SIGSPATIAL: G: Computational Steering for Geosimulations

Ashwin Shashidharan*
North Carolina State University
Raleigh, NC 27695
ashdharan@ncsu.edu

ABSTRACT

Geosimulations using computer simulation models provide researchers an effective way to study complex geographic phenomena and their outcomes. These simulations allow for scenario based exploration by capturing spatial and temporal relationships between various geographic processes in a region. However, current approaches to geosimulation limit manipulating model input and exploring alternative scenarios by controlling the simulation at runtime. This paper proposes a computational steering system for geosimulations and presents a prototype, *tFUTURES*, developed for the FUTURES Urban Growth Model (UGM). By allowing users to specify *steering input* and *steering actions* from a web browser, the system solves the problem of lack of user-interactivity experienced by a user at runtime during a geosimulation. We leverage the OGC web services to enable user interaction and visualization of the geosimulation results from a web browser. Further, the *versioning* and *checkpointing* features in the system support: (i) concurrent execution paths of a geosimulation based on varying inputs; and (ii) controlled execution with the ability to *pause*, *advance* or *rollback* a geosimulation. Finally, we support execution of geosimulations in *local* and *distributed* computing environments and demonstrate our computational steering system with the FUTURES UGM simulation.

CCS Concepts

•**General and reference** → *Design;* •**Computing methodologies** → *Real-time simulation; Interactive simulation; Scientific visualization;* •**Computer systems organization** → *Special purpose systems;*

Keywords

Geosimulation, Computational steering, Visualization

*The author acknowledges the support and guidance received from his advisors Dr. Ranga Raju Vatsavai and Dr. Ross K. Meentemeyer.

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA

DOI: http://dx.doi.org/10.1145/2996913.3002146

1. INTRODUCTION

Modeling and simulation has revolutionized many scientific and engineering fields in the past two decades. In recent years, geosimulation has emerged at the intersection of Geographic Information Science, Complex Systems Theory and Computer Science. Geosimulations [2], where real-world processes are modeled and studied over time, have been successfully applied in urban studies, epidemiology, land use and land cover changes, and climate change studies. Using geosimulations, "what if" scenarios can be studied to understand potential impacts of geographic events. However, such scenario analyses rely on static inputs prepared beforehand by GI scientists.

Computational steering [7, 14] is a mechanism that supports interactivity in simulations while they are in progress. Specifically, it allows for manipulation of the internal state of a simulation and its inputs during execution. For instance, in a UGM geosimulation, computational steering mechanisms could be used to specify new zoning regulations and transportation networks to an in-progress simulation. Further, the ability to visualize the impact on development patterns in real-time, could be used to tweak the inputs for subsequent time-steps or in retrospect. Such interactivity helps improve the quality of simulations, allows on-the-fly "what if" scenarios, and improves computational efficiency. However, little work has been carried out to integrate computational steering and geosimulations with visualization support. In our system, tFUTURES, we attempt to bridge this gap by supporting computational steering for geosimulations from a javascript enabled web browser. We enable *versioning* and *checkpointing* in geosimulations and support *steering actions* that can *pause*, *advance* or *rollback* a geosimulation from any such browser.

2. tFUTURES SYSTEM

tFUTURES is a computational steering system that we develop for user-driven simulation of urbanization under varying urban growth policies. The system enables users to specify experimental development constraint policies to the FUTURES simulation at runtime and analyze its outcomes in real-time. The system aids policy makers with formulation of new urban growth policies.

The tFUTURES computational steering system comprises of three components as shown in Fig. 1, namely (i) Monitoring Server; (ii) Steering Client; and (iii) Visualization Service. The system supports two modes of operation based on its deployment, a *local* and a *distributed* mode. The *local* mode of operation works on a single machine with the Mon-

