407*, *Bessarion*, *Ellesmere*, *Midland*, at around a few thousand per day. For similar reasons to termini, observations at these stations would have high likelihoods and provide a lot of information to our model. In fact, sometimes these outliers would correct previously erroneous predictions.

By the same token, stations with very high ridership, such as *Union* and the interchanges such as *St. George (1/2)* and *Bloor-Yonge (1/2)*, also predictably provide samples that have high identifiable likelihood. In particular, these stations provide more meaningful posterior values at their current iteration and the approximate position overall.

However, because they dwarf their neigbours by so much, sometimes, they introduce more uncertainty afterwards. In Appendix 6.8 and 6.9, we demonstrate that at an interchange such as *St. George (2)*, the model is accurately certain of the true position in that iteration but becomes less sure immediately after leaving. This could end up being counter-intuitive because in an abstract way, these interchange stations essentially "erase" most of the information from the previous history of observations.

4.1.3 The Downtown "U"

The middle portion of Line 1, colloquially known as the "U" and known for being out of service frequently as a group, is notably symmetrical. The stations appear in pairs on the same streets and support the same streetcars. This symmetry can sometimes confuse the model into thinking the true position is on the opposite side. For example in Appendix 6.10, on a path on the downtown "U", the predictions are mostly correct, but sometimes make "phantom" predictions on the same direction but the other side of the "U". In the following simulation, the path started at *Queen's Park* but the model may thought *College* was also plausible, so when the true position move South to *St. Patrick*, it was predicted to be North at *Wellesley*, which is understandable given the model got the direction right.

5 Conclusion

In general, Bayesian Filters are a good way to approximate the state of a system, and geographical modelling, whether discrete or continuous, is a sound application. But we did not get the results we were expecting. Bayesian Filters (including particle filters) should be increasingly more confident about the likelihoods of state estimations. The intuition is that starting out with many (perhaps uncountable) states and high entropy, with more observations of measurements over time, the model could recursively filter the state likelihoods to a small manageable number of possible states.

The model we constructed and simulations we ran have produced surprising results on the Toronto Subway System. The model's accuracy did not improve much beyond a few iterations and sometimes even got worse. This may be attributed to the low dimensionality of data we collected in observations, or perhaps the small number of edges and the linearity of the graph. The data of idle times and ridership are naturally Gaussian, but that is only with respect to each individual station. Overall, the means of all the stations, especially the ridership statistics, do not distribute smoothly over a Gaussian, and had many sharp low/high outliers. As a consequence, our model was able to be more correctly confident at these outliers, but it is a double-edged sword as some of them, especially the interchange stations, added more confusion to the model.

However, we have also constructed a model that is more accurate *earlier*, and its state estimation is quite accurate at most iterations; it has even shown to sometimes self-correct from sharp errors. The plus side is that the model is efficient at estimating shorter with shorter paths, which make more sense in the scheme of rapid transit as opposed to mountain range elevations. The model might be improved by gathering larger and more datasets to derive statistically sampled parameters. It could even be applied to other transit systems, particularly in larger systems in Europe and Asia.

References

[1] Chen, Zhe, *Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond*, Statistics, vol. 182, 2003, doi: 10.1080/02331880309257.

[2] N.J. Gordon, D.J. Salmond, and A.F.M. Smith, *Novel approach to nonlinear/non-Gaussian Bayesian state estimation*, IEE PROCEEDINGS-F, vol. 140, no. 2, 1993, doi:10.1049/ip-f-2.1993.0015.

[3] V. Fox, J. Hightower, Lin Liao, D. Schulz and G. Borriello, *Bayesian filtering for location estimation*, in IEEE Pervasive Computing, vol. 2, no. 3, pp. 24-33, 2003, doi: 10.1109/MPRV.2003.1228524.

