Proceedings of the 2011 Winter Simulation Conference
S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, eds.

PERENNIAL SIMULATION OF A LEGACY TRAFFIC MODEL: IMPLEMENTATION, CONSIDERATIONS, AND RAMIFICATIONS

Seth N. Hetu
Gary Tan

National University of Singapore
Department of Computer Science, School of Computing
Computing 1, 13 Computing Drive, SINGAPORE 117417

ABSTRACT

A prototype implementation of a "perennial" simulation framework, previously introduced for crisis management studies, is detailed and applied to a simple traffic dynamics study. In addition to proving the framework, this study also demonstrates a method for encapsulating legacy models and simulations to allow them to interact compatibly with the framework. Finally, the traffic application itself features a straightforward and logically pipelined image processing algorithm intended to analyze traffic logistics data from security cameras in real-time –an important prerequisite for symbiotic simulation. On a single-processor machine, traffic data are extracted at an acceptable 500ms/frame, with a few caveats.

1 INTRODUCTION

Modeling and simulation is a mature, extensively studied and generally well understood field. The process of creating, validating, and running a model has been presented by several different authorities (Law 2007, Sargent 2007), and there is a general consensus on what comprises a quality simulation study. To put it another way, a simulation engineer who wishes to perform a study of a given phenomenon has a wealth of information to avail herself of.

The purpose of any simulation is to answer a domain-specific set of questions. Over the years, simulation methodology and computer hardware have both improved in great strides, allowing ever more complex questions to be answered. Modern systems may incorporate multiple models and simulations and a variety of data sources; as a result, the meta-level of organizing simulations themselves has become increasingly more complex. Technologies such as the HLA (IEEE 2000) have greatly improved interoperability and time management, while others have focused on syntactic and semantic composability (Pidd 2002). The advent of symbiotic simulation, however, led to an all-new challenge in simulation organization: that of tracking a huge variety of sensor inputs and simulation outputs. This was necessary because of symbiotic simulation's focus on "What If?" analysis and a constantly-running model, both of which demand large amounts of specifically targeted data (Fujimoto et al. 2002). We have argued that the lack of a reasonable conceptual approach to organizing simulations in a symbiotic context has limited both the power and the applicability of symbiotic simulation, in particular restricting it to domains with very simple performance metrics. We have further made the case that symbiotic simulation is suitable for crisis management, as a means of answering complex and time-critical questions. We introduced the concept of a "perennial" simulation (detailed in Section 2.3) to broaden symbiotic simulation's domain (Hetu and Tan 2010). Our previous work in perennial simulation introduced the framework at a conceptual level and listed several recommendations, but did not provide an implementation.

This paper builds off our previous work and provides several key pieces of new research. First, it provides a prototype implementation of a perennial simulation framework written in Java. To demonstrate

this implementation, a traffic simulation study is performed on a small university campus traffic network. The goal of the study is to estimate the time taken by security vehicles to respond to incidents under different levels of congestion and at different areas on the campus. Second, we present our research in the direction of extending perennial simulation to support legacy systems. One of the concerns voiced over our original framework was its requirement that all models be agent-based and run in discrete time steps, which would obviate many existing (and otherwise acceptable) models. Support was added for these "legacy" models through an abstraction mechanism that allowed them to be used for *hindsight* and *foresight* studies. Third, through careful configuration of existing technologies, we attempt to mine traffic data from security cameras in real-time. An abundance of such data are required for non-legacy models running symbiotic simulations, and our automated image processing algorithm achieved reasonable results. (It should be noted that there are some limitations to our solution; these are presented in Sections 3.4 and 4.3 along with their workarounds.) Finally, several secondary aspects of the perennial system prove interesting and useful in their own right, such as history windows and sensor dependency trees. These components might prove useful in other simulation studies unrelated to our work.

The remainder of this paper is split up as follows. Section 2 covers related work in the field. Section 3 details our implementation, traffic simulation, and image processing algorithm. Section 4 covers the results of our study, and Section 5 concludes the paper.

2 RELATED WORK

2.1 Traffic Models

Our work benefits from existing traffic models, of which the field is well-saturated. Boxill and Yu (2000) provides a thorough overview of the most-used traffic simulation packages as of the turn of the century. Broadly speaking, these models operate at one of three resolutions. *Microscopic* models are the slowest and most accurate, operating at the level of each vehicle. *Macroscopic* models calculate congestion based on each "link" (road and intersection). They are less accurate but much faster. *Mesoscopic* models operate in between the two extremes, both in terms of accuracy and performance. Some studies operate at multiple resolutions, such as Claes and Holvoet (2009), which leverages both micro- and mesoscopic models simultaneously. Recently, microscopic models have received a great deal of research; Chowdhury, Santen, and Schadschneider (2000) explores them in more detail.

The most feature-complete simulation packages were AIMSUM, Paramics, and DYNAMIT. Other interesting models like DRACULA and NEMIS focused on smaller tasks such as driver behavioral patterns over time or evaluating the impact of route choice. Some newer research attempts to combine microscopic traffic models with agent-based simulation techniques. The black-box nature of agents is an attractive mechanism for abstracting driver behavior, and provides a modular means of adding new types of agents such as pedestrians or traffic lights. Tao and Huang (2009) provides a good example, using the FIPA-compliant JADE framework (Bellifemine, Caire, and Greenwood 2007) to design a traffic simulation which represents cars as agents and can dynamically shuffle agents onto different networked computers if the resources of one system become exhausted.

