Eurographics Workshop on Urban Data Modelling and Visualisation (2016)
F. Biljecki and V. Tourre (Editors)

Interactive Low-Cost Wind Simulation For Cities

Eduard Rando, Imanol Muñoz & Gustavo Patow

ViRVIG - Universitat de Girona
Girona, Spain

Abstract
Wind is an ubiquitous phenomenon on earth, and its behavior is well studied in many fields. However, its study inside a urban landscape remains an elusive target for large areas given the high complexity of the interactions between wind and buildings. In this paper we propose a lightweight 2D wind simulation in cities that is efficient enough to run at interactive frame-rates, but also accurate enough to provide some prediction capabilities. The proposed algorithm is based on the Lattice-Boltzmann Method (LBM), which consists of a regular lattice that represents the fluid in discrete locations, and a set of equations to simulate its flow. We perform all the computations of the LBM in CUDA on graphics processors for accelerating the calculations.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Computer Graphics]: Simulation and Modeling—Animation

1. Introduction

Wind is well known and long-studied problem. Wind has been well studied in many fields, including meteorology, climate, energy, human comfort studies and even urban planning. For an accurate simulation, however, it is needed a large Computational Fluid Dynamics (CFD) implementation that uses a numerical version of Navier-Stokes equations, and that is limited to a small area around a point of interest [BJvH12, Blo15].

However, interactive simulations in a real urban landscape remains an elusive target for large areas given the high complexity of the interactions between wind and buildings. Thus, the challenge of achieving an interactive simulation, but accurate enough to be used for different simulation purposes, remains an elusive target [Blo14].

In this paper we propose a lightweight 2D wind simulation in cities that is efficient enough to run at *interactive* frame-rates (around 10 fps), but also accurate enough to provide some prediction capabilities. The method is based on the well established Lattice Boltzmann Method (LBM), which has been previously used for a large variety of fluid simulations. We also present a GPU-friendly implementation that makes the simulation amenable for interactive applications, including wind direction and strength variations, resulting in an overall interactive, low-cost, reliable simulation method for wind in an urban environment.

2. Previous Work

Urban physics is the science and engineering of physical processes in urban areas. Its main field of study is the heat and mass transfer in urban environments, both outdoor and indoor [Blo15]. In spite of being quite a recent subject of research, urban physics is establishing itself as a key factor in understanding many urban dynamics and being a field of research which could provide effective answers to many challenges in modern society. Its main tools are field measurements, full-scale and reduced-scale laboratory measurements and numerical simulation methods including Computational Fluid Dynamics (CFD) [BJvH12].

Fluid simulations tend to be based on one of two main approaches, namely empiric models and accurate simulations. The first ones are usually based on a simplified model that treats the fluid as particles under Newtonian laws. These particles flow through a grid placed over the object's surface. This way, particles are moved by different forces (e.g., pressure differences), but collisions are avoided by accumulating fluid in the grid cells and using the resulting fluid quantity for further calculations.

Full-blown physical simulations include molecular mechanics, direct Navier-Stokes simulations and the Lattice-Boltzmann Method. Molecular simulations constitute a family of computer simulation techniques that involve the exact solution of statistical mechanics problems [Sad99]. In these approaches, which include molecular dynamics (MD) and Monte Carlo (MC) techniques, the properties of the studied systems are determined solely by the evaluation of intermolecular forces and energies. Molecular dynamics utilizes Newton's equations of motion, while the Monte Carlo techniques use a statistical mechanics approach, using the concept of configuration space [BS01]. Unfortunately, these techniques can only be used to simulate a fluid up to a few millimeters scale.