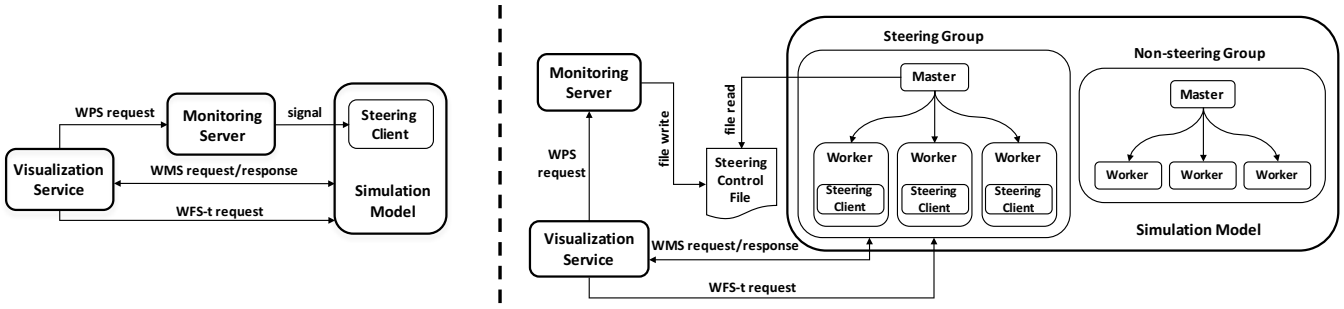

Figure 1: tFUTURES Local and Distributed Execution Modes

itoring Server, Visualization Service and Steering Client, all executing in the context of a single operating system. While, the *distributed* mode of operation extends the local implementation to an environment, where the Monitoring Server, Visualization Service, and Steering client can be different processes on different machines. The two different modes are illustrated in Figure 1. In Section 2.1, we begin with a brief overview of the geosimulation in both these modes followed by a description of the features, capabilities and mechanisms of each system component.

2.1 Operational Modes

2.1.1 Local Execution

The local execution mode is a simple scenario of a single process executing a geosimulation for a single county on a single machine. In this mode of operation, the geosimulation, steering and visualization all run on a user's local machine. The input dataset and user-generated steering input correspond to a single county, and is stored locally on the user's machine for execution. In tFUTURES, the geosimulation is an urban growth model for a single county which executes in discrete time-steps. At the end of every time-step, the steering system pauses the geosimulation to receive: (i) a user-triggered steering action and, (ii) an optional steering input from the user. Thus, the geosimulation is steered by a user in discrete time-steps till completion. At the end of the simulation, a user can setup the dataset for another county for execution within the same environment.

2.1.2 Distributed Execution

In the distributed mode of execution, the geosimulation relies on a HPC cluster with a MPI framework for distributed task processing. The geosimulation executes on a HPC cluster, while steering and visualization are controlled from a user's local machine. The steering actions and steering input are generated from a web browser on the user's local machine. The distributed execution environment assumes the availability of two pre-partitioned sets of counties of the study region: (i) steering counties, consisting of counties in the study region which will be steered by the user and, (ii) an optional set of non-steering counties which will execute as a typical data parallel HPC job, independent of user interaction. The geosimulation is then set up as two distinct groups with a master-slave mode of execution in both groups.

Steering group: The master distributes the execution of the steering set of counties among its workers in an HPC cluster. It assigns a single county from the steering set to a worker and controls the geosimulation at the worker in discrete time-steps. At the beginning of every time-step, the master executes the following pipeline: (i) wait and read a user steering action from a steering action file; (ii) prepare the steering input by clipping the user steering input along county boundaries and rasterizing the data per-county for execution at the workers; and (iii) enforce data synchronization among all workers by receiving and propagating spatial updates from workers across county boundary lines for neighboring regions. At the beginning of every time-step, along with the spatial updates, the workers receive a steering action which corresponds to the user-triggered action in the visualization client.

Non-steering group: The main responsibility of the master is to map a task on to a worker till all non-steering group counties have been processed. Once a county has been loaded at the worker, the worker executes the simulation on the county for all time-steps. Unlike the steering group, the master doesn't enforce data synchronization in every time-step. Thus, spatial effects across county boundaries aren't preserved in the non-steering set of counties.

Figure 2 illustrates the boundary communication pattern in these simulation groups.