Our traffic study relies on the work of Kraus, Wagner, and Gawron (1997) and Kraus (1998), which defines a microscopic, collision-free, space-continuous traffic model with a focus on driver imperfections and jam modeling. Drivers respond to congestion through "dynamic user assignment", which is implemented as described in Gawron, Kraus, and Wagner (1997). The SUMO simulation package provides an integrated framework for all of these approaches. (SUMO 2011) In addition to being backed by sound reasoning and mathematical rigor, Kraus's work was chosen because it fits both requirements of a "legacy" model: it is not an agent-based model, and it cannot be run in real-time in discrete time-steps. (In fact, SUMO *does* use a discrete timer, but the dynamic user assignment strategies require multiple simulation runs, and cannot be performed in sync with real time.) Thus, Kraus's model is an ideal example of a high-quality legacy model that one might wish to incorporate into a perennial simulation.

2.2 Traffic-Related Image Processing

Early work in automated traffic-related image processing focused on identifying traffic violations and congestion, with little thought given to using the mined data as input into a real-time simulation. Atkociunas et al. (2005) describes the various independent steps of an image processing approach: background elimination, edge detection, and object labeling, as well as various filtering steps are considered. This "region-based" approach is criticized by Beymer et al. (1997) and Tan, Sullivan, and Baker (1996) as inaccurate under conditions of heavy traffic. Both achieve acceptable real-time performance despite hardware limitations at the time. Beymer's group proposes tracking sub-components of cars called "features," and clustering features with similar trajectories. Tan, Sullivan, and Baker, however, uses pre-computed orthographic projections of existing 3-D car models combined with real-time hypothesis generation to predict and group cars efficiently. One weakness with this approach is its reliance on existing vehicle models.

Later work broadened the application of mined traffic data to include, among other things, automated signal control. Lin, Chao, and Lo (2010) provide a good example of this sub-domain, which studies the possibility of controlling traffic signals dynamically in response to actual traffic data. They leverage Dynamic, Data-Driven Application Systems (DDDAS) (Darema 2005) –a simulation-driven feedback loop very similar to symbiotic simulation– to predict the optimal traffic signal patterns in a series of complex intersections. The author implies that traffic cameras will provide the input data, but it is unclear how much of this is actually occurring in real-time. The study is notable for its emphasis on modern techniques such as DDDAS and the Service-Oriented Architecture, which is increasingly becoming a popular means of simulation interoperability. Other studies which attempt to dynamically adjust traffic signals include Hewage and Ruwanpura (2004), which sadly does not focus on real-time data or symbiotic response.

Our work also benefited from the ViBe background elimination algorithm (Barnich and Droogenbroeck 2010) and Perreault's median filter (Perreault and Hebert 2007). ViBe is notable for its performance, resistance to camera jitter, and need for only 1 frame of startup data. Perreault's approach uses an array of past kernel values to compute each pixel's filtered value in constant time with respect to kernel size. Using these techniques, we chose to create our own region-based tracking algorithm (described in Section 3.4). Region-based tracking seemed acceptable, as our traffic network experienced only moderate levels of congestion. Unlike most related work, our footage comes from security cameras, which are often badly positioned with respect to traffic. This presents a challenge to our object tracking algorithm, and ultimately puts a limit on the quality of traffic data obtainable from our cameras.

2.3 Real-Time and Symbiotic Simulation, and Virtual Worlds

Until very recently, computer hardware was not powerful enough to run complex simulations in real-time. Around the turn of the century, however, the military began exploring how one might use real-time simulation for training. Distributed Interactive Simulation (DIS) (IEEE 1993) and the High Level Architecture (HLA) (IEEE 2000) are standards that attempted to encompass all things relevant to the emerging field. Despite their general success, several academics noted drawbacks with these systems from the perspective of research (e.g., Davis and Moeller 1999). Work such as Boukerche and Zhang (2008) explored using a non-military technology, the Service-Oriented Architecture (SOA) to create real-time simulated virtual worlds, while McGrath et al. (2005) created a completely new protocol with the goal of being agile.

The idea of leveraging virtual worlds for research is by no means new (e.g., Cruz-Neira, Sandin, and DeFanti 1993), and is something we have explored as well (see Hetu and Tan 2008). Generally speaking, virtual worlds are used to train real users, to gather data from real users, or to visualize a simulation (either in real-time or as a playback). "What If?" symbiotic analysis may operate on the physical world during an actual threshold event, or it may operate on a virtual world during a training event.

DDDAS, mentioned earlier, pushed past the boundaries of DIS and the HLA by attempting to drive the simulation process in response to real-time sensors. This technique also provided a means of allowing simulation feedback to relocate sensors, based on the assumption that the input space would be too vast to

map in its entirety (Kennedy et al. 2007). Likewise, symbiotic simulation focuses on applying multiple "What If?" analysis to a real-time simulation in an attempt to optimize the system. It was originally intended to deal with shipping, manufacturing, and other domains that were easily automated (Fujimoto et al. 2002).

