Procedural Urban Forestry

TILL NIESE, University of Konstanz, Germany
SÖREN PIRK, Google Research, USA
MATTHIAS ALBRECHT, University of Konstanz, Germany
BEDRICH BENES, Purdue University, USA
OLIVER DEUSSEN, University of Konstanz, Germany

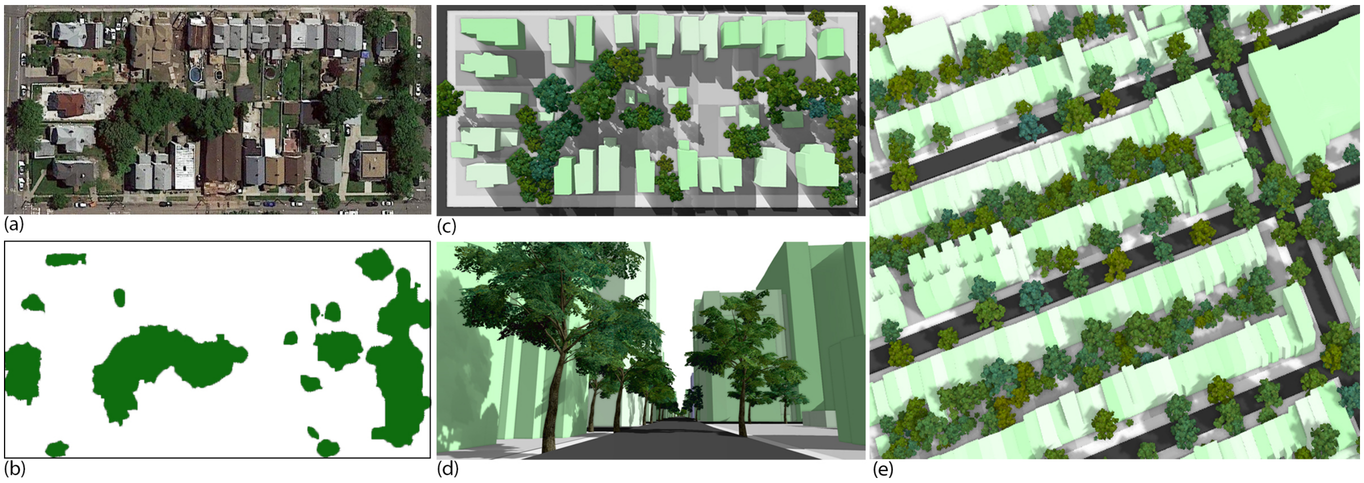


Fig. 1. Steps of our learning-based plant population method: we use satellite images (a) and predict coverage maps for vegetation (b). We use these maps to identify regions for placing plants (c) and to learn the parameters for our procedural models when populating new virtual cities with complex plants (d), which significantly increases the realism of urban landscapes (e).

The placement of vegetation plays a central role in the realism of virtual scenes. We introduce procedural placement models (PPMs) for vegetation in urban layouts. PPMs are environmentally sensitive to city geometry and allow identifying plausible plant positions based on structural and functional zones in an urban layout. PPMs can either be directly used by defining their parameters or learned from satellite images and land register data. This allows us to populate urban landscapes with complex 3D vegetation and enhance existing approaches for generating urban landscapes. Our framework's effectiveness is shown through examples of large-scale city scenes and close-ups of individually grown tree models. We validate the results generated with our framework with a perceptual user study and its usability based on urban scene design sessions with expert users.

CCS Concepts: • **Computing methodologies → Shape analysis**; • **Theory of computation →** *Grammars and context-free languages*; Rewrite systems.

Authors' addresses: Till Niese, University of Konstanz, Germany; Sören Pirk, Google Research, USA; Matthias Albrecht, University of Konstanz, Germany; Bedrich Benes, Purdue University, USA; Oliver Deussen, University of Konstanz, Germany.

Additional Key Words and Phrases: Urban Models, Vegetation, Procedural Generation, Urban Forestry

ACM Reference Format:
Till Niese, Sören Pirk, Matthias Albrecht, Bedrich Benes, and Oliver Deussen. 2021. Procedural Urban Forestry. *ACM Trans. Graph.* 1, 1 (November 2021), 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The visual simulation of urban models and the generation of their 3D geometries are fundamental open problems in computer graphics that have been addressed by many approaches. Existing methods range from modeling façades, buildings, city block subdivisions, to entire cities with viable street and road systems. Synthetically generated city models already exhibit a high degree of realism. However, cities are immersed in vegetation, but only very little attention was dedicated to the interplay of urban models and vegetation in computer graphics. Many approaches have considered ecosystem simulations. The prevailing algorithms use plant competition for resources as the main driving factor of their evolution either on the level of entire plants [Deussen et al. 1998] or on the level of branches [Makowski et al. 2019]. Unfortunately, these approaches fail in urban areas because urban trees have only limited space to compete for resources. They are heavily affected by surrounding urban areas and human intervention.

The term urban forest refers to vegetation in urban areas [Miller et al. 2015]. Vegetation has many practical functions: it controls air movement, solar radiation, heat, humidity, and precipitation. It can also block snow and diminish noise. Moreover, an essential function of vegetation is to increase city aesthetics. Urban forests are not planted at once but managed over time. Dead trees are removed, and new trees are planted. Living trees are pruned for visibility or utility services. In contrast to real cities, we face a different situation in computer graphics. An existing algorithm generates a city model without vegetation, and we need to find suitable locations for individual trees. Simulating urban forest evolution, *i.e.,* by using the algorithm by Benes et al. [2011], is time-consuming and challenging to control.

We introduce a procedural method for the advanced placement of vegetation to increase urban models' overall realism. We are inspired by urban rules that control which trees and bushes can be planted and how tall they can grow. These rules vary for individual areas; they are relaxed in industrial zones. People also have more flexibility in their properties, but they are enforced in public zones of a city and around important landmarks. Therefore, we introduce procedural placement models – strategies for generating plant positions – along with parameters to enable an automatic placement of vegetation, faithful to the characteristic features of plant distributions within the different municipality zones of a city. We show that placement models and parameters together provide an efficient means of controlling urban landscapes' interactive modeling.

Moreover, we can populate city models with static tree geometry and dynamic models of plants that can grow and change their shape in response to environmental changes or human intervention. This allows us to apply simulation models that describe how a city or its areas would change if more or less effort could be spent on maintaining them. Such dynamic urban ecosystems allow users to visually predict and control gardening effects in a city and make such models more realistic since they inhibit decay and different order levels.

While procedural placements can be used directly to populate urban layouts, we also show that placement models can be used to learn plant distributions of real cities. We use satellite images and land register data to train deep neural networks to learn trees and other plants' distributions in our procedural placement models' parameter space. While placement models act as a strong prior to regularize finding plausible placements, learning parameter values also enable users to efficiently author scenes through intuitive parameters. The example in Fig. 1 shows a satellite image (a) and the predicted coverage map (b). We use coverage maps to identify areas where to place vegetation (c) and learn the procedural models' parameters. Once the parameters are obtained, we can automatically populate city models with complex models of plants (d) to increase their realism (e).

Our main contributions are: (1) we advance the state-of-the-art in modeling vegetation in urban landscapes by introducing a procedural modeling framework that is based on the idea to factorize the complexity of plant placement into manageable components; (2) we introduce a set of procedural placement models along with their parameterization to capture a large variety of placement patterns; (3) we use a novel pipeline for learning plant distributions in cities from satellite data; we convert satellite images into coverage maps and then learn the placement parameters of our procedural models.

2 RELATED WORK

Only recently, researchers started exploring approaches to model virtual environments with realistic traits of real urban landscapes [Smelik et al. 2014]. Here, we focus on the problematic aspects of plant and urban modeling, ecosystems, and learning-based methods.

Urban Modeling: urban structures are often modeled proceduraly [Watson et al. 2008]. In their seminal paper, Parish and Müller [2001] used L-systems to model complex cities, and Wonka et al. [2003] applied split grammars to procedurally define buildings that were later extended by using subdivision [Müller et al. 2006] and by more advanced operations [Schwarz and Müller 2015]. Purely procedural models of infinite cities were introduced by Merrell and Manocha [2008; 2011], the procedural modeling of street layouts has been described by using vector fields [Chen et al. 2008]. Similarly, procedural approaches have been successfully applied to modeling façades [Müller et al. 2007]. Urban modeling has been combined with urban simulation to generate viable cities [Vanegas et al. 2009, 2010b], and city growth [Weber et al. 2009]. However, most of the related work focuses solely on urban structures and considers vegetation only as a decorative add-on.

Inverse Procedural Modeling: our approach is related to inverse procedural models, in that it learns plant placement from real cities and attempts to transfer it to synthetic ones by fitting parameters of a procedural model. An inverse procedural model for façades has been introduced by AlHalawani et al. [2013] and Wu et al. [2014]. Variations from a procedurally encoded single layout can be generated by the work of Bao et al. [2013], the layered nature of façades has been used for inverse procedural modeling in [Ilčík et al. 2015; Li et al. 2011b], exploiting structural symmetries was done in [Dang et al. 2014; Zhang et al. 2013]. Interactive alterations of shape grammars were utilized in [Dang et al. 2015]. Buildings can be encoded as L-systems by using the inverse procedural approach from [Vanegas et al. 2010a], modeled by using a procedural connection of structures [Bokeloh et al. 2010], or through binary integer programs [Kelly et al. 2017]. Finding the procedural models' parameters from existing data was investigated by Talton et al. [2011]. They used expressions of L-system strings of modules to fit a generated structure to an input. Ritchie et al. [2015] attempt to control procedural programs and procedural models using stochastic Monte Carlo methods. Structural patterns can be encoded by using the approach of Yeh et al. [2013] or encoded as L-systems by the work of Šťava et al. [2010]. Recently, trained deep neural networks have been combined with inverse procedural modeling to allow for the interactive design of buildings by using sketches [Nishida et al. 2016], to find urban models from real world images [Zeng et al. 2018], and for large-scale reconstruction [Kelly et al. 2017]. Inverse procedural modeling has also been used to generate entire urban layouts in [Martinovic and Van Gool 2013; Vanegas et al. 2012]. Inverse procedural models primarily deal with regular structures (facades, buildings, cities), and only a few focus on stochastic problems (trees, distributions). Our approach is inspired by previous works in that it attempts to define an inverse procedural model (urban forest) and finds its parameters. This, in effect, is used to augment an input urban model with vegetation.

Plant Modeling: research has long focused on defining plausible branching structures based on fractals [Aono and Kunii 1984; Oppenheimer 1986] or L-Systems [Lindenmayer 1968; Prusinkiewicz 1986]. Other methods focus on rule-based modeling [Lintermann and Deussen 1999], inverse procedural modeling of trees [Stava et al. 2010, 2014], and finding L-system for branching structures [Guo et al. 2020]. Moreover, sketch-based modeling techniques allow artists to produce plant models interactively and in more nuanced ways [Ijiri et al. 2006; Okabe et al. 2007; Wither et al. 2009]. Alternative approaches attempt to reconstruct plant models automatically either from images [Li et al. 2021; Tan et al. 2008, 2007], videos [Li et al. 2011a], or scanned 3D point clouds [Livny et al. 2011; Xie et al. 2016]. Only just recently, several approaches also focus on the dynamic and realistic behavior of plant models, including growth [Longay et al. 2012; Pirk et al. 2012a], the interaction with wind or fire [Pirk et al. 2017, 2014], or as established through realistic materials [Wang et al. 2017; Zhao and Barbič 2013].

Modeling the plants' response to its environment is of utmost importance to obtain realistic branching structures when positioned in groups or alongside obstacles [Měch and Prusinkiewicz 1996]. Approaches exist to model this phenomenon by considering the self-organization of plants [Palubicki et al. 2009; Runions et al. 2007], through explicitly modeling the plasticity of branches [Pirk et al. 2012b] or through the dynamic adaptation to support structures, as can be observed for climbing plants [Benes and Millán 2002; Hädrich et al. 2017]. The growth, decay, and pruning of buds and branches play an essential role in plant development [de Reffye et al. 1988]; a phenomenon that is often parameterized in procedural models to develop convincing branching structures [Stava et al. 2014]. In our approach, once the location of the tree has been established, we grow the trees in the given location and adapt their shape by laws of competition for resources.

Ecosystems: various works focus on ecosystem simulation. The seminal paper of Deussen et al. [1998] introduced a competition for resources on the plant level, and this approach has been recently extended towards the competition of individual trees in ecosystems [Makowski et al. 2019]. Various techniques attempt to simulate ecosystems considering different phenomena, such as erosion [Cordonnier et al. 2017] or wildfires [Hädrich et al. 2021], and even by locally learning plant distributions and using them as interactive brushes [Emilien et al. 2015; Gain et al. 2017]. Closely related to our approach is the work of Benes et al. [2011] that models urban ecosystems by combining wild ecosystem growth from [Deussen et al. 1998] with controlled plant management. However, contrary to our work, the initial plant placement is purely ad hoc, and their approach does not allow for procedural plant placement that could be connected with real cities. Our approach defines the procedural models and learns their distributions to populate an empty urban layout. Moreover, their approach is a simulation that seeds new trees and eliminates others by competition for resources over time. Our approach populates the entire city at once.

