USING INFOPIPES TO ANALYZE STREAMED TRAFFIC DATA

Catherine Vilhauer
Portland State University

July, 2006

Table of Contents

1. INTRODUCTION

Highways across the country are monitored by Civil Engineers to try to ensure good traffic flow. Often traffic is measured using traffic loop detectors, installed under city highways, which can sense when vehicles move on and off them. In the Portland area traffic loop detectors have been in operation for a long time and provide useful data. However, the Portland traffic loop detectors are less sophisticated than many, being ‘single loop detectors’ rather than ‘dual loop detectors’. Single loop detectors have the drawback that they are unable to distinguish between different types of vehicles, thus there is no differentiation between cars and trucks travelling down the highway, which can often be an important metric.

Several algorithms address this problem and allow us to estimate the volume of car traffic versus truck traffic with single loop detectors. One well-known algorithm was developed by Kwon (2003) in a paper entitled “Estimation of Truck Traffic Volume from Single Loop Detector Outputs Using Lane-to-lane Speed Correlation”. The project that I will outline in this paper builds upon work already completed by Emerson Murphy-Hill in which he implemented a Smalltalk version of the Kwon traffic algorithm using the Infopipes abstraction. Infopipes are an object-oriented approach streaming extensively researched by Professor Andrew Black (2002) and colleagues at Portland State University and which I will outline in more detail later. The data that the project will draw upon is Portland traffic data as supplied by Portland State University’s PORTAL project.

My aim in this paper is to give the reader a general background to the project, including an introduction to Infopipes, the principles behind how traffic data is analyzed as well as the data sets we are using. I will discuss project goals, what has been implemented to date, and the problems that I encountered when trying to change the existing implementation. I will then lay out the architecture for a possible implementation in Smalltalk. The final section of this paper contains several suggested experiments.

1. BACKGROUND

To give a general overview of this project I must first describe the technologies involved. This includes the Infopipe abstraction which has been a focus of research of Professor Andrew Black and colleagues for several years, as well as an overview of the streamed media data that will be used for the project, namely traffic data collected from data stations on highways around the Portland area.

The original implementation was carried out by Emerson Murphy-Hill in 2005. His implementation took an algorithm developed by Kwon et al (2002) to estimate the number of trucks versus cars on highways that are equipped with single-loop detectors. He used California traffic data from Berkeley Highway Laboratories, which is freely available on their website, to simulate streaming.

The project outlined in this paper builds upon the existing implementation and aims to use data from the Portland area. Portland State University's Intelligent Transportation Systems Laboratory (PORTAL), has a repository of traffic data from the Portland area. As a result the goal of this project was to convert the existing implementation for Kwon's algorithm to also accept Portland State University data.

2.1. Infopipes

Professor Andrew Black and colleagues at Portland State University have been researching an abstraction called Infopipes for several years. Infopipes are an object-oriented abstraction for multi-media streaming. They are designed to simplify the task of building distributed streaming applications. With use of a series of components, or Infopipes, such as sinks, buffers, filters, broadcasting pipes, and multiplexing pipes, complex systems can be built up easily to form a complex Infopipeline. For an in depth overview of Infopipes see Black et al (2002). I will give a brief overview of Infopipes here.

Each Infopipe has a set of Inports through which data flows in and a set of Outports through which data flows out. Data can be either pushed or pulled through the Infopipes depending on the polarity of the port. We say that a port that invokes a push or pull method on another port is of positive polarity, and conversely a port that is invoked has negative polarity.

Take for example the following simple Infopipeline:

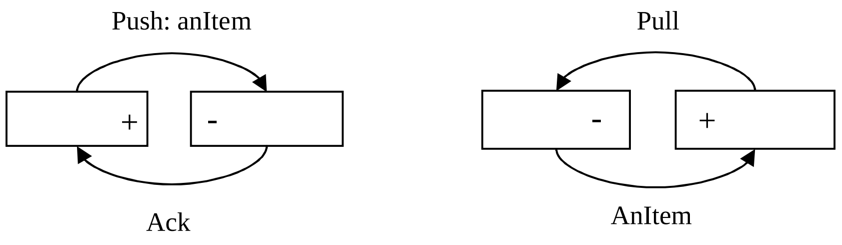

Figure 1: Push and Pull mode communication

In the first example, the downstream port invokes the method push: anItem on the upstream port. The downstream port has positive polarity since it sends a message to the upstream port. The upstream port has negative polarity since the downstream port invokes it.