This paper draws heavily from our past work developing a perennial simulation framework, which will be discussed more fully in Section 3. Briefly, a perennial simulation is defined as a meta-level organization of models, simulations, sensors, and effectors that attempts to maximize the ease at which one can perform *foresight*, *hindsight*, and *symbiotic decision support* studies. This framework was motivated by our observation in Hetu and Tan (2009) that symbiotic simulation had a potential positive effect on crisis management. Its complete description is provided in Hetu and Tan (2010). Details will be provided throughout the remainder of this paper as they become relevant to the discussion of our implementation.

3 IMPLEMENTATION PROTOTYPE

3.1 High-Level Overview

Our framework implementation is detailed in Figure 1, which is based off an early diagram found in Hetu and Tan (2010). The *sensescape* is composed of multiple sensors which may provide real-time, estimated, or saved historical data. Sensors may depend on each other for input (for example, one sensor may provide an average of multiple other sensors), and the sensescape contains a *dependency tree* which records dependency relationships between sensors and ensures that sensors compute their input in a consistent order. Sensors are grounded to a *world* and a *target*, and the combined timespans of all sensors for a given world and target form the *history window*, which describes which data are actually cataloged by the sensescape. Real-time sensors' data points are saved automatically into a database, thus extending the history window. *Effectors* function similarly to sensors, except that instead of reading the target of a given world, they modify it. Effectors are much simpler than sensors, and do not contribute to either the dependency tree or the history window. Effectors are not shown in Figure 1, as they are not relevant to the current study.

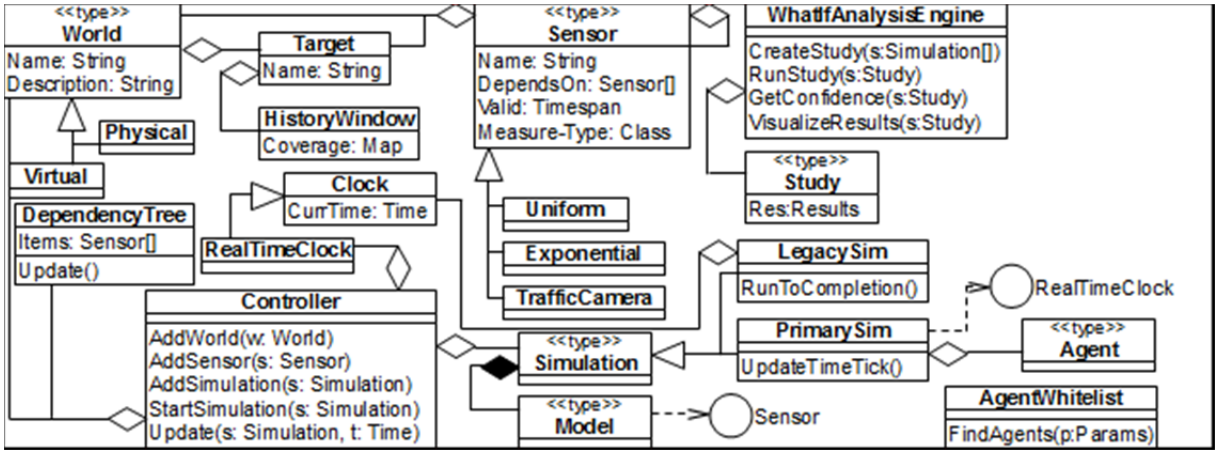

Figure 1: UML diagram of our perennial simulation implementation in Java.

A *world* may be either physical or virtual. There may be many disjoint virtual worlds, but usually only one physical world is modeled. Each world has a number of *targets*, which generally represent locations in that world. The *controller* manages the complex relationships between worlds, sensors, and effectors, in addition to managing the *global clock*. This controller is also aware of a variety of *models* and *simulations*, and provides an interface for simulations to request data from a given timespan without worrying which sensor (or combination of inter-dependent sensors) will provide that data. Each *primary sim-*

ulation is known by the controller, which manages that simulation's *local clock*. Simulations and models are left loosely defined, under the assumption that developers will prefer to create their own implementations of these components; however, each is expected to expose a set of *agents*, which the *Agent Whitelist* can index in the event that a new model requests an agent of a particular type.

"What If?" analysis is also performed by the controller, which maintains a "What If?" analysis engine. This allows the creation of "What If?" analysis studies, along with a few other features.

Support for legacy simulations was added by abstracting S*imulation* into its own class and then having a sub-class called *LegacySim* which does not have access to the global clock. The controller can only request that this simulation "run to completion", which effectively limits its interaction with the "What If?" analysis engine. In addition, legacy simulations do not expose their agents (if they even use agents at all) which limits their communicability with other simulations. In short, legacy simulations are isolated, both for their own good and for the benefit of the primary simulations.

3.2 Traffic System: Non-Legacy Components

Despite focusing on legacy system compatibility, our traffic system has several components which interact with the remainder of the system in a non-legacy way. These include the world and sensors, as well as their associated history window and dependency tree.

Our traffic system contains a single world, "nus", with five targets as described in Table 1. These targets cover several thoroughfares, intersections, entrances/exits, and roundabouts, and serve to completely describe the major traffic influences on campus. (Several minor but essential sources of traffic, such as shuttle buses and city transit, are modeled using schedules and route maps.) Each target is represented using several sensors, as shown in Figure 2. These components will be described in more detail in a later section.