Learning-based Approaches: some works have started to explore the capabilities of learning-based methods for scene generation and object placement. While neural networks have shown paramount performance on image classification, synthesis [Khan et al. 2019; Wu et al. 2017], or inverse texture modeling [Guehl et al. 2020; Hu et al. 2019] tasks, properly placing objects into meaningful configurations is still a challenging problem. For arranging scenes, methods need to coherently generate plausible and continuous poses (translation and orientation) of objects and to one another. However, most neural network architectures only allow operating on fix-sized in- and outputs, which makes placing arbitrary numbers of objects challenging. To this end, a number of approaches introduce convolutional neural networks for scene generation [Li et al. 2019; Ritchie et al. 2018; Wang et al. 2019] and Zeng et al. [2018] learn to reconstruct buildings by learning parameters of a procedural model. For outdoor scenes, Guerin et al. [2017] and Kelly et al. [2018] use generative adversarial networks to author textures for terrain and building details. While these methods only tangentially relate to our work, they show the capabilities of neural networks for scene generation. We also combine the advantages of image-based learning techniques with procedural modeling as we aim at learning the parameters of procedural models with neural networks that place plants realistically.

3 OVERVIEW

Generating plausible vegetation models for virtual urban landscapes faces two significant challenges: first, plant placement varies across different functional and demographic zones (Fig. 2a)– an industrial zone may only have a small number of non-managed plants, while residential areas not only have regularly placed trees alongside roads but also in gardens and parks. The planting rules depend on culture, habits, city rules, etc. They are difficult to quantify. Second, plant models need to simulate growth and interaction with their environment to generate vegetation with high visual fidelity. Moreover, urban trees are often pruned or may lack resources (water or light), which hinder their growth and affect their structure.

To address these challenges, we propose a two-stage procedural modeling pipeline. First (Fig. 2), we introduce PPMs (b) to generate plausible plant positions based on placement strategies and known planting rules for vegetation. A PPM can be defined for each functional or demographic zone of a city (*e.g.,* residential, commercial, or industrial) and operates on single lots of land (realty). Each PPM has a different set of rules parameterized by structural and positional parameters to capture the various kinds of planting patterns found in real cities. Second, once the plant positions are generated, we use a state-of-the-art developmental model (Fig. 2, e) for growing plants. Given the plant's location and environment, the growth process generates unique and realistic branching structures.

Finally, we have developed a novel learning-based pipeline for populating models of real cities with vegetation. First, we convert satellite images of urban landscapes to *vegetation coverage maps* by using a style-transfer network (Fig. 13, b). The coverage maps represent areas that are covered with above-ground vegetation. Second, we learn a mapping from the coverage maps to the parameters of our PPMs (Fig. 13, d). Given our pipeline and the parameter values obtained from real satellite images, we can generate vegetation similar to what can be observed in the satellite images.

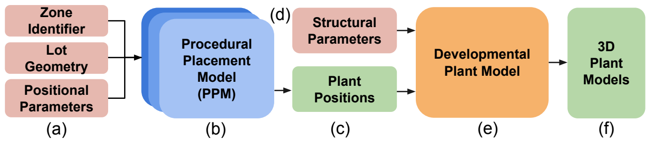


Fig. 2. To place vegetation in urban environments we propose procedural placement models (b) that implement placement strategies for vegetation based on the geometry of individual lots, positional parameters, and a zone identifier (a). After plant positions (c) have been generated we use a developmental model (e) along with structural parameters (d) to jointly grow plants, which results in realistic 3D plant models.

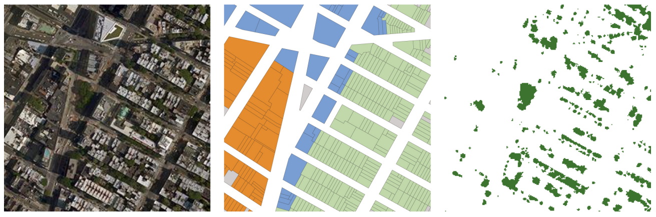


Fig. 3. Urban layout: satellite images (left), zone data for individual lots (middle), and coverage maps (right) are available in public datasets. We use zone data and lot geometry as inputs to our procedural models and learn to predict their parameter values from the coverage maps.

4 PLANTING RULES

Road networks define landscapes as administrative or functional zones [Waddell et al. 2007], and they can be further classified into rural, exurban, suburban, and urban areas [Miller et al. 2015]. All the involved plants form the urban forest, an umbrella term referring to trees, shrubs, and bushes found in urban and suburban areas.

A common way of introducing vegetation into an urban forest is by replacing a dead tree. Only newly created developments have large areas directly populated by vegetation. When a new neighborhood is built, a city will plant regularly spaced trees and bushes parallel to roads and sidewalks by applying municipal tree ordinances [Grey 1995] (see also [Miller et al. 2015, pg 254]). The neighborhood is subdivided into blocks and blocks into individual lots left to the owners to plant the vegetation as needed. Typically, the city only defines specific planting rules such as the distance between individual trees should depend on the tree height, or the distance is derived from the soil the tree requires to survive [Endreny 2018]. Trees should not obstruct views at intersections. They should have a certain distance from the curb and sidewalks [Bloniarz and Ryan 1993]. Vegetation must not block house entrances for emergency purposes. These functional restrictions are also combined with aesthetic constraints: vegetation should not be planted in the proximity of windows [Miller et al. 2015]. Most of these rules are incorporated into a so-called building activity area (or building envelope) that is an extension of the building's 2D projection by about 600cm perpendicularly from each building wall and 150cm from each driveway.

At a higher level, we aim to generate vegetation for the various types of zones procedurally. Therefore, we assume that each urban layout, either real or synthetically generated, can be divided into such zones. Specifically, we use a zonal layout commonly used in urban planning [Waddell 2002; Waddell et al. 2007] and urban simulations [Vanegas et al. 2009, 2010b; Weber et al. 2009] and divide an urban layout into five zones: 1) *residential* includes houses and buildings where people live, 2) *commercial* consists of businesses such as department stores, malls, and small stores, 3) *industrial* zones include factories and other production services, 4) *street* zones, which describe areas next to roads. We add a category (5) *other* that includes parks, non-managed areas, areas close to railroads, unassigned regions, etc.

As shown in Fig. 3, we further assume that a city layout is organized as individual lots, where each lot represents a property that may be occupied by a building. Given a lot and its zone type, we then define a PPM that places vegetation individually into each lot.

Based on the observations from municipal tree ordinances [Grey 1995; Miller et al. 2015] and previous work on urban forests [Benes et al. 2011], we define six tree planting rules and show them in different real-world images in Sec. 5.1. *Semi-Random* placement within a lot follows a Poisson-disc distribution preventing trees from being in close proximity. Trees are often planted along lot *boundaries* as a noise barrier, but can also be *clustered* forming areas with grass and shade. Along streets trees often serve as a barrier and are planted in an *equidistant* manner along the medial axis of a lot. We can also observe *a single* tree within a lot or a *regular* placement.

In addition, trees are rarely planted at once and their distributions are mostly an emergent phenomenon of growth over long periods of time. Our objective is to populate an empty urban model at once. Thus we define the planting rules as geometrical distributions that allow us to encode plant populations as procedural models, as shown in the next section.

5 PROCEDURAL URBAN VEGETATION

Vegetation for an urban landscape is generated in two steps: first, we apply a PPM to seed plants individually for each zone according to their functional types. After the virtual plants have been planted, we use a developmental model that dynamically grows them in their locations while interacting with the surrounding environment. This allows plant adaptation to their environment, such as bending and shedding of branches due to the competition for resources, resulting in vegetation with high visual fidelity.

5.1 Procedural Placement Models - PPMs

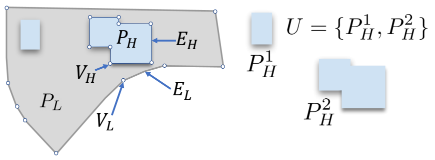
We seek to model plant morphology and the variance of plant placement across different municipality zones to distribute vegetation in an urban landscape realistically. Defining and parameterizing rules for obtaining plausible plant positions, while adhering to urban features such as buildings and streets, is intractable. Therefore, we factorize the problem into specifying placement models for the different zones as denoted in Sect. 4 (industrial, commercial, residential, street, and other) and for each lot.

The factorization allows us to define a manageable parameterization along with placement strategies for the different zones. Each placement model defines a concise strategy to place vegetation into a single lot. For example, we have models to place vegetation randomly, along the edges of a lot, equidistantly, etc. Moreover, we define the PPMs in a context-sensitive way. This means to maintain a global appearance, a PPM can query adjacent lots to adjust its parameters (*e.g.*, the distance between trees alongside a road in one lot should be the same in the neighboring lot). A PPM is a tuple

$$\mathcal{M} = \langle \mathcal{S}_g, \mathcal{P}_p, \mathcal{P}_s \rangle, \tag{1}$$

where $\mathcal{S}_g$ is a function implementing a *placement strategy* (rules) with $g \in \{R, B, C, E, S, I\}$ (see Sect. 5.2 and Tab. 2), $\mathcal{P}_p$ is a set of *positional* parameters to define the placement of plants, and $\mathcal{P}_s$ is a set of *structural* parameters for the morphological appearance of vegetation within the lot.

Lots and buildings are defined as 2D polygons possibly concave and with holes (see Fig on the right): $P_L = \{V_L, E_L\}$, $P_H = \{V_H, E_H\}$, where V_L and V_H denote the vertices of a lot (L) and buildings (H) and E_L and E_H the edges of the polygon for lot and building, respectively. A lot can include multiple buildings (or other structures): $U = \{P_H^i\}$. The polygon $P = P_L - \cup P_b^i, \forall P_b^i \in U$, defines the area of a lot that can be covered by vegetation; the PPM only places vegetation within the geometric shape of the polygon P. A set of plant positions for a single lot is then generated as



$$\mathcal{X} = \mathcal{S}_g(\mathcal{V}_p, \mathcal{V}_s, P, Z, \mathcal{K}), \tag{2}$$

where $\mathcal{V}_p$ and $\mathcal{V}_s$ denote the parameter values for positional $\mathcal{P}_p$ and structural $\mathcal{P}_s$ parameters, P is the polygon of a single lot, Z is a zone identifier, and $\mathcal{K}$ is the context of a lot. We use Z to select parameter values for each lot. For example, a residential and a commercial lot may use the same strategy (*e.g.*, boundary) but differ in their parameter values (*e.g.*, different species are used). This is illustrated in Fig. 4. Generating vegetation with the same value for Z produces a uniform appearance (the same settings are used for every lot), while varying Z with the functional zones generates a diverse yet coherent appearance. Put differently, Z allows us to control the placement of vegetation on a global scale. Finally, we use $\mathcal{K}$ to modify the input parameters according to the neighbors of a lot to allow for consistent global appearance as detailed in Sect. 5.5.

To summarize: a PPM defines a placement strategy and structural and positional parameters for populating single lots. Varying these parameters' values generates different plant positions within the constraints of the strategy at a local scale while changing the parameters jointly – *e.g.*, based on zoning types – allows us to vary vegetation at a more global scale.

5.2 Placement Strategies

A placement strategy $g \in \{R, B, C, E, S, I\}$ (Semi-Random, Boundary, Cluster, Equidistant, Single, and Individual tree) defines rules for placing the plants and how the parameters are used.

To implement the different placement strategies, we compute active areas within each lot that define where the vegetation can be placed. For the strategies *semi-random* and *single* the entire lot polygon P_L is used, while for the strategies *boundary* and *cluster* we define active areas within the polygon; *i.e.*, we define a boundary along the edge of the polygon towards its center for *boundary* and

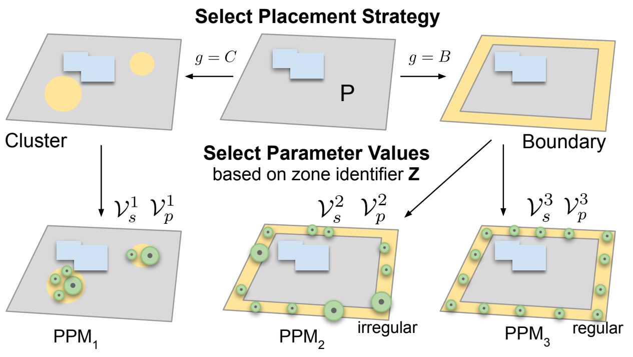

Select Placement Strategy
$g = C$
$g = B$
P
Cluster
Boundary
Select Parameter Values
based on zone identifier **Z**
$\mathcal{V}_s^1$ $\mathcal{V}_p^1$
$\mathcal{V}_s^2$ $\mathcal{V}_p^2$
$\mathcal{V}_s^3$ $\mathcal{V}_p^3$
PPM$_1$
PPM$_2$
irregular
PPM$_3$
regular

Fig. 4. Given a lot, we use a placement strategy to define the placement of vegetation. The zone identifier Z is used to select parameter values for structural $\mathcal{V}_s$ and positional parameters $\mathcal{V}_p$. Together, strategies and parameters allow us to generate vegetation with globally similar appearance depending on the municipality zones within a city.