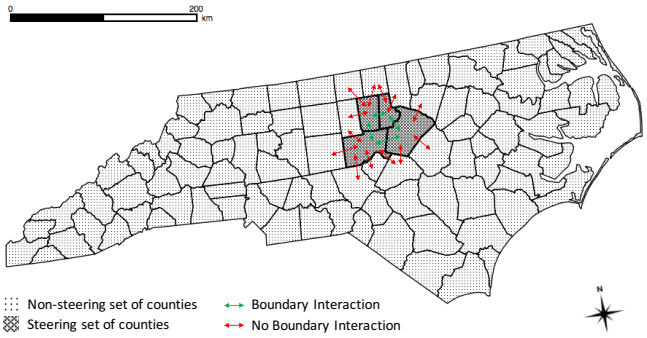

Figure 2: Illustration of the state of North Carolina partitioned as sets of steering and non-steering counties

2.2 Features

2.2.1 Versioning

We support multiple concurrent execution paths of a single program based on steering input which has varied in a given time-step. Let us say, in a particular time-step, a user decides to change the steering input for the geosimu-

lation and execute a steering action. In the next time-step, the geosimulation would read the new steering input and continue execution. However, in response to this change in input, our system also spawns another background worker process with the old steering input and runs the geosimulation till completion. Thus, we execute two versions of the simulation allowing the user to explore the outcomes due to changes in inputs at the end of the simulation. To retrieve a specific output version pertaining to a change in steering input at a time-step, we implement a unique output filename naming scheme which lets the user identify the time-step and the the input used to generate the output. In the newly spawned version of execution, our output file versioning scheme appends the time-step to the output filename to indicate the time-step at which steering input differed from the previous time-step. Versioning is essential to recall specific versions of output during a post-processing step. In both modes of execution, versioning is carried out by the process running the geosimulation in a county. Along with the checkpointing data generated at each time-step (Section 2.2.2), an analyst can obtain the complete input set used to generate the output.

2.2.2 Checkpointing

Checkpointing is a feature available in both modes of operation. In our system, checkpoints help in rolling back the simulation to a previous step for which computation has already been completed. The checkpointing system serializes the data necessary to restart each iteration. It eliminates the need to re-start the simulation from the beginning to reach a simulation time-step which was previously computed. To enable this feature, a developer must identify all output files and input files that change in each time-step during adoption of a geosimulation to our framework. This data saved at each time-step our computational steering system is then used to obtain the entire dataset necessary to rollback or replay the simulation.

2.3 System Components

In this section, we describe the system components in our Computational Steering System, tFUTURES, namely: (i) Monitoring Server; (ii) Steering Client; and (iii) Visualization Service. We provide an overview of each component, their capabilities and mechanism.

2.3.1 Monitoring Server

The Monitoring Server acts as an interface between the Visualization Service and the Steering Client in the system. It receives WPS requests generated by the Visualization Service and forwards them to the Steering Client in the geosimulation.

Capabilities.

The Monitoring Server features a (i) OGC Web Processing Service (WPS) interface to receive XML or URL-encoded requests from a web client; and (ii) an IPC mechanism to trigger the execution of a process in *local* or *distributed* mode of operation. The WPS interface enables web-enabled steering actions through an online resource URL while, the underlying IPC mechanism allows the Steering Client to consume and respond to the steering action. To support the *local* and *distributed* modes of operation, we implement two variations of the IPC mechanisms in our framework.

Mechanism.

User selection of a *steering control* in the web interface, generates a WPS request to the Monitoring Server. The WPS interface in the Monitoring Server receives this request and identifies the corresponding steering action for the steering control. The Monitoring Server then interrupts the control flow of the geosimulation and delivers the steering action for execution by the geosimulation.

Local: The IPC mechanism is setup as a signal handler hub capable of issuing signals to the geosimulation. During server initialization, the Monitoring Server defines unique signal mappings for steering actions to signal handlers in the Steering Client. Then, at runtime, on receiving a WPS request from the Visualization Service, the Monitoring server triggers a signal corresponding to the steering action for execution by the geosimulation.