The equations of motion of an incompressible, Newtonian fluid (usually called Navier-Stokes equations [Bri08]) have been tradi-

DOI: 10.2312/udmv.20161421

tionally used for accurate simulations, but their complexity makes them infeasible for real-time simulations of large volumes or surfaces [WMT05]. For instance, for the simulation of an airplane wing, using a $21m \times 10m \times 12m$ domain discretized into 3.4M cells, computed with optimized GPU-based code (SpeedIT Flow (SITF)) running on a modern machine (CPU: 2x Intel(R) Xeon(R). CPU E5649 2.53GHz, GPU: NVIDIA Quadro K6000 12GB RAM, RAM: 96GB. OS: Ubuntu 12.04.4 LTS 64bit), reported simulation times of around 2 hours, while standard, CPU-bassed OpenFoam simulation takes more than 7 hours to compute the same problem [VRa15]. As we can see, these times even using hardware-accelerated computations, are far from interactive.

In this paper we use the lattice Boltzmann method, which is derived from the Boltzmann equation [Wag08, LFWK05, MZ88] and that provides a low-cost, reliable substitute of the full Navier-Stokes equations. This method will be described in detail in Section 4.

3. Overview

Our objective in this paper is to perform an approximate, but reliable simulation of the wind flow in an urban environment at interactive frame-rates, allowing the dynamic change of wind parameters. To achieve these objectives we will use a technique based on the Lattice-Boltzmann Method (LBM), which consists of a regular lattice that represents the fluid in discrete locations, and simulates how the fluid flows on it. At high spatial resolutions, it can be demonstrated that these methods converge into the continuous equations of Navier-Stokes, which are the most important to describe the macroscopic behavior of a fluid. We approximate the wind in an urban landscape as a 2D grid, which allows us to represent the fluid as a layer with respect to the ground. To accelerate all calculations we implement our system on current graphic hardware.

4. Lattice Boltzmann

The Lattice Boltzmann method (LBM) derives from the Boltzmann equation [Wag08, LFWK05], which describes the behavior of gas on a microscopic level using kinetic theory. First proposed by McNamara and Zanetti [MZ88], the Lattice Boltzmann method replaced the boolean particle number in a lattice direction with the density distribution function to reduce statistical noise. The practical viability of simulating in three dimensions came with the work of Higuera and Jimenez [HJ89], while Quian et al. [QdL92] suggest enhanced collisions for the LBM that allow simulations with low viscous fluids. They eliminate collisions from the LBM so that only their consequences matters.

A final improvement to the collision operator is known as the Bhatnagar-Gross-Krook approximation [QdL92, CCM92]. This version of the LBM is known as the lattice-BGK model (LBGK) and provides a single time relaxation. The LBGK is the most popular LBM used today due to its simplicity and efficiency.

The LBM is inherently compressible. Consequently, it models the compressible Navier-Stokes equation. Fluid compressibility is a main feature of the LBM and is what gives it a performance advantage over other methods.

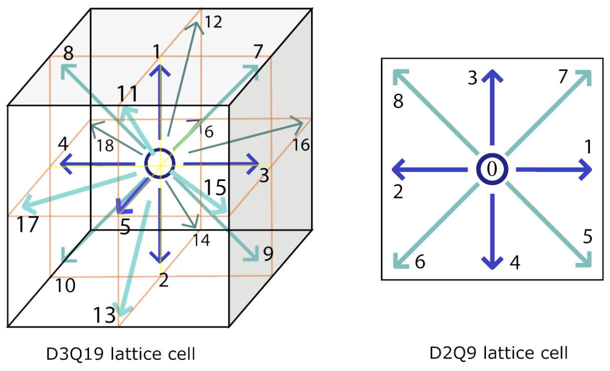

Figure 1: *Left: the D3Q19 model. Right: D2Q9 Model*

4.1. LBM details

The lattice Boltzmann methods work on a lattice. Several variations of the LBM exist, and are named $DXQY$, where X is the dimension and Y is the number of lattice velocities or vectors, as shown in Figure 1. All cells are updated at each time step by simple rules, taking into account the state of the surrounding cells. In our project we use the $D2Q9$ model.