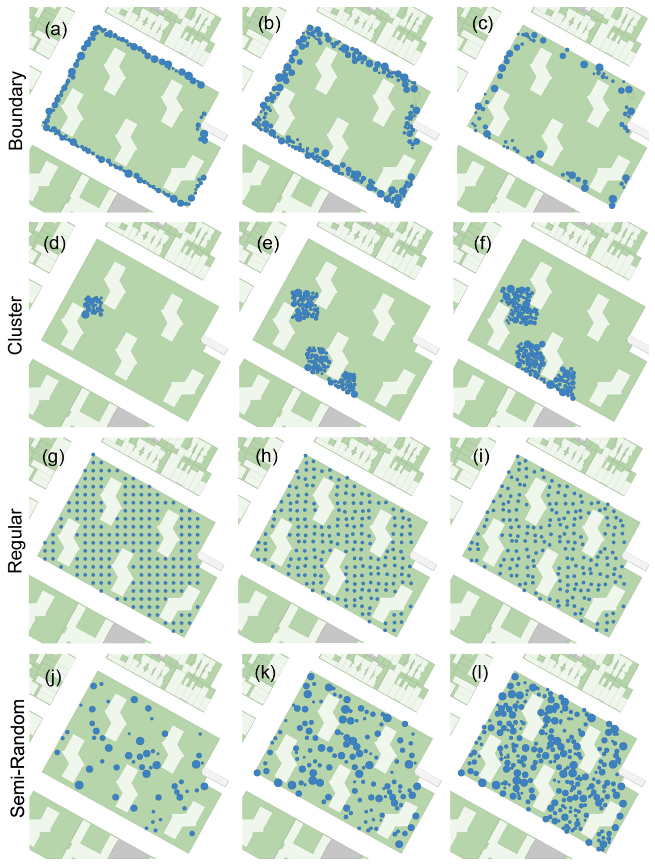


Fig. 5. Variations of positional parameters on a single lot with different placement strategies. (a)-(c): strategy *boundary* with narrow (a) and wide (b) boundary size, and less density (c). (d)-(f): strategy *cluster* with a single cluster (d) and multiple clusters (e) of different sizes (f). (g)-(i): strategy *regular* with no (g), medium (h), and high (i) jitter. (j)-(l): strategy *semi-random* with low (j), medium (k), and high (l) density.

a circular area around a randomly selected point within the polygon for *cluster*. For *equidistant*, we compute the medial axis of the polygon and then generate equidistant plant positions along the axis. The strategy *single* defines a single plant's random placement within the entire lot. Finally, for *regular* we compute a lot-aligned lattice and place plants at the center of each cell. Fig. 5 shows four of our six placement strategies and their parameter variations.

5.3 Positional Parameters

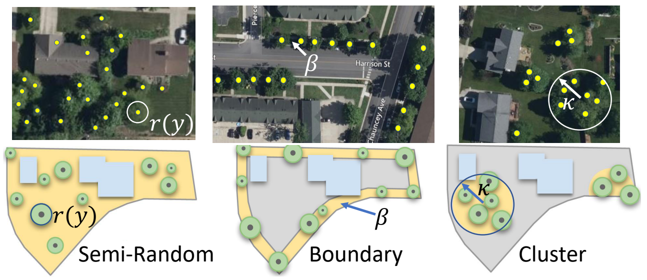
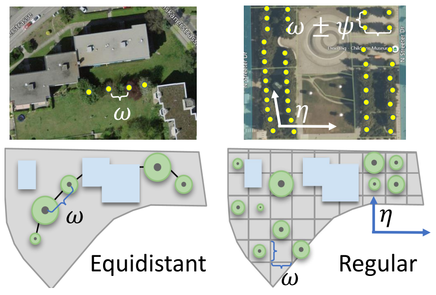


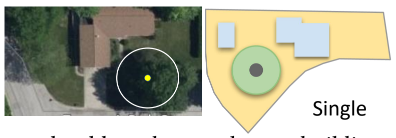
Fig. 6. Semi-Random, Boundary, and Cluster placement strategies use Variable Radii Poisson-Disk Sampling to position trees.

Semi-Random, Boundary, and Cluster: placement strategies are parameterized by the positional parameters shown in Tab. 1. We use the Variable Radii Poisson-Disk Sampling [Mitchell et al. 2012] to generate plant positions within active areas of a lot (see Fig. 6). More specifically, we are interested in generating a set of points $\mathcal{X}$ with spatially varying point density. A new position sample y is assigned a radius $r(y) : \Omega \longrightarrow \mathcal{N}(\mu, \sigma)$, where $\mathcal{N}$ denotes a normal distribution with mean μ and variance σ. The new position sample y is accepted and added to the set if $|y - x| \geq r(x) + r(y) \forall x \in \mathcal{X}$.

For the *boundary* placement strategy we define the boundary size as parameter β that defines an area along the normal of the edge of a polygon towards its center. To implement the *cluster* strategy, we randomly sample points in a lot and define the cluster area as a circle with a radius κ. A lot can have a variable number of clusters with the maximum number defined by π. For both strategies, *boundary* and *cluster*, we first compute the active regions (boundary, cluster circles) before generating sample positions.



Regular and Equidistant: allow for semi-regular vegetation placement. For the *regular* strategy, we compute a regular lattice based on the bounding box of a lot and define the size of cells with ω and their orientation with η. We optionally jitter the positions using ψ within each cell. To implement the *equidistant* strategy, we first compute the medial axis of the lot polygon P_L [Choi et al. 1997] and then equidistantly place plants along the axis based on the distance parameter δ. We model the *density of vegetation* for all placement strategies by defining the parameter τ, which deactivates position samples in $\mathcal{X}$. A value of $\tau = 1$ activates all samples, while a value of $\tau \leq 1$ randomly deactivates them until all samples are deactivated ($\tau = 0$). Finally, we define the radius ξ for the context $\mathcal{K}$ of a lot. The context is defined as the adjacent lots, and we use it to model context-sensitivity (see Sec. 5.5).



Single: Finally, we sample one random position in the lot for the strategy *single* (see right).

Building Envelope: Trees should not be too close to buildings and should not obstruct doors and windows. We adopted the concept of building envelopes [Miller et al. 2015] that defines the clearance distances from the buildings. Moreover, we extend the envelope in front of doors and windows to avoid their blockage (see Fig. 7).

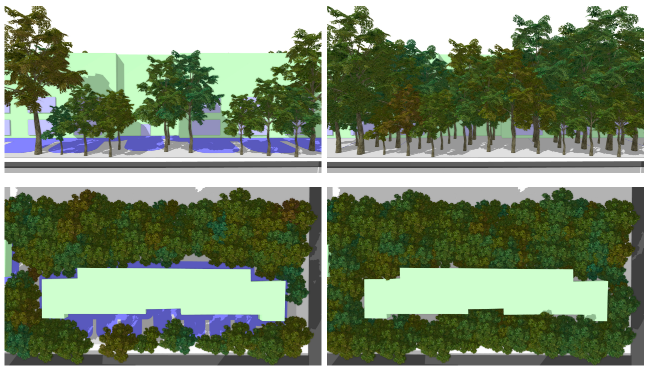


Fig. 7. Left: the building envelope (blue) defines a zone where plants cannot be planted to avoid proximity to walls and blockage of door and windows. Right: plant placement without considering the building envelope.

Tab. 1 summarizes the positional parameters along with their ranges, and Tab. 2 shows the placement strategies and their corresponding positional parameters. Examples of changing the values of positional parameters are shown in Fig. 5.

5.4 Structural Parameters

We define *structural parameters* to model the morphology of individual trees as well as the plant population within a lot. Based on the computed plant positions we define a plant seed as the tuple

$$\mathcal{T} = \langle p, \alpha, \phi, \gamma \rangle, \tag{3}$$

where $p \in \mathcal{X}$ is the plant position, α its maximum age, ϕ denotes a species identifier, and γ is a pruning factor. To generate branching structures we grow a plant with a developmental model (see Sec. 5.6) and jointly simulate its growth with all other plants in a lot.

We define several species ($n = 10$) for the whole urban landscape by selecting parameter values for our developmental model [Palubicki et al. 2009]. We then use the species identifier ϕ to associate one of the species to a seed. We further control this selection by using the parameter ρ, which defines the tree vs. shrub ratio in a lot. A value of $\rho = 1$ assigns all seeds tall-growing species, while a value of $\rho = 0$ only associates short growing ones.

Table 1. Positional and Structural Parameters for PPMs

Parameters		Meaning	Range/Dimensions
Positional	μ	Plant envelope mean	[1m - 10m]
	σ	Plant envelope variance	[0.1 - 2]
	τ	Vegetation density	[0-1]
	β	Boundary size	[0m - 5m]
	κ	Cluster radius	[1m - 20m]
	π	Max number clusters	[0 - 5]
	ω	Regularity grid size	[5m - 50m]
	ψ	Regularity jitter	[0 - 1]
	η	Regularity orientation	[0 - 180°]
	δ	Equidistant spacing	[0m - 10m]
	ξ	Radius of context	[0m - 300m]
Structural	α	Max plant age	[0 - 100 years]
	ρ	Tree vs shrub ratio	[0 - 1]
	θ	Species diversity	[0 - 1]
	γ	Pruning factor	[0 - 1]
	λ	Num. species	[1 - 10]

Table 2. Placement Strategies and used Positional Parameters.

Strategy	Symbol	μ	σ	τ	β	κ	π	ω	ψ	η	δ	ξ
Semi-Random	R	✓	✓	✓								✓
Boundary	B	✓	✓	✓	✓							✓
Cluster	C	✓	✓	✓		✓	✓					✓
Equidistant	E	✓	✓	✓							✓	✓
Single	S	✓	✓	✓								✓
Regular	I	✓	✓	✓				✓	✓	✓		✓

To vary the number of used species in a lot, we use the parameter θ. We randomly select one of the species as the dominant species in a lot and use θ as a ratio to control the number of seeds associated with the dominant species and all other available species. A value of $\theta = 0.5$ sets half of the available seeds to the dominant species and the other half with randomly selected ones.

Finally, we may prune a plant by a bounding volume for the tree crown of a fully developed model. This allows us to generate a more organized appearance of vegetation, *e.g.,* along avenues or highways. Branches that reach out of the volume are cut off. We scale this volume by γ; a value of $\gamma = 1$ will leave a plant unpruned, while a value of $\gamma \leq 1$ scales the bounding volume and therefore results in a pruned plant. After pruning, we again simulate the plant growth to develop smaller branches and leaves. Fig. 11 shows an example of the pruning of trees; other variations of structural parameters are shown in Fig. 8.

5.5 Context-Sensitive Rules

So far, lots have been treated as individual units without any mutual relationship. However, each lot has its context that are its surrounding roads and neighboring lots. The neighbors often share similar planting rules provided by the applying municipal tree ordinances [Grey 1995; Miller et al. 2015]. To account for the context of lots we want to adjust planting rules.

Let us recall that each PPM from Eqn. (1) has associated a placement strategy $\mathcal{S}_g$ and two sets of parameters $\mathcal{P}_p$ and $\mathcal{P}_s$. Each lot has a set of parameter values from Eqn. (2) $\mathcal{V}_p$ and $\mathcal{V}_s$. Moreover, it considers the context (*i.e.,* the neighborhood) $\mathcal{K}$ of the lot that is being populated with plant positions Eqn. (2). Further, let us denote a particular lot L and its parameter values as $\mathcal{V}^L$. In the following text, we will omit the lower index s and p because the parameters are calculated in the same way. The context is the set of lots within radius ξ centered on the lot L and weighted by a 2D Gaussian. The

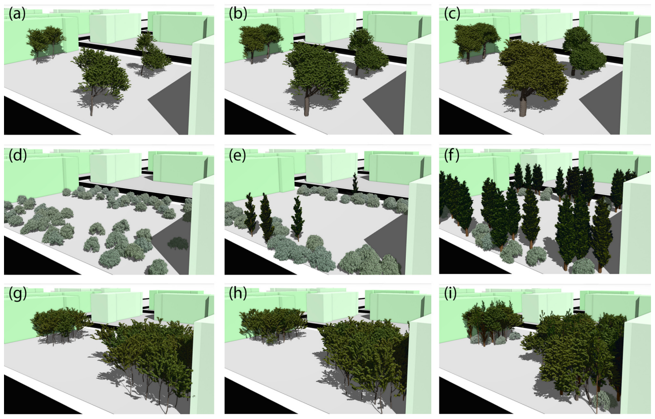


Fig. 8. Variations of structural parameters. Top row: variations of age parameter from young (left) to old (right). Middle row: changes of tree to shrub ratio from only shrubs, to mostly trees. Bottom row: variations of species diversity from a single species (left) to multiple species (right).