We see a similar thing in the second example. The upstream port sends a pull message to the downstream port and is therefore of positive polarity. The downstream port is invoked by the upstream port and therefore is of negative polarity.

Some commonly used Infopipes are as follows:

- Buffer. This Infopipe has a negative Inport, a negative Outport and some storage. Data flows into the buffer by being pushed by downstream Infopipes, and only flows out of Outport when given a pull message by upstream ports.
- Pump. This Infopipe has a positive Inport and a positive Outport. It pulls data into it at a rate that we can set within the Infopipe and pushes it out of its Outport.
- Split tee. This is an Infopipe that has one Inport but several Outports. Data that flows in is then split and forwarded out the Outports.
- Merge tee. This Infopipe has several Inports, but only one Outport. Data flows in through the Inports and is then forwarded out the single Outport.

These simple Infopipes can be combined to form large and complex Infopipelines. We aim in this project to be able to construct a reasonably complex Infopipeline to analyze our streamed traffic data from simple Infopipe components such as those described above.

2.2. Streamed Traffic Data

Traffic sensors are installed on many highways in the United States in order to monitor the volume and flow of traffic and stream this data back to data centers for analysis. These sensors come in the form of loop detectors, which are lain under the surface of highways and can detect when vehicles moving on and off them. These loop detectors are either:

- Single-loop detectors: a single sensor laid under the highway which can sense when a vehicle goes on the loop and subsequently leaves the loop.
- Dual-loop detectors: 2 single loops laid in series a known distance apart in some lane on the highway. These provide more data than single-loop detectors as they can measure the time between when a vehicle moves onto the first loop and then onto the second can. This can be extrapolated to provide data on how long the vehicle is as well as vehicle velocity.

In the Portland area, single-loop detector data is collected. The PORTAL project, within Portland State University’s Intelligent Transportation Systems Laboratory, has a repository of this data and no access to the more informative dual-loop data.

2.2.1 How Single-Loop Traffic Detectors Work

Figure 2 illustrates the data from a single-loop traffic detector. The loop detector emits a series of on-off pulses, which register when a vehicle passes on and off the sensor. As you can see from Figure 2 we can easily count the number of vehicles that pass over the loop, but further analysis is more complicated. As an example, let us attempt to analyze the data from Figure 2 and make assumptions about vehicle classification and velocity. I refer to the data in the figure as being a series of lanes, with the data in the top third of the graph being lane 1, the middle data being lane 2, and the bottom third being lane 3.

The data from lane 1 could be interpreted in a number of ways. The short duration of each on-off pulse in lane 1 could indicate that:

a) Several vehicles of short length (i.e. cars) had passed over the loop detector.
b) The velocity of the vehicles in lane 1 is high.

In addition, comparing lane 1 and lane 2 we could argue either that:

a) The velocity of vehicles in lane 1 is constant and fairly fast in comparison with the velocity of lane 2 where the long on-pulse indicates that the traffic had slowed down temporarily
b) A truck traveled along lane 2 resulting in the long on-pulse, followed by a car, resulting in the short on-pulse that followed.

From these contrasting conclusions it is evident that without further information we are unable to make any conclusions. The single-loop traffic data alone simply does not provide us with enough information.