A lattice vector is referred to as e_i where i is the lattice vector number. In our model the lattice vectors are $e_0..e_8$. At each lattice site $\vec{x}$ and time t, fluid particles moving at arbitrary velocities are modeled by particle distribution functions $f_i(\vec{x},t)$. Each $f_i(\vec{x},t)$ is the expected number of particles moving along a lattice direction vector e_i. Note that the particles are allowed to move only along the lattice velocity vectors. The magnitude of the velocity vectors e_1 through e_4 is 1 lattice unit per time step. The magnitude of velocity vectors e_5 through e_8 is $\sqrt{2}$ lattice units per time step. Also, the vector with index 0 has zero length, and indicates particles that are not moving anywhere in the next time step, although some of them may be accelerated due to collisions with other particles. Thus, the number of resting particles may change at each time step.

From the particle distribution functions two important physical values can be calculated. By summing up all 9 distribution functions, the density for the volume of this cell can be calculated, assuming that all particles have the same mass of 1. As the distribution functions contain the number of particles moving in a certain direction for each cell, the sum of all particles in a single cell is its density:

$$\rho = \sum_{i=0}^{8} f_i \tag{1}$$

Another important attribute for each cell is the velocity and overall direction in which the particles of one cell move:

$$\vec{v} = \frac{1}{\rho_0} \sum_{i=0}^{8} f_i \vec{e}_i \tag{2}$$

A simulation consists of two steps, namely *streaming* and *collision*, that are repeated for each time step. The streaming step is

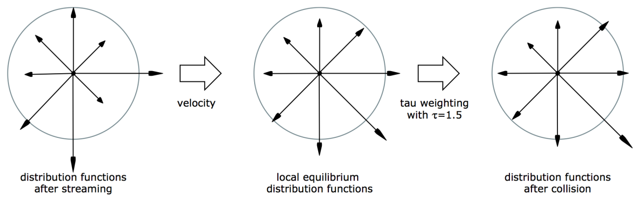
velocity
tau weighting with τ=1.5
distribution functions after streaming
local equilibrium distribution functions
distribution functions after collision

Figure 2: *Collision step*

the simplest one, and consists of moving particles from one cell to another.

In the collision step, particles arrive at a cell and collide with other particles, as shown in Figure 2. The collision step does not change the density or velocity of a cell, but only changes the cell's particle distribution.

To model this behavior, the equilibrium distribution function, f_i^{eq} and new distribution functions must be calculated. For incompressible flows the following equilibrium distribution function is a good choice [HL97]:

$$f_i^{eq}(\rho,\vec{v}) = w_i\left[\rho+\rho_0\left(\frac{3}{c^2}(\vec{e}_i\cdot\vec{v})+\frac{9}{2c^4}(\vec{e}_i\cdot v)^2-\frac{3}{2c^2}(\vec{v}\cdot\vec{v})\right)\right] \quad (3)$$

The weights w_i depend on the length of the velocity vector.

The streaming and collision models are usually combined into one formula, which is known as the Lattice-Boltzmann equation:

$$f_i(\vec{x}+\vec{e}_i,t+1)-f_i(\vec{x},t)=\omega(f_i^{eq}(\rho,\vec{v})-f_i(\vec{x},t)) \quad (4)$$

where $\omega = 2/(6\nu+1)$. The left hand side of this equation accounts for the stream step. The right hand side is a combination of the current distribution function and the local equilibrium.

The simulation of the LBM then proceeds by repeating the following steps: First, the density distribution is computed according to Equation 1. Then, velocities are computed using Equation 2, following by the equilibrium distribution given by Equation 3. Finally, the distribution is updated using Equation 4.

4.2. Implementation of the LBM

As discussed in Section 4, the lattice-Boltzmann method requires the program to keep track of 9 particle distribution functions f_i, as well as density ρ, and velocity $\vec{v}$ for each cell. We therefore need to store 12 floats per cell and per time step. This amount of data can be accommodated by 3 RGBA floating point textures, as shown in Figure 3.