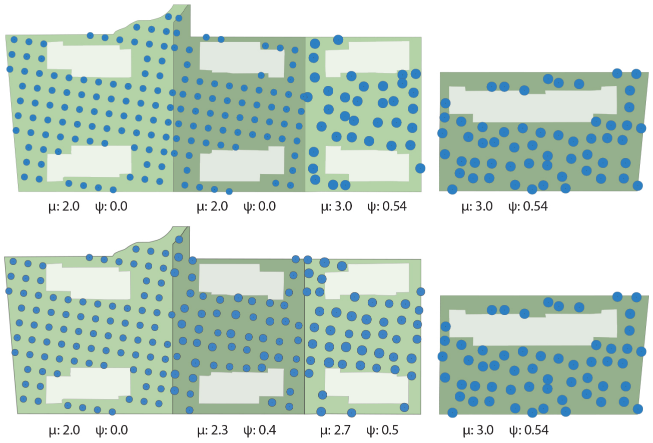


Fig. 9. Context-sensitivity: we calibrate the parameter values of a lot with those of adjacent lots (context). Here we show two lot configurations with regular placement strategy and variations over the parameters μ and σ. For the lots shown in the top row context-sensitivity is turned off and plant placement changes abruptly from one lot to another, while for the bottom row we show context-sensitivity across lots and the resulting calibration of parameters (context radius: $\xi = 180m$).

values of the corresponding parameters (see Tab. 1) of the neighbors and the lot L are weighted according to the distance resulting in a context-updated parameter set $\tilde{\mathcal{V}}^L$ as:

$$\tilde{\mathcal{V}}^L = \sum_{\forall \mathcal{V}^K \in \mathcal{K}} w\left(d(L, L^K)\right) \mathcal{V}^K, \tag{4}$$

where $w\left(d(L, L^K)\right)$ is the Gaussian-weighted distance between the center of the lot L and L^K within the investigated context, and $\mathcal{V}^K$ are the values of the parameters of the lot L^K. The updated parameter values $\tilde{\mathcal{V}}^L$ are then used for the PPM.

Note that this process can be considered as a diffusion of the parameters within radius ξ. Fig. 9 shows the effect of using context-sensitivity on a regular placement of trees. The first row shows two lots with regular tree placement with an abrupt change to a random placement in neighboring lots that is smoothed out into a semi-random transition when the context is used (bottom row).

5.6 Developmental Plant Model

After generating plant positions, we jointly grow the plants in the computed locations of a single lot. Our developmental model is based on the work of Palubicki et al. [2009]; a tree is a modular system (leaves, buds, stems, and internodes). An internode is a plant stem between two or more leaves, and a tree is composed of a succession of internodes.

The primary plant development is controlled by the expansion of buds that are either apical (terminal) or lateral (axial). Branches expand at their tips by expanding their apical buds or on sides by growing lateral buds. Buds use signaling by the growth hormone Auxin to prevent overgrowth and to control apical dominance [Kebrom 2017]. Secondary plant development (cambial growth) is the thickening of a tree trunk and branches [Kratt et al. 2015] simulated by expanding their radii using da Vinci's rule (see [Minamino and Tateno 2014] for a discussion).

Trees compete for space by seeking light (phototropism) and avoiding collisions and overcrowding. Many different algorithms have been implemented to capture plant competition for resources (see [Měch and Prusinkiewicz 1996; Runions et al. 2007] and [Pirk et al. 2016] for an overview). We use the space occupation approach of [Palubicki et al. 2009; Runions et al. 2007], which controls the growth by randomly scattered particles that attract growing branches. We also simulate phototropism by computing buds' illumination and bending the growth direction towards the brightest spot visible from a bud. Apical control and branching parameters are simulated by using the growth model from [Stava et al. 2010] with the set of parameters.

6 LEARNING VEGETATION PLACEMENT

Learning plant positions directly from image data is a challenging problem that cannot be easily addressed by existing neural network architectures or other methods. To obtain plant positions in an end-to-end manner, a network would have to either output a variable number of plant positions or operate on a fixed size domain, such as an image. The latter requires to obtain plant positions as a post-processing step, which is error-prone. Furthermore, generating ground truth data pairs of satellite images and plant positions (*e.g.,* GPS coordinates) for training a neural network is challenging (see Sect. 8.3 for a discussion). Moreover, an end-to-end deep learning-based system would sacrifice an in-depth understanding of the underlying mechanisms. It would not allow for low-level control that is needed in interactive editing.

Therefore, to recover the placement and appearance of natural urban landscapes, we aim to learn plant distributions in our parameter space of positional parameters. This has the advantage that our above-defined PPMs act as a prior, which helps to regularize the training of our network and, in turn, to generate plausible plant positions. Furthermore, learning the procedural model parameters maps images to comprehensible and intuitive parameters, providing an efficient way to further edit plant placements.

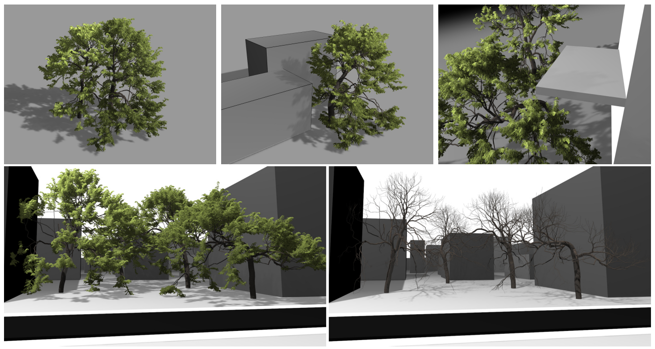


Fig. 10. Top row: trees grown in different environmental conditions. From left to right: two trees close to each other, close to a set of buildings, and underneath a balcony. Bottom row: the growth response of a group of trees in an urban environment generates complex and unique branching structures.

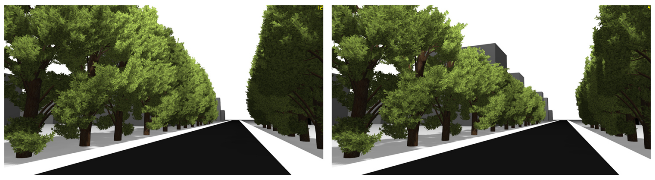


Fig. 11. Branch pruning allows for the adjustment and organization of tree form. Here trees along a street are severely pruned to form a hedge ($\gamma = 0.7$).

6.1 Learning Plant Placements

We use a two-stage neural network pipeline to learn the parameters of our PPMs: first, we translate satellite images to semantic maps that describe vegetation coverage (Fig. 13, a-c). Second, we learn the positional parameters from coverage maps with a lightweight convolutional neural network (Fig. 13, d, e). This pipeline has the advantage that we do not need to rely on pairs of satellite images and positional parameters for training, but instead on pairs of coverage maps and positional parameters, which can be generated synthetically with our PPMs.

To translate satellite images to coverage maps, we used a style-transfer deep neural network [Isola et al. 2016]. A coverage map is a flat-colored image where every pixel color is based on whether the corresponding pixel in a satellite image represents vegetation. Coverage maps have less complex visual traits and are similar to real and synthetic data. Therefore, the network can learn this transfer. We used pairs of satellite images and coverage maps publicly available for some cities [NYCOpenData 2019] to train the style-transfer network to learn coverage maps from satellite images. This allows us to obtain coverage maps of cities for which coverage data does not exist. Fig. 23 (Appx. B) shows examples of training data and generated coverage maps.

We then train a neural network to obtain positional parameter values ($\mu, \sigma, \tau, \beta, \kappa, \pi$, see Tab. 1) from the coverage maps. Training is done on synthetically generated pairs of coverage maps and positional parameters obtained from our PPMs. Specifically, we define the generated coverage maps as $q \in \mathcal{Q}$ for which we know the corresponding positional parameters $\mathcal{P}_p \in \mathcal{U}$. The network can thus be defined as

$$f(q) : \mathcal{Q} \rightarrow \mathcal{U}.$$

To summarize: stage one of our pipeline learns coverage maps from satellite images, which – in stage two – enable us to obtain the positional parameters of our PPMs. Together this allows us to generate vegetation positions for individual lots with similar characteristics as observed in the satellite imagery (*e.g.,* plant distance, density, etc.). Once the parameters are generated, we stencil the coverage map with each lot's geometry and identify areas to place vegetation for a reconstruction. We convert the regions into polygons and then use our *semi-random* placement strategy to generate plant positions within the covered areas of a lot (Fig. 12). Please note that the *semi-random* strategy is regularized by the positional parameters values (Tab. 1) learned by the CNN neural network (see Fig. 13, d, e). As a coverage map defines the areas where vegetation should be placed within a lot, it is sufficient to only rely on a *semi-random* placement here.

6.2 Data and Training

For training the Pix2Pix style-transfer network, we rely on the publicly available implementation of the original model implemented in Python. We train the network on 20K pairs of satellite images and coverage maps. This data is generated in three steps: first, we obtain satellite images from Google maps with a resolution of 256×256 pixels per image. The images correspond to the lowest level of the tile graph. Second, we use the vector data of streets, buildings, and lots from NYCOpenData [2019] and render them into image tiles of resolution 256x256. Third, we generate coverage maps by converting the vegetation coverage data provided by NYCOpenData (total rasterized resolution of 316K x 312K pixels) by reprojecting the data – provided in the geospatial data format: EPSG:2263 - NAD83 / New York Long Island – for each tile to match the Mercator projection used by Google Maps. We use the default hyperparameter settings for Pix2Pix [Isola et al. 2016]; the network converged after training for 200 epochs. We then use the network to convert satellite images of urban landscapes to coverage maps. The geometry of single lots is also obtained from the NYC Open Data. Our urban modeling framework operates on longitudinal and latitudinal coordinates, which allows us to register satellite images, lot data, and coverage maps, enabling us to render satellite images and publicly available maps (*e.g.,* Open Street Maps) in the same framework. Our regression CNN consists of five convolutional layers (32 units) followed by two dense layers (64 units) with relu activations for all except the last layer. We use our PPMs to synthetically generate 21K pairs of (coverage map, positional parameter value)-pairs to train the network. To regress the positional parameters, we use mean squared error as loss function and can achieve 95% accuracy for predicting the validation data's parameters. We use an 80% – 20% split for training and testing data. All results shown in the paper are generated from validation data.

7 IMPLEMENTATION AND RESULTS

Our interactive framework for modeling and rendering urban landscapes was implemented in C++ and OpenGL. All results have been generated on an Intel(R) Core i7-7700K, 8x4.2GHz with 32GB RAM, and an NVIDIA GeForce RTX 2080 GPU with 12 GB RAM.

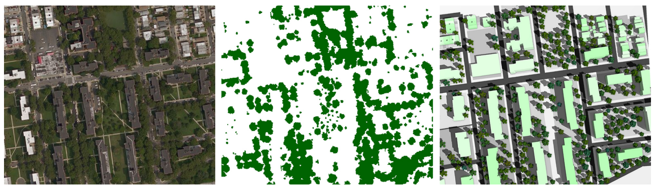


Fig. 12. Vegetation placement based on real data: we use vegetation coverage maps (middle) to identify active regions for individual lots and populate them with our PPMs. This allows us to generate plant distributions (right) similar to what can be observed in satellite images (left).

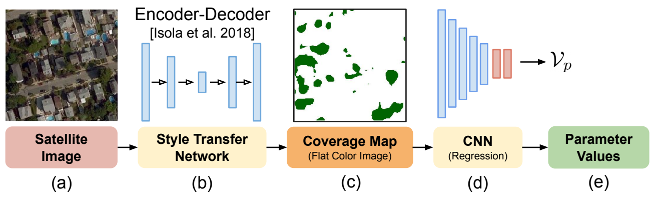


Fig. 13. Neural network pipeline: we use a style-transfer network (b) trained on data pairs from NYCOpenData [2019] to convert satellite images (a) to coverage maps (c). To learn parameter values for our PPMs (for which no ground truth data for satellite images exist) we generate pairs of coverage maps and parameter values with the PPMs of our framework. We then train a CNN (d) to obtain parameter values (e) for the estimated coverage maps of the real satellite images.

The most demanding online task is the generation of tree geometry. We simplify this by representing trees by their skeletons that are generated on the CPU. We further offload the mesh generation of the branch surfaces into a geometry shader on the GPU. Similarly, leaves are generated as textured quads that are also generated on the fly. Buildings and other structures are rendered as extruded outlines. While we cannot render large plant populations in real-time, our framework allows us to explore placement strategies and parameter settings. To render large scenes (*e.g.,* Fig. 22), we use a level-of-detail scheme that successively replaces tree geometry with billboards and point primitives according to the distance from the camera. Appx. A (Tab. 4) shows parameter values for most figures shown in the paper.

7.1 Interactive Authoring

We demonstrated that PPMs can automatically place vegetation into urban landscapes based on the lot data. The geometry of individual lots can either be obtained from publicly available datasets or as a part of the modeling process, for synthetically generated layouts.

However, PPMs operate on polygons, and they were designed with interactive authoring in mind. The user can use a brush tool to draw an area on a map. We then convert the sketch to a polygon and assign a PPM. Depending on its placement strategy, the PPM will then generate plant positions according to the geometry of the polygon and its associated placement strategy (Fig. 14). Furthermore, a user can directly draw the vegetation coverage for individual lots or polygons. Like learning the coverage maps from satellite images, sketching a coverage map replaces the placement strategy for a lot. The PPM then places plants based on the positional and structural

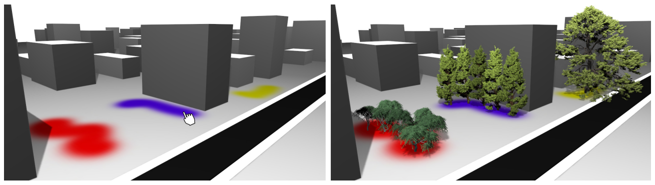


Fig. 14. A user can interactively sketch placement zones with a brush tool (left). Each placement zone is converted to a polygon and assigned a placement strategy to grow plants (right). Here we show the strategies medial axis (blue), single (yellow), and semi-random (red).

