

Procedural generation of flood-sensitive urban layouts

Environment and Planning B: Urban Analytics
and City Science
0(0) 1–21

© The Author(s) 2018

Article reuse guidelines:

sagepub.com/journals-permissions

DOI: 10.1177/2399808318812458

journals.sagepub.com/home/epb



Ahmed Mustafa 

Liège University, Belgium

Xiao Wei Zhang and Daniel G Aliaga

Purdue University, USA

Martin Bruwier

Liège University, Belgium

Gen Nishida

Purdue University, USA

Benjamin Dewals, Sébastien Ericum ,
**Pierre Archambeau, Michel Piroton and
Jacques Teller**

Liège University, Belgium

Abstract

Aside from modeling geometric shape, three-dimensional (3D) urban procedural modeling has shown its value in understanding, predicting and/or controlling effects of shape on design and urban planning. In this paper, instead of the constructing flood resistant measures, we create a procedural generation system for designing urban layouts that passively reduce water depth during a flooding scenario. Our tool enables exploring designs that passively lower flood depth everywhere or mostly in chosen key areas. Our approach tightly integrates a hydraulic model and a parameterized urban generation system with an optimization engine so as to find the least cost modification to an initial urban layout design. Further, due to the computational cost of a fluid simulation, we train neural networks to assist with accelerating the design process. We have applied our system to several real-world locations and have obtained improved 3D urban models in just a few seconds.

Corresponding author:

Ahmed Mustafa, Liège University, LEMA, Allée de la Découverte 9, Quartier Polytech I, 4000 Liège, Belgium.

Email: a.mustafa@ulg.ac.be

Keywords

Inverse procedural modeling, urban layout, urban flooding, neural network, Markov Chain Monte Carlo

Introduction

Three-dimensional (3D) urban modeling has gained more and more research attention in last decades. Besides modeling geometric shape, urban modeling has shown its research value in understanding, predicting and/or controlling effects of shape on many urban concerns such as urban planning (e.g. Vanegas et al., 2012; Weber et al., 2009), urban vehicular traffic (e.g. Garcia-Dorado et al., 2014; Sewall et al., 2011), crowd simulation (e.g. Feng et al., 2016), urban weather simulation (e.g. Garcia-Dorado et al., 2017) and water/fluid simulations (e.g. Bridson, 2015; Enright et al., 2002). In this paper, we establish a link between geometric urban layout design and urban flooding, assist in designing improved flood-sensitive cities, and generate more realistic simulations of flooding in an urban environment.

For several decades, structural flood protection measures such as levees, dams and dikes have been widely used to control floods. These measures have been criticized because they often interrupt natural flooding processes by removing natural land cover, reduce natural water storage capacity and disrupt water flow paths (Lennon et al., 2014; O'Neill, 2013). In the recent years, there is a shift in focus from hard flood controls towards a more strategic approach characterized by mitigating flood risk and increasing resilience during the urban design process (Lennon et al., 2014; White, 2008). In particular, several soft solutions have been introduced at different spatial scales ranging from regional scale to building scale. For example, storing water in farmland, where the land remains property of the farmer and is used for temporary water storage in extreme flooding (Fokkens, 2006). Other case-studies relocate the most sensitive land-use types; e.g. suggest moving residential areas to zones with a lower flood risk (Satterthwaite, 2007). However, the natural advantages of the floodplain and the trend towards urban densification and expansions has fomented that floodplain development continued apace, regardless of potential planning policies to control these factors (Moel and Aerts, 2010; Mustafa et al., 2018; White, 2008). Consequently, the potential damages as a result of floods will continue to rise. Within this context, digitally enhanced urban design has the potential to construct flood resistant urban patterns. Our approach is concerned with accommodating the unavoidability of floods through modifications to urban and architectural design.

Within computer graphics and geometric modeling, existing tools model either fluid simulation, which has a long history in computer graphics (e.g. Bridson, 2015; Enright et al., 2002; Losasso et al., 2004) or geometric modeling (e.g., forward procedural modeling (Müller et al., 2006; Vanegas et al., 2009), and inverse procedural modeling (Demir et al., 2015; Vanegas et al., 2012)).

Our key motivation is to create a procedural generation system that automatically generates 3D urban layouts that consider the influence of geometric urban characteristics (e.g. road width, orientation, curvature, etc.) on flow properties during flood water simulations (and in reality) (Figure 1). The urban model layout influences the distribution of water discharges between roads as well as the flow depths and velocities. In a sense, we explore urban geometric grammars that help reduce flooding: i.e. what urban design rules produce a passive barrier against natural floods? Our methodology is concerned with accommodating



Figure 1. Flood-sensitive urban layouts. We introduce a procedural urban design system that reduces flood water depth based on the geometrical characteristics of an urban layout. The user sets desired parameter values or building coverage, then our system optimizes the initial layout to reduce flood water depth. In this example, a layout in Frankfurt ((a) and (b)) shows an average water depth of 0.73 m and (c) is the initial layout which produces a simulated flood water level within 0.03 m of the actual layout's. The optimized layout ((d) and (e)) reduces the water level by 0.19 m (26%). All layouts retain the same building coverage.

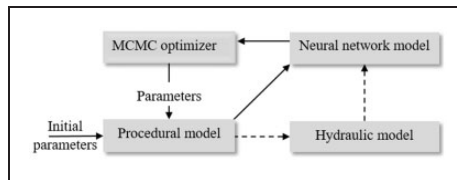


Figure 2. System pipeline.

the unavoidability of floods by pre-emptive modifications to urban design. Designing and evaluating different urban layouts in terms of flood damage requires considerable time and resources. Any effort to reduce the design time or required resources is a profitable investment. Visual computing design tools, such as ours, can greatly help urban designers and architects assess the implications of different design decisions on expected flood damage, reduce design time, and produce practical urban patterns.

Our proposed interactive and automatic approach consists of four components (Figure 2). First, we represent an urban area by dividing it into representative cells, typically 1×1 km. For each cell, we define a parameterized procedural model that can generate a wide range of possible urban layout configurations which mimic actual real-world urban patterns. Second, a porosity-based hydraulic model computes the water flow characteristics of a proposed urban layout cell. Third, to yield an interactive design process, we approximate the relationship between urban layout and flood flow characteristics with a trained neural network (NN). Fourth, we provide an inverse modeling engine to suggest the best urban layout that yields desired building coverage – the area of buildings at ground level divided by the area of the entire cell and expressed as a percentage – and water depth.

Our major contributions include:

- a parameterized procedural urban model of the urban layout within a 1 km^2 area controlling road network, blocks, parcels, and buildings geometries;
- integrating our procedural urban model with a porosity-based hydraulic model that computes the water flow characteristics based on 2D shallow-water equations; and

- an optimization-based and NN-based inverse-modeling engine that can interactively propose the optimal urban layout to satisfy a specified reduction in flood water depth while maintaining a specified building coverage.

We have implemented our approach in many virtual urban layouts as well as three real-world locations located in Liege (Belgium), Paris (France), and Frankfurt (Germany).

Previous work

Our work builds on flood flow modeling within urban areas as well as forward and inverse procedural modeling. Floods claim many lives and cause major economic losses. A recent report by the UN, entitled “Human Cost of Weather Related Disasters”, shows that 157,000 people have died as a result of floods between 1995 and 2015. Additionally, floods damages cost US\$104 billion/year globally according to the UN report “Global Assessment Report on Disaster Risk Reduction”. The magnitude and occurrence of flooding probability in many areas are currently increasing (Moel and Aerts, 2010; Pfister et al., 2004). Indeed, flood risks affect the day-to-day concerns of households, economic sectors and local governments in many areas around the world. Flood depth reductions, even if mild, can have a huge impact. For example, if the expected water height during a flood is 1.2 m referenced to ground level, installing polymer treated lumber panels may be necessary. If inundation water depth decreased by 20%, then at city or country-scale the resulting savings will be very significant.

Three-dimensional models have been employed for assessment and visualization of urban flood flow and characteristics. Mioc et al. (2012) used observed data to generate 3D city models and integrates it with an inundation model. This integrated model helps city managers determine unsafe buildings and infrastructure in case of flooding. Amirebrahimi et al. (2016) proposed an integrated framework to measure flood damage to buildings based on a detailed 3D building information model. More related to our work is the work proposed by Christensen (2016) who introduced a dynamic approach to understanding the effect that natural disaster emergencies can generate in urban areas, with a particular emphasis on flooding events in residential areas. This approach also enables the end-user to simulate multi-scenarios based on different flood or water levels and to evaluate the impact of policy or structural interventions on reducing flood damages using an interactive interface. However, the objective disregarded the impact of urban layout configurations on flood damage.