Distributed: A file-based IPC mechanism is used to relay a steering action to the geosimulation. During server initialization, the Monitoring Server defines a shared steering control file to communicate with the master process of the steering group. The Monitoring Server also defines a mapping of unique values corresponding to the steering action. At runtime, on receiving a steering action, the Monitoring server writes the corresponding unique value for the steering action to the file. The steering group master reads the file and delivers the steering action to the Steering Client embedded in the workers. The Monitoring Server then blocks till the steering action has been executed on all steering group workers before accepting new steering actions from the user. Thus, the read/write operations to the steering control file are synchronized based on the return status of execution.

2.3.2 Steering Client

The Steering Client augments the geosimulation code to handle steering actions and user-defined steering input. It implements logic routines to service the steering actions forwarded by the Monitoring Server. It also implements the logic to load the steering input for the simulation received directly from the Visualization Service. Specifically, the Steering Client embedded in a geosimulation: (i) modifies geosimulation state; (ii) alters the control flow of the geosimulation at runtime; and (iii) periodically checkpoints the simulation state to enable rollback of the geosimulation.

Capabilities.

Fig. 3a illustrates a few steering actions that we believe must be supported in most geosimulations. The *steering controls* are defined as follows: (i) **skipPrev:** rollback the simulation by a single time-step; (ii) **restart:** reset the steering input and restart the existing simulation run; (iii) **play:** run the simulation from the current time-step till completion; (iv) **skipNext:** advance the simulation by a single time-step; and (v) **pause:** pause the simulation at the end of the current time-step.

Mechanism.

Local: The Steering Client implements signal handler routines to handle steering actions delivered to it as signals from the Monitoring Server. When a signal is received by the Steering Client, the handler routine corresponding to the steering action is executed. The signal handling routine modifies the control flow of the geosimulation during execu-

tion and loads the simulation data necessary to execute the next time-step.

Distributed: In a distributed environment, a Steering Client is embedded in all the steering group workers. A steering action generated by a user is delivered to the steering group master process using a shared steering control file. The steering group master process monitors the steering control file and delivers the steering action to its workers at the beginning of every time-step. The Steering Client in each worker then receives the steering action, following which the worker reads the new steering input, if any, and executes the steering action as the next time-step in its simulation.

2.3.3 *Visualization Service*

The Visualization Service enables a user to visualize and interact with the geosimulation from a web browser on a user's local machine. It provides the end-user with: (i) web controls for interacting with the simulation; and (ii) real–time online visualization of the simulation results.

Capabilities.

The Visualization Service combines three OGC Web Services to provide online visualization and interaction from a web browser: (i) Web Map Service (WMS); (ii) Web Processing Service (WPS); and (iii) Transactional Web Feature Service (WFS-t). Specifically, WMS enables rendering output raster maps from the geosimulation, WPS supports executing steering actions in a geosimulation and, WFS-t allows drawing steering input as vector data in a browser. The Visualization Service also provides two sets of control widgets as shown in in Fig. 3: (i) "Steering Controls" for a user to select a steering action; (ii) "Map Controls" for a user to edit the geosimulation's steering input. These control widgets in the web interface of a geosimulation enable user interaction from a web browser.

(a) Steering Controls (b) Map Controls

Figure 3: Map and Steering Controls Menu

Mechanism.

To experiment with various development scenarios, a user defines new steering input using the Map Controls shown in Fig. 3b. The Map Controls trigger WFS-t requests directly modifying the geosimulation input while, the Steering Controls allows a user to execute the UGM in time-steps based on the steering input. At the end of every steering action, the resulting output map is refreshed in the browser.

Local: In the local mode of execution, the Visualization Service runs on the same node as the Monitoring Server and the Steering Client. The output maps generated by the geosimulation are available for rendering from the same node.

Distributed: In the distributed mode of execution, the Visualization Service must be configured for displaying the resulting maps by retrieving the data from the HPC network shares over the network. Thus, in both modes, the Visualization Service enables an endpoint that accepts user input and displays simulation outputs.

3. APPLICATION

FUTURES [6] is a spatio-temporal urban growth model used to simulate land changes in environments. The simulation executes in discrete time-steps with urban growth in each time-step influencing further growth in subsequent time-steps. As a simulation tool, it is well suited to policy makers and city planners who wish to study development of new urban regions and existing cities under varying development policies. In developing a computational steering system, one of our goals was to allow city planners to use tFUTURES to determine optimal zoning information, like size, shape, and it's influence on urban development.

