Volume 31 (2012), Number 2

Procedural Generation of Parcels in Urban Modeling

Carlos A. Vanegas[2,1], Tom Kelly[† 3,1], Basil Weber[1], Jan Halatsch[4], Daniel G. Aliaga[2,4], and Pascal Müller[1]

[1]Esri R&D Center Zurich, Switzerland
[2]Department of Computer Science, Purdue University, USA
[3]University of Glasgow, UK
[4]ETH Zurich, Switzerland

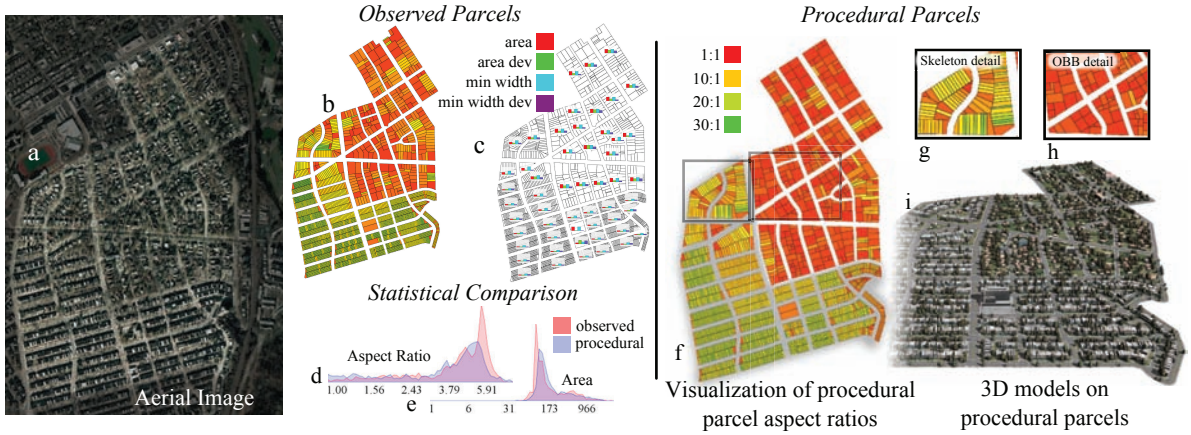

Figure 1: *Procedural Parcel Generation. Our method creates parcels inside city blocks (f,i) using two different subdivision techniques —* skeleton *(g, shaded part of f) or* OBB *(h, unshaded part of f). The subdivision attributes are automatically extracted from observed real-world cities (a,b,c) or determined by the user. The resulting parcel configurations closely resemble real-world subdivisions, as shown by our statistical and visual comparison of procedural and observed parcel datasets (d,e).*

Abstract

We present a method for interactive procedural generation of parcels within the urban modeling pipeline. Our approach performs a partitioning of the interior of city blocks using user-specified subdivision attributes and style parameters. Moreover, our method is both robust and persistent in the sense of being able to map individual parcels from before an edit operation to after an edit operation - this enables transferring most, if not all, customizations despite small to large-scale interactive editing operations. The guidelines guarantee that the resulting subdivisions are functionally and geometrically plausible for subsequent building modeling and construction. Our results include visual and statistical comparisons that demonstrate how the parcel configurations created by our method can closely resemble those found in real-world cities of a large variety of styles. By directly addressing the block subdivision problem, we intend to increase the editability and realism of the urban modeling pipeline and to become a standard in parcel generation for future urban modeling methods.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—I.3.6 [Computer Graphics]: Methodology and Techniques—

1. Introduction

Interactive large-scale urban modeling is becoming increasingly popular in computer graphics research and in applications to several fields including gaming, urban planning, and navigation tools. A key reason for the popularity of interactive urban modeling is the ability to quickly create large complex models. The major steps of a typical modeling task

† joint first author

include (i) creating the underlying terrain and general land-use pattern of the city or urban area, (ii) generating an interconnected road network, (iii) subdividing the space in between roads (i.e., blocks) into parcels, and (iv) populating parcels with buildings, parks, and other urban structures. In this paper, we focus on the third major step in this pipeline: providing a comprehensive and fully interactive approach for subdividing blocks into parcels, a task which has been largely ignored in previous computer graphics systems potentially resulting in unrealistic and implausible results.