An LBM implementation consists of four operations: streaming, boundary detection, velocity and density computation, and collision handling (Figure 4). All these operations can be easily implemented in CUDA:

- **Streaming** The streaming step for non-boundary cells is accomplished by swapping channels in a texture. This operation is straightforward to carry out in a simple kernel: we just need to read the data stored in the correct neighbor. For f_0, we simply use: `fNew0 = tex2D(texture1,coords.r)`.

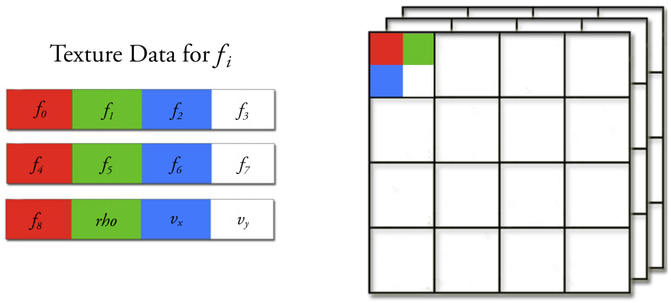

Figure 3: *Particle Distribution Function, velocity and density storage in 3 textures*

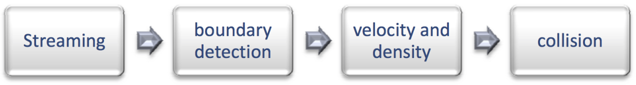

Figure 4: *Operations for computing the LBM*

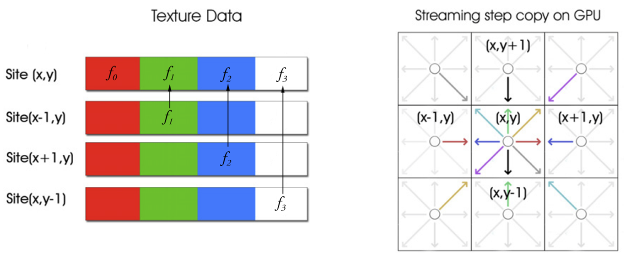

Figure 5: *Streaming step on the GPU*

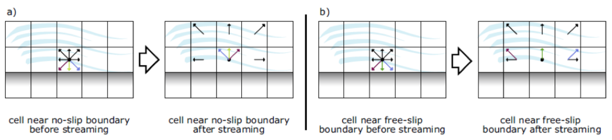
a)
b)
cell near no-slip boundary before streaming
cell near no-slip boundary after streaming
cell near free-slip boundary before streaming
cell near free-slip boundary after streaming

Figure 6: *Boundary conditions implemented: No-slip (left) and free-slip (right).*

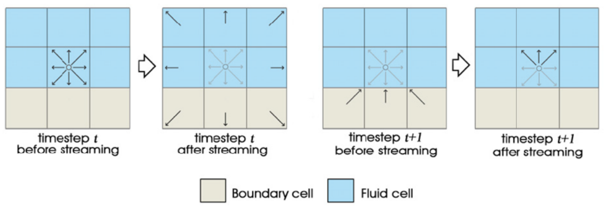
timestep t before streaming
timestep t after streaming
timestep t+1 before streaming
timestep t+1 after streaming
Boundary cell
Fluid cell

Figure 7: *Boundary conditions implemented: Half-Way Bounce Back.*

- **Velocity and density computations** After streaming all the particle distribution functions, we calculate the new velocity and density using a direct implementation of equations 1 and 2.
- **Collisions** The final step in the fluid simulation is calculating the new distribution functions after collision handling. The kernel uses the incompressible variant of the LBM, given by Equation 3. First we calculate the equilibrium distribution function with equation 4 to find the new distribution functions. Then the kernel saves the new f_i into the texture for use in the next time step.