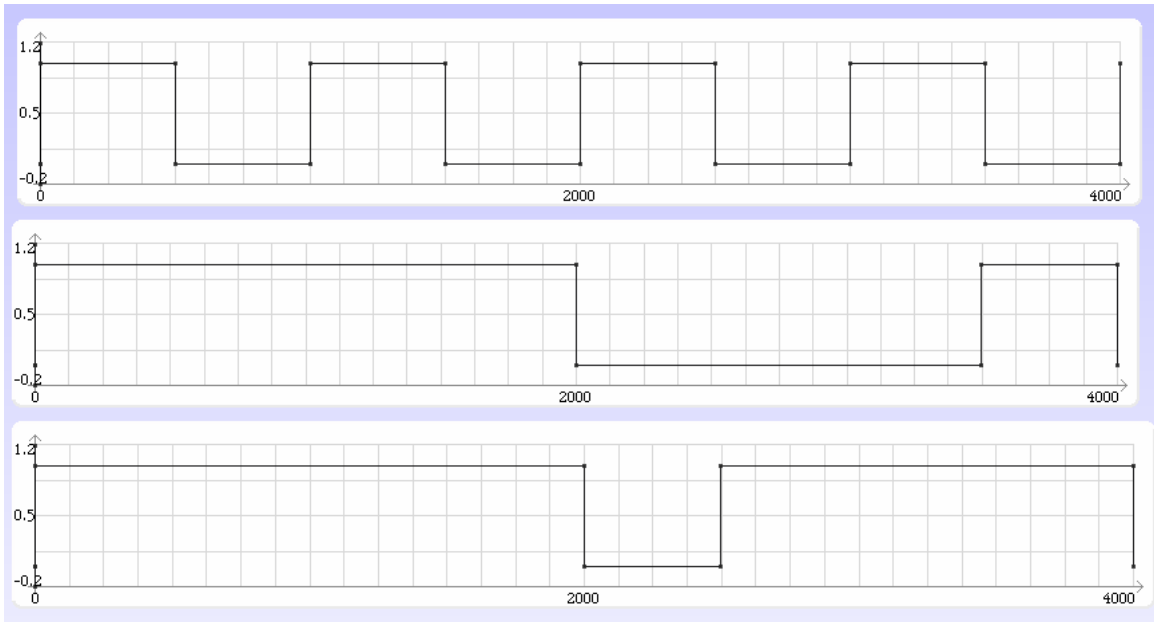

Figure 2: An illustration of data retrieved from Single-loop traffic sensors (courtesy of Emerson Murphy-Hill)

Luckily there are several algorithms which take data from single-loop traffic sensors and estimate the breakdown of cars versus trucks. The Kwon algorithm, which we are implementing in this project, does this very thing and bases its estimations on two key assumptions:

1. Trucks are not allowed on the inner (fast) lane of the highway being monitored
2. The mean velocity of each of the outer lanes is assumed to be a certain percentage less than the velocity of the inner lane.

These assumptions allow us to interpret the data in a more meaningful way. For example, if we assume that no trucks are travelling along the innermost lane and we know the average length of cars, then we can find the average velocity of the vehicles travelling along the inner lane. By assuming that the velocity of vehicles in Lane 2 is 5% less than the speed of Lane 1 we can extrapolate the velocity of vehicles in Lane 2. We can do the same with Lane 3. Once we know the mean velocity of each lane, we can then calculate the length of each of the vehicles that crosses the sensor.

A more detailed overview of Kwon's algorithm is left until [xxxx].

2.2.2. Berkeley Highway Laboratories Data versus PORTAL Data

Berkeley Highway Laboratories Data
The original Infopipe implementation of Kwon's algorithm on streamed traffic data uses data from Berkeley Highway Laboratories which has the advantage of providing both single-loop and dual-loop traffic data. Our proposed implementation uses PORTAL data which is single-loop, and which differs slightly from the Berkeley data set.

Berkeley Highway Laboratories data is of reasonably fine granularity. It is emitted as one second aggregates of on / off pulses which indicate when a vehicle passes over a loop. Each data station has a number and emits a stream of data to the data center. It encodes within it the time, then name of the station and the lane number. A sample raw data file entry can be seen in Figure 3.

```
1014956125000          1014956125000  0       7150    8       27100EF14892E44
```

Figure 3: Sample Raw data file entry

Fields of interest to us in Figure 3 are:
- First series of digits – refer to the time stamp for that data entry
- Fifth series of digits – refer to the station identification number
- Final series – Controller Data Packet as seen in more detail in Figure 4.

```
27100EF14892E44
```

Figure 4: Sample Controller Data Packet

To decipher the Controller Data Packets, the BHL web site says the following:

Each block of three characters, starting from the 8th position and continuing through all but the final two characters of Field 6, are encoded loop transitions. Each three-character hexadecimal substring, when converted to its equivalent twelve bits in base 2, describes a single loop transition. The first six bits count the number of 1/60 second intervals ("ticks") that have elapsed at the time of the loop transition. The seventh bit indicates the transition type ("on" or "off"). The final five bits indicate the controller port from which the transition was observed; the port corresponds to a particular loop at that station, as indexed in the Controller configuration files.

From http://bhl.its.berkeley.edu:9006/bhl/data_guide/reference/controller_packet.html

The encoding for this Controller Data packet is described at the Berkeley Highway Laboratories web site.