[4] P. Abbeel, *Bayes Filters*. Available at https://people.eecs.berkeley.edu/~pabbeel/cs287-fa13/slides/bayes-filters.pdf

[5] Chen, Z., *rob301proj deli2 bayesian.ipynb*. Available at https://colab.research.google.com/drive/1EZMb3rjnYIE7Q7J5oH5Apu4hgYhqkRS_.

[6] Toronto Transit Commission, *Subway*. Available at http://ttc.ca/Subway/.

[7] Toronto Transit Commission, *Toronto Transit Commission Subway ridership – 2018*. Available at http://www.ttc.ca/PDF/Transit_Planning/Subway%20ridership%20-%202018.pdf.

6 Appendix

6.1 Derivation of the Bayes' Filter Procedure

First, recall that the Markov Assumption is

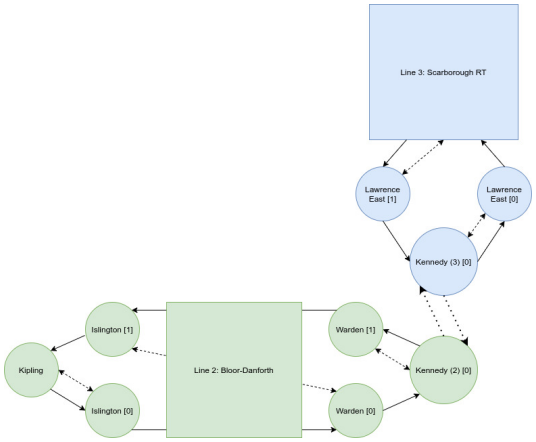
$$(z_t|x_1, \ldots, x_t, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) = P(z_t|x_t)$$
$$(x_t|x_1, \ldots, x_{t-1}, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) = P(x_t|x_{t-1}, u_t)$$

The following is the proof of Bayes' Filter Procedure using Bayes' Theorem, renaming the denominator to λ, Chain Rule, Markov Assumption, Law of Total Probability, Chain Rule, and Markov Assumption.

$$\begin{aligned}
Bel(x_t) &= P(x_t|u_1, \ldots, u_t, z_1, \ldots, z_t) \\
&= \frac{P(x_t, z_t|u_1, \ldots, u_t, z_1, \ldots, z_{t-1})}{P(z_t|u_1, \ldots, u_t, z_1, \ldots, z_{t-1})} \\
&= \lambda P(x_t, z_t|u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) \\
&= \lambda P(z_t|x_t, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) P(x_t|u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) \\
&= \lambda P(z_t|x_t) P(x_t|u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) \\
&= \lambda P(z_t|x_t) \int P(x_t, x_{t-1}|u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) dx_{t-1} \\
&= \lambda P(z_t|x_t) \int P(x_t|x_{t-1}, u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) P(x_{t-1}|u_1, \ldots, u_t, z_1, \ldots, z_{t-1}) dx_{t-1} \\
&= \lambda P(z_t|x_t) \int P(x_t|x_{t-1}, u_t) Bel(x_{t-1}) dx_{t-1}
\end{aligned}$$

6.2 Line 2: Directed Graph

The following is the graph between *Kipling* and *Lawrence East*. The solid lines represent forward edges (with weight w_f), the dashed lines represent backward edges (with weight w_b), and the dotted lines (with weight w_t) between the two *Kennedy* nodes represent transfer connections at interchanges.



6.3 Code Implementations on GitHub

The code repository can be found at https://github.com/rusbridger/Reasonable-Subway-Surfing.

Code for the traditional model can be found at
Commit 131d68236262f3f67f1a2b60bbee0c019d6f1d2a.