The subdivision of blocks and generation of parcels has been partially addressed in previous works. Within computer graphics research, Parish and Müller [PM01] pioneered a procedural approach to urban modeling and many derivative works have since been created. Vanegas et al. [VAW*10] provides a recent state-of-the-art report of related urban modeling methods. While previous automatic block subdivision methods take into account providing egress (i.e., ensuring a parcel has street access), the methods do not always produce plausible parcel shapes, sometimes only support simple block shapes, and can yield areas within a block that are not assigned to any parcel. Within urban design and planning research, the focus of parcel generation methods has been on satisfying the major interests of real-estate investors and complying with zoning and building law regulations [PCDS04]. While automatic subdivision is well exploited in computer graphics, in urban design blocks are partitioned according to desired patterns (e.g., [TKD10, PPC08]) but the labor is often performed manually, a task which does not scale well to large urban modeling applications. Moreover, in either graphics or urban design, altering an existing city's geometry can cause unexpected changes in a block's subdivision, leading to difficult shape control and editing. Further, this challenge is only exacerbated during interactive editing. For example, a small change to road geometry can cause the subdivision for an entire neighborhood of parcels to be unwillingly altered both in number and in shape – this lack of parcel persistence can cause the loss of prior customizations.

The key motivation behind our work is to develop a generalized block subdivision method inspired by urban design guidelines, suitable for intuitive large-scale interactive editing, and able to reproduce the parcel shapes and configurations observed in many cities. In urban design, a block usually consists of one of two parcel varieties [Car03]:

- The first variety has parcels whose front-side is along a street and rear-side is adjacent to another instance of the same parcel variety.
- The second variety includes parcels that may also be adjacent to streets but can include a string of interior parcels separated by small pathways or alleys, instead of formal streets.

Moreover, parcels usually have a rather regular or uniform structure that is typically a deep rectangle, wide rectangle, approximate square, quadrilateral, or sometimes polygonal [Cur97]. We seek to automate the subdivision of arbitrary block shapes, ensure the aforementioned set of urban characteristics are met, and provide user-controlled realistic parcels that approximate the forms used in practice. This level of automation gives more time to designers to concentrate on high-level design decisions, including during virtual world content creation.

Our approach for parcel generation uses a combination of two subdivision methods to reproduce the aforementioned two parcel varieties, including mixed-types, and to ensure a set of subdivision attributes are satisfied. The input to our method is a set of interconnected roads where an ordered sequence (loop) of road segments defines a block to be subdivided into parcels (Section 3). Any block can have potentially one or both prototypical parcel varieties. The overall regularity and shape of all parcels is controlled by user-specified subdivision attributes that ensure: (i) the parcels collectively partition the block (i.e., there should be no unused/unassigned space), (ii) all parcels have the option of street access (i.e., egress), (iii) parcels have a simple exterior boundary, often nearly rectangular, and the parcel's size and aspect ratio is controllable, and (iv) parcels are aligned as best as possible with the adjacent street segment, if any (Section 4). Further, our solution includes a robust mechanism to map individual parcels from before an interactive edit operation to after the edit operation — this enables transferring most, if not all, customizations, despite there being significant changes to the underlying road network and block geometry (Section 5). Our framework also supports the generation of urban models of up to half a million parcels of arbitrary shapes. As shown in our results (Section 6), the styles afforded by our method combined with our expressive set of attributes allow for a large variety in the subdivision results and support the generation of many subdivision styles found in real world urban layouts.

Succinctly, our solution improves the urban modeling pipeline for all future methods by achieving:

- realism – we focus on providing an automatic subdivision algorithm that is able to produce the patterns used by urban designers in real-world cities;
- persistence – we address the challenge of consistently labeling blocks and parcels so as to enable a best mapping of parcels from before an edit operation to after an edit operation, thereby enabling persistence of a priori customizations; and
- interactivity – we support fully interactive editing (i.e., move, copy, transform) of intersection points, road geometry, and parcel attributes at local and citywide scales.