Portal Data

The Portal data, in comparison, is aggregated into 20s chunks and is emitted as Total Occupancy and Volume sums over this 20s time period. The raw data is analyzed for us by a script that is run each night and aggregated into Excel files which maintain the following information

- Station identification number
- Time
- Vehicle count
- Occupancy

This pre-processed information is a double-edged sword. While in some ways it is convenient be able to ignore the problems of converting raw data to meaningful information, there are a number of drawbacks.

a) We do not know the details of how these values are calculated, as they are calculated in a script which is run every 24 hours which we have not been able to analyze.
b) Are the aggregation procedures accurate?
c) Is data lost when the data is aggregated?

The final question is an interesting one. As explore it further, let us examine Lane 1 data in Figure 1.

At the beginning of the time period in Figure 1, a vehicle is already on the sensor for Lane 1, characterized by the fact that the time period starts out registering an on-pulse. At the very end of the time period a vehicle moves onto the sensor, as we register an on-pulse with no corresponding off-pulse. When analyzing the raw data we can take into account these orphan pulses and if we register an on-pulse without a corresponding off-pulse in one time period, we can roll the on-pulse over into the next time period. It is interesting to speculate whether the PORTAL data aggregation script does this more sophisticated analysis when it processes the streamed data. If it doesn’t how much does this affect our results?

3. PROJECT GOALS

As a result of the above work as well as speculation over the accuracy of information received from an aggregated stream of data as described above, the goals of this project were as follows:

- To convert the existing Infopipeline which calculates truck and car volume for single-loop data for the Berkeley Highway Laboratories (BHL) to accept data from the Portal web site.
- To aggregate BHL data to the format used by PORTAL and compare the results of both formats to analyze whether accuracy had been lost in the aggregation.

Secondary goals of this project were:

- To understand the Infopipes abstraction
- To learn Smalltalk (to implement the Infopipeline)
- To analyze the existing pipeline and come up with an algorithm to allow it to accept BHL data.

4. PROBLEMS ENCOUNTERED

It seemed, initially, to be a fairly trivial task to convert the existing Infopipeline to accept a slightly different data format. However, there were three main problems.

Firstly, as I mentioned earlier, the data available from BHL and Portal are intrinsically different: BHL outputs data as a one second aggregation of on/off pulses for each loop; Portal data is outputted as pre-calculated Occupancy and Vehicle Counts which is aggregated over a 20 second period. The existing pipeline designed for the BHL data set passes a stream of on/off pulses (a Controller Data object) through the entire pipeline and calculations are performed on this stream throughout. Thus the entire pipeline relies on a string, or a Controller Data object. The Portal data doesn’t need Controller Data objects as they represent the raw data from the traffic sensor. As a result, integrating a very new and different data format into the existing pipeline is a non-trivial problem.

Secondly, the existing pipeline uses at its heart two objects that calculate the Occupancy and Vehicle Count based upon the Controller Data objects that are passed into them. These Occupancy Pipes, and VehicleCounters occur throughout the pipeline in different places. As I mentioned earlier, Portal data already consists of aggregate Occupancy and Vehicle counts for each 20-second period.

To use the existing pipeline to integrate both a new data format which already calculates the very heart of the existing pipeline in an elegant, object-oriented fashion seemed to be a non-trivial task.

A third and final problem exists when considering the problem of whether accuracy is lost in the process of aggregating data into 20 second chunks. We do not know how the PORTAL aggregation script works. If the orphan pulses that I described in the section above are not taken into account, our results could be significantly skewed.

5. ALGORITHMS

I devised the following alternative procedures to deal with the above problems. Here I discuss the pros and cons of each of the algorithms in turn.

1. Convert BHL data to Portal format and rewrite the existing implementation to understand this format.

   It would be possible to rewrite the existing implementation to understand a different format. This was my original intention before I had thoroughly examined the existing implementation. However, due to the problems outlined above and the myriad differences that would have to be made to the implementation, I decided that this would be a hack and a non-trivial one at that.

2. Build a new pipeline which uses a generic data format and calculates the total Occupancy and Vehicle Count at the beginning of the pipeline.

   Since Occupancy and Vehicle Count totals have already been calculated for us, the pipeline would be much simpler than the original implementation. We would then simply have to convert the BHL data to Portal format and compare data from the original implementation to the new implementation.

   The benefit of this approach is that it would result in a pipeline that allowed generic data to be passed into it.

   Problems with this approach are that it is entirely reinventing the wheel. Very little code reuse would be possible.

3. Convert Portal Data to BHL format and run it through a slightly modified pipeline.

   This is probably the simplest solution of all three. The idea would be to convert the Occupancy and Vehicle Counts back into the raw data format of on / off pulses. This would enable almost total code reuse.