Table 1: Targets for the world "nus".

Target	***Description***
Upper	NW entrance, upper main road and small roundabout.
Round	Large roundabout; connects upper, east, and south main roads.
East	NE entrance, east main road.
South	S entrance, south main road, major traffic light intersection leading to SE entrance.
SEast	SE entrance, south-east main road, intersection to 2 other campus locations (which are not modeled)

3.3 The Legacy Traffic Model

The SUMO simulation package requires a road network, a list of vehicle types, and a list of routes in order to run a predictive simulation. The road network lists all "nodes" (intersections, lane splits and merges) and "edges" (paths between nodes) in a simple XML format. Rules on U-turns, number of lanes, lane priorities, and lane usage (e.g., car lanes versus bus lanes) are all specified as properties of edges, or through the use of "connections" (specialization templates describing what happens when two edges meet). In addition, bus stops are represented by automatically segmenting an edge at a given point. Buses themselves require additional logic in the vehicle types list to make proper use of bus stop edges.

The road network for our traffic study was created using a variety of GIS data sources. Particularly tricky road segments were sketched, paced, and modeled manually. Figure 2 presents just such an example: the roundabout connecting the campus's three major roadways, with cars represented in silver, taxis in blue, and shuttles in orange. Since SUMO expects traffic to flow on the right (and in Singapore the reverse is true), the road network was flipped about the Y-axis before being sent to the simulation engine. Care was taken to ensure that this did not lead to an invalid model; see Section 3.5 on validation. Only one intersection had a traffic signal, and its timing was extracted from camera footage at that location.

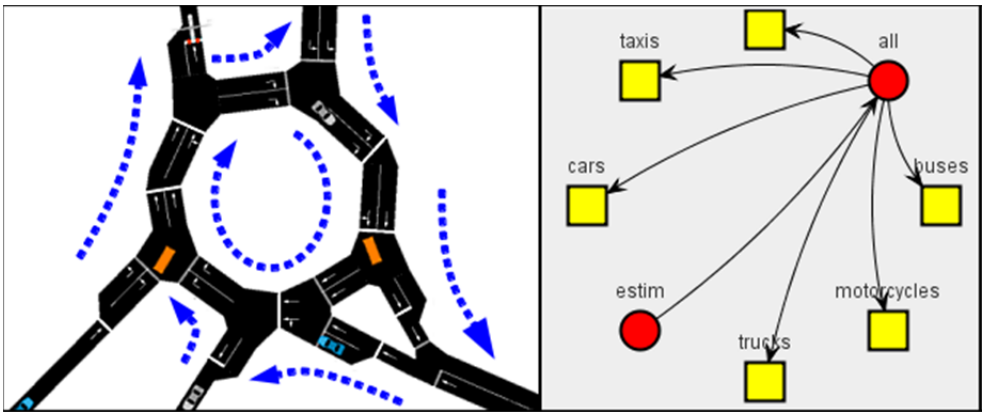

Figure 2: *Left*: The main roundabout on campus, with buses, cars, and taxis navigating it. Dashed arrows indicate general direction of traffic flow. Small 2-lane segments complicate the modeling of traffic flow. *Right*: The roundabout camera's sensor dependency tree. Squares (colored yellow) represent "leaf" sensors, which must be processed first.

Vehicles were grouped into categories with similar properties. Cars, taxis, motorcycles, and trucks each had different following characteristics as defined in (Kraus 1998), which were found to be consistent with our observed traffic data. City buses and campus shuttles functioned as larger vehicles with the additional property of periodically stopping to release and pick up pedestrian passengers. A vehicle's behavior was modified by the driver's "imperfection" parameter; this is a common technique for introducing driver variability into the model.

The route list was generated each time a simulation was requested from the *Controller*. As explained below, each simulation run could avail itself of a series of arrival times and route choices for each sensor location. Buses and shuttles were generated first, as they appeared on a regular schedule. For each shuttle/bus arrival, the nearest arrival time from the input data was tagged as being a public transport vehicle, and given a predefined route including bus stops. Then, the remaining entrance arrivals were tagged as one of the remaining car types. Each car type was given a route based on the probability of node choice from each intersection.

3.4 Traffic System: Automated Image Processing

One week's worth of traffic footage, recorded daily from 3pm till 4pm, was processed to retrieve the arrival time, estimated velocity, and on-screen entrance of each vehicle that passed within range of each camera. Available to this algorithm was a list of entrance locations (in screen co-ordinates), the distance of each possible route through that location, and a mask image which was generated by hand to help reduce the impact of obvious noise sources, such as a nearby highway, trees blowing in the breeze, or a pedestrian crosswalk.

The image processor was implemented in GStreamer, a multimedia processing framework built around the idea of a software pipeline. Audio and video streams flow from "sources" through a series of "nodes" until they reach a "sink" and terminate. This approach is highly flexible and, as we shall see later, capable of being parallelized.