2. Previous Work

We provide a brief review of previous and related work for generating the 2D and 3D geometry of urban spaces. Parish and Müller [PM01] introduced an initial approach in which L-systems were adapted to resemble the growth of streets. Subsequent block subdivision was implemented as an algorithm that recursively divides the longest pair of approxi-

mately parallel edges until parcel sizes are under a user-specified threshold area. Parcels with no street access are discarded. This simple algorithm does not necessarily produce realistically shaped parcels and leaves odd-shaped empty areas inside city blocks that do not belong to any parcel.

Later work has built upon Parish and Müller's paper and further developed different components. Several papers have focused on providing realistic building content (e.g., [WWSR03,MWH*06,LWW08]). Other works have concentrated on road networks. For example, Chen et al. [CEW*08] use hyper-streamlines and tensor fields to generate a road network but do not provide a novel automatic block subdivision algorithm. Galin et al. [GPMG10] provide a methodology to generate realistic roads between a source and destination but do not address parcel generation. Vanegas et al. [VABW09a] describe a block subdivision algorithm which assumes parcels are mostly rectangular. Their method computes the oriented bounding box of the parcel and uses the middle (long) axis to optionally divide the block into two areas, which are then partitioned into the same number of parcels. While all parcels will have egress, other subdivision styles are not supported, nor is parcel persistence addressed.

More recently, Lipp et al. [LSWW11] proposed a method that enables editing an urban layout. While their approach is interactive and does support a type of editing persistence, it does not focus on block subdivision. They assume the subdivision is present in the initial layout and use very simple heuristics to subdivide small blocks that may appear during an editing process (e.g., around the fringes of a street or neighborhood that is moved or transformed interactively). Several works have integrated urban simulation engines into the urban modeling pipeline. For example, Weber et al. [WMWG09] describe a geometrical simulation of a growth of a city of overtime. Vanegas et al. [VABW09b] integrate an urban behavioral simulator with procedural modeling. Neither paper innovates block subdivision – rather they re-implement the methods in [PM01, AVB08] or [VABW09a].

In urban design and planning, automatic methods are rarely used. The few available solutions (e.g., [WD11, HKS08, Mar09, WCP*11]) do not go beyond basic implementations and have very limited stylistic control (Figure 2).

In contrast to previous work, we propose a guided space-partitioning based scheme. Since we perform a space-partitioning, we avoid the presence of unassigned areas. The automatic processing of our method, which seeks to satisfy a set of subdivision attributes, supports a range of block subdivision styles, and produces regular and nicely-shaped parcels despite partitioning arbitrarily shaped blocks (i.e., not only nearly rectangular blocks). Finally, our solution is designed to support parcel persistence and fast interactive editing for both local and city scale operations.

3. System Overview

In this section, we provide an overview of our interactive editing system and underlying data structure. In particular, we explain our graph-based road network data structure and our procedure for the extraction and updating of blocks from a given road graph. To support (live) interactive editing of road and block attributes, we also demonstrate how to trigger updates to block subdivisions. Note that we do not explicitly discuss how to compute road geometry nor building geometry, since this is not the objective of our work.

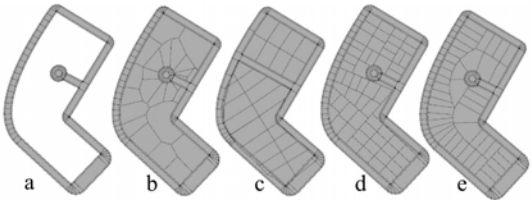

Figure 2: *Comparison to existing approaches. Given a street network (a), several systems have been proposed to create parcels. The Voronoi tessellation of points near the border of the street is one geometrical approach (b), while a minimum area bounding box approach has also been suggested [WCP*11] (c). Cube packing is another approach that leads to badly formed parcels in concave areas (d). We show our result (e) which gives a statistically realistic result. A comparison to [WMWG09] is presented in Appendix C.*

3.1. Data Structure

A city, or fragment of a city, is represented by road network graph and its dual containing all city blocks. The road network is a planar graph (V,E) with nodes V and edges E. Each node $n \in V$ is defined by its position in $\mathbb{R}^2$ and logically corresponds to a road intersection point. Each edge $e \in E$ stores the indexes of its source and target nodes and logically corresponds to a road segment. In our implementation, the edge also stores the width of the street that it represents, and an interpolation type that determines whether the edge is a straight line segment or a curve. In the latter case, the adjacent nodes are interpolated using Bezier splines.