   This is an unsatisfactory solution in several ways, the greatest of these being the amount of replication that this would incur. Occupancy and Vehicle Counts would be converted to raw data and passed to the pipeline which would calculate Occupancy and Vehicle Counts once again. It is a temporary hack for quick results but not the best solution.

As a result I decided to implement the second solution and rewrite the pipeline. I attempt to make the algorithm as simple and as reusable as possible.

6. IMPLEMENTATION

6.1. Kwon's Algorithm

A brief description of the algorithm that we will implement follows.

Kwon's algorithm takes single loop traffic data from a number of lanes on a highway. It is based on two assumptions.

- The inner lane on a highway doesn't have any trucks on it. This is a law in some states and just best practice in others.
- Highway lanes have lane to lane velocity correlation. It has been observed by manual counting that the velocity in each lane decreases about 5% from the previous where lanes are numbered from innermost to outermost with the inner lane being the fastest of them all.

As a result of these two assumptions, Kwon et al are able to use single loop data to figure out the number of trucks on the road in a given period of time.

There are a few algebraic equations that we need to define before we go into Kwon's algorithm, and a few basic definitions.

- Occupancy, O, is the proportion of time a given point is occupied by vehicles.
- Flow, F, is the number of vehicles that pass a given point in a given time period.'
- Number of vehicles, n
- Time, t
- Period kth vehicle occupies the loop, t^k

The PORTAL data supplies these values pre-computed for us but algebraically we could show them as:

$F = n/t$ (Flow = number of vehicles / time)

$O = \sum_{l}^{n} t^k / t$ (Occupancy = Sum of time each vehicle spends on loop)

Rather than going into the specifics of Kwon's algorithm, which can easily be read in his paper, I will instead give a general overview of how it works and give an example of the calculations.

The basic idea of Kwon's algorithm is as follows. We know the number of vehicles in a given time period over lanes on a particular point on the highway, and we know how long they spend over the single-loop detector. Using the two assumptions above with estimations of car and vehicle length, we can separate cars and trucks from each other when observing single-loop data.

Estimated lengths for cars and trucks is:

- Length of cars, L_c = 18.6 ft
- Length of trucks, L_t = 61.2 ft

6.2. General Example

Lane 1 (Truck-free)
We take the Count and the Occupancy from the PORTAL data set, the mean vehicle length of that lane (18.6ft) and then calculate the mean velocity, V_1, based on these figures:

$$V_1 = L_c \times F / O$$

Lane 2 (5% slower than Lane 1, with trucks)

In this case we do not know the length of the vehicles but we do know the velocity of the vehicles as it is 95% the value of the velocity of lane 2 which we derived from our initial assumptions. Therefore we can calculate the Number of trucks as follows:

$$V_2 = V_1 \times \frac{95}{100}$$ Mean Velocity of Lane 2

$$L_2 = V_2 \times \frac{O_2}{F_2}$$ Mean Vehicle Length of lane 2. O_2 is Occupancy of lane2, F_2 is Flow of Lane 2

$$PT_2 = \frac{L_2 - L_c}{L_t - L_c}$$ Proportion of trucks for Lane 2, PT_2

To get the actual number of trucks that have passed we can use the following equation:

$$T_2 = PT_2 \times n$$ Number of trucks for lane 2, T_2

The subsequent lanes can be calculated in a similar way to lane 2. We would estimate the velocity of the lane based on the 5% decrease from the previous lane. We would then calculate the Proportion of trucks based on our length calculation above, and then the overall number of trucks based on the fourth equation above.

6.3. Infopipes Implementation

The implementation of this algorithm using Infopipes should be reasonably simple. The pipeline in figure 4 demonstrates how it should work.

Each pipeline corresponds to the equations above and is designed to take PORTAL data, which is correlated, into 20-second aggregates.