Prior literature on geometric urban modeling for flood-sensitive urban areas has tended to focus on architectural design and general site layout recommendations (e.g. Lennon et al., 2014; Watson and Adams, 2010; White, 2008). The adaptation measures presented in these literatures include usage of wet-proofing, construction based on elevated ground, building on stilts, using temporal flood defenses, increasing urban green areas and increasing the distance between buildings and water bodies. Existing literature did not make a comprehensive analysis to identify the relationship between urban layout parameters and flood damage such as geometrical arrangement of road network, parcels, and buildings. Nonetheless, computational approaches such as ours can play a distinctive role in urban geometric design and flood damage assessment.

There are already existing models that enable urban designers to procedurally alter and explore urban configurations. An in-depth review of urban procedural modeling can be found in Kelly and McCabe (2017), Smelik et al. (2014), Vanegas et al. (2009), and Watson et al. (2008). Parish and Müller (2001) introduced a system to model cities using

a procedural approach based on L-systems. Aliaga et al. (2008) proposed an interactive example-based approach that synthesizes urban patterns based on procedural modeling and image-based modeling. Vanegas et al. (2009) tackled an interdisciplinary research problem which considers modeling of behavioral and geometrical aspects of cities interactively. Yang et al. (2013) applied an optimized hierarchical splitting method to design urban layout automatically and interactively. Vanegas et al. (2012) proposed an inverse procedural system to generate urban 3D models according to several high-level parameters, such as average distance from buildings to closest parks and sun exposure, which motivate our work. While their approach uses a black-box procedural modeling, we explicitly define a set of parameters for our procedural modeling engine to make it suitable for training the NNs with a constant number of output values. In addition, we use a tensor field approach (Chen et al., 2008) to support a wide variety of road structures, which is crucial to increase the variation of the urban layouts.

The work of Whiting et al. (2009) is one of the first examples of iterative adjusting a procedural model's parameters to meet some design constraints. A range of later examples exist as well, such as the iteratively adjusted procedural model of Talton et al. (2011) who used Markov Chain Monte Carlo (MCMC) optimization to guide a procedural model to produce a geometric shape that conforms to a specification and, more recently, Garcia-Dorado et al. (2017) explore how to alter an urban design so as to yield a desired local weather pattern.

Compared to previous work, a major differentiation of our work is providing a computer system that integrates geometric rules and water flow assessment in order to provide urban layout designs that reduce flood water depth considering only changes to the urban geometrical configuration.

Methodology

Figure 2 provides a summary of our system pipeline. The system consists of four main modules (i) parameterized procedural urban model, (ii) a porosity-based hydraulic model that computes the water flow characteristics, (iii) an NN that used to compute hydraulic simulation, for a reason of speed, and (iv) an MCMC optimization-based inverse-modeling engine that can interactively propose the optimal urban layout.

The hydraulic model is used to train an NN to accelerate performance. Our hydraulic model focuses on river-based flooding scenarios. Although the dynamics of flooding vary with terrain, characterized by varying degrees of slope (Kirkby et al., 2002; Meraj et al., 2015), we do not consider slope in our current implementation. The two main reasons for this assumption include (1) many floodplain areas across the globe are relatively flat (e.g. Berthelot et al., 2015; CSIRO, 2000) and (2) slope could lead to underestimating the effect of urban geometry changes.

Our flooding depth simulations are performed by WOLF 2D model (Ercicum et al., 2010). WOLF 2D is widely implemented to compute flood damages in urban areas (e.g. Beckers et al., 2013; Bruwier et al., 2015; Ernst et al., 2010). In 2003, WOLF 2D was selected by Wallonia (Belgium) authorities to perform all detailed 2-D flow simulations to produce official inundation maps, including in the framework of the European Floods Directive (Arrault et al., 2016). WOLF 2D solves the fully 2D shallow-water equations on Cartesian grids based on a conservative finite volume scheme with a flux vector splitting technique (Ercicum et al., 2010). To make more practical the training time of our NN, we accelerate the computational performance of this model by using a lower resolution mesh.

Then, to compensate for the lower-resolution we include an anisotropic porosity model (Bruwier et al., 2018) to yield more accuracy.

Parameterized urban model

We represent an urban layout using a parameterized procedural model. While cities can occupy hundreds of square kilometers (e.g. a typical European city is 180 km² in area (Eurostat, 2012)), we are only concerned with the part of a city near a river. Since the search space of all possible urban patterns is quite large, our procedural tool (1) divides the urban area into grid cells and provides design guidelines for each cell and (2) reduces complexity by generating urban layouts based on a few number of rules and parameters (i.e., 12 parameters). Our tool also enables users to import and export typical GIS data which facilitates migrating the tool's outcomes into a wide range of GIS software.

Generation

Our procedural generation tool is inspired by Parish and Müller (2001), Vanegas et al. (2009) and CityEngine (from ESRI). Our tool first generates roads, then parcels, and finally buildings. Roads are divided into a two-level hierarchy (i.e., major and local roads). We adopt the tensor field approach (Chen et al., 2008) as it supports a wide variety of road network patterns using a single set of production rules. However, the original work by Chen et al. (2008) requires the user to specify the constraints by manually drawing the curves in order to generate the tensor field. To make the process fully automatic, we generate the constraints automatically from the parameters, and the tensor fields (Chen et al., 2008) are produced based on the generated constraints. Afterward, the tensor fields guide the generation of the road network. Once the road network is generated, we take each area surrounded by roads, called a block, and subdivide them into parcels, define parks, decide where to place buildings, and instantiate 3D building envelopes.

Parameters

Altogether, our procedural model is controlled by a 12-dimensional parameter vector $P = \{p_1, \dots, p_{12}\}$. These parameters are selected according to a literature survey of common parameters involved in previous studies (e.g. Aliaga et al., 2013; Sarralde et al., 2015; Vanegas et al., 2012). In the following, we describe each parameter (Figure 3):

- average road length p_1 — the distance between two adjacent intersections,
- road orientation p_2 — orientation of the initial radially outward road relative to lower-left corner,
- road curvature p_3 — rotation of a road segment when it passes through an intersection,
- major road width p_4 , and
- minor roads width p_5 .

Using p_2 , p_3 , and p_4 we generate two orthogonal major roads such that the orientation of the first major road at the center of the cell is determined by p_2 and the second major road is orthogonally intersecting with the first one at the center of the cell. Next, the two major roads are used as constraints to generate the tensor fields, and the minor roads are generated based on the tensor fields with parameters p_1 , p_3 , and p_5 . This approach can generate both the grid and radial patterns of roads by changing the road curvature p_3 . Furthermore, the overall orientation and density of roads can be controlled by p_2 and p_1 , respectively.

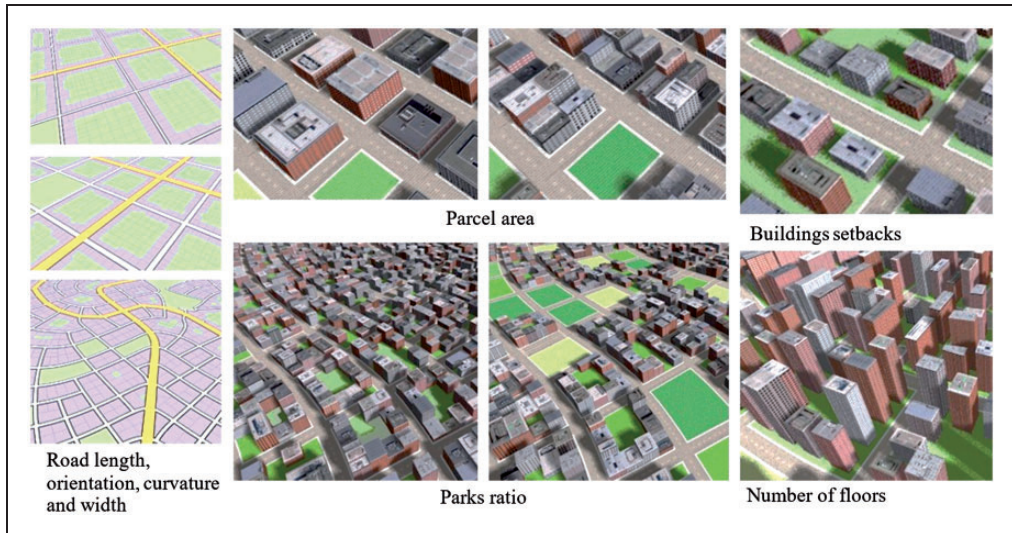


Figure 3. Urban procedural parameters.