4.2.1. Boundary Conditions

Building walls: A standard boundary condition between the simulation space (the empty space between buildings) and the simulation boundaries (the buildings themselves) is the no-slip boundary condition, illustrated in Figure 6(left). Free-slip conditions, as shown in Figure 6(right), are also popular and they suppose no friction between the fluid and the solid boundary. Finally, for a more realistic behavior, we have also implemented the so called half-way bounce back condition, which reflects the incoming particle distribution into their opposite directions, see Figure 7.

Simulation Domain: In our implementation there are basically three different conditions for the simulation domain: inlet, outlet and simulation sides [Moh11]. At the entrance boundary (inlet), which we will specify the speed, and thanks to Zhu and He's method [ZH97], we can find the density and the cell's initial vectors. At the exit boundary (outlet), where the output speed is not known, we take the values of the cells immediately before and extrapolate their values to calculate the missing directions. These are called open boundary conditions in the literature. For the sides there are two possibilities. One, called *pipe* boundary condition, which set the scene as the walls of cells occupied by solid simulate as if they were building. This is to make simulations of tubes and pipes. This can be used with cities as long as we leave enough margin of empty space between the simulation borders and the city, so that these conditions do not affect the simulation result. The second option is to use a *periodic* boundary condition. In this case the implementation connects one side of the simulation with what happens at the opposite side, and vice versa. This simulates that there are no rigid boundaries, although it is not 100% correct if there are buildings close to the simulation boundaries, because their presence can affect the simulation on the opposite boundary. There is a third option that would fix walls as outlets, but we did not find any support for this option, and in our implementation it resulted in an overall loss of numerical stability. In general, we observed that periodic boundaries are the most used in the literature. These conditions are shown in Figure 8.

4.2.2. Negative densities

During an LBM simulation, some cells may end up with zero density. As a result, and after applying the equilibrium distribution function (Equation 3), there may be cells with negative densities. It is possible to add a parameter α to the advection term to reduce advection in areas with low densities [CT05], and thereby avoid negative densities. The new equilibrium distribution function is then given by:

$$f_i^{eq}(\rho,\vec{v}) = w_i\left[\rho + \alpha\rho_0\left(\frac{3}{c^2}(\vec{e}_i\cdot\vec{v}) + \frac{9}{2c^4}(\vec{e}_i\cdot v)^2 - \frac{3}{2c^2}(\vec{v}\cdot\vec{v})\right)\right] \tag{5}$$

The variable α is defined by the well known *Smoothstep(0,λ,ρ)*, where λ is a user specified value: *Smoothstep()* will set α to 0 in cells with no wind, causing no advection to occur. Otherwise α will be greater than zero. We have found that $\lambda \in [0.1, 0.6]$ to work well.

5. Results

We implemented our system in a MacBook Pro Retina 15inch Late 2013, with an Intel Core i7 2.6GHz processor, 16GB DDR3, and a NVIDIA GeForce GT 750M 2048MB graphics card. For the implementation, it was used OpenGL 4.1 and Cuda Driver Version 7.5.20 with a Gpu Driver Version 10.10.5.2 310.42.25f01.

In Figure 8 we can see four stages of a simple city with a simple pipe and periodic boundary conditions (wind entering from the left towards the right of the city).

In Figure 9 we can see the velocities of the wind in a real example, the city of Girona (Catalonia, Spain), again with different simulation boundary conditions, and under different wind directions. As we can see, the only requirement of our system is a 2D slice of the city map with the building profiles separated from the background, something quite easy to acquire from public databases and local city councils.

In Table 1 we can observe the performance of our technique for two different simulation resolutions for the same area, which imply two different spatial resolutions, in our case, $1m^2$ and $10.2m^2$ for each simulation pixel. We can see that the coarser simulation is faster than the higher resolution one, at the cost of providing less accurate results.