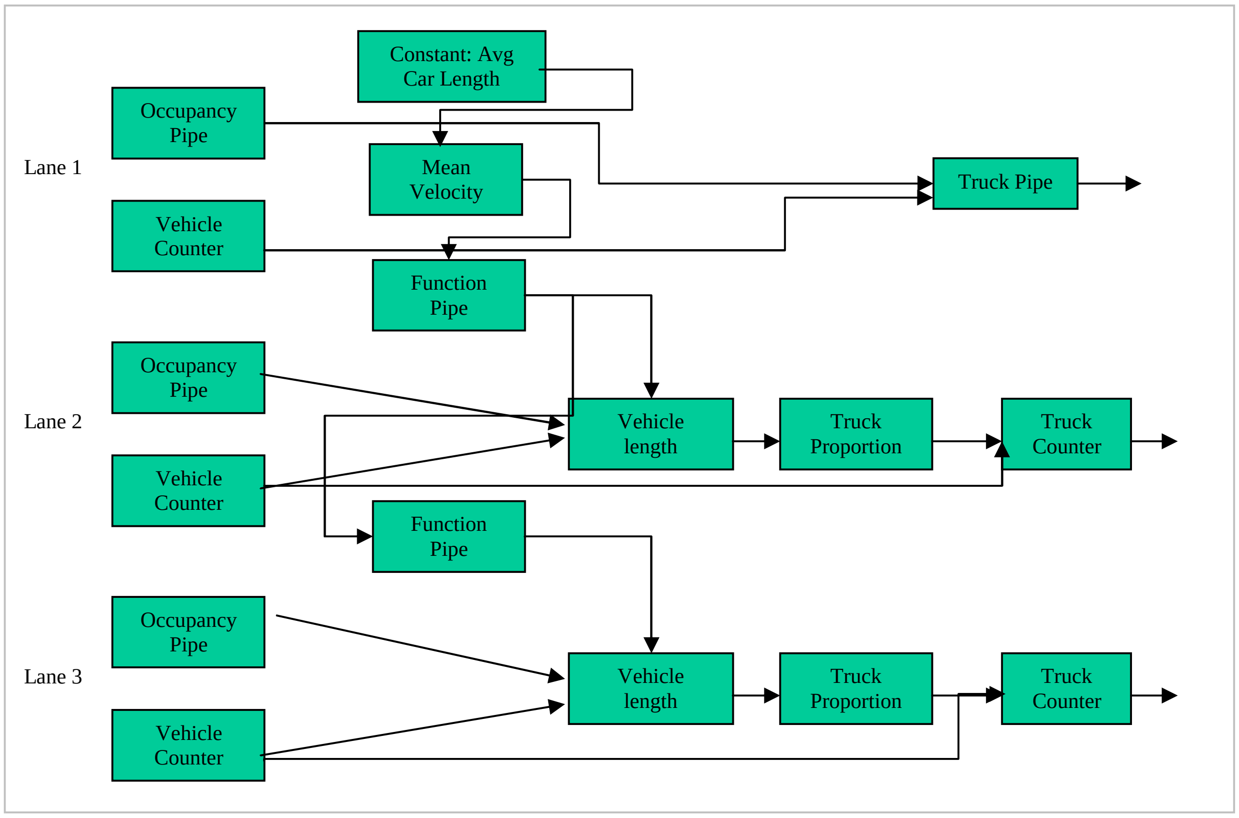

Figure 5: Proposed Pipeline

6.4. Description of the Pipeline

- **Vehicle Counter**
  This pipe would take PORTAL Flow data from the data file and convert it to numeric format. This pipe could then be easily modified to form a converter pipe which converted any data format to a generic vehicle count format.

- **Occupancy Pipe**
  This would simply take PORTAL data and convert it to numeric format. As with the Vehicle Counter, this pipe could also be easily be replaced or adapted to form a converter pipe which converted any data format to a generic vehicle count format.
- **Mean Velocity**
  This pipe would take as its inputs the occupancy, flow and car length for lane 1 and output the mean velocity for that lane.

- **Function Pipe**
  This pipe performs the velocity correlation function, takes the velocity of the previous lane and computes the velocity of the next lane, which will be 5% less than the previous lane based on our assumptions. This

- **Vehicle Length Calculator**
  The vehicle length calculator takes as inputs the occupancy and vehicle count for that lane as well as the average speed for that lane and outputs the mean vehicle length.

- **Truck Proportion Pipe**
  This pipe takes as input the mean vehicle length and calculates the proportion of trucks to cars within that.

- **TruckCounter**
  This pipe takes the proportion of trucks to cars and the total number of vehicles counted and converts that into a numeric count of how many trucks and cars passed along that lane.

- **Truck Pipe**
  The truck pipe would simply take the vehicle count from lane 1 and output the number of trucks and cars for that lane (0 trucks and x cars).