Parcels are defined based on a recursive subdivision of oriented bounding boxes (OBB) fit around each city block, as in Vanegas et al. (2012). Parcels are controlled by the following parameters:

- average parcel area p_6 , and
- percentage of parcels selected as parks p_7 .

Buildings are generated with the following parameters:

- minimum number of floors p_8 ,
- maximum number of floors p_9 , and
- front p_{10} , rear p_{11} , and side p_{12} building setbacks.

The building footprint is defined by calculating the offset from the parcel using the setback parameters p_{10} , p_{11} , and p_{12} . The height of the building is determined by a uniform sampling between p_8 , and p_9 .

The variance of the input parameters representative of real-world locations are obtained with the assistance of our urban planning collaborators and by inspecting the cadastral data for 500 km² of Wallonia (Belgium). Table 1 lists a summary of these findings.

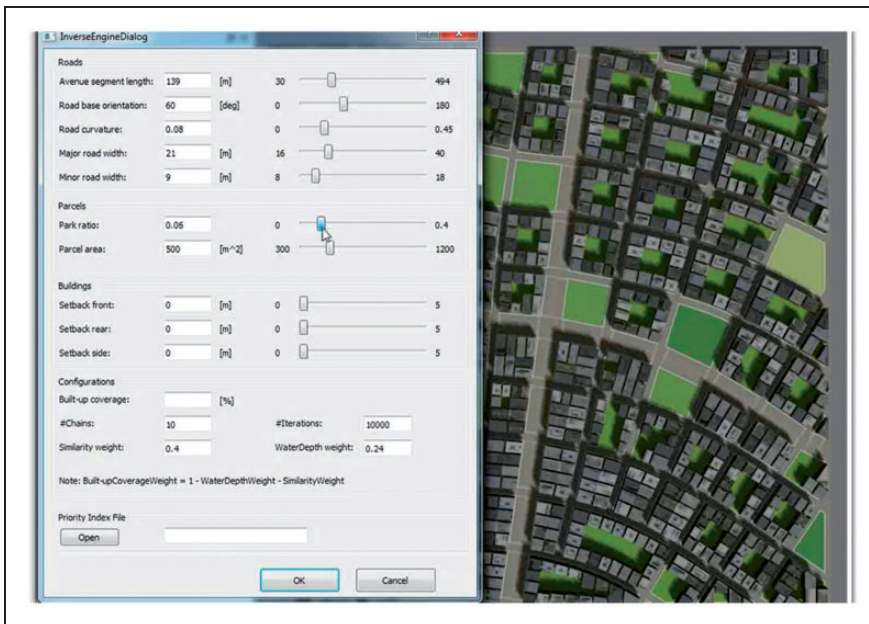
Inverse engine

Our inverse modeling engine searches for an urban configuration similar to the initial layout but producing a lower flooding depth. To search through the solution space we make use of a controlled random walking optimization, an error function, and an NN-based acceleration.

Solution searching. Our inverse modeling engine provides a Graphical User Interface (GUI) that enables the user to interactively explore urban layout configurations that yield a desired

Table 1. Input parameters for the parameterized urban model engine.

Parameter	Min value	Max value
Average street length	40 m	400 m
Road orientation	0°	180°
Road curvature	0 rad.	0.42 rad.
Major road width	16 m	34 m
Minor road width	8 m	16 m
Park coverage	5%	40%
Average parcel area	300 m ²	1100 m ²
Building front setback	0 m	5 m
Building rear setback	0 m	5 m
Building side setback	0 m	5 m

**Figure 4.** GUI of the inverse modeling engine.

water depth, Figure 4. The user interactively “paints” the desired water depth or uploads an image containing per cell values. As one option, we provide a mechanism to create such images with each cell encoded by 1 (low importance), 2 (medium importance), or 3 (high importance). For example, the user may specify certain key areas to have relatively less water depth (e.g. residential areas) and other areas may have higher water depth (e.g. parks). Alternatively, the user could specify a desire to lower the average water depth of the entire area but at the price of not being able to control where the reduction is lowest. Our method also enables the user to control the tradeoff between similarity of the optimized urban model to the initial model and the amount of water depth reduction.

MCMC-based optimization. The optimization consists of a Monte Carlo based search through the solution space. Our optimization uses MCMC (Gilks et al., 1995) and the

Metropolis–Hasting algorithm (Hastings, 1970; Metropolis et al., 1953) to seek a parameter vector P^* that yields desired spatially varying water depth values W^* . The vector W^* corresponds to desired water depth measurements over the G grid cells of the simulation area.

This optimization seeks P^* by attempting n state changes from s different initial seeds. Each initial seed is chosen by a sampling process based on the initial model defined by parameter vector P_0 . Given a current parameter vector P_t , for $t \in [1, n]$, a candidate state change P_t' is computed by sampling for each parameter p_i a value from a Gaussian distribution whose mean and standard deviation are estimated from Table 1. Since the distribution of proposed moves is symmetric (i.e., it consists of a sum of weighted Gaussians – thus, the probability of moving from state t to $t + 1$ is the same as from state $t + 1$ to t), the acceptance probability a of a move from a current state P_t to a new state P_{t+1} is given by the Metropolis ratio. Hence, the new state $P_{t+1} = P_t'$ with probability a and $P_{t+1} = P_t$ with probability $1 - a$.

Error function. The optimization error function is a weighted sum of three objectives: maintaining a similar building coverage between the initial model and the optimized urban layout, achieving the desired water depth values in the optimized urban layout, and keeping the optimized parameter vector as similar as desired to the initial one. The use of building coverage is to ensure the new configuration can house the same number of people (note: we set number of floors at 4 so the gross floor area of all buildings is maintained in all test-cases presented in this paper); simply removing buildings will decrease flood level but of course leave no space for people.

The optimization error function E can be written as **IAQ1**

$$E_t = \alpha \frac{\|B_t - B_0\|}{B_0} + \beta \frac{1}{G} \sum_{i=1}^G \left(\frac{w_{it} - w_i^*}{w_d} \right)^2 + \gamma \frac{\|P_t - P_0\|}{\|P_0\|}$$

where the variables

- B_0 and B_t correspond to the initial model's and current model's building coverage,
- w_{it} and w_i^* correspond to the current model's water depth for cell i and to the desired water depth for cell i ,
- w_d corresponds to a tolerance value which is used to balance the small values in the water depth part; if w_d is too small, it will take much weight in the total optimization error function,
- α , β and γ correspond to the weight for building coverage, water depth, and layout similarity such that $\alpha + \beta + \gamma = 1.0$, and
- G is the number of total outputs.

The similarity level is calculated as follows

$$\frac{1}{N} \sum_{i=1}^N \frac{|p_i - p_{ti}|}{p_{ti}}$$

where

- p_i : The values of the current parameter vector.
- p_{ti} : The values of the original parameter vector.

Acceleration. To accelerate produce an interactive system, we make use of two trained NNs. First, during each step of the optimization, a hydraulic simulation must be computed which can quickly lead to a very time consuming process. A typical hydraulic simulation run requires about two to three hours on a desktop PC. We accelerate performance significantly by using an NN. The hydraulic model (Results and discussion section) is used to train the NN.

With the Caffe framework (Jia et al., 2014), we build a single-layer fully connected NN (including Dropout layer and ReLU Activation layer) and train it to predict a grid of water depth values W given the parameter vector P . Our training dataset consists of 7000 hydraulic simulations performed offline. Each simulation uses a random parameter vector P with values sampled within the ranges shown in Table 1. We used 10 desktop computers with multiple cores to train the NN. Therefore, we ran 39 simulations simultaneously. The hydraulic computations of the 7000 configurations lasted in about three weeks. The loss function in the loss layer, which is used to measure the error between an NN's output and the actual output, is EuclideanLoss. It is a standard function in the loss layer in Caffe framework and computes the sum of squares of differences between its two inputs. A second similar NN is used to predict the building coverage resulting from a parameter vector P . While generating urban layouts is significantly fast (e.g. approximately 0.5 seconds) it still prohibits an interactive system when needing many MCMC iterations. Thus another dedicated NN is trained for this objective. For the first NN, we trained 5000 iterations. For the second NN, we trained 20,000 iterations. The results section contains additional information and evaluation of our methods.