We now describe the changes to the FUTURES simulation code to adopt it within our steering framework. Firstly, we instrument FUTURES to accept steering actions and to pause the simulation till an explicit steering action has been provided by a user. Secondly, we implement the *advance* and *rollback* steering actions to execute a single time-step i.e., the previous or next time-step in the FUTURES simulation, respectively. Thirdly, we modify the FUTURES code to accept new steering input, at every time-step of the simulation, in our case user-defined zoning policies. Finally, we develop a web front-end to enable interaction and visualization of the urbanization maps generated at every time-step from a user's web browser. Thus, we enable the FUTURES urban simulation to be steered with interactive adjustment of constraints and visualization at every time-step.

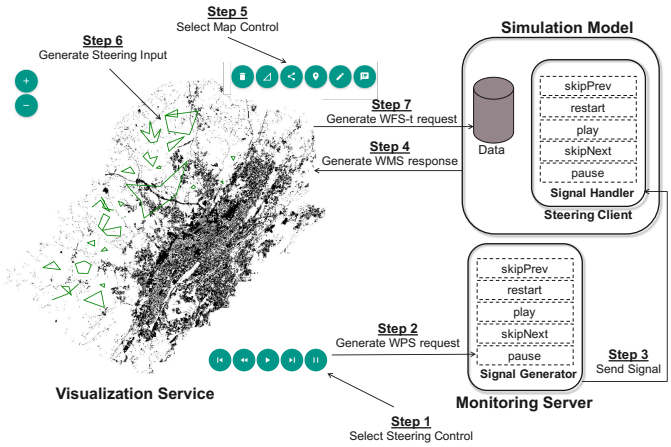

Figure 4: Steps involved in steering a geosimulation

Local: In the local mode of operation, we configure a standalone version of FUTURES for simulating growth in a single county on a single machine. As part of the setup, a locally deployed GeoServer is used to host the urbanization maps generated by the simulation at every time-step. We create a simple web application that includes the steering controls shown in Fig. 3 and leverage the WMS service with the GeoServer to display the urbanization map. A PostGIS vector layer based on the WFS-t service with GeoServer provides an editable zoning constraints layer within the web application. A Jetty based web server on the local machine is used to make the steering web application accessible locally from a web-browser. Finally, the steering components are implemented as separate processes on the local machine and the communication between the application and the simulation are done via UNIX signals as described in Sec 2.1.1.

Distributed: In our distributed mode of operation, we setup the parallel version of FUTURES [13] in Approach 1 and Approach 2 for the non-steering group and steering group, respectively. We use simulation data for the state of North Carolina, partitioning the dataset as steering and non-steering counties as shown in Fig. 2. Similar to the local mode, we use a GeoServer for hosting the application maps and a Jetty webserver for hosting the web application. However, the servers are setup on a remote machine accessible using a web URL. The other steering components are implemented in the NCSU henry2 HPC center. To support WFS-t transactions on the constraint layer, we maintain a PostGIS database on the GeoServer site as our datastore. The PostGIS database and WPS services read/write to a network mountable share in the HPC. A city planner accesses the the simulation from a remote web client using the web URL and interacts with the steering set of counties. All steering actions for the steering counties are processed by MPI processes in a distributed environment at the HPC center. The non-steering set of counties also execute in the same HPC environment, but without user intervention.

4. FUTURE WORK

Support for a multi-user environment: We would like to implement a multi-user environment to support simultaneous steering of a geosimulation from multiple web clients. This would enable collaborative decision making by allowing multiple stakeholders to engage simultaneously, for e.g., in developing new zoning policies for urban development.

Steering capabilities: To further extend the capabilities in our computational steering framework, we would like to support run-time modification of model parameters during a simulation. This would enable policy makers to alter the mathematical model of a geosimulation and analyze the outcomes of different simulation scenarios on-the-fly.