A city block B corresponds to exactly one face of the road network graph. City blocks are connected by a shared road segment. Thus the set of city blocks form the dual of the road network graph. This dual graph can be stored explicitly or recomputed each time it is needed from the road graph (as is the case in our system - see Section 3.2). Each city block B stores several attributes that determine how the block is subdivided into parcels, including subdivision style, parcel area and width bounds and split irregularity - see Section 4.1. Some of these parameters are common to groups of blocks, while others are specific to individual blocks.

3.2. Live Interactive Editing System

As with most urban modeling systems, our implementation supports editing the road geometry and block parameters, either for individual roads/blocks or for large areas of the city. The editing system determines that the contour geometry of the blocks needs to be updated (i) after road graph topology changes and/or (ii) after any road/block attribute changes.

If a topology change occurs (e.g., addition or removal of a road segment), the set of blocks is updated through a planar face traversal method that efficiently extracts the cycles of the graph. For every new block the contour geometry is computed. If an attribute change occurs (e.g. a node translation or a street width change), the set of affected blocks is computed and the contour geometry of the respective blocks is updated. Block subdivision (Section 4) is triggered for a given block B whenever there is a change in either the geometry of the contour $C(B)$ of the block or in the values of the subdivision attributes for the block. The parameters for individual blocks are mapped to the corresponding set of blocks as determined by our parcel consistency method (Section 5).

4. Block Subdivision

This section provides an overview of our block subdivision method, describes the sought after subdivision attributes, and explains our two main subdivision algorithms.

4.1. Algorithm Overview

Our approach consists of two main subdivision algorithms that enable each of the two prototypical parcel varieties inspired from urban design work (see Section 1). The subdivision scheme to model the first parcel variety is based on the straight skeleton [AAAG95] of $C(B)$. The straight skeleton of a polygonal block ensures the following design requirements: (i) the resulting regions always have street access, which is one of the main constraints for this parcel variety; and (ii) the rear side of the generated parcels is always abutting another parcel – a characteristic of this parcel variety. The rear-side neighbor parcels can be either parcels with street access themselves, or a central *patio* parcel with no street access. The latter case is known as *perimeter block* (Figure 3 right), which according to urban planners is an optimal design pattern to achieve very high urban densities without high-rise buildings. Intuitively, such a perimeter layout can be obtained by creating parcels "around" the medial axis of the block contour, and we exploit the fact that the straight skeleton is a linear approximation of the medial axis. Our choice of the straight skeleton is further supported by our observations of how urban planners often, out of intuition, draw an approximate medial axis to homogeneously subdivide blocks, especially in residential areas.

The second subdivision scheme is based on a recursive splitting concept as introduced in [PM01] and performs an adaptive spatial partitioning of $C(B)$ using oriented bounding boxes (OBBs). The result is parcels resembling the second variety, namely the presence of parcels on the perimeter of the block with egress and parcels interior to the block presumably separated by small pathways or alleys. Hybrid uses of these two methods is also possible. For example, a large block can be subdivided using skeleton-based subdivision to yield parcels of a chosen maximum depth around the perimeter and the interior space can be optionally partitioned using OBB-based subdivision.

The algorithms are controlled by high-level user-specified stylistic control parameters and also support mixed-type configurations. In all cases, the partitioning process is performed independently for each block. Let B be a block to be subdivided. Let $C(B) = \{b_1, b_2, \ldots, b_m\}$ be the ordered set of m vertices describing the contour of B. The vertices are specified in a counter-clockwise direction. In general, our subdivision algorithms partitions B into a set $L = \{l_1, l_2, \ldots, l_n\}$ of n regions such that $\bigcup_{i=1}^{n} l_i = B$ and $\forall_{l_i, l_j \in L}(l_i \cap l_j = \emptyset)$.