Results and discussion

We implement our system in C++, using Qt and OpenCV, and running on a desktop computer clocked at 3.20 GHz with an NVIDIA GTX780 graphics card and 24.0 GB RAM. This section shows several analysis and case-study results. Although some urban patterns might not be represented accurately, our urban generation system supports a wide variety of typical urban patterns, which enables us to effectively find a desired pattern from an otherwise huge search space (Figure 5).

Flooding simulation: Our porosity WOLF 2D model runs for an average of 2.5 hours to calculate the water for each urban layout versus about 10 days using the classical WOLF 2D model. The relative average water depths errors between the classical WOLF 2D and porosity WOLF 2D for three test cases is less than 1% (≈ 1 cm). However, the optimization process may last for several days to find the optimal urban layout using porosity WOLF 2D model – thus we train an NN.

Neural networks: We train a hydraulic-model NN and a building coverage NN. The hydraulic-model NN is trained using our dataset of 7000 precomputed flood depth solutions for a random set of urban configurations. After training, we compare the output of the NN to that produced by the porosity WOLF 2D model for two layouts (Figure 6). In this comparison, we use an NN trained to produce 20×20 water depth values, or 400 outputs. The comparison demonstrates that the average water depth of the trained NN closely resembles the ones produced by the porosity WOLF 2D model.

We also compare the performance of our hydraulic-model NN to the use of a standard nearest neighbor linear interpolation (Figure 7). The multiple graphs show the number of test cases for different relative error rates (3%, 4%, and 5%) and for different output spatial resolutions – for example, 1 corresponds to obtaining a single depth value for the entire urban layout, 16 corresponds to obtaining 4×4 depth values, 100 corresponds to obtaining

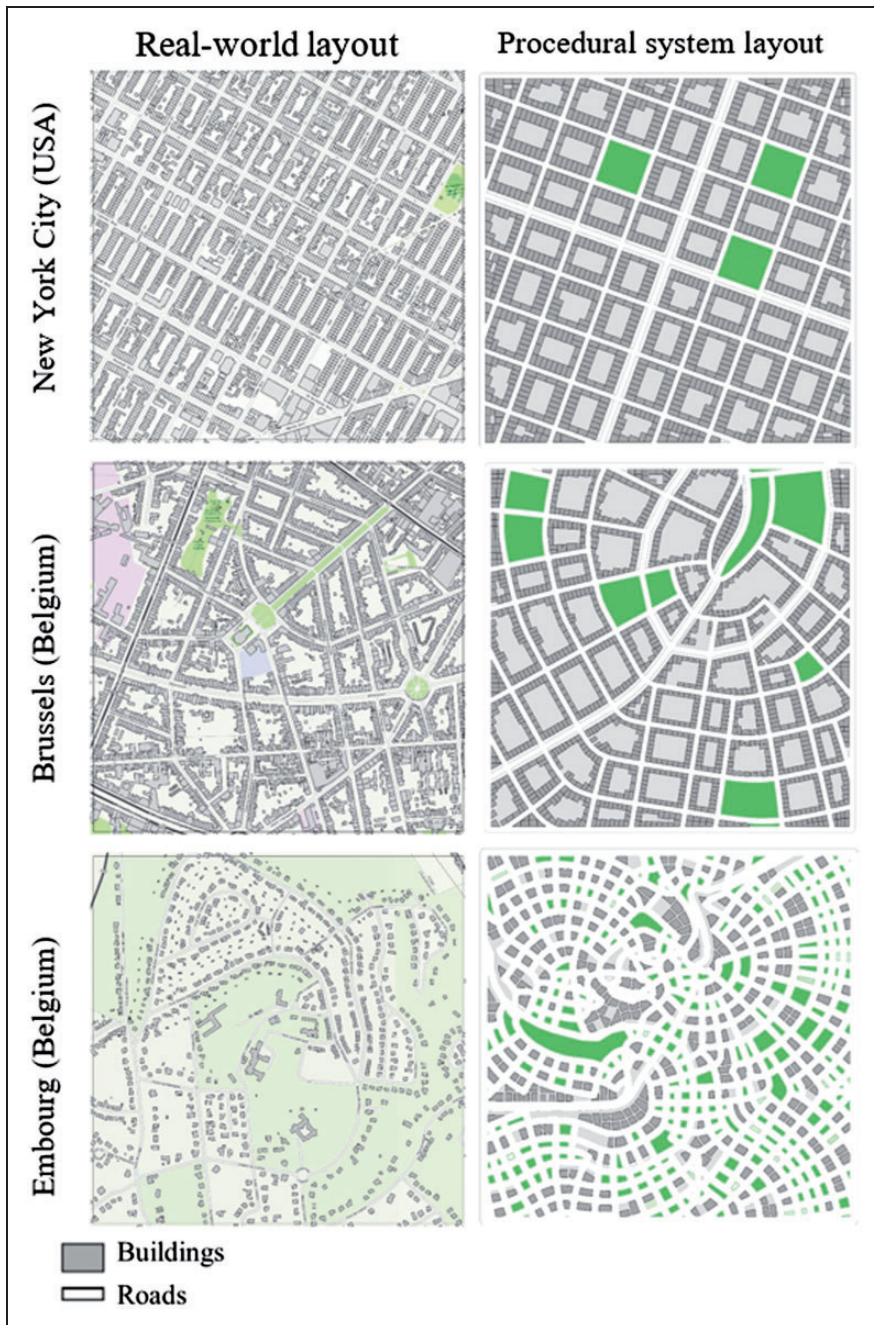


Figure 5. Real-world layouts versus procedural urban generator layouts.

10×10 depth values, and so forth. Overall, the NN performs much better than k-nearest neighbor (KNN) interpolation implying that is able to “learn” the function well. We choose the 20×20 output resolution for most of our results because it yields a good balance between spatial resolution and accuracy.

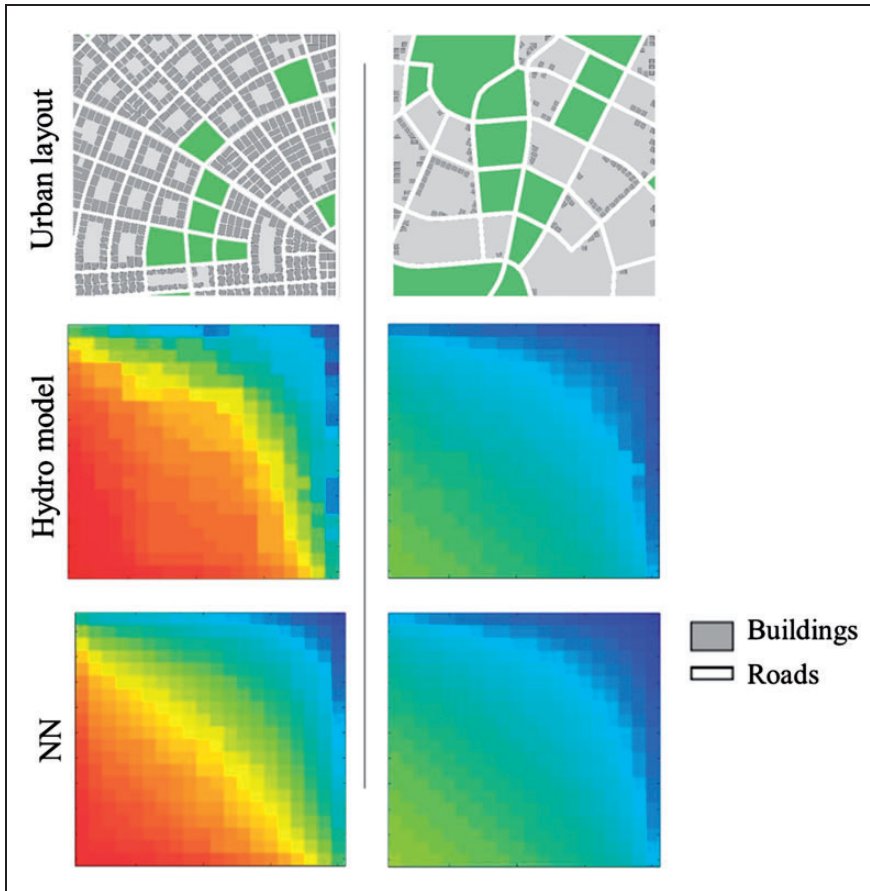


Figure 6. Water depth result maps from hydro model and NN for 400 outputs. Each cell represents average water depth for 50 m² area.

Figure 8 shows the error rate and validation of our building coverage NN. This NN is trained using 80,000 randomly sampled procedural models; overall the network yields very good results.