6.5. Converting Berkeley Highways Laboratory Data to PORTAL format

In order to test the validity of the PORTAL aggregated data versus the fine granularity Berkeley Highways Laboratory data, we need to convert the BHL data to PORTAL format and see whether any accuracy is lost.

As mentioned in earlier sections, the method by which the PORTAL data is aggregated is a mystery. We don’t know whether orphan on- or off-pulses which occur at the boundaries of the aggregation periods are matched up with their corresponding off- or on-pulses in the subsequent time period. Because we don’t know the aggregation method for the PORTAL data I suggest that a few experiments with the BHL conversion be carried out. These experiments should make it clear to us how the PORTAL aggregation script works and whether we are losing accuracy or not. The experiments I suggest are as follows:

1. Convert the data in an optimum way. This means when the 20 second aggregation boundaries are met, ensure that any on-pulses that have not yet received corresponding off-pulses, are recorded and then aggregated into the following 20 second time period. This should ensure that no cars / trucks are lost in the period between the two 20 second time intervals.

2. Convert the data and discard all on-pulses that do not have corresponding off-pulses.

3. Convert the data and discard all off-pulses that do not have corresponding on-pulses

4. Convert the data and discard all on-pulses that do not have corresponding off-pulses and off-pulses that do not have corresponding on-pulses.

Data from these experiments when compared with the data resulting from calculations deduced from the BHL raw dataset should give us some interesting results which will show one of the following:

- Much accuracy is lost when aggregating data into 20-second chunks and therefore raw data such as BHL's data set is highly preferable. If we discover that the PORTAL data aggregates according to algorithm number 4 in the section above, then this result may be the case.
- Some accuracy is lots due to a particular flaw in the aggregation algorithm, corresponding to case 2 or 3 above.
- No accuracy is lost at all, which will be an interesting discovery in its own right.

6.6. Implementation of the Pipeline for these Further Experiments

In order to carry out these experiments it is necessary for us to change our pipeline somewhat. The data we will be using is BHL data which is in raw-data format: a series of on off pulses. Emerson Murphy-Hill has an excellent data converter in his Traffic implementation where he converts these on-off pulses in a unit called a ControllerData. This Controller Data takes the on-off pulses and converts them to Occupancy and Vehicle Counts. As a result we can change our pipeline slightly to incorporate these elements. See Figure 6.