The complete image processing pipeline is presented in Figure 3, but was roughly broken into four phases. First, the "input" phase read a source MPEG file, removed the audio track, and balanced the video stream. Next, the "filter" phase applied a mask to each frame, followed by the ViBe background eliminator and finally Perreault's median filter. The filtered video stream was then duplicated. The "display" phase operated on one copy of the stream and simply presented the candidate objects to the user. Figure 4 shows a sample frame from the "display" phase. Finally, the "identification" phase converted the color-

space from YUV to RGB, and performed segmentation, object identification, and object tracking in that order. After each object left the camera's range, its path and travel information was saved to a file.

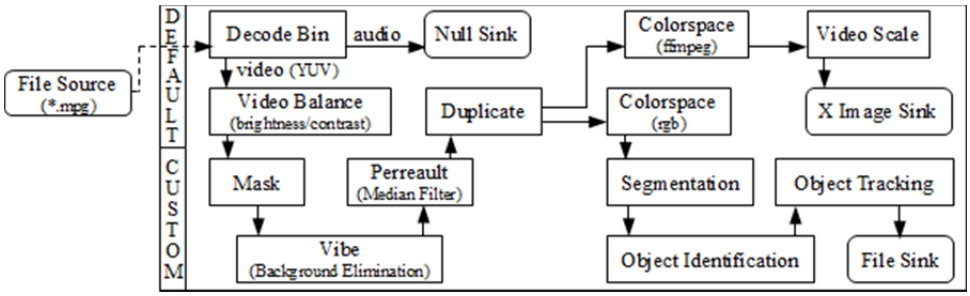

Figure 3: GStreamer pipeline for our image processing algorithm, with sources and sinks as rounded rectangles. “Default” plugins are provided by GStreamer; the “Custom” plugins were manually added.

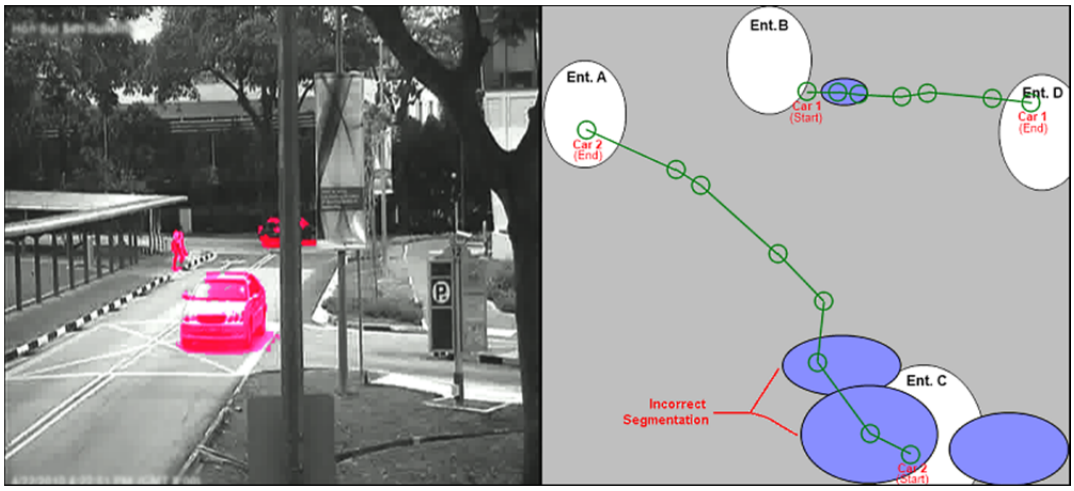

Figure 4: *Left*: Output of the GStreamer X Image Sink node; possible objects are highlighted in pink. A pedestrian (left) will be removed in the “object identification” phase. *Right*: Screenshot of image tracking system, showing a false segmentation: the two marked nodes are actually one car.

The segmentation algorithm we used grouped pixels together which shared a common minimum threshold number of neighbor pixels. Object identification was performed by first checking if each segment's size was above a certain minimum. If so, the segment was promoted to an object. If not, the segment was conditionally promoted depending on the histogram of color values it contained, under the assumption that vehicles tended to contain narrower histograms than other moving objects like pedestrians. Finally, the object tracking algorithm operated in two steps. First, new objects were detected by checking if they overlapped one of the entrance locations. Second, objects from the previous time-step were compared against objects in the current time-step by size and position; the nearest match became part of a “tracked” object. When a tracked object overlapped an exit location, the tracking of that object was considered complete.

When operating on real-world data, the system exhibits a few weaknesses. First, the traffic cameras had a tendency to shake in the wind, and also to exhibit temporary fluctuations in their histograms. Individually, either of these was filtered by Vibe. If both occurred at the same time, however, they resulted in a small number of frames consisting solely of numerous fragmented artifacts. Second, a car's windows tended to get eliminated along with the background, which would occasionally cause the segmentation algorithm to split a car into four or five separate pieces. Figure 4 depicts an example of this false segmen-

tation; in this case, the object tracking algorithm was able to recover, but if the false segmentation continued for several frames, or if two cars overlapped during a region of high background noise, then the image tracking information would need to be manually corrected.

3.5 Verification and Validation

Verification was performed on the entire traffic system, checking that the output of a sample run did not result in cars skipping road segments or similar behavior. Complicating this was the fact that SUMO expects cars to drive on the right side of the road. Thus, our road network had to be flipped twice: once before processing traffic and once after all cars had exited the simulation. This process was carefully scrutinized to ensure that all output was valid with respect to driving on the left-hand side of the road.