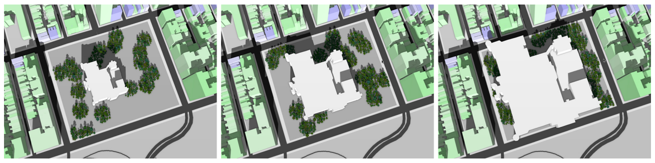


Fig. 15. The placement of vegetation changes with the size of the active areas within a lot. While the used cluster strategy initially generates plants in the entire lot, transitioning to less available space due to a larger building (white) generates more organized plant positions at the boundary of the lot.

parameters, which provides a convenient way for more nuanced vegetation placement.

This process also allows us to generate even more diverse zones if necessary. For example, it is possible to define individual zones for back and front yards, the vegetation along streets, or even parks. Our approach's key idea is to factorize the complexity of defining a complex procedural model into more manageable placement strategies. A PPM only works on a single polygon and generates plant positions for this geometry. This way, it is easy to extend our approach by new placement strategies.

7.2 Results

Figs. 1, 17, and 22 show perspective and top-down renderings of urban landscapes along with the vegetation generated by our framework. For these results, we used coverage maps to reproduce vegetation placement similar to the real scenes. Fig. 16 shows results where we only used our procedural model, without additional coverage maps. For both cases, the produced plant populations show characteristic visual traits found in real vegetation distributions at the city-scale. Based on our placement strategies, we can generate complex urban vegetation patterns in combination with the positional and structural parameters.

Moreover, we show vegetation placements for the different municipality zones (residential, park, commercial) in Fig. 16. Positional parameters allow us to generate planting patterns as commonly found in these areas. At the same time, we can also produce structural variations by selecting the number of species, their height, and their age (Fig. 8). Additionally, we can control the pruning of plants to generate more organized plant shapes (Fig. 11). In an urban setting, buildings often shade larger areas. Trees growing in these regions strive to grow out of the shadow toward the light. This interaction of a tree with other trees and close-by buildings generates complex and unique branching structures. Fig. 10 shows the modeling result of trees grown in varying environmental conditions.

Figs. 14 and 15 show the capabilities of our framework for the interactive authoring of urban landscapes. In Fig. 14, a user drew regions for vegetation onto the ground of an urban layout; each brush tool was assigned a different placement strategy and set of parameter values. Our method then converted the sketched areas to polygons and applied different PPMs. Fig. 15 shows how the placement of vegetation changes when the size of a building on a lot increases. While with a small building, there is more space for random plant configurations, the placement transitions to more organized plant positions when the building's size increases.

Figs. 21 and 22 show vegetation placement results for large scenes generated with our framework. To generate the results in Fig. 21 we manually defined the entire park as a single lot (middle) or generated multiple smaller lots (right). For the result shown in Fig. 22 we rely on the lot data provided by NYCOpen Data [2019] for the vegetation placement. When lot data is provided, our framework enables the efficient generation of realistic urban scenes.

Fig. 17 shows a comparison of using different placement strategies. Given the satellite images of different urban scenes (a), our method is able to closely approximate the real scenes by using coverage maps and PPMs (c). Additionally, we compare our results to different variation as ablation studies. In (d) we place plants randomly without any parameter regularization (*fully random*), but we use the coverage maps to define areas where vegetation can be placed. This setup does not account for the coherent parameterization of plants, which results in less realistic populations of plants. Tree species and their age are not selected consistently and plants are placed too close to buildings and to other plants. In (e) we show the result of our *semi-random* placement strategy that generates random plant positions with regularized structural parameters. The result of placing plants *fully random*, without coverage map and without any regularization across the positional and structural parameters is shown in (f). To validate our results we also manually labeled plant positions and used their longitude and latitude coordinates to render them at their real positions in our framework (b). This allows us to evaluate the visual quality of synthetically generated plant positions compared to real plant distributions.

Finally, Fig. 18 shows the result of generating a scene by carefully fine-tuning the parameters of our PPMs (semi-random, boundary, and cluster strategies) against a real urban environment.

8 EVALUATION, DISCUSSION, AND LIMITATIONS

To validate point distributions generated with our placement models, we performed a user study to evaluate the perceived realism of plant distributions generated with our PPMs and real data. Additionally, we asked expert users to compare the usefulness of our modeling approach for authoring plant distributions compared to the manual placement of individual plants. Finally, we measured the distance of generated and ground truth point sets of plant positions.

8.1 Perceptual User Study

We generated two sets of images for the user study, one with trees placed by our PPMs and another based on real data. It is difficult to generate images resembling satellite photographs, because of varying lighting, scattering, etc. To avoid this bias and to maintain

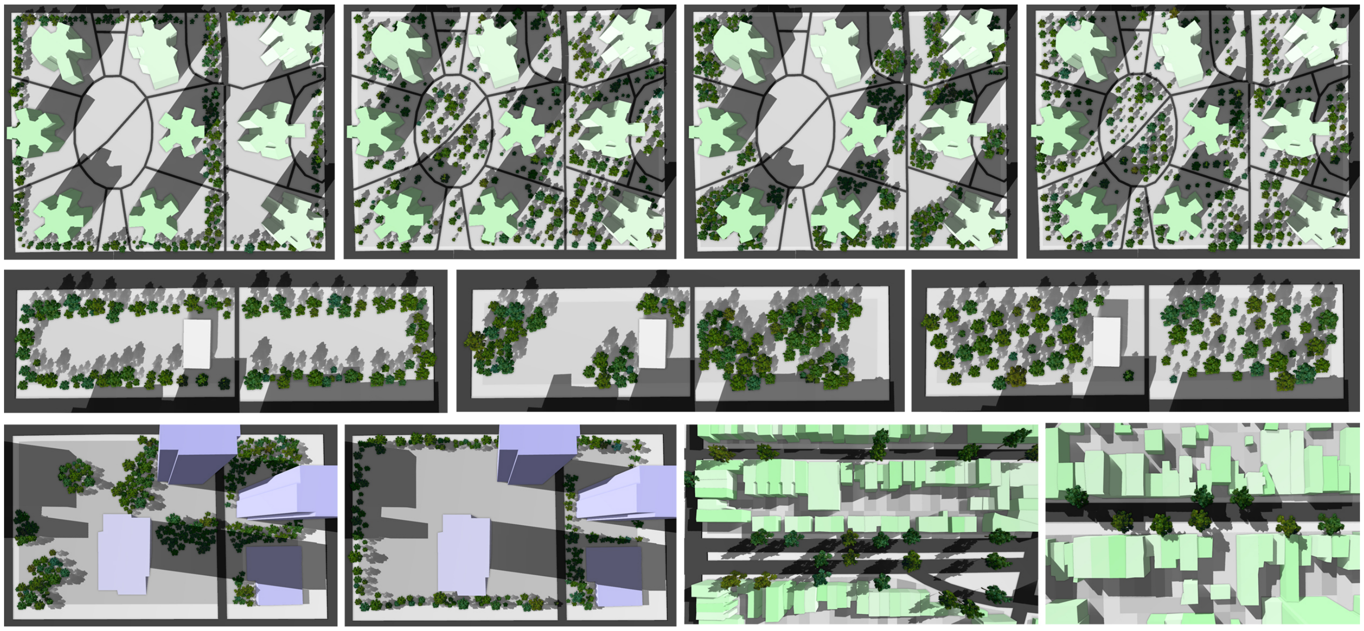


Fig. 16. Top-down renderings of plant distributions for three municipality zones generated with different placement strategies. Top row: the placement strategies boundary, semi-random, cluster, and regular for a residential lot of buildings. Middle row: the placement strategies boundary, cluster, and regular for the lot of a public park. Bottom row: the placement strategies cluster, boundary for a commercial lot (left) and the placement of trees with medial axis along streets with equidistant spacing set to: $\delta = 13m$ (right).

a similar appearance, we rendered real and synthetic plant distributions by using our framework (see Fig. 17f). We identified 30 lots with varying plant placements and produced plant positions using all placement strategies for these lots and rendered them as top-down images using our framework. We generated the real data by manually identifying plants in satellite images of these lots and marked their positions. We then loaded these positions into our framework and rendered them in the same rendering style for both categories. Furthermore, we chose top-down views for evaluating placement strategies, as this allows us to assess the respective distributions of plants.

We then performed a two-alternative force check (2AFC) on the images, for which we generated pairs of images showing real and synthetic plant distributions. The synthetic data shows distributions generated with our placement strategies (semi-random, boundary, cluster, regular), the reconstruction based on the coverage maps (reconstructed), using the planting strategy introduced by [Benes et al. 2011], and an entirely random placement (fully random). The *semi-random* placement strategy places plants based on Poisson Disk sampling, which generates random plant positions. However, only the positions are randomly generated, and the other parameters are selected in the same way as for the other strategies. This allows us to create plant distributions with a similar visual appearance. On the other hand, placing plants *fully random* means that all parameter values are sampled fully randomly, which results in incoherent and thus unrealistic plant distributions.

For *fully random* parameters of the PPM are not regularized at all but instead each parameter value is chosen randomly in its defined range, which may result in a very unrealistic appearance (*e.g.,* trees may stand unrealistically close to each other). The methods for plant placement of [Benes et al. 2011] are based on blocks and not individual lots. Consequently, these placement strategies assume a specific layout of buildings that cannot be used for individual lots (see Appx. Fig. 24) - the majority of areas selected for the user study have a layout that reflects that limitation. In total, we have selected 30 city blocks in New York City of up to 26 lots and populated them with our strategies (see Fig. 19). We randomly shuffled their arrangement (left-right) and their order. The image pairs (see Fig. 19 a) were shown to 107 users from Mechanical Turk (MT), and we made sure that only MT masters (reliable users) were answering the study. We asked the users, "Which plant distribution looks more realistic (left or right)?" The user had to choose one image. Each PPM category and real data received multiple rankings from every user.

The results of this evaluation are shown in Fig. 20 (left plot). The green bar shows the selection of scenes that were generated by reconstructing vegetation based on coverage maps. The blue bars show the selection results of placements with our pipeline, and the red bars shows the result for the baseline [Benes et al. 2011] and *fully random* placement. When comparing the results, the placement with coverage maps and our *semi-random* PPM strategy was selected to be realistic in 50% of the cases. This indicates that our method generates plant distributions that cannot be distinguished from real distributions. Out of our PPM strategies, *semi-random* was selected as the most realistic. In 53% of the cases, it was perceived as more realistic compared to the real placement. The strategy *boundary* was preferred in 44% of the cases, *cluster* placement in 48%, and *regular*

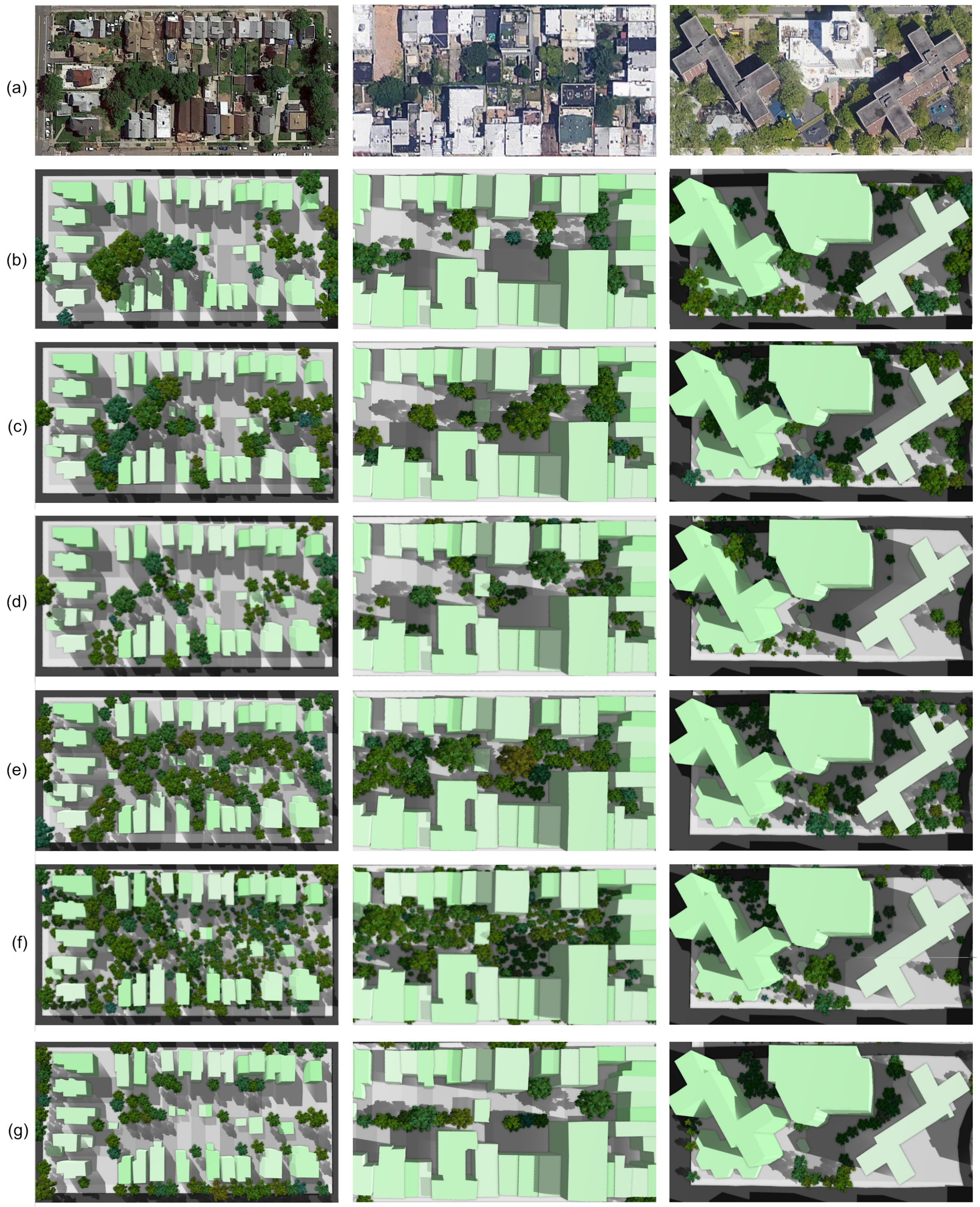


Fig. 17. We seek to generate plant populations as observed in satellite images of urban scenes (a). To validate our results, we manually labeled plant positions and used their longitude and latitude coordinates to render them at their real positions in our framework (b). This allows us to evaluate the visual quality of synthetically generated plant positions compared to real plant distributions. By using coverage maps and PPMs (*semi-random* strategy) our method is able to generate highly similar plant populations (c). Additionally, we compare our results to different variations as ablation studies. In (d) we place plants randomly without any parameter regularization (fully random), but we use the coverage maps to define areas where vegetation can be placed. The result of using no coverage map along with our *semi-random* placement strategy is shown in (e). In (f), we show the result of placing trees without a coverage map and without any regularization of parameters (fully random). In (g), the trees have been placed using the method of [Benes et al. 2011].