Urban layouts: In order to find the optimal layout that reduces water depth for an initial urban layout, we use our MCMC-based optimizer (Figure 4). The optimizer allows users to set initial parameter values and/or desired building coverage ratio. It also allows us to set the number of iterations, threads, and weights for similarity to the initial parameter values, reducing the average water depth, and building coverage weight. If the user does not explicitly provide a per-cell desired water depth, then with our GUI the user can define three levels of water-depth priority (i.e., low (L), medium (M), or high (H)) which can be defined for one or more target zones.

We tested our system on three different synthesized urban layouts. To set the initial parameter values (i.e., original layouts), we randomly selected scenarios from the urban generator. For all layouts, the system runs for about one minute to find the optimal layout that meets desired configurations (using a single core and without GPU acceleration). Our optimizer generates layouts that reduce the average water depth by up to 19%, 25% and 22% for cases 1, 2 and 3 respectively. Figure 9 shows original layouts and optimized

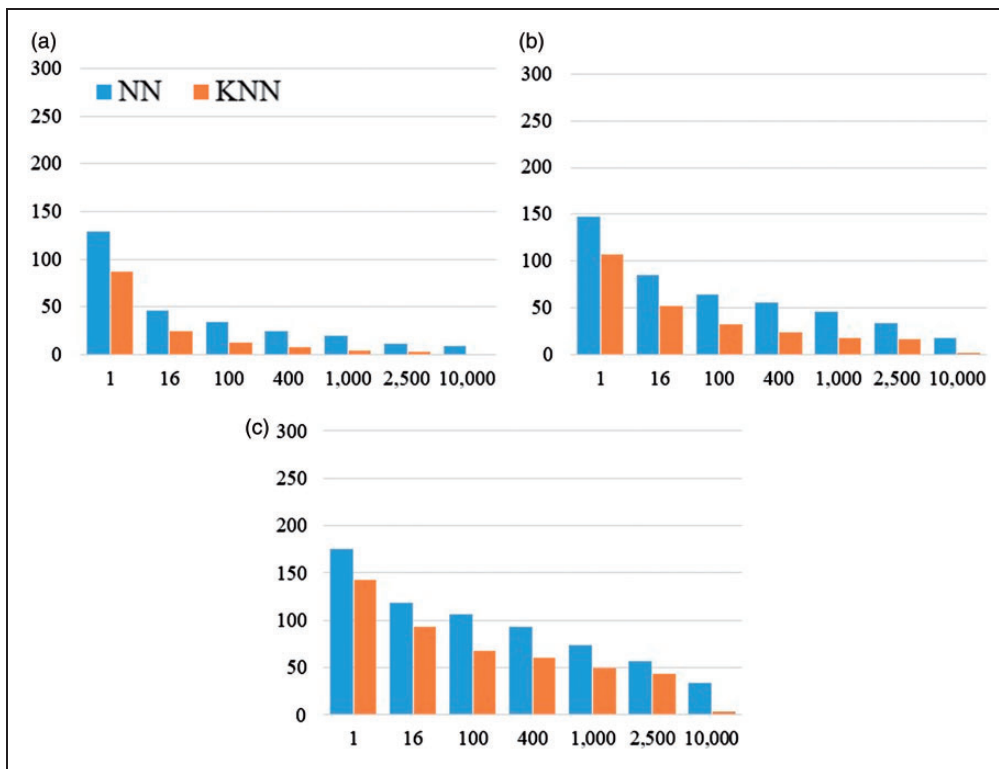


Figure 7. Water depth error plots for 300 test cases considering several outputs. X-axis represents number of output and Y-axis represents number of test cases. (a) corresponds to 3%, (b) corresponds to 4% and (c) corresponds to 5% relative error.

layouts using a value for γ so as to obtain layouts with medium-level similarity to the original. Figure 10 presents the convergence curve of the objective function of our MCMC-based optimizer for case 3 as an example. The figure reveals that the optimizer converges faster when we increase the number of threads.

Figure 11 presents water depth reduction for different layout similarity levels. As expected, decreasing similarity will reduce water depth as the optimizer can search within a larger space of parameter combinations. It is also worth noting that case 2 shows the highest water reduction. Since this case has a relatively lower building coverage, the optimizer has more freedom to re-locate buildings so as to reduce flood levels.

For the same test cases, we set three priority (or importance) zones with regards to desired water level. Figure 12 gives all results for different zones. It is clear that the ability to reduce water depth for each zone depends on the test case. However, our system is still able to reduce water depth in a specific area. For instance, the water depth reductions for low similarity case within the high importance zone are 20%, 24% and 29% for cases 1, 2 and 3 respectively.

Figure 13 contains the layouts used in Figures 11 and 12. Specifically, the top row shows the original synthetic urban layouts. The next nine scenarios correspond to the three different layout similarities for three test cases shown in Figure 9. The bottom row of scenarios in Figure 13 corresponds to the low-similarity solutions, each at three water depth priority

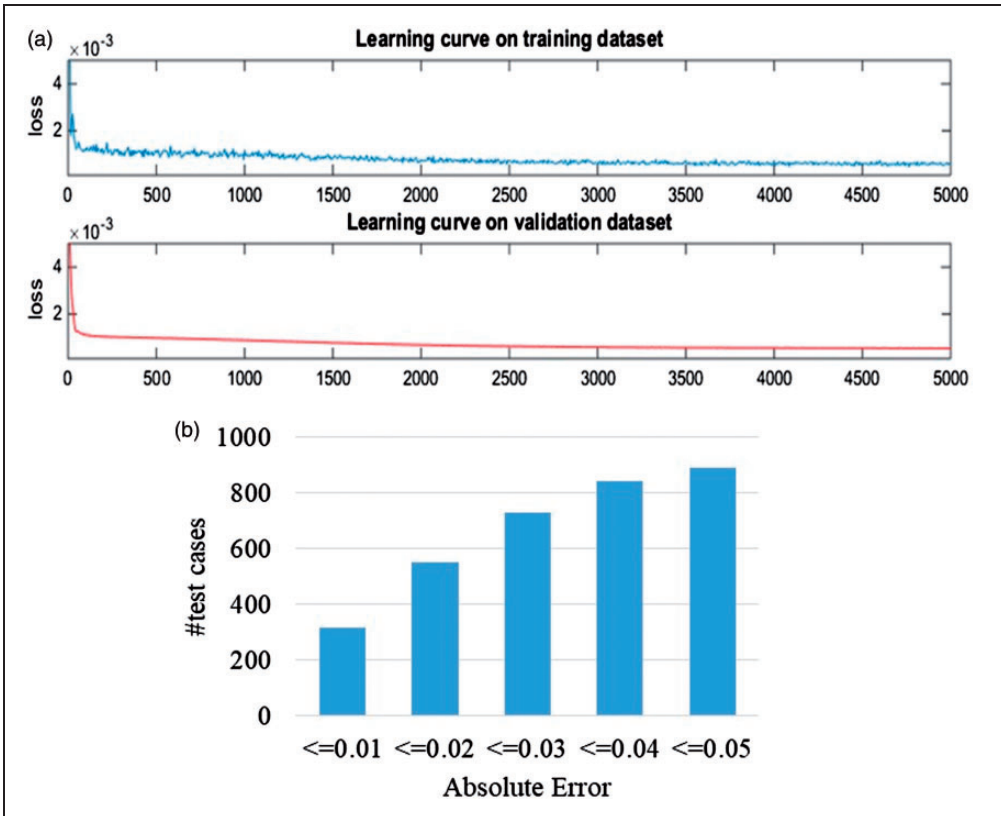


Figure 8. (a) learning curves presents loss (x-axis) versus iterations (y-axis) for training and validation datasets, and (b) the absolute error for a number of test cases (out of 1000) that occur between a range of absolute errors [AQ19].