6. Conclusions and Future Work

We have presented a low-cost, reliable interactive wind simulation technique for large urban environments, including the possibility to change wind direction and strength. This simulation is based on a Lattice-Boltzmann implementation, running on the GPU thanks to a CUDA implementation. This implementation allows to capture general wind behavior, and allows its usage in general urban physics implementation. However, its accuracy is limited by the size of the cell grid used, and the fact that the simulation runs only in a 2D environment (i.e., a layer parallel to the ground, and the 2D model neglects different wind velocity profiles and inequalities in the ground surface). If a more accurate simulation is needed, a full 3D CFD simulation should be implemented [Blo15, Blo14].

Future lines of research include hierarchical, adaptive multigrid techniques to allow the simulation to run at different scales, allowing a more accurate study in certain areas of interest, while preserving the overall advantage of the large simulation domain of this method. Another promising, but open line of research is the study of the interaction of wind, as done in this paper, with other urban elements, like trees, heat, humidity, pollutants or rain. The complex interplay of all these factors at the urban level require a granularity

Model	Resolution (px)	Resolution (m/px)	#buildigns	fps	SimulationTime (ms)
MyBlocks	400x290	1x1	18	50	10.83
Girona	248x508	3.4x3	226	50	12.21

Table 1: *Comparison of the results for two different simulation resolutions*

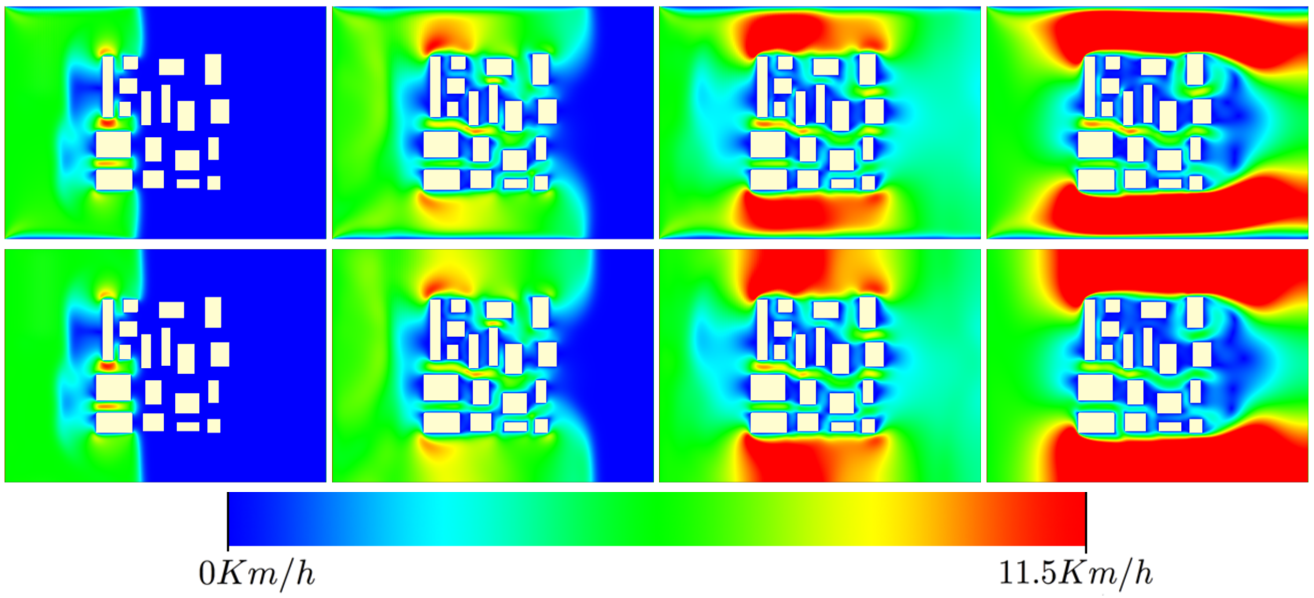

Figure 8: *Velocity profiles for different boundary conditions on a simple city, wind entering from the left simulation boundary and leaving on the right. Top: a pipe boundary condition. Bottom: a periodic boundary condition. Although differences are subtle, we can appreciate them at the upper and lower borders of the images*

degree like the technique we have presented in this paper in order to keep interactivity at the center of our requirements.