While our methods support an infinite number of subdivisions for a given block, only some such subdivisions are adequate for mimicking land subdivision in real-world cities. Hence, several constraints are considered when determining suitable subdivisions, including a fairly homogeneous area distribution among the resulting parcels, parcel areas, aspect ratios and angles that meet structure construction constraints, and access from the parcels to the road network. The set of subdivision attributes that our system enforces for all parcels is provided below.

- Parcel area bounds (A_{min}, A_{max}): The upper and lower bounds on the area of the resulting parcels. For all l_i, $A_{min} < A(l_i) < A_{max}$.
- Minimum parcel width (W_{min}, W_{max}): The upper and lower bounds on the length of the sides of the oriented bounding box of a parcel.
- Split irregularity ω: The deviation of a split edge from its default position, normalized in $[0, 1]$. Larger values result in the split being further away from the mid-point, and generally, in a higher variance in the parcel areas.

The detailed pseudocode for both styles is presented in Algorithm 1 of Appendix A.

4.2. Skeleton-based Subdivision

This subdivision style creates parcels resembling the first prototypical parcel variety, i.e., parcels whose front-sides are along a street, and rear-sides are adjacent to other parcels.

We made several observations which directed the design of our algorithm. The first was that every parcel should have street access, and that the direction of this street access at intersections varied. This lead us to the concept of *strips* of parcels, which follow roads. Further observations were that the initial split was frequently the common centerline between rows of parcels on either side of the block, and the rear edges of the parcels were normally straight.

A polygonal skeleton supports these observations as it identifies both the potential strips and the centerline. We found the straight skeleton the most realistic, since other techniques, such as the medial axis, introduced curved segments. Therefore our fist subdivision method computes the initial split contour via the straight skeleton, identifying the depth of the parcels via strips. Subsequent splits cast rays from the street edges to position the sides of the parcels.

4.2.1. Initial Split

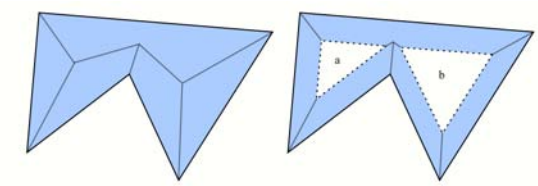

Figure 3: *Left: The straight skeleton of a polygon (black) consists of* arcs *(dark blue) which define* faces *adjacent to each polygon edge. Right: A partial application of the straight skeleton computes the offset contour $C'(B)$ (dashed lines). Note that the contour may split the innermost, or* patio*, region into several portions (a,b). The individual faces of the straight skeleton conform to our definition of a* strip*, and we may take the supporting edges to be portions of the boundary of input polygon. We take these faces to be the initial set of α-strips.*

To define the initial split contour, the user defines a perpendicular distance d_{offset} from the block contour $C(B)$ to an inwards offset contour $C'(B)$. Intuitively, this value corresponds to the maximum depth (distance from the road to the rear) of the parcels. If the d_{offset} value is sufficiently large (e.g., infinity), then the area enclosed by the initial split contour collapses and the rear side of any resulting parcel will be directly adjacent to another parcel. If the d_{offset} value is sufficiently small, the initial split contour defines a closed inner *patio* region, with no direct access to roads. This inner region may be disconnected if the initial block is non-convex, and can be further partitioned using an arbitrary subdivision style. While setting an infinite value for d_{offset} is typical and better complies with the first parcel variety, inner *patios* are not uncommon and we designed our skeleton-based subdivision to also support them. The contour $C'(B)$ is calculated (via the CGAL library [cga]) by a partial application of the straight skeleton to $C(B)$ (Figure 3), i.e., by computing the intersection of the roof model of [AAAG95] with a horizontal plane at a specific height.

The arcs of the skeleton application specify the division of the region between $C(B)$ and $C'(B)$ into a set of strip polygons. We initially refer to these as *α-strips*, to differentiate them from the *β-strips*, from which we have removed some diagonal edges. These strips are an intermediate value in our algorithm, representing a group of parcels with their primary frontage on the same logical street. Collectively they form a single connected region.