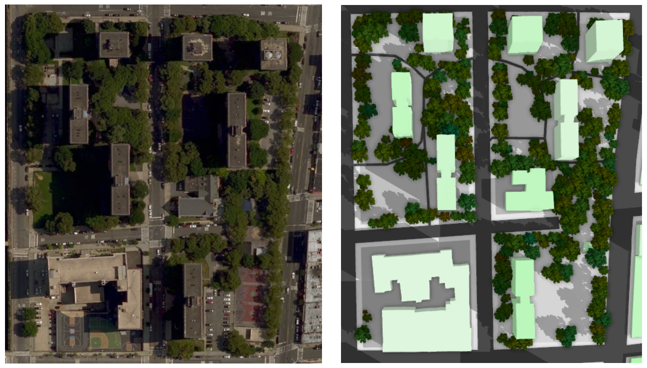


Fig. 18. Given a reference satellite image (left), we can create a similar distribution of plants by manually choosing the strategies and the corresponding parameters. Here we used the strategies *semi-random*, *boundary*, and *cluster* for the different lots (right).

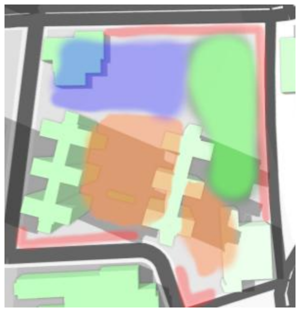
was perceived as more realistic in 30% of the cases. The *fully random* placement – as the lower bound baseline, without any PPM strategy involved – was perceived as realistic only for 16% , the baseline of [Benes et al. 2011] was chosen only in 35% of the cases of the shown image pairs.

8.2 Usability for Content Creation

To validate our method's effectiveness for content creation, we asked expert users (five 3D artists) to compare our placement strategies to random placement of plants. These strategies serve as presets to generate a realistic plant placement. In contrast to manually defining the details of every single plant (*e.g.,* the position, species, age, size, etc.), they automatically generate plant positions while respecting structural and positional constraints. We created a simple GUI with two brush tools: one for the manual placement of plants and one for the placement with PPM strategies. When using the manual brush tool, users have to specify the brush radius and the tree parameters. The user then interactively (by clicking with the mouse) generates multiple trees in the radius of the brush with random unconstrained positions. To precisely place a single tree, a small radius with a single mouse click can be used. The second brush tool uses the PPM strategies. Here a user defines the PPM parameters and sketches an area in a lot to automatically place plants with the selected and configured strategy.

We asked five experts to populate a given lot (inset figure) based on a predefined set of requirements: (1) ensure that the trees are not too close to the buildings; (2) populate the border of the lot with trees in the red marked area with consistent space and width; (3) populate the orange areas with randomly placed trees – with high density in the right and low density in the left area; (4) populate the green area regularly; and (5) create 2-3 clusters of trees within the blue area. The participants first received a brief introduction to the system and were then tasked to familiarize themselves with the UI without knowing the problem definition. After this introductory phase, the subjects were shown the problem definition and asked to solve the task once with manual placement using the manual brush tool and another time with the PPM brush tool. Whether to start with the manual or the PPM brush tool was changed for each expert. We then asked the experts to rank their experience based on several questions (Tab. 3). We used five-level Likert scale (*Strongly Agree, Agree, Undecided, Disagree, Strongly Disagree.* The results of this assessment are shown in Fig. 20 (right plot). For each question (a-j), we show the distribution of answers as box plots for the manual placement (left, red) and the PPM placement (right, blue).

These results show that our method provides an effective means to populate urban scenes with vegetation efficiently, since PPMs were rated as favorable for vegetation placement compared to manual placement. In addition we performed a qualitative study, in which the users commented that while the manual placement provides more control to precisely place plants, it takes much longer time to populate larger areas. The users also stated that generating realistic distributions is more difficult with manual placement.



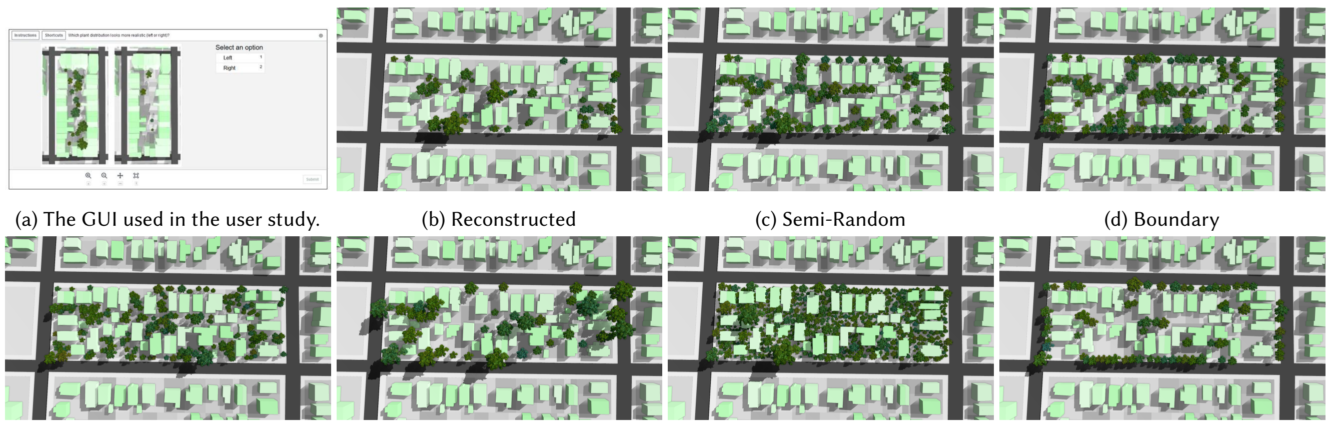
Finally, we asked the experts to automatically place trees by selecting a strategy and a lot, without brushing placement regions. For this setup, the experts unanimously stated that this technique provides less control but allows for fast and realistic vegetation placement in large areas. They further mentioned that the PPM brush tool and the automatic PPM placement produced excellent overall results and were preferred over manual placement to quickly achieve realistic-looking results. They also noted that a combination of manual and PPM-based placement would be desired when vegetation needs to be placed toward specific objectives.

	Question
a)	I think that I would like to use this system frequently.
b)	I found the system unnecessarily complex.
c)	I thought the system was easy to use.
d)	I think the outcome of the result was easy to control.
e)	I was satisfied with the outcome of the result.†
f)	I think the results look convincing/realistic.†
g)	I think it was able to meet the constraints.†
h)	I was satisfied with the outcome of the result.‡
i)	I think the results look convincing/realistic.‡
j)	I think it was able to meet the constraints.‡

Table 3. Expert users were asked to rate their experience between 1 (strongly disagree) to 5 (strongly agree) with respect to the questions above. †: before showing real images; ‡: after showing real images.

8.3 Discussion and Limitations

Our framework allows us to place and simulate vegetation in urban landscapes. To this end, our focus was on generating convincing distributions of plants for synthetic and real city models. Because defining rules for all possible variations of plants in urban landscapes is intractable, we factorized the problem of placing plants


Instructions
Shortcuts
Which plant distribution looks more realistic (left or right)?
Select an option
Left 1
Right 2
Submit

(a) The GUI used in the user study. (b) Reconstructed (c) Semi-Random (d) Boundary

(e) Cluster (f) Regular (g) Fully Random (h) [Benes et al. 2011]

Fig. 19. Examples of images shown to the participants of the user study. Random pairs of images were selected, and the participants were asked which plant populations looks more realistic.

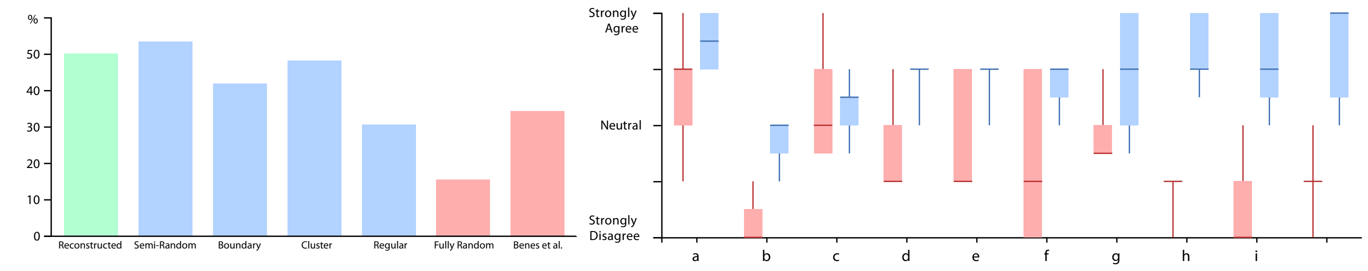


Fig. 20. Left: the results of a user study. Subjects were asked to select the more realistic vegetation placement based on various strategies compared to real plant distributions. The green bar shows the selection of scenes generated by reconstructing vegetation based on coverage maps, placement based on the PPM strategies (blue), and the baselines *fully random* placement and [Benes et al. 2011] (red). Right: experts' rating of manual plant placement (red bars) compared to using the PPM strategies (blue bars). Questions a-j are listed in Tab. 3.

into several placement strategies. Each strategy provides a concise set of rules and parameters to describe the positional and structural properties of vegetation within individual lots. Together, placement strategies and parameters allow us to generate realistic distributions of plants within an urban layout's functional zones. Besides, we use a state-of-the-art developmental model for plants to simulate their environmental response.

We generate distributions of vegetation that resemble what can be observed in satellite imagery; our focus was not on precisely reconstructing every plant of a real environment. While this is arguably important, it would require further analysis (*e.g.,* through deep learning) of satellite images and additional data sources, such as coverage maps. To this end, we think that procedurally generated vegetation can help to generate training data for more advanced analysis pipelines. Compared to manually placing vegetation, our method provides more control and capabilities for the efficient authoring of vegetation placement for city models.

As an alternative to learning parameters with the neural network pipeline from Fig. 13, we experimented with learning plant positions with Pix2Pix [Isola et al. 2016] in an end-to-end manner. We used satellite images as input and images with plant positions and building geometry as an output for this setup. The goal was to obtain the plant positions from the images in a post-processing step. Training this network was not successful for two reasons: it is challenging to get ground truth data pairs of satellite images and plant positions. While some datasets contain trees' GPS positions, they only store these positions for trees along streets, which is not useful for learning plant positions of an entire city. Second, the results of the network produced were not satisfactory. We suspect that the ground truth images were too sparse (*i.e.,* too few tree positions and building geometry) to provide a meaningful training signal.

A limitation of our current implementation is that we cannot obtain structural parameters with our learning pipeline. Such parameters cannot be learned from coverage maps; learning them from top-down satellite images was unsuccessful. Another limitation of our current approach is that we focus on medium and large trees and do not place smaller plants, such as flowers, bushes, or grass. While fixed models of flowers could be placed with our placement strategies (for example, by using agent-based models [Benes et al. 2003]), there exists no integrated developmental model that would allow us to develop trees and flowers jointly. Therefore, we decided only to simulate the growth response of trees to their environment.

Furthermore, we do not model plants that are shaped through advanced topiary. More research would be required to explore how pruning affects growth, *e.g.,* for hedges.

9 CONCLUSION AND FUTURE WORK

We have presented a novel framework for populating synthetic and real urban landscapes with vegetation. To this end, we introduced procedural placement models that allow us to generate plant positions realistically and grow individual plants into individual lots jointly. The key idea to our approach is that complex vegetation patterns among different zoning types of a city can be factorized into a set of simple placement rules. A PPM implements these rules and – together with their parameterization – allows to generate complex vegetation patterns with high visual fidelity. Moreover, the PPMs are context-sensitive and read the immediate neighborhood, enabling us to smooth out abrupt changes in placement.