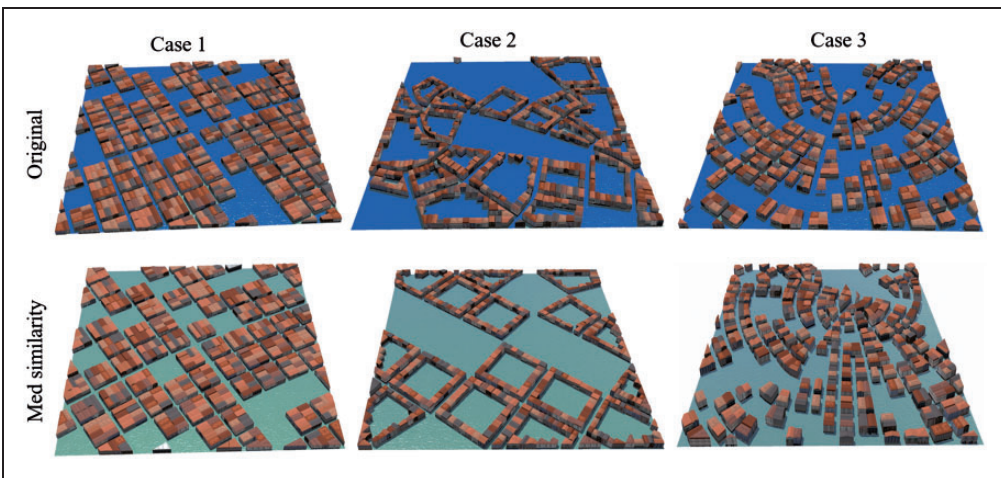


Figure 9. Three-dimensional simulations of the three test cases.

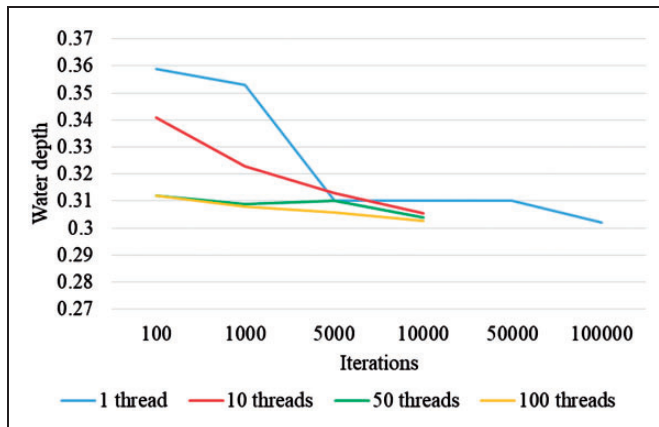


Figure 10. The convergence curve for case 3.

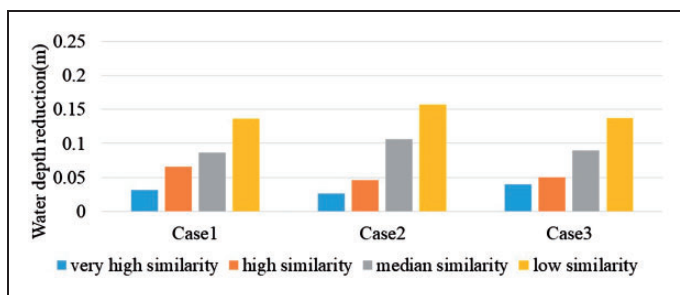


Figure 11. Water reduction (m) referenced to the original layouts.

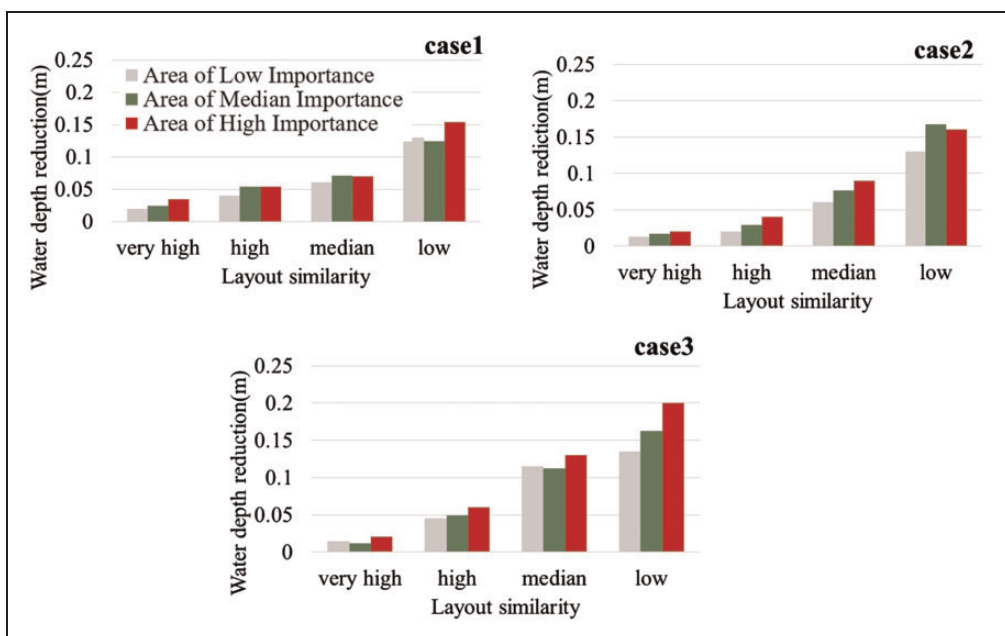


Figure 12. Water reduction in meters (x-axis) for various layout similarities (y-axis) considering different priority zones referenced to the original layouts.

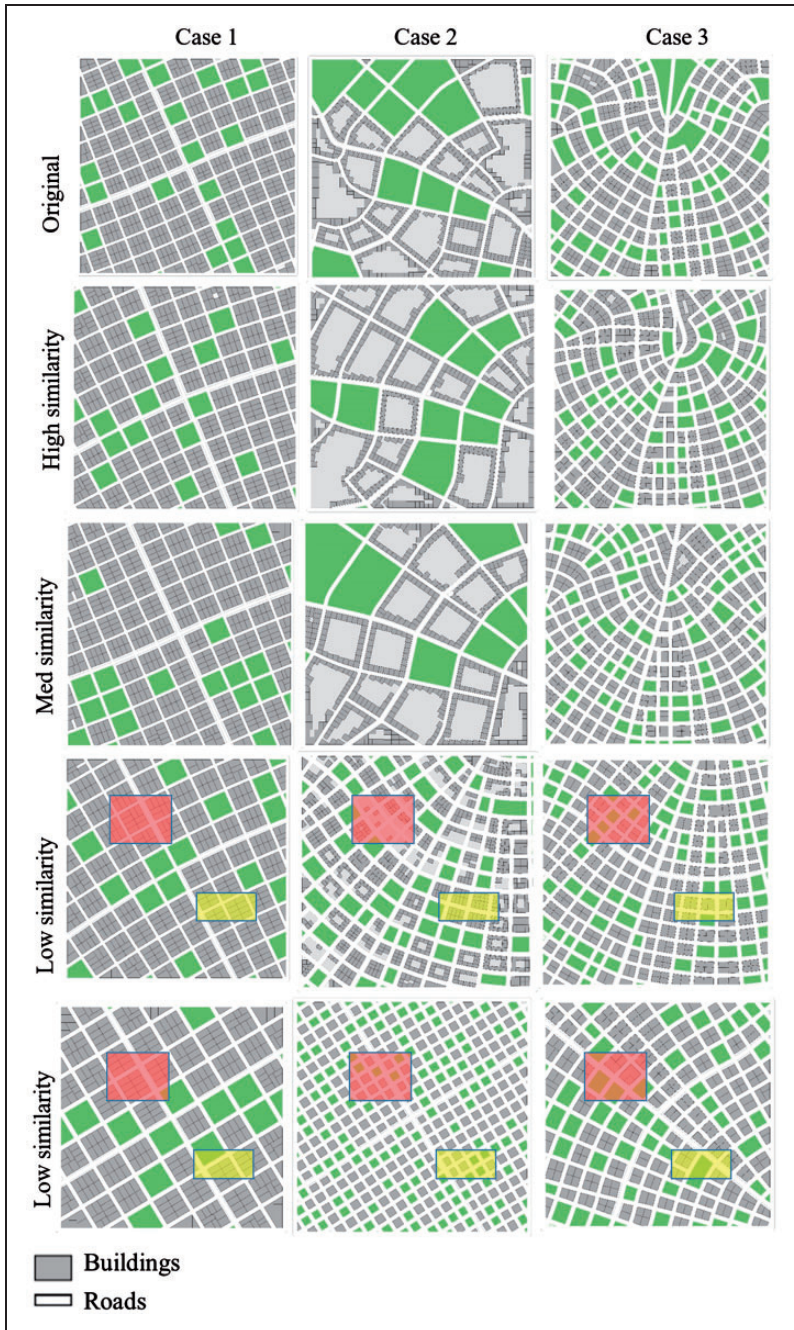


Figure 13. Resulted layouts from our system considering different similarity levels. It also shows low similarity layouts for importance zones (IZ) considering three zones: low importance (background), med importance (yellow zone) and high importance (red zone).