6.4 Code for Constructing the Probabilistic Adjacency Matrix (PAM)

```python
import numpy as np
from lib.data.toronto import *
from .hparams import *

def build_probabilistic_adjacency(stations):
    n = len(stations)
    prob_adj_matrix = np.zeros((n, n))
    for i in range(n):
        stn_i = stations[i]
        total_weight = compute_weight(stn_i)

        for j in range(n):
            stn_j = stations[j]

            def compute_matrix_val(connection, weight):
                if stn_j == connection:
                    prob_adj_matrix[i][j] = weight / total_weight

            compute_matrix_val(stn_i.backward, backward)
            compute_matrix_val(stn_i.forward, forward)
            compute_matrix_val(stn_i.opposite, opposite)
            for t in stn_i.transfers:
                compute_matrix_val(t, transfer)

        prob_adj_matrix[i][i] = 1 / total_weight

    return prob_adj_matrix

def compute_weight(station):
    weight = 1
    if station.backward: weight += backward
    if station.forward: weight += forward
    if station.opposite: weight += opposite
    weight += len(station.transfers) * transfer
    return weight
```

6.5 Constructed PAMs for Line 4 (Undirected)

Note that because the first node is *Sheppard-Yonge (4)*, which has edges to *Sheppard-Yonge (1)*, so its row in this PAM for Line 4 does not sum to 1.

```
[0.25 0.25 0.00 0.00 0.00]
[0.40 0.20 0.40 0.00 0.00]
[0.00 0.40 0.20 0.40 0.00]
[0.00 0.00 0.40 0.20 0.40]
[0.00 0.00 0.00 0.66 0.33]
```

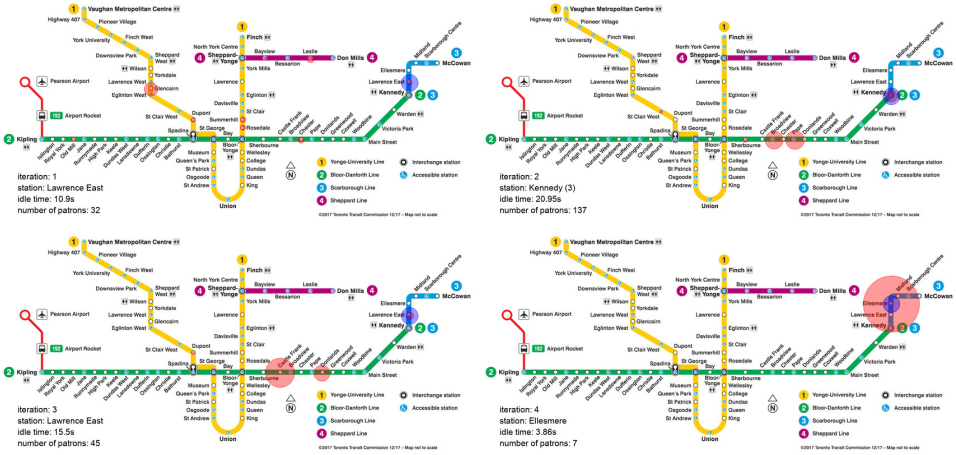
6.6 Constructed PAMs for Line 4 (Directed)

Note that because the first node is *Sheppard-Yonge (4)*, which has edges to *Sheppard-Yonge (1)*, so its row in this PAM for Line 4 does not sum to 1.

```
[[0.17 0.17 0.   0.   0.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25 0.   0.   0.   0.  ]
 [0.25 0.25 0.25 0.25 0.   0.   0.   0.  ]
 [0.   0.   0.25 0.25 0.25 0.25 0.   0.  ]
 [0.   0.   0.25 0.25 0.25 0.25 0.   0.  ]
 [0.   0.   0.   0.   0.25 0.25 0.25 0.25]
 [0.   0.   0.   0.   0.25 0.25 0.25 0.25]
 [0.   0.   0.   0.   0.   0.   0.25 0.25]]
```