Adapted Pipeline to Convert BHL Data to PORTAL Format

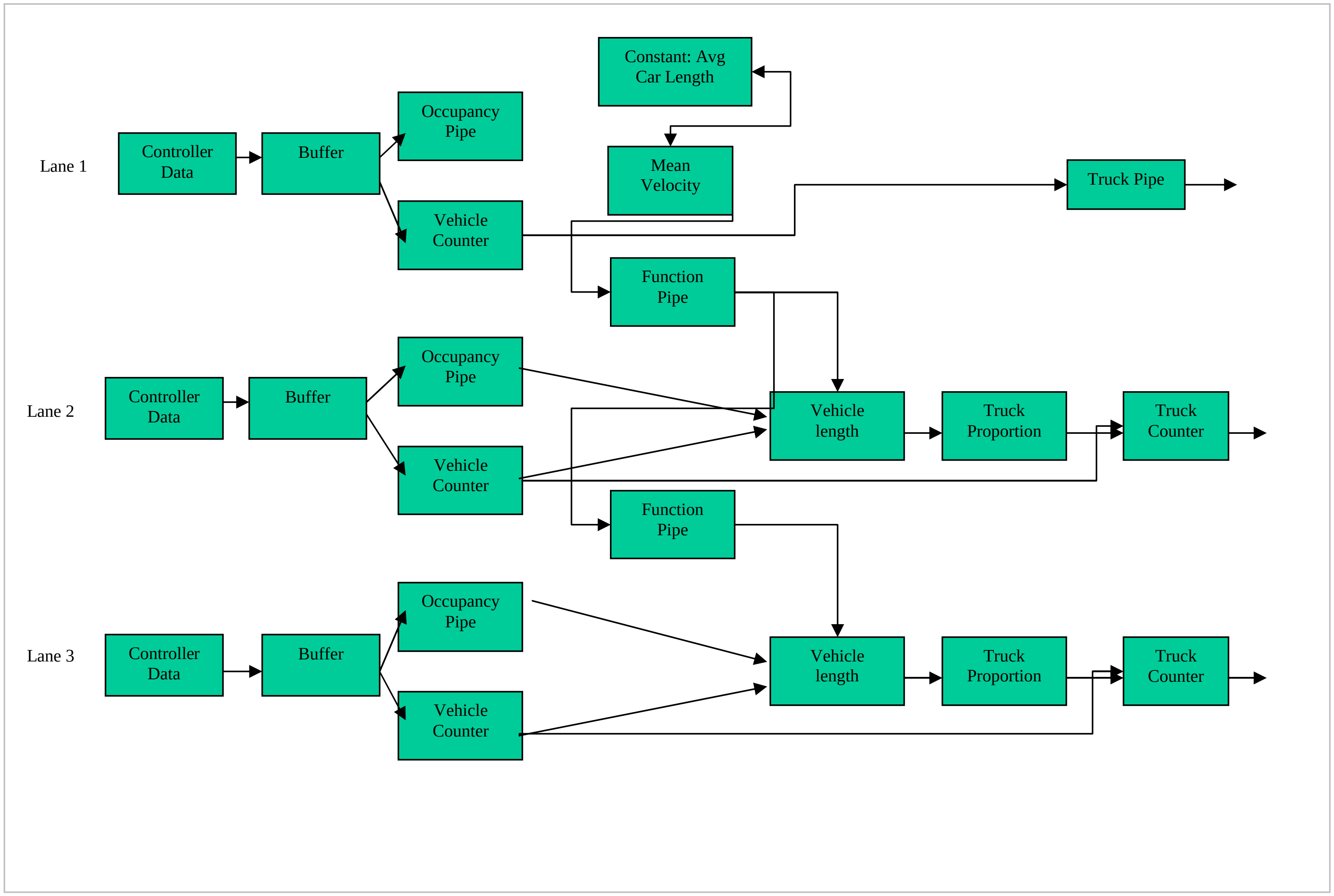

Figure 6: Adapted Pipeline to Convert BHL data to PORTAL format

6.7. Further Experiments

Once we have detailed the accuracy of the PORTAL data set versus the BHL data set we can then start looking at how accurate the data from the single-loop traffic sensors is in comparison with data from the dual-loop traffic sensors. These experiments could be carried out with Berkeley Highway Laboratories data, with our pipeline as described above, and then compared with data from dual loops which can also be found on their website. This will give an interesting insight into whether or not single-loop data can be used to reasonably accurately measure highway volume.

7. SUMMARY

In this paper I have outlined the initial project as it was assigned to me in Fall Term 2006. I then discussed the background to the project, the Infopipes abstraction, Kwon's traffic algorithm, and problems that I saw with the current implmentation.

I then have outlined a pipeline that I think should be straightforward to implement that will integrate well with the excellent graphing framework that has been set up by Emerson Murphy-Hill in his implementation.

Overall there are some interesting conclusions that could be drawn from the experiments that I have outlined in this paper. Whether or not it is necessary to keep data of such fine granuarlity such as that which Berkeley Highway Laboratories is an interesting question. Also it is interesting to see whether the single-loop data is reasonably accurate when compared to data from dual-loops or whether much information is lost. This could be of great importance when considering whether to upgrade entire highway systems with dual-loop traffic sensors, or to simply make do with a single-loop sensor.

8. USEFUL INFORMATION

Visit http://www.cs.pdx.edu/~cv/infopipes for a list of sources for the experiments, as well as the papers that I have mentioned.

Visit Emerson Murphy-Hill's website at http://www.cs.pdx.edu/~emerson for details of his work into the Infopipes traffic implementation.

Berkeley Highway Laboratories: http://bhl.its.berkeley.edu:9006/bhl/research/index.html

PORTAL: http://portal.its.pdx.edu/

9. REFERENCES

Jaimyoung Kwon, Pravin Varaiya, and Alexander Skabardonis, "Estimation of Truck Traffic Volume from Single Loop Detectors Using Lane-to-Lane Speed Correlation" (July 1, 2003). California Partners for Advanced Transit and Highways (PATH).

A. P. Black, J. Huang, R. Koster, J. Walpole, and C. Pu, "Infopipes: An Abstraction for Multimedia Streaming" (2002). Multimedia Systems 8: 406 - 419

R. Koster, A. P. Black, J. Huang, J. Walpole, and C. Pu, "Infopipes for Composing Distributed Information Flows," Oregon Graduate Institute, Department of Computer Science, Beaverton, OR 01-005, 2001

E. Murphy-Hill, Report on his implementation, Portland State University, 2005