Ease of adoption: In our case study, we take a manual code instrumentation approach to adopt a geosimulation for processing new steering input and steering actions. To allow end users with limited programming background to use our steering framework, we would like to automate the code generation process for easier adoption of a geosimulation within our framework. Finally, to support a wider range of geosimulations, we would like to port tFUTURES to new shared memory architectures like the Apache Spark framework.

5. CONCLUSIONS

In this paper, we implement a computational steering system that allows a human-in-the-loop for interactive simulation. We show that computational steering capabilities can be easily extended to geosimulations with a small set of interacting components and minimal changes to legacy model code. At a bare minimum, this set of interacting components must include: (i) Visualization Service to support *steering input* and *steering actions* from a web browser; (ii) Monitoring Server to intercept and relay steering actions to the simulation; and (iii) Steering Client embedded in the legacy simulation code to execute the steering actions. We demonstrate a typical use of such a system in a *local* and *distributed* mode of operation. In conclusion, by intertwining user interactions with geosimulations in a computational steering system, we empower practitioners to dynamically vary model inputs and produce desired simulation results at runtime.

6. REFERENCES

[1] I. Ba, C. Malon, and B. Smith. Design of the ALICE Memory Snooper, 1999.

[2] I. Benenson and P. M. Torrens. *Geosimulation: Automata-based modeling of urban phenomena*. John Wiley and Sons, 2004.

[3] W. Gu, G. Eisenhauer, E. Kraemer, K. Schwan, J. Stasko, J. Vetter, and N. Mallavarupu. Falcon: On-line monitoring and steering of large-scale parallel programs. In *Frontiers of Massively Parallel Computation, 1995. Proceedings. Frontiers' 95., Fifth Symposium on the*, pages 422–429. IEEE, 1995.

[4] D. J. Jablonowski, J. D. Bruner, B. Bliss, and R. B. Haber. VASE: The visualization and application steering environment. In *Supercomputing'93. Proceedings*, pages 560–569. IEEE, 1993.

[5] J. A. Kohl, T. Wilde, and D. E. Bernholdt. CUMULVS: Interacting with high-performance scientific simulations, for visualization, steering and fault tolerance. *International Journal of High Performance Computing Applications*, 20(2):255–285, 2006.

[6] R. K. Meentemeyer, W. Tang, M. A. Dorning, J. B. Vogler, N. J. Cunniffe, and D. A. Shoemaker. FUTURES: multilevel simulations of emerging urban-rural landscape structure using a stochastic patch-growing algorithm. *Annals of the Association of American Geographers*, 103(4):785–807, 2013.

[7] J. D. Mulder, J. J. van Wijk, and R. van Liere. A survey of computational steering environments. *Future generation computer systems*, 15(1):119–129, 1999.

[8] S. G. Parker, C. R. Johnson, and D. Beazley. Computational Steering Software Systems and Strategies. *IEEE Comput. Sci. Eng.*, 4(4):50–59, Oct. 1997.

[9] S. G. Parker, M. Miller, C. D. Hansen, and C. R. Johnson. An integrated problem solving environment: The SCIRun computational steering system. In *System Sciences, 1998., Proceedings of the Thirty-First Hawaii International Conference on*, volume 7, pages 147–156. IEEE, 1998.

[10] S. Pickles, R. Haines, R. Pinning, and A. Porter. A practical toolkit for computational steering. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 363(1833):1843–1853, 2005.

[11] B. Reitinger. *On-line program and data visualization of parallel systems in a monitoring and steering environment*. PhD thesis, Citeseer, 2001.

[12] E. Shaffer, D. A. Reed, S. Whitmore, and B. Schaeffer. Virtue: Performance visualization of parallel and distributed applications. *Computer*, 32(12):44–51, 1999.

[13] A. Shashidharan, D. B. van Berkel, R. R. Vatsavai, and R. K. Meentemeyer. *pFUTURES: A Parallel Framework for Cellular Automaton Based Urban Growth Models*, pages 163–177. Springer International Publishing, Cham, 2016.

[14] R. Van Liere, J. D. Mulder, and J. J. Van Wijk. Computational steering. In *International Conference on High-Performance Computing and Networking*, pages 696–702. Springer, 1996.