To populate vegetation into real city models, we have used a state-of-the-art style-transfer network to translate satellite images to vegetation coverage maps. These coverage maps allow us to determine the distribution of vegetation within individual lots of a city, which allows us to reconstruct vegetation similar to what can be observed in real data. Instead of reconstructing vegetation at city scale precisely – which is intractable – our goal is to generate convincing and plausible details for reconstructing existing cities or populating entirely new virtual cities with vegetation.

We see several avenues for future work. First, it would be interesting to explore physical functions in an urban context affected by vegetation, such as heat transfer, shading, wind, and sound barriers. Second, further exploring how neural networks can generalize to more diverse urban data and use them to learn parameters for scene generation seems like a promising direction for future research. Tree position detection in a satellite image is an open problem. If we could detect the tree position, we could automatically detect what kind of procedural model should be applied to a lot. Our user study focused on validating the placement strategies. Another study should focus on aesthetic criteria, such as the building envelope and window visibility. Finally, we want to explore enhanced placement strategies to capture more of the variation of vegetation placements that can be observed in real cities.

9.1 Acknowledgments

This project was partly funded by the Deutsche Forschungsgemeinschaft (DFG) project (DE 620/27-1). This research was funded in part by National Science Foundation grant #10001387, *Functional Proceduralization of 3D Geometric Models* and by the Foundation for Food and Agriculture Research Grant ID: 602757. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the FFAR.

REFERENCES

S. AlHalawani, Y.-L. Yang, H. Liu, and N. J. Mitra. 2013. Interactive Facades Analysis and Synthesis of Semi-Regular Facades. *CGF* 32 (2013), 215–224.

M. Aono and T.L. Kunii. 1984. Botanical Tree Image Generation. *IEEE Comput. Graph. Appl.* 4(5) (1984), 10–34.

F. Bao, M. Schwarz, and P. Wonka. 2013. Procedural facade variations from a single layout. *ACM Trans. on Graphics* 32, 1, Article 8 (2013), 13 pages.

B. Benes, J. A. Cordóba, and J. M. Soto. 2003. Modeling Virtual Gardens by Autonomous Procedural Agents. In *Proceedings of TPCG*. IEEE Computer Society, 58.

B. Benes, M. A. Massih, P. Jarvis, D. G. Aliaga, and C. A. Vanegas. 2011. Urban Ecosystem Design. In *I3D*. 167–174.

B. Benes and E. U. Millán. 2002. Virtual Climbing Plants Competing for Space. In *CA '02: Proceedings of the Computer Animation*. IEEE Computer Society, 33.

D.V. Bloniarz and H.D.P. Ryan. 1993. Designing alternatives to avoid street tree conflicts. *Journal of Arboriculture* 19 (1993), 152–152.

M. Bokeloh, M. Wand, and H.-P. Seidel. 2010. A connection between partial symmetry and inverse procedural modeling. In *ACM Trans. on Graphics*, Vol. 29. ACM, 104.

G. Chen, G. Esch, P. Wonka, P. Müller, and E. Zhang. 2008. Interactive Procedural Street Modeling. In *ACM SIGGRAPH 2008*. 10.

H. I. Choi, S. W. Choi, H. P. Moon, and N.-S. Wee. 1997. New Algorithm for Medial Axis Transform of Plane Domain. *CVGIP: Graphical Model and Image Processing* 59 (1997), 463–483.

G. Cordonnier, E. Galin, J. Gain, B. Benes, E. Guérin, A. Peytavie, and M.-P. Cani. 2017. Authoring landscapes by combining ecosystem and terrain erosion simulation. *ACM Trans. on Graphics* 36, 4 (2017), 134.

M. Dang, D. Ceylan, B. Neubert, and M. Pauly. 2014. SAFE: Structure-aware Facade Editing. *CGF* (2014).

M. Dang, S. Lienhard, D. Ceylan, B. Neubert, P. Wonka, and M. Pauly. 2015. Interactive Design of Probability Density Functions for Shape Grammars. *ACM Trans. on Graphics* 34, 6, Article 206 (2015), 13 pages.

P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. 1988. Plant Models Faithful to Botanical Structure and Development. *SIGGRAPH Comput. Graph.* 22, 4 (1988), 151–158.

O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. 1998. Realistic Modeling and Rendering of Plant Ecosystems. In *Proc. of Sigg. (SIGGRAPH '98)*. ACM, 275–286.

A. Emilien, U. Vimont, M.-P. Cani, P. Poulin, and B. Benes. 2015. WorldBrush: Interactive Example-Based Synthesis of Procedural Virtual Worlds. *ACM Trans. on Graphics* 34, 4, Article Article 106 (2015), 11 pages.

T. A Endreny. 2018. Strategically growing the urban forest will improve our world. *Nature communications* 9, 1 (2018), 1160.

J. Gain, H. Long, G. Cordonnier, and M-P Cani. 2017. EcoBrush: Interactive Control of Visually Consistent Large-Scale Ecosystems. In *CGF*, Vol. 36. 63–73.

G. W Grey. 1995. *The urban forest: Comprehensive management.* John Wiley & Sons.

P. Guehl, R. Allegre, J.-M. Dischler, B. Benes, and E. Galin. 2020. Semi-Procedural Textures Using Point Process Texture Basis Functions. *Comp. Graph. Forum* 39, 4 (2020), 159–171. https://doi.org/10.1111/cgf.14061

É. Guérin, J. Digne, É. Galin, A. Peytavie, C. Wolf, B. Benes, and B. Martinez. 2017. Interactive Example-based Terrain Authoring with Conditional Generative Adversarial Networks. *ACM Trans. on Graphics* 36, 6, Article 228 (2017), 13 pages.

J. Guo, H. Jiang, B. Benes, O. Deussen, D. Lischinski, and H. Huang. 2020. Inverse Procedural Modeling of Branching Structures by Inferring L-Systems. *ACM Trans. Graph.* (2020), 1.

T. Hädrich, D. T. Banuti, W. Pałubicki, S. Pirk, and D. L. Michels. 2021. Fire in Paradise: Mesoscale Simulation of Wildfires. *ACM Trans. Graph.* 40, 4, Article 163 (July 2021), 15 pages.

T. Hädrich, B. Benes, O. Deussen, and S. Pirk. 2017. Interactive Modeling and Authoring of Climbing Plants. *Comput. Graph. Forum* 36, 2 (2017), 49–61.

Y. Hu, J. Dorsey, and H. Rushmeier. 2019. A novel framework for inverse procedural texture modeling. *ACM Tran. on Grap. (TOG)* 38, 6 (2019), 1–14.

T. Ijiri, S. Owada, and T. Igarashi. 2006. Seamless Integration of Initial Sketching and Subsequent Detail Editing in Flower Modeling. *CGF* 25, 3 (2006), 617–624.

M. Ilčík, P. Musialski, T. Auzinger, and M. Wimmer. 2015. Layer-Based Procedural Design of Façades. *CGF* 34, 2 (2015), 205–216.

P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. 2016. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2016), 5967–5976.

T. H. Kebrom. 2017. A Growing Stem Inhibits Bud Outgrowth – The Overlooked Theory of Apical Dominance. *Frontiers in Plant Science* 8 (2017), 1874.

T. Kelly, J. Femiani, P. Wonka, and N. Mitra. 2017. BigSUR: large-scale structured urban reconstruction. *ACM Trans. on Graph.* 36, 6 (2017).

T. Kelly, P. Guerrero, A. Steed, P. Wonka, and N. J. Mitra. 2018. FrankenGAN: Guided Detail Synthesis for Building Mass Models Using Style-Synchonized GANs. *ACM Trans. Graph.* 37, 6 (2018), 1:1–1:14.

A. Khan, A. Sohail, U. Zahoora, and A. Saeed. 2019. A Survey of the Recent Architectures of Deep Convolutional Neural Networks.

J. Kratt, Mark Spicker, A. Guayaquil, M. Fišer, S. Pirk, O. Deussen, J. C. Hart, and B. Benes. 2015. Woodification: User-Controlled Cambial Growth Modeling. *CGF* 34, 2 (2015), 361–372.

B. Li, J. Kałużny, J. Klein, D. L. Michels, W. Pałubicki, B. Benes, and S. Pirk. 2021. Learning to Reconstruct Botanical Trees from Single Images. *ACM Transaction on Graphics* 40, 6, Article 231 (12 2021).

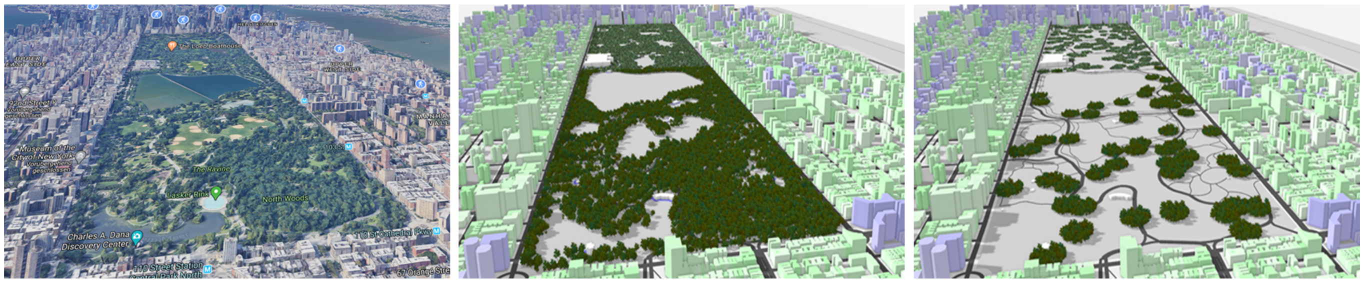


Fig. 21. Left: Google maps view of New York (Central Park). Our framework generated two variations of plant placements (middle, right) for an initially empty city model. Middle: 54k plant positions were generated in about 60 seconds with a *random* strategy. Right: a different plant population generated with the strategy *cluster* (16k plants).

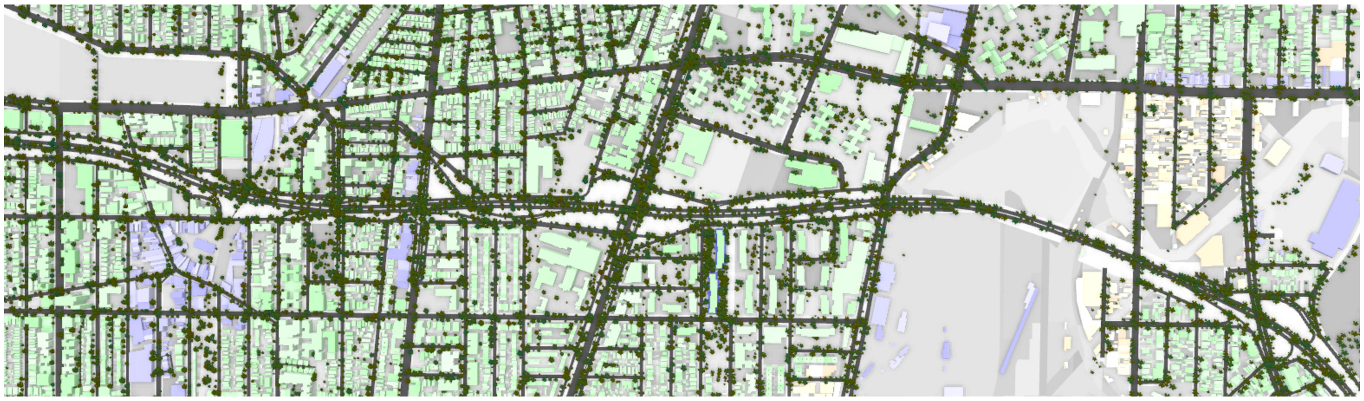


Fig. 22. Our framework enables to efficiently place vegetation for large urban areas. To reconstruct vegetation for larger urban areas we predict coverage maps and populate the detected areas with our *semi-random* strategy for each lot.

C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall. 2011a. Modeling and Generating Moving Trees from Video. *ACM Trans. on Graphics* 30, 6, Article 127 (2011), 127:1–127:12 pages.

M. Li, A. G. Patil, K. Xu, S. Chaudhuri, O. Khan, A. Shamir, C. Tu, B. Chen, D. Cohen-Or, and H. Zhang. 2019. GRAINS: Generative Recursive Autoencoders for INdoor Scenes. *ACM Trans. Graph.* 38, 2, Article 12 (Feb. 2019), 16 pages.

Y. Li, Q. Zheng, A. Sharf, D. Cohen-Or, B. Chen, and N. J. Mitra. 2011b. 2D-3D fusion for layer decomposition of urban facades. In *ICCV*. 882–889.

A. Lindenmayer. 1968. Mathematical models for cellular interaction in development. *Journal of Theoretical Biology* Parts I and II, 18 (1968), 280–315.

B. Lintermann and O. Deussen. 1999. Interactive Modeling of Plants. *IEEE Comput. Graph. Appl.* 19, 1 (1999), 56–65.

Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. 2011. Texture-lobes for Tree Modelling. *ACM Trans. on Graphics* 30, 4, Article 53 (2011), 10 pages.

S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. 2012. TreeSketch: interactive procedural modeling of trees on a tablet. In *Proc. of the Intl. Symp. on SBIM*. 107–120.

