

GPU Accelerated Dispersion Simulation For Urban Security

Feng Qiu Ye Zhao Zhe Fan Xiaoming Wei Haik Lorenz Jianning Wang
Suzanne Yoakum-Stover Arie Kaufman Klaus Mueller

Center for Visual Computing and Department of Computer Science
Stony Brook University

<http://www.cs.sunysb.edu/~vislab/projects/urbansecurity>

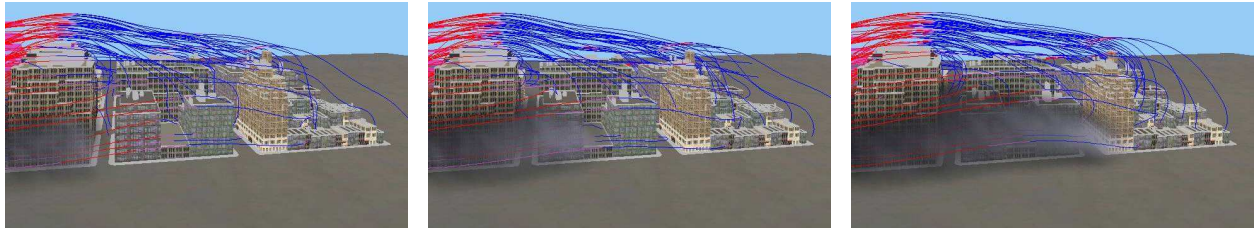


Figure 1: Smoke and streamlines representing dispersion simulation results in the West Village area of New York City.

We developed a system for simulating and visualizing the propagation of dispersive contaminants for urban security in open environments characterized by sky-scrapers and deep urban canyons. Our approach is based on the Multiple Relaxation Time Lattice Boltzmann Model (MRTLBM), which can efficiently handle complex boundary conditions such as buildings. The readings from various sensors distributed in the environment are used to adapt the simulation accordingly. We accelerate the computation on the GPU as well as on the GPU efficiently render many buildings using small textures. We render the contaminant smoke with self-shadowing composited with the textured buildings.

The LBM models Boltzmann particle dynamics on a lattice. Particles stream in discrete time steps to neighboring sites. Between streaming steps, they undergo collision. The MRTLBM uses a general collision model in which many of the hydrodynamic moments relax toward their equilibria independently. The additional freedom afforded by the decoupled relaxation parameters, gives the model better stability and facilitates the coupling of additional physics compared with single relaxation time LBM (SRTLBM).

In urban environments, some sensors are installed for recording the wind velocity, temperature, etc. at a real-time speed. Our system provide two methods of adapting the simulation's numerical models to accommodate live-sensor input. In the first method, the effect of the sensor data is incorporated as a body force. In the second method, we try to trace those boundary nodes that will affect the sensor points and modify those boundary nodes directly to match the sensor data.

We exploit the GPU to accelerate the MRTLBM simulation because of the locality, and hence parallelizability of LBM operations. To layout the LBM data, we divide the lattice sites into several volumes. Each volume contains data associated with a given state variable and has the same resolution as the LBM lattice. To use the GPU vector operations and save storage space, we pack four volumes into one stack of 2D textures (note that a fragment or a texel has 4 color components). The LBM operations (e.g., streaming, collision, and boundary conditions) are translated into fragment programs that can be executed in one render pass. For each fragment in a given pass, the fragment program fetches any required current state information from the appropriate textures, evaluates the result, and renders it to a pixel buffer. When the pass is complete, the results are copied back into another texture.

To visualize the simulation results, we render the buildings of the

city model with textures and plume or smoke with self-shadows in real-time. Because most of the texture memory is used for simulation, a small number of high resolution textures are used for buildings. To reduce the repetitiveness, we take advantage of the programmable fragment shading capability of GPU by implementing a texture-aging-and-variation shader. This shader changes the overall appearance of a facade texture by adding dirt and cracks without affecting major features like windows. The smoke volume is splatted in slices perpendicular to the half angle $h = v + l$. The reconstruction results are used to not only render the current slice but also attenuate the lighting for the next slice.

Fig. 1 shows the simulation results in the West Village area of New York City, which has 10 blocks and more than 100 buildings. The simulation runs on single GeforceFX 5800 GPU with resolution of 4.5m ($90 \times 30 \times 60$) at interactive speed. Compared with CPU implementation, the simulation is 8.02 times faster. Fig 2 shows the simulation results in the Time Square area of New York City, which has 91 blocks and 851 buildings. The simulation runs on a GPU cluster of 30 nodes with resolution of 3.8m ($480 \times 80 \times 400$) and each simulation step consumes 0.31 seconds, which is 4.6 times faster than a CPU cluster implementation.

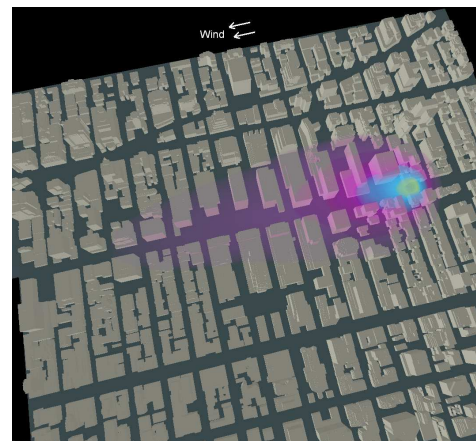


Figure 2: Plume in the Time Square area of New York City.