Estimations of input data, where applicable, were accurate with $p<0.01$. Vehicle inter-arrival time was exponential in nature, while intersection behavior tended to be uniform. Output from traffic camera image processing algorithm was double-checked by hand for all recorded segments.

Validation was performed against the traffic camera data whenever possible. In particular, two copies of the simulation were run: one based directly on the camera data and another based on the estimations mined from the camera data. Both were found to have similar traffic patterns.

4 RESULTS

Our results fall into three categories: results from the traffic study itself, results from implementing our perennial simulation system, and results from the automated image processing algorithm.

4.1 Traffic System Results

The results from our traffic system center around the idea of a "response radius". Any vehicle which can reach the area of incident with a given time limit is considered within the response radius. For example, Figure 5 depicts the average (plus variance) response radius around **Region A** with a response time of 60s. Since our assumption is that incidents will be responded to by patrol cars (instead of a central dispatch), we do not inject cars into the simulation when an incident occurs. Rather, we simply track all existing vehicles throughout the time limit, and flag which ones pass through the area of incident. This creates an approximation of the response radius, and repeated simulations under similar conditions increase the accuracy of this approximation.

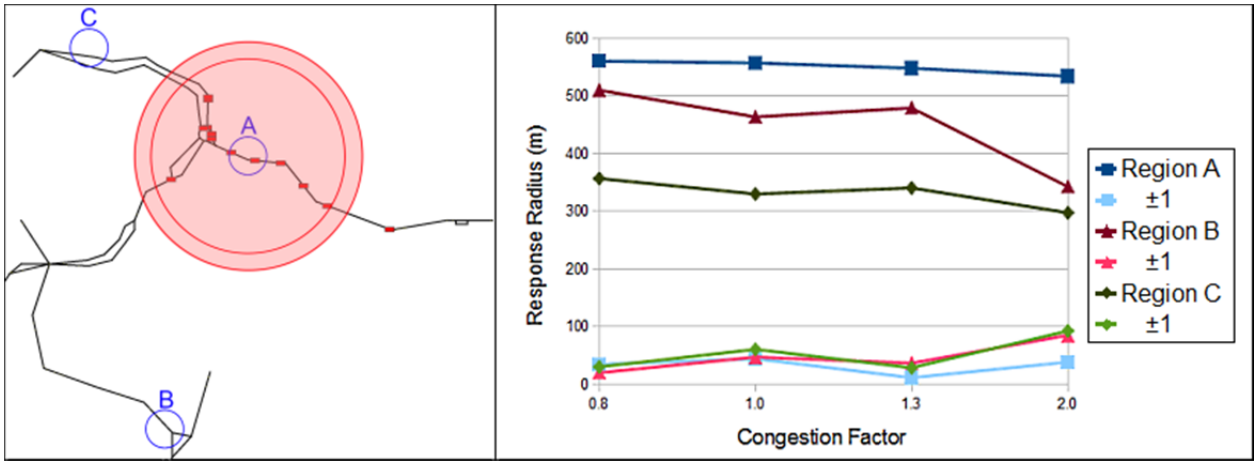

Figure 5: *Left*: Campus road network, with three regions marked A, B, and C. Region A's incident response radius and variance are depicted for congestion factor 1.0. *Right*: Mean response radius (±1 standard deviation) for each region under difference congestion factors.

Three areas were analyzed for congestion. **Region A** was centrally located near the main campus roundabout, while **Region B** and **Region C** were located near the edge of campus. To alter congestion levels, the inverse of the mean inter-arrival time was modified by a constant factor at each entrance, while route choice probabilities at each node remained unchanged. Figure 5 shows the effect of congestion on the response radius. Region A's response radius decreased slightly as congestion increased, but remained relatively unaffected. This is likely due to its central location near three major roadways. Regions B and C each noticed a steady drop in response radius, reflecting the remoteness of these locations. Region B's response radius plummeted under a congestion factor of 2.0, which is particularly troubling as 2.0 only represents a medium-high level of congestion. (Recall that the default congestion factor of 1.0 was extracted from non-peak traffic patterns from 3pm till 4pm.)

4.2 Perennial Implementation and Legacy Model Results

Although the traffic results were valuable in their own right, the primary goal of this study was to explore the simulation framework's ability to incorporate legacy models. This was a marked success; the legacy model was cleanly incorporated into the framework through the use of inheritance, thus giving it access to all real-time and historical sensor information. For *foresight* or *hindsight* studies (including "What If?" analyses), this was sufficient. Decision-support studies (of the symbiotic and non-symbiotic variety) cannot currently be performed on our traffic network. Non-symbiotic decision-support would be possible by extending the simulation to be able to take a set of cars already on campus and insert them into the model. Although SUMO supports absolute positioning of cars, we were concerned that inserting cars directly might invalidate some of our modeling assumptions, and we did not have the additional time available to validate such a system. In addition, placing cars directly would require guessing their routes based on their previous locations. These problems are by no means intractable; rather, they serve to illustrate the boundaries that one invariably ends up pushing against when attempting to shoe-horn a legacy model into a modern framework.