Table 2. Average water depth for each case in Figure 10.

		Average water depth	Building similarity level
Case 1	Original	0.72	–
	High similarity	0.66	0.26
	Med similarity	0.64	0.47
	Low similarity	0.59	0.87
Case 2	Original	0.63	–
	High similarity	0.59	0.23
	Med similarity	0.53	0.50
	Low similarity	0.48	0.72
Case 3	Original	0.64	–
	High similarity	0.59	0.28
	Med similarity	0.55	0.31
	Low similarity	0.50	0.47

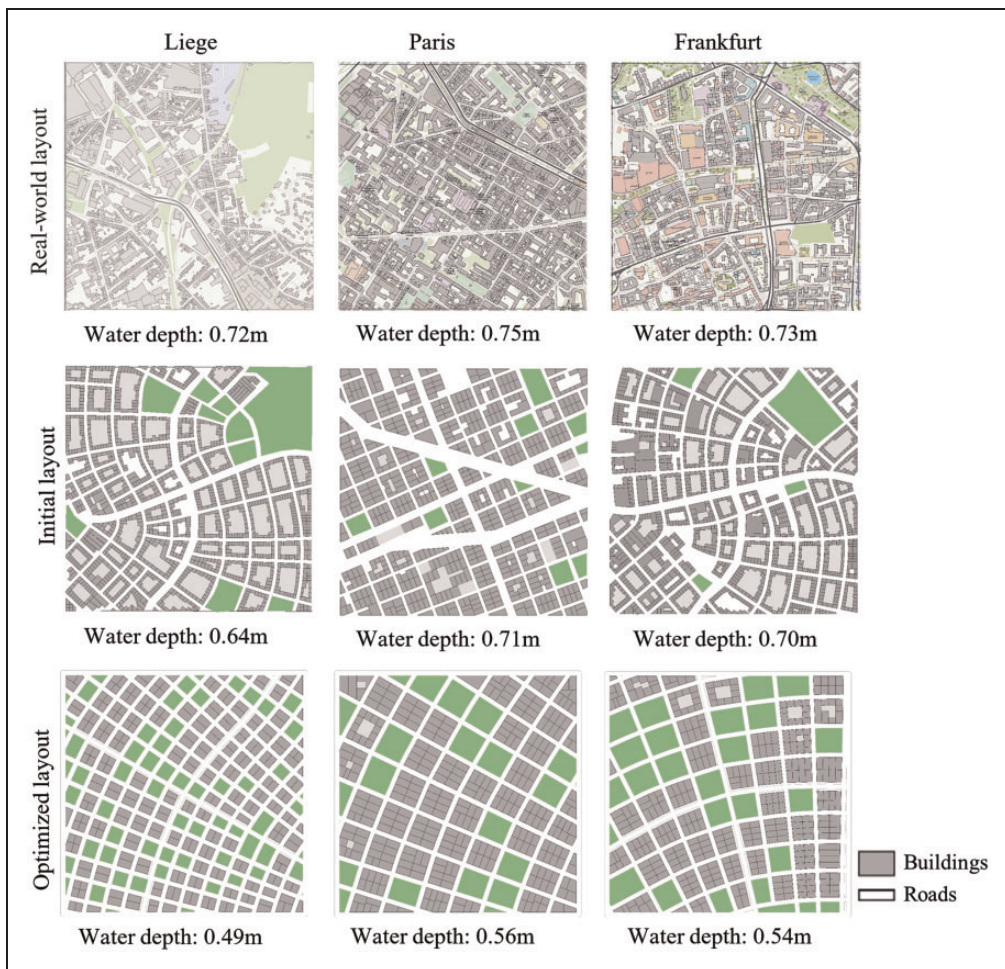


Figure 14. Real-world test cases.

Table 3. Average water depth for the real-world case (Figure 14) calculated by the porosity WOLF 2D model and our NN.

Case	Layout	Hydro model	NN
Liege	Initial	0.66	0.64
	Optimized	0.46	0.49
Paris	Initial	0.69	0.71
	Optimized	0.59	0.56
Frankfurt	Initial	0.66	0.70
	Optimized	0.57	0.54

levels, shown in Figure 12. The bottom two rows visually show the zones at the various selected priority levels.

Table 2 lists the water depth and layout similarities for three test cases shown in Figures 9 to 13.

We test three real-world layouts located in Liege, Paris and Frankfurt. All three cities are subject to riverine floods. Figure 14 presents the real-world, initial and optimized layouts. Table 3 compares the average water depth for each case calculated by the porosity WOLF 2D model and our trained NN.

Our system cannot automatically extract parameter values from existing layouts; therefore, we manually determine such parameter values to get layouts as close as possible to reality. Thereafter, the system automatically searches for the optimal layouts.

In general, our system keeps almost the same building coverage as in the original layout, and tends to increase parcel area and road width, and decreases road length and rear setback in order to achieve the optimal solutions. Road curvature and orientation do not play an important role. The optimal solutions always show larger parcel area; however, parcels can be subdivided into several buildings in the reality.

Conclusions and future work

We have coupled an automatic design approach for urban procedural modeling with a hydraulic model. Further, to achieve interactive feedback, our approach has employed NN and MCMC so that we get results in a few seconds. Based on several case studies, our findings highlighted that the impact of geometric characteristics of urban patterns (e.g. street width, park ratio, etc.) on flow properties during urban floods is significant. This is especially important for accurate flood/water simulations and for city planners concerned with flooding. Our approach can reduce water depth during the design phase of an urban layout and without other flood-mitigation costs.

There are some open problems remaining for future exploration. While our approach enables the user to design a flood-resistant urban area, the output controllability is limited. Recently, several interactive sketching frameworks for 3D modeling are proposed in computer graphics literature (e.g. Nishida et al., 2016, Guerin et al., 2017). **[AQ2]** Their approaches offer more controllability for the output 3D model by sketching. It would be interesting to add an interactive sketching interface to our framework. Another important next step in the research is the analysis of real-world case studies, which would showcase the operationality of the system and hence increase the impact of this assessment tool for urban planning practice. Specifically, the use of a deep learning network should be explored to directly take a real-world urban layout as input and produce estimated flood depth values.

In addition, we would like to process larger geographic areas as well as consider other flooding scenarios (e.g. coastal floods, flash floods, etc.).

Declaration of conflicting interests


The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The research was funded through the ARC grant for Concerted Research Actions for project number 13/17-01 financed by the French Community of Belgium (Wallonia-Brussels Federation) and through the European Regional Development Fund – FEDER (Wal-e-Cities Project).

ORCID iD

Ahmed Mustafa  <http://orcid.org/0000-0002-1592-6637>

Sébastien Erpicum  <http://orcid.org/0000-0002-7094-9604>

References

- Aliaga DG, Vanegas CA and Benes B (2008) Interactive example-based urban layout synthesis. In: *ACM SIGGRAPH Asia 2008 Papers*, New York, NY, USA, 2008, pp. 160:1–160:10. SIGGRAPH Asia '08. ACM. **[AQ3]**
- Aliaga DG, Vanegas C, Lei M, et al. (2013) Visualization-based decision tool for urban meteorological modeling. *Environment and Planning B: Planning and Design* 40(2): 271–288.
- Amirebrahimi S, Rajabifard A, Mendis P, et al. (2016) A framework for a microscale flood damage assessment and visualization for a building using BIM–GIS integration. *International Journal of Digital Earth* 9(4): 363–386.
- Arrault A, Finaud-Guyot P, Archambeau P, et al. (2016) Hydrodynamics of long-duration urban floods: Experiments and numerical modelling. *Natural Hazards and Earth System Sciences* 16(6): 1413–1429.
- Beckers A, Dewals B, Erpicum S, et al. (2013) Contribution of land use changes to future flood damage along the river Meuse in the Walloon region. *Natural Hazards and Earth System Sciences* 13(9): 2301–2318.
- Berthelot J-S, Saint-Laurent D, Gervais-Beaulac V, et al. (2015) A comparison of the composition and diversity of tree populations along a hydrological gradient in floodplains (Southern Québec, Canada). *Forests* 6(4): 929–956.
- Bridson R (2015) *Fluid Simulation for Computer Graphics*. 2nd ed. ■: CRC Press. **[AQ4]**
- Bruwier M, Erpicum S, Piroton M, et al. (2015) Assessing the operation rules of a reservoir system based on a detailed modelling chain. *Natural Hazards and Earth System Sciences* 15(3): 365–379.
- Bruwier M, Mustafa A, Aliaga DG, et al. (2018) Influence of urban pattern on inundation flow in floodplains of lowland rivers. *Science of the Total Environment* 622–623: 446–458.
- Chen G, Esch G, Wonka P, et al. (2008) Interactive procedural street modeling. In: *ACM Transactions on Graphics (TOG)*, p. 103. ACM. Available at: <http://dl.acm.org/citation.cfm?id=1360702> (accessed 2 February 2017).
- Christensen PH (2016) Understanding the potential of 4D geospatial modeling to enhance flood mitigation planning in residential areas. *Inventing the House: Case-Specific Studies on Housing Innovation* ■: 6. **[AQ5]**
- Csiro (2000) *Floodplain Management in Australia: Best Practice Guidelines*. ■: Csiro Publishing. **[AQ6]**
- Demir I, Aliaga DG and Benes B (2015) Coupled segmentation and similarity detection for architectural models. *ACM Transactions on Graphics* 34(4): 104:1–104:11.