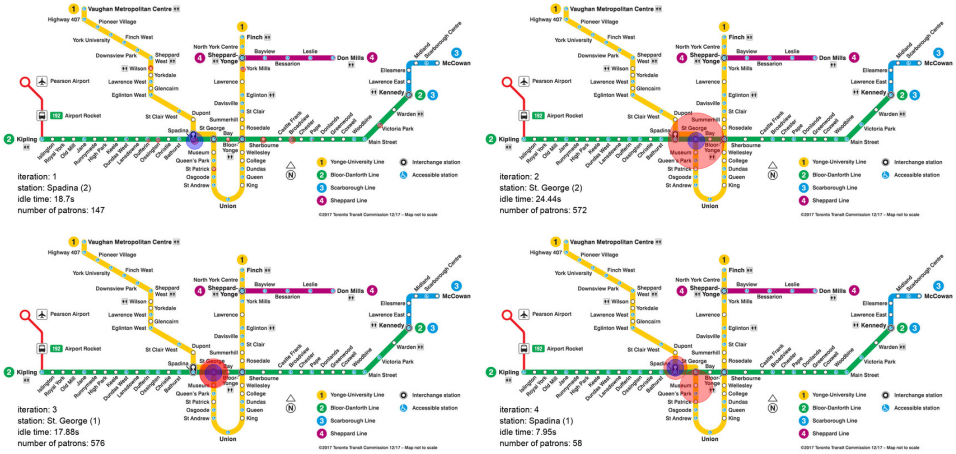
6.7 Line 3: Around Kennedy and Back

The true path begins at *Lawrence East*, moves to the terminus *Kennedy (3)*, back to *Lawrence East*, and the model believes the position to be around *Chester* until the true path reaches *Ellesmere*, whose exceptionally low ridership gives away the true position easily.



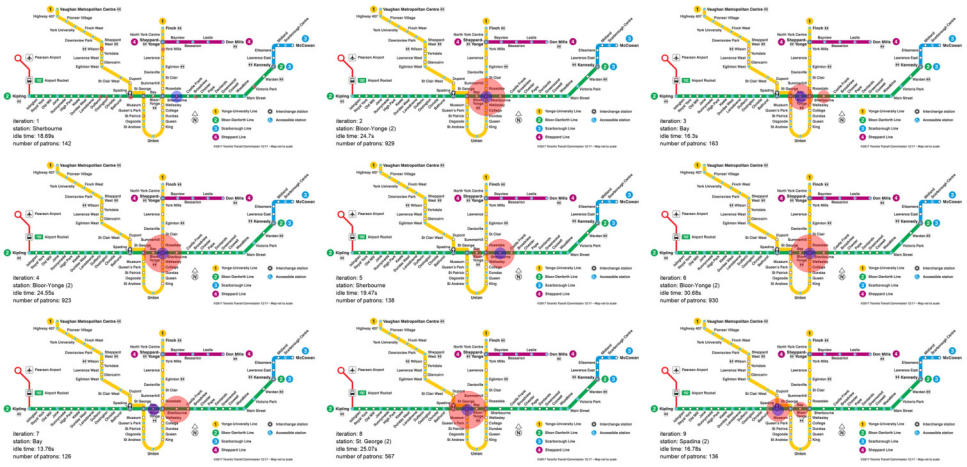
6.8 Lines 1/2: Around Spadina & St. George

The simulated path starts at *Spadina (2)* and transfers to Line 1 via *St. George* and turns North to *Spadina (1)*. However, the model predicts that it is more likely that the last turn was South towards *Museum*.



6.9 Line 2: Yorkville Area

When simulating a path of observations in the Yorkville area around *Bloor-Yonge (2)* and *St. George (2)*, the model is pretty correctly certain about the position at the interchange stations, but the observations immediately after leaving to neighbouring stations introduce a random amount of uncertainty.



6.10 Line 1: Downtown "U"

On a path on the downtown "U", the predictions are mostly correct, but sometimes make "phantom" predictions on the same direction but the other side of the "U". In the following simulation, the path started at *Queen's Park* but the model may have thought *College* was also plausible, so when the true position move South to *St. Patrick*, it was predicted to be North at *Wellesley*, which is understandable given the model got the direction right.