The benefits of enabling legacy interaction with a perennial system are manifold. The legacy model now has access to the complex network of sensors that perennial systems organize, which brings with it the power to automatically generate "What If?" analysis. In addition, any generated input could be saved and used by other simulations in the system. Another benefit was noticed during the design stage: the system was significantly easier both to plan and develop, since the perennial framework handled most of the difficulty of managing sensors and worlds. Finally, if a non-legacy model were connected to the system, that model could benefit from spawning "What If?" analysis using the traffic legacy model we developed. A small amount of work would be required to enable this feature, but that would further enable the possibility of symbiotic decision support.

4.3 Automated Image Processing Results

The automated image processing algorithm performed quite well, with a few caveats. Processing a single frame takes 500ms on average. The bulk of the processing time is spent at or after the median filtering step in the pipeline; that is to say, Vibe performs its background elimination very efficiently. As mentioned previously, this could be sped up by running each step of the pipeline on a different processor and communicating through shared memory. Even without any speedup, however, a 500ms latency is still perfectly acceptable for an arterial traffic network such as a university campus.

In terms of accuracy, the system performs well with three minor caveats. The first is the previously mentioned "scattering" of artifacts that occurs due to wind or noise. This could be fixed by inserting a dynamic histogram balancing step before background elimination. As mentioned, Vibe can handle jitter due to wind, so long as the histogram does not also fluctuate at the same time. The second flaw is the false segmentation of a car's features, combined with a general inability to segment cars during periods of heavy congestion. This shortcoming is best addressed by replacing our region-based segmentation algorithm with a feature-based one such as Beymer et al. (1997). Finally, the cameras themselves tend to be

placed in locations with a poor perspective to monitor traffic. Most were located near ground level, and large vehicles in the nearer lanes could completely obscure those in the farther lanes. As the camera network utilized was put in place for campus security and not specifically designed for monitoring traffic, this roadblock represents a stopping point. Better camera placement is the most obvious solution, but that would eliminate the benefit of using a pre-existing camera network. Another solution would be to augment the camera network with additional sensors designed specifically for measuring traffic and counting vehicles.

5 CONCLUSIONS AND FUTURE WORK

We clearly demonstrated the benefits of a perennial simulation framework by providing an implementation of the framework and a sample study using that implementation. The cost to develop using this framework was far outweighed by the benefit in creating *hindsight* and *foresight* studies, as well as easy management of "What If?" analyses. The model we implemented was a "legacy model" not originally designed to work within a perennial context, yet this limitation was partially overcome using a very simple inheritance abstraction.

The traffic simulation study we performed demonstrated the tradeoffs between high congestion and response to security incidents. The system designed for this task used traffic data extracted from cameras, which leads to the possibility of estimating a response radius during the event of an actual incident. Overall, our approach for automatically extracting traffic data from an existing network of security cameras was a success. Manual processing of camera data was required, but most of this was due to minor issues that the system could, in time, be tuned to avoid automatically. Processing could be performed efficiently, and using a pipelined system opened up the possibility of parallelizing each processing step, most likely leading to a boost in performance.

Future work might include a better pre-processing step for the automated image processing algorithm, or exploring opportunities for parallelism within the image processing pipeline. In addition, we would like to expand our "What If?" analysis to include a multi-objective analysis step, most likely using a Pareto front approximation via the highly modular PISA framework (PISA 2011). Another obvious extension to the traffic study would be to add support for pedestrian dynamics. As the legacy model doesn't currently have this capability, we consider this a good opportunity to augment the system with an agent-based pedestrian model. This would allow our system to demonstrate its full potential for symbiotic-guided decision support, while also leveraging the legacy model for "What If?" analyses.

ACKNOWLEDGMENTS

This research was supported by SMART grant, sub-contract R-252-000-459-592.

REFERENCES

Atkociunas, E., R. Blake, A. Juozapavicius, and M. Kazimianec. 2005. "Image Processing in Road Traffic Analysis." *Nonlinear Analysis: Modelling and Control* 10(4):315–332.

Barnich, O., and M. Droogenbroeck. 2010. "ViBe: A Universal Background Subtraction Algorithm for Video Sequences." *IEEE Transactions on Image Processing*, Volume 20, Issue 6: 1709 – 1724.

Bellifemine, F., G. Caire, and D. Greenwood. 2007. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology.

Beymer, D., P. McLauchlan, B. Coifman, and J. Malik. 1997. "A Real-time Computer Vision System for Measuring Traffic Parameters." In: *Proceedings of Computer Vision and Pattern Recognition* 495–501. IEEE Computer Society.

Boukerche, A., and M. Zhang. 2008. "Towards Peer-to-Peer Based Distributed Simulations on a Grid Infrastructure." In *Proceedings of 41st Annual Simulation Symposium (ANSS 2008),* edited by S. Kawada*,* 212–219. IEEE Computer Society.

Boxill, S., and L. Yu. 2000. "An Evaluation of Traffic Simulation Models for Supporting ITS Development." Center for Transportation Training and Research. Texas Southern University.

Chowdhury, D., L. Santen, and A. Schadschneider. 2000. "Statistical Physics of Vehicular Traffic and Some Related Systems." *Physics Reports* 329:199–329.