- Enright D, Marschner S and Fedkiw R (2002) Animation and rendering of complex water surfaces. In: *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2002, pp. 736–744. SIGGRAPH '02. ACM. **[AQ7]**
- Ernst J, Dewals BJ, Detrembleur S, et al. (2010) Micro-scale flood risk analysis based on detailed 2D hydraulic modelling and high resolution geographic data. *Natural Hazards* 55(2): 181–209.
- Ercicium S, Dewals BJ, Archambeau P, et al. (2010) Dam break flow computation based on an efficient flux vector splitting. *Journal of Computational and Applied Mathematics* 234(7): 2143–2151.
- Eurostat (2012) European cities. Available at: <http://ec.europa.eu/eurostat/web/cities/spatial-units> (accessed 16 May 2017).
- Feng T, Yu L-F, Yeung S-K, et al. (2016) Crowd-driven mid-scale layout design. *ACM Transactions on Graphics* 35(4): 132:1–132:14.
- Fokkens B (2006) The Dutch strategy for safety and river flood prevention. In: *Extreme Hydrological Events: New Concepts for Security*. Dordrecht: Springer, pp. 337–352.
- Garcia-Dorado I, Aliaga DG and Ukkusuri SV (2014) Designing large-scale interactive traffic animations for urban modeling. *Computer Graphics Forum* 33(2): 411–420.
- Garcia-Dorado I, Aliaga DG, Bhalachandran S, et al. (2017) Fast weather simulation for inverse procedural design of 3D urban models. *ACM Transactions on Graphics* 36(2): 21:1–21:19.
- Gilks WR, Best NG and Tan KKC (1995) Adaptive rejection metropolis sampling within Gibbs sampling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 44(4): 455–472.
- Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57(1): 97–109.
- Jia Y, Shelhamer E, Donahue J, et al. (2014) Caffe: Convolutional architecture for fast feature embedding. Available at: <http://arxiv.org/abs/1408.5093> (accessed 27 October 2018).
- Kelly G and McCabe H (2017) A survey of procedural techniques for city generation. *The ITB Journal* 7(2): ■. **[AQ8]**
- Kirkby M, Bracken L and Reaney S (2002) The influence of land use, soils and topography on the delivery of hillslope runoff to channels in SE Spain. *Earth Surface Processes and Landforms* 27(13): 1459–1473.
- Lennon M, Scott M and O'Neill E (2014) Urban design and adapting to flood risk: The role of green infrastructure. *Journal of Urban Design* 19(5): 745–758.
- Losasso F, Gibou F and Fedkiw R (2004) Simulating water and smoke with an octree data structure. In: *ACM SIGGRAPH 2004 Papers*, New York, NY, USA, 2004, pp. 457–462. SIGGRAPH '04. ACM. **[AQ9]**
- Meraj G, Romshoo SA, Yousuf AR, et al. (2015) Assessing the influence of watershed characteristics on the flood vulnerability of Jhelum basin in Kashmir Himalaya. *Natural Hazards* 77(1): 153–175.
- Metropolis N, Rosenbluth AW, Rosenbluth MN, et al. (1953) Equation of state calculations by fast computing machines. *The Journal of Chemical Physics* 21(6): 1087–1092.
- Mioc D, Anton F, Nickerson B, et al. (2012) Flood progression modelling and impact analysis. In: *Efficient Decision Support Systems – Practice and Challenges in Multidisciplinary Domains*. InTech, pp. 227–246. **[AQ10]**
- Moel H de and Aerts JCJH (2010) Effect of uncertainty in land use, damage models and inundation depth on flood damage estimates. *Natural Hazards* 58(1): 407–425.
- Müller P, Wonka P, Haegler S, et al. (2006) Procedural Modeling of Buildings. In: *ACM SIGGRAPH 2006 Papers*, New York, NY, USA, 2006, pp. 614–623. SIGGRAPH '06. ACM. **[AQ11]**
- Mustafa A, Bruwier M, Archambeau P, et al. (2018) Effects of spatial planning on future flood risks in urban environments. *Journal of Environmental Management* 225: 193–204.
- O'Neill E (2013) Neighbourhood design considerations in flood risk management. *Planning Theory and Practice* 14(1): 129–134.
- Parish YIH and Müller P (2001) Procedural modeling of cities. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 2001, pp. 301–308. SIGGRAPH '01. ACM. **[AQ12]**
- Pfister L, Kwadijk J, Musy A, et al. (2004) Climate change, land use change and runoff prediction in the Rhine–Meuse basins. *River Research and Applications* 20(3): 229–241.

- Sarralde JJ, Quinn DJ, Wiesmann D, et al. (2015) Solar energy and urban morphology: Scenarios for increasing the renewable energy potential of neighbourhoods in London. *Renewable Energy* 73: 10–17.
- Satterthwaite D (2007) *Adapting to Climate Change in Urban Areas: The Possibilities and Constraints in Low- and Middle-income Nations*. ■: IIED. [AQ13]
- Sewall J, Wilkie D and Lin MC (2011) Interactive hybrid simulation of large-scale traffic. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*, New York, NY, USA, 2011, pp. 135:1–135:12. SA '11. ACM. [AQ14]
- Smelik RM, Tutenel T, Bidarra R, et al. (2014) A survey on procedural modelling for virtual worlds. *Computer Graphics Forum* 33(6): 31–50.
- Talton JO, Lou Y, Lesser S, et al. (2011) Metropolis procedural modeling. *ACM Transactions on Graphics* 30(2): 11:1–11:14.
- Vanegas CA, Aliaga DG, Benes B, et al. (2009) Interactive design of urban spaces using geometrical and behavioral modeling. In: *ACM SIGGRAPH Asia 2009 Papers*, New York, NY, USA, 2009, pp. 111:1–111:10. SIGGRAPH Asia '09. ACM. [AQ15]
- Vanegas CA, Garcia-Dorado I, Aliaga DG, et al. (2012) Inverse design of urban procedural models. *ACM Transactions on Graphics* 31(6): 168:1–168:11.
- Watson B, Müller P, Vervovka O, et al. (2008) Procedural urban modeling in practice. *IEEE Computer Graphics and Applications* 28(3): 18–26.
- Watson D and Adams M (2010) *Design for Flooding: Architecture, Landscape, and Urban Design for Resilience to Climate Change*. ■: John Wiley & Sons. [AQ16]
- Weber B, Müller P, Wonka P, et al. (2009) Interactive geometric simulation of 4D cities. *Computer Graphics Forum* 28(2): 481–492.
- White I (2008) The absorbent city: Urban form and flood risk management. *Proceedings of the Institution of Civil Engineers – Urban Design and Planning* 161(4): 151–161.
- Whiting E, Ochsendorf J and Durand F (2009) Procedural modeling of structurally-sound masonry buildings. In: *ACM SIGGRAPH Asia 2009 Papers*, New York, NY, USA, 2009, pp. 112:1–112:9. SIGGRAPH Asia '09. ACM. [AQ17]
- Yang Y-L, Wang J, Vouga E, et al. (2013) Urban pattern: Layout design by hierarchical domain splitting. *ACM Transactions on Graphics* 32(6): 181:1–181:12.

Ahmed Mustafa ■ [AQ18]

Xiao Wei Zhang ■

Daniel G Aliaga ■

Martin Bruwier ■

Gen Nishida ■

Benjamin Dewals ■

Sébastien Erpicum ■

Pierre Archambeau ■

Michel Piroton ■

Jacques Teller ■