M. Makowski, T. Hädrich, J. Scheffczyk, D. L. Michels, S. Pirk, and W. Palubicki. 2019. Synthetic Silviculture: Multi-scale Modeling of Plant Ecosystems. *ACM Trans. on Graphics* 38, 4, Article 131 (2019), 14 pages.

A Martinovic and L Van Gool. 2013. Bayesian grammar learning for inverse procedural modeling. In *CVPR*. 201–208.

P. Merrell and D. Manocha. 2008. Continuous Model Synthesis. *ACM Trans. on Graphics* 27, 5, Article Article 158 (2008), 7 pages.

P. Merrell and D. Manocha. 2011. Model Synthesis: A General Procedural Modeling Algorithm. *TVCG* 17, 6 (2011), 715–728.

R. W Miller, R. J Hauer, and L. P Werner. 2015. *Urban forestry: planning and managing urban greenspaces.* Waveland press.

R. Minamino and M. Tateno. 2014. Tree branching: Leonardo da Vinci's rule versus biomechanical models. *PloS one* 9, 4 (2014), e93535.

S. Mitchell, A. Rand, M. Ebeida, and C. Bajaj. 2012. Variable Radii Poisson-Disk Sampling. *CCCG* (01 2012).

P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. 2006. Procedural modeling of buildings. *ACM Trans. on Graphics* 25, 3 (July 2006), 614–623.

P. Müller, G. Zeng, P. Wonka, and L. Van Gool. 2007. Image-based Procedural Modeling of Facades. *ACM Trans. on Graphics* 26, 3, Article 85 (2007).

R. Měch and P. Prusinkiewicz. 1996. Visual models of plants interacting with their environment. In *ACM SIGGRAPH 96*. ACM, New York, NY, USA, 397–410.

G. Nishida, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and A. Bousseau. 2016. Interactive sketching of urban procedural models. *ACM Trans. on Graphics* 35, 4 (2016), 130.

NYCOpenData. 2019. The Next Decade of Open Data. (2019). https://data.ny.gov/

M. Okabe, S. Owada, and T. Igarashi. 2007. Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing. In *ACM SIGGRAPH Courses*. ACM, Article 26.

P. E. Oppenheimer. 1986. Real time design and animation of fractal plants and trees. *Proc. of SIGGRAPH* 20, 4 (1986), 55–64.

W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz. 2009. Self-organizing Tree Models for Image Synthesis. *ACM Trans. on Graphics* 28, 3, Article 58 (2009), 10 pages.

Y. I. H. Parish and P. Müller. 2001. Procedural Modeling of Cities *(ACM SIGGRAPH 2001)*. 301–308.

S. Pirk, B. Benes, T. Ijiri, Y. Li, O. Deussen, B. Chen, and R. Měch. 2016. Modeling Plant Life in Computer Graphics. In *ACM SIGGRAPH 2016 Courses*. Article Article 18, 180 pages.

S. Pirk, M. Jarząbek, T. Hädrich, D. L. Michels, and W. Palubicki. 2017. Interactive Wood Combustion for Botanical Tree Models. *ACM Trans. on Graphics* 36, 6, Article 197 (2017), 12 pages.

S. Pirk, T. Niese, O. Deussen, and B. Neubert. 2012a. Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. on Graphics* 31, 6, Article 169 (2012), 10 pages.

S. Pirk, T. Niese, T. Hädrich, B. Benes, and O. Deussen. 2014. Windy Trees: Computing Stress Response for Developmental Tree Models. *ACM Trans. on Graphics* 33, 6, Article 204 (2014), 11 pages.

S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen. 2012b. Plastic trees: interactive self-adapting botanical tree models. *ACM Trans. on Graphics* 31, 4, Article 50 (2012), 10 pages.

P. Prusinkiewicz. 1986. Graphical Applications of L-systems. In *Proceedings on Graphics Interface '86/Vision Interface '86*. Canadian Information Processing Society, 247–253.

D. Ritchie, B. Mildenhall, N. D. Goodman, and P. Hanrahan. 2015. Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo. *ACM Trans. on Graphics* 34, 4 (2015), 105.

D. Ritchie, K. Wang, and Y.-A. Lin. 2018. Fast and Flexible Indoor Scene Synthesis via Deep Convolutional Generative Models. In *CVPR*.

A. Runions, B. Lane, and P. Prusinkiewicz. 2007. Modeling Trees with a Space Colonization Algorithm. In *Conference on Natural Phenomena (NPH–07)*. 63–70.

M. Schwarz and P. Müller. 2015. Advanced Procedural Modeling of Architecture. *ACM Trans. on Graphics* 34, 4 (Proceedings of SIGGRAPH) (2015), 107:1–107:12.

R. M. Smelik, T. Tutenel, R. Bidarra, and B. Benes. 2014. A Survey on Procedural Modelling for Virtual Worlds. *CGF* 33, 6 (2014), 31–50.

O. Stava, B. Benes, R. Měch, D. G. Aliaga, and P. Kristof. 2010. Inverse Procedural Modeling by Automatic Generation of L-systems. *CGF* 29, 2 (2010), 665–674.

O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen, and B. Benes. 2014. Inverse Procedural Modelling of Trees. *CGF* 33, 6 (2014), 118–131.

J. O. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun. 2011. Metropolis procedural modeling. *ACM Trans. on Graphics* 30, 2 (2011), 11.

P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan. 2008. Single Image Tree Modeling. *ACM Trans. on Graphics* 27, 5, Article 108 (2008), 7 pages.

P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. 2007. Image-based Tree Modeling. *ACM Trans. on Graphics* 26, 3, Article 87 (2007).

C. A. Vanegas, D. G. Aliaga, and B. Benes. 2010a. Building reconstruction using manhattan-world grammars. *CVPR* 0 (2010), 358–365.

C. A. Vanegas, D. G. Aliaga, B. Benes, and P. A. Waddell. 2009. Interactive Design of Urban Spaces Using Geometrical and Behavioral Modeling. In *ACM SIGGRAPH Asia*. Article Article 111, 10 pages.

C. A. Vanegas, D. G. Aliaga, P. Wonka, P. Müller, P. Waddell, and B. Watson. 2010b. Modelling the Appearance and Behaviour of Urban Spaces. *CGF* 29, 1 (2010), 25–42.

C. A. Vanegas, I. Garcia-Dorado, D. G. Aliaga, B. Benes, and P. Waddell. 2012. Inverse Design of Urban Procedural Models. *ACM Trans. on Graphics* 31, 6, Article 168 (Nov. 2012), 11 pages.

P. Waddell. 2002. UrbanSim: Modeling urban development for land use, transportation, and environmental planning. *Journal of the American planning association* 68, 3 (2002), 297–314.

P. Waddell, G. F. Ulfarsson, J. P. Franklin, and J. Lobb. 2007. Incorporating land use in metropolitan transportation planning. *Transportation Research Part A: Policy and Practice* 41, 5 (2007), 382–410.

B. Wang, Y. Zhao, and J. Barbič. 2017. Botanical Materials Based on Biomechanics. *ACM Trans. on Graphics* 36, 4, Article 135 (2017), 13 pages.

K. Wang, Y.-A. Lin, B. Weissmann, M. Savva, A. X. Chang, and D. Ritchie. 2019. PlanIT: Planning and Instantiating Indoor Scenes with Relation Graph and Spatial Prior Networks. *ACM Trans. on Graphics* 38, 4, Article Article 132 (2019), 15 pages.

B. Watson, P. Müller, O. Veryovka, A. Fuller, P. Wonka, and C. Sexton. 2008. Procedural Urban Modeling in Practice. *IEEE Computer Graphics and Applications* 28, 3 (2008), 18–26.

B. Weber, P. Müller, P. Wonka, and M. Gross. 2009. Interactive Geometric Simulation of 4D Cities. *CGF* 28, 2 (2009), 481–492.

J. Wither, F. Boudon, M.-P. Cani, and C. Godin. 2009. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *CGF* 28, 2 (2009), 541–550.

P. Wonka, M. Wimmer, F. Sillion, and W. Ribarsky. 2003. Instant Architecture. *ACM Trans. on Graphics* 22, 3 (2003), 669–677.

F. Wu, D.-M. Yan, W. Dong, X. Zhang, and P. Wonka. 2014. Inverse Procedural Modeling of Facade Layouts. *ACM Trans. on Graphics* 33, 4, Article 121 (2014), 10 pages.

X. Wu, K. Xu, and P. Hall. 2017. A survey of image synthesis and editing with generative adversarial networks. *Tsinghua Science and Technology* 22, 6 (2017), 660–674.

K. Xie, F. Yan, A. Sharf, D. Deussen, H. Huang, and B. Chen. 2016. Tree Modeling with Real Tree-Parts Examples. *TVCG* 22, 12 (2016), 2608–2618.

Y.-T. Yeh, K. Breeden, L. Yang, M. Fisher, and P. Hanrahan. 2013. Synthesis of Tiled Patterns Using Factor Graphs. *ACM Trans. on Graphics* 32, 1, Article 3 (2013), 13 pages.

H. Zeng, J. Wu, and Y. Furukawa. 2018. Neural procedural reconstruction for residential buildings. In *Proceedings of the ECCV*. 737–753.

H. Zhang, K. Xu, W. Jiang, J. Lin, D. Cohen-Or, and B. Chen. 2013. Layered Analysis of Irregular Facades via Symmetry Maximization. *ACM Trans. Graph.* 32, 4, Article 121 (July 2013), 13 pages.

Y. Zhao and J. Barbič. 2013. Interactive Authoring of Simulation-ready Plants. *ACM Trans. on Graphics* 32, 4, Article 84 (2013), 12 pages.

A PARAMETERS

Table 4. Parameter values we used to generate the figures in the paper.

Fig.	Strategy	μ	σ	τ	β	κ	π	ω	ψ	η	α	ρ	θ	λ
5 a	B	3.13	0.35	1.0	4.0									
5 b	B	3.13	0.35	0.8	10.0									
5 c	B	3.13	0.35	0.4	10.0									
5 d	C	3.13	0.35	1.0		10.0	1							
5 e	C	3.13	0.35	1.0		10.0	3							
5 f	C	3.13	0.35	1.0		15.0	3							
5 g	I	3.13	0.00	1.0				9.0	0.00	30°				
5 h	I	3.13	0.00	1.0				9.0	0.30	30°				
5 i	I	3.13	0.00	1.0				9.0	0.55	30°				
5 j	R	3.13	0.35	0.2										
5 k	R	3.13	0.35	0.4										
5 l	R	3.13	0.35	1.0										
8 a	R	3.4	0.15	0.2							14	0.0	0.0	4
8 b	R	3.4	0.15	0.2							20	0.0	0.0	4
8 c	R	3.4	0.15	0.2							30	0.0	0.0	4
8 d	B	2.2	0.0	0.9	10						20	1.0	0.0	4
8 e	B	2.2	0.0	0.9	10						20	0.7	0.0	
8 f	B	2.2	0.0	0.9	10						20	0.5	0.0	4
8 g	C	3.2	0.25	1.0		10.0	3				16	0.0	0.0	4
8 h	C	3.2	0.25	1.0		10.0	3				16	0.1	0.3	4
8 i	C	3.2	0.25	1.0		10.0	3				16	0.2	0.4	4
15	C	3.5	0.35	1.0		20.0	18				16	0.0	0.0	1

B SATELLITE AND COVERAGE MAP DATA

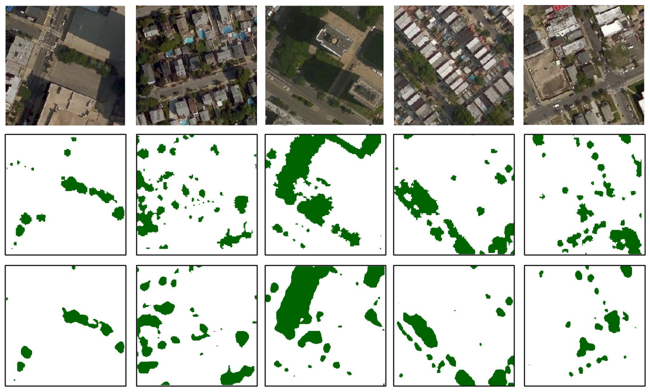


Fig. 23. Learning of coverage maps: we use satellite images (top) and ground truth coverage maps (middle) from NYC Open Data to train a neural network for style-transfer. After training the network is able to predict coverage maps (bottom) from satellite images.

C PLACEMENT WITH [BENES ET AL. 2011]

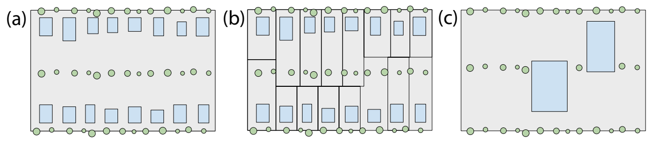


Fig. 24. The method presented in [Benes et al. 2011] places the trees according to procedural rules by using a single strategy for a whole block. Managed ecosystem simulation then makes the trees grow, seed, and die by competition leading to semi-random distributions. Our method works on individual lots, places all plants at once and does not require simulation to populate the urban model. a) With the method of [Benes et al. 2011] trees planted at the front and back of the block and along its main axis. b) An overlay of the same block with the actual lots of individual properties. This indicates that the method does not consider lot boundaries. c) When the method of [Benes et al. 2011] is used for a single lot plants are placed in an unrealistic manner as buildings and other lot features (e.g. lot shape) are not considered.