Claes, R., and T. Holvoet. 2009. "Multi-Model Traffic Microsimulations." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls. 1113–1123. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Cruz-Neira, C., D. Sandin, and T. DeFanti. 1993. "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE." In *SIGGRAPH'93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, edited by M. C. Whitton, 135–142. ACM.

Darema, F. 2005. "Grid Computing and Beyond: The Context of Dynamic Data-Driven Applications Systems." *Proceedings of the IEEE* 93(3):692–697.

Davis, W., and G. Moeller. 1999. "The High Level Architecture: Is there a Better Way?" In *Proceedings of the 1999 Winter Simulation Conference*. edited by P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans. 1595–1601. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Fujimoto, R., D. Lunceford, E. Page, and A. Uhrmacher. 2002. "Grand Challenges for Modeling and Simulation: Dagstuhl Report." Technical Report 350, Schloss Dagstuhl. Seminar No. 02351.

Gawron, C., S. Kraus, and P. Wagner. 1997. "Determining the Dynamic User Equilibrium in a Traffic Simulation Model." University of Cologne, Center for Parallel Computing, Koln, Germany.

Hetu, S., and G. Tan. 2008. "MMOHILS: A Simpler Approach to Valid Agents in Human Simulation Studies" In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler. 909-913. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Hetu, S., and G. Tan. 2009. "Potential Benefits of Symbiotic Simulation to Pedestrian Evacuation." In *2009 Asia Simulation Conference.* Japan Society for Simulation Technology.

Hetu, S., and G. Tan. 2010 "The Big Picture of Symbiotic Decision Support: Designing a "What-If" Simulation Framework for Crisis Management." In *FISAT: Second International Conference on Advanced Computing and Communications Technologies for High Performance Applications.* Keynote.

Hewage, K., and J. Ruwanpura. "Optimization of Traffic Signal Light Timing Using Simulation." In *Proceedings of the 2004 Winter Simulation Conference*, Edited By R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 367–372. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

IEEE. 1993. (Updated 1997) "IEEE Standard for Information Technology - Protocols for Distributed Interactive Simulations Applications. Entity Information and Interaction." Standard: IEEE 1278.

IEEE. 2000. (Updated 2010) "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules." Standard: IEEE 1516.

Kennedy, C., G. Theodoropoulos, V. Sorge, E. Ferrari, P. Lee, and C. Skelcher. 2007. "AIMSS: An Architecture for Data Driven Simulations in the Social Sciences." In *Proceedings of the Workshop on Dynamic Data-Driven Applications Simulation at ICCS 2007*, edited by Y. Shi, G. D. van Albada, J. Dongarra, P. M. A. Sloot, 1098–1105. Beijing, China. Springer.

Kraus, S. 1998. "Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics". Ph.D. thesis, Department of Mathematics, University of Cologne, Koln, Germany.

Kraus, S., P. Wagner, and C. Gawron. 1997. "Metastable States in a Microscopic Model of Traffic Flow." *Physical Review E* 55(304):55–97.

Law, A. 2007. *Simulation Modeling and Analysis*. 4th Edition. New York: McGraw-Hill Education.

Lin, S., K. Chao, and C. Lo. 2010. "Service-Oriented Dynamic Data Driven Application Systems to Traffic Signal Control." In *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*. 3463–3470. Istanbul, Turkey. Kudret Press & Digital Printing.

McGrath, D., A. Hunt, and M. Bates. 2005. "A Simple Distributed Simulation Architecture for Emergency Response Exercises." In *Proceedings of the 9th IEEE International Symposium on Distributed Simulation and Real-Time Applications*, edited by A. Boukerche, S.J. Turner, D. Roberts, and G.K. Theodoropoulos. 221–228. Washington, DC, USA. IEEE Computer Society.

Perreault, S., and P. Hebert. 2007. "Median Filtering in Constant Time." *IEEE Transactions on Image Processing* 16(9):2389–2394.

Pidd, M. 2002. "Simulation Software and Model Reuse: A Polemic." In *Proceedings of the 2002 Winter Simulation Conference*, edited by E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 772–775. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

PISA. 2011. "ETH – SOP – PISA." Accessed March 20. http://www.tik.ethz.ch/~sop/pisa.

Sargent, R. 2007. "Verification and Validation of Simulation Models." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton. 124–137. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

SUMO. 2011. "Simulation of Urban Mobility." Accessed March 30. http://sumo.sourceforge.net.

Tao, C., and S. Huang. 2009. "An Extensible Multi-agent Based Traffic Simulation System." In *International Conference on Measuring Technology and Mechatronics Automation*, 713–716. Los Alamitos, CA, USA. IEEE Computer Society

Tan, T. N., G. D. Sullivan, and K. D. Baker. 1996. "Efficient Image Gradient-Based Object Localisation and Recognition." In *Computer Vision and Pattern Recognition, Proceedings CVPR '96*.

AUTHOR BIOGRAPHIES

SETH N. HETU is working towards his Ph.D. in Computer Science at the National University of Singapore, with a focus on Crisis Management and Symbiotic Simulation. His email address is seth.hetu@gmail.com.

GARY TAN is an Associate Professor at the Department of Computer Science, School of Computing, National University of Singapore. His current research interests are Parallel and Distributed Simulation, Model Composability, Crisis Management Simulation and Symbiotic Simulation. His email address is gtan@comp.nus.edu.sg.