Acknowledgments

This work was partially funded by the TIN2014-52211-C2-2-R project from Ministerio de Economía y Competitividad, Spain.

References

[BJvH12] Blocken B., Janssen W., van Hooff T.: Cfd simulation for pedestrian wind comfort and wind safety in urban areas: General decision framework and case study for the eindhoven university campus. *Environmental Modelling & Software 30* (2012), 15 – 34. 1

[Blo14] Blocken B.: 50 years of computational wind engineering: Past, present and future. *Journal of Wind Engineering and Industrial Aerodynamics 129* (2014), 69 – 102. 1, 4

[Blo15] Blocken B.: Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations. *Building and Environment 91* (2015), 219 – 245. Fifty Year Anniversary for Building and Environment. 1, 4

[Bri08] Bridson R.: *Fluid Simulation for Computer Graphics*. AK Peters, 2008. 1

[BS01] Bukowsky R., Szalewicz K.: Monte carlo simulations for gaseous and liquid argon with complete ab initio nonadditive potential. *Journal of Chemical Physics* (2001). 1

[CCM92] Chen H., Chen S., Matthaeus W. H.: Recovery of the navier-stokes equations using a lattice-gas boltzmann method. *Physics Review A 45*, 8 (1992), R5339–R5342. 2

[CT05] Chu N. S.-H., Tai C.-L.: MoXi: real-time ink dispersion in absorbent paper. *ACM Transactions on Graphics 24*, 3 (2005), 504–511. 4

[HJ89] Higuera F., Jimenez J.: Boltzmann approach to lattice gas simulations. *Europhysics Letters 9*, 663 (1989). 2

[HL97] He X., Luo L.-S.: Lattice boltzmann model for the incompressible navier-stokes equation. *Journal of Statistical Physics 88*, 3-4 (1997), 927–944. 3

[LFWK05] Li W., Fan Z., Wei X., Kaufman A.: Flow simulation with complex boundaries. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, Pharr M., Fernando R., (Eds.). Addison-Wesley Professional, 2005. 2

[Moh11] Mohamad A.: *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*. SpringerLink : Bücher. Springer London, 2011. 4

[MZ88] McNamara G. R., Zanetti G.: Use of the boltzmann equation to simulate lattice-gas automata. *Physics Review Letters 61*, 20 (1988), 2332–2335. 2

[QdL92] Quian Y. H., d'Humières D., Lallemand P.: Lattice bgk models for navier-stokes equation. *Europhysics Letters 17*, 6 (1992). 2

[Sad99] Sadus R. J.: *Molecular Simulation of Fluids: Theory, Algorithms and Object-Orientation*. Elsevier Science Inc., New York, NY, USA, 1999. 1

[VRa15] VRatis: Higher productivity of single-phase flow simulations thanks to gpu acceleration, August 2015. 2