A strip, s_i, is a simple polygonal area within B, such that a single connected length of the polygon's boundary forms part of $C(B)$. These lengths are the *supporting edges*, $\psi(s_i)$, of s_i. A block's cyclic list of strips, $LS(B) = s_1 \dots s_n$, is such that it covers the area between $C(B)$ and $C'(B)$ without overlap. The list $LS(B)$ is ordered counter clockwise, such that the last supporting edge of s_i and first of s_{i+1} are adjacent edges of $C(B)$. Note that we take $i+1$ to mean $(i+1) \bmod n$ in the context of the cyclic list.

We initialize $LS(B)$ from the faces of the straight skeleton

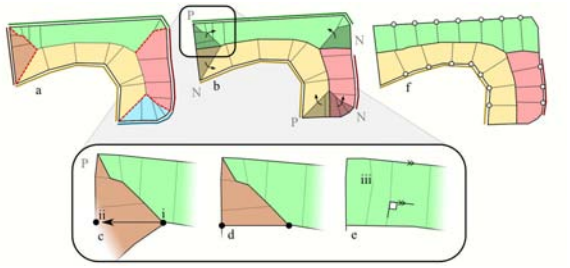

Figure 4: *The α-strips (solid colors, a) are recovered from the skeleton and logical streets (bold lines, a). This leaves undesirable diagonal edges (red dashed lines, a). Given the classification $T(v_i) \in$ None, Previous (P) or Next (N), we reassign regions (shaded, b), to create the set of β-strips. In the example (c-e) we use the classification scheme StreetLength, specifying the direction Previous. The subsequent splits are computed over these β-strips (f).*

used to compute $C'(B)$. We observe that these faces fulfill the strip properties — bounded by the arcs of the skeleton, and supported by an edge of $C(B)$. Any strip in $LS(B)$ may be combined with either of its neighbors and the union retains the strip properties; Therefore we may union adjacent faces in $LS(B)$ according to whether they lie on the same logical street. In this manner we create a single α-strip for every logical street (Figure 4a).

4.2.2. Removing Diagonal Edges

The α-strips computed from the skeleton faces suffer from diagonal edges at the intersection of logical streets (Figure 4, a). To correct these edges, we modify $LS(B)$ (Figure 4, b-e) to transfer a near-triangular region from the strip on one side of an offending edge to the strip on the other side. We refer to these corrected strips as β-strips.

Let the shared supporting vertex between each pair of α-strips s_i and s_{i+1}, be designated v_i. The shared boundary of these two strips forms the diagonal edges we are concerned with, one end of which is v_i. We provide a classification $T(v_i) \in \{Previous, Next, None\}$, to specify which of the pair of strips will gain the region, and which will lose the same region. A vertex with the property *Previous* (respectively *Next*), will assign a region to the previous (next) strip, given the counter-clockwise vertex ordering. No action is taken with a value of $T(v_i) = None$.

The values of $T(v_i)$ may be assigned by one of two schemes, determined by a per-parcel parameter specified by the user. When the angle of the supporting edges at v_i is reflex we always assign $T(v_i) = None$. For the remaining v, we choose between the following two schemes:

- *StreetWidth*. If the average width of the supporting edges of s_i is greater (respectively lesser) than those of s_{i+1} then $T(v_i) = Previous$ ($Next$).
- *StreetLength*. If the length of the supporting edges of s_i is greater (respectively lesser) than those of s_{i+1} then $T(v_i) =$ *Previous* (*Next*).

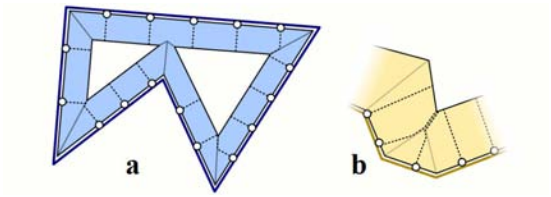

Figure 5: *The slice operation divides a β-strip's area (solid color) into parcels using the supporting edges (bold lines) to position rays (dashed lines) (a). An example showing the skeleton arcs guiding the rays (