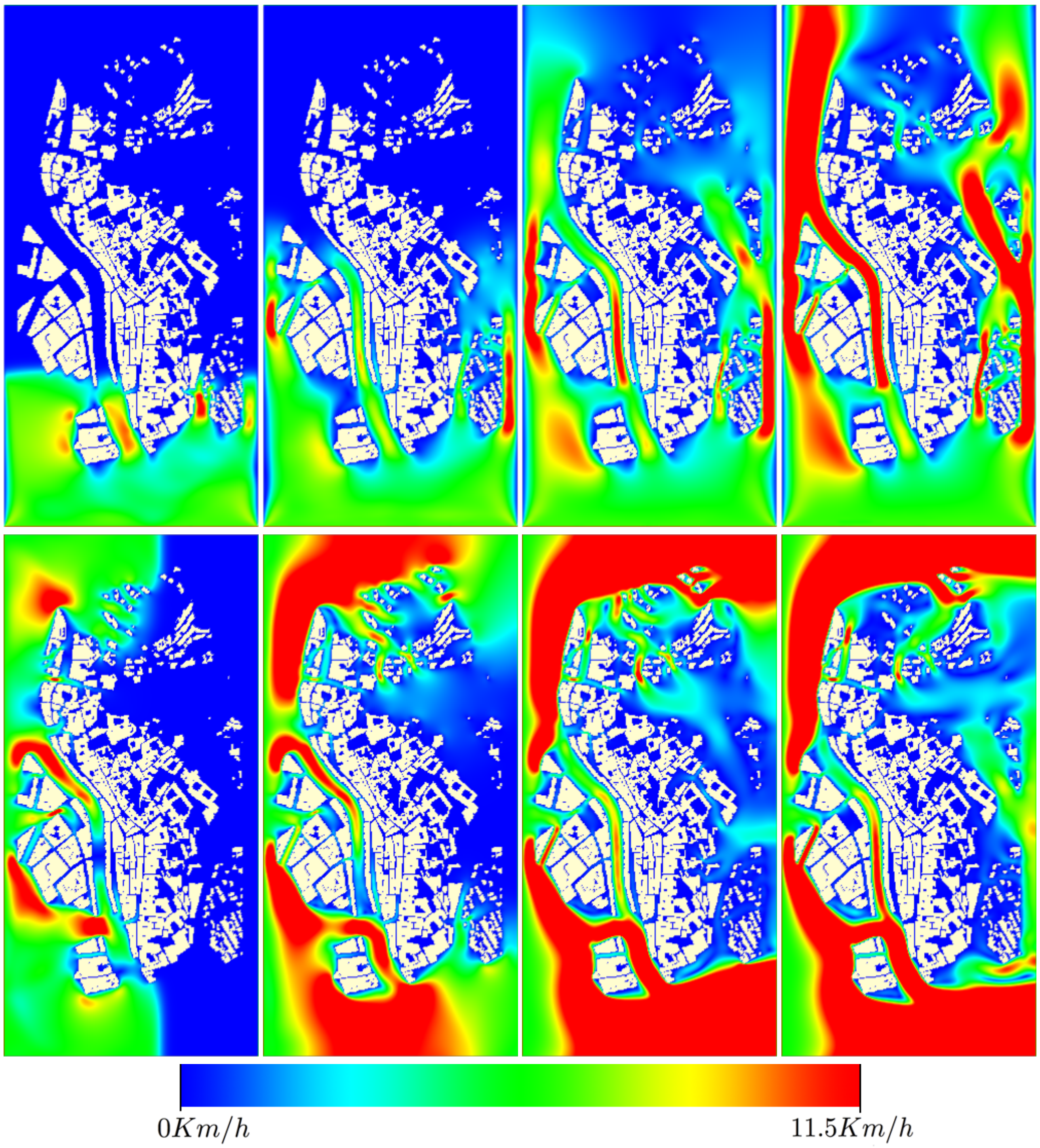

Figure 9: *Velocity profiles for different boundary conditions on the city of Girona. Top: pipe boundary conditions, wind entering from the bottom (actual north) towards the top. Bottom: periodic boundary conditions, with wind entering from the left (actual east).*

[Wag08] WAGNER A. J.: *A Practical Introduction to the Lattice Boltzmann Method*. Department of Physics North Dakota State University, 2008. 2

[WMT05] WANG H., MUCHA P. J., TURK G.: Water drops on surfaces. *ACM Transactions on Graphics 24*, 3 (2005), 921–929. 2

[ZH97] ZOU Q., HE X.: On pressure and velocity boundary conditions for the lattice boltzmann bgk model. *Physics of Fluids 9*, 6 (1997), 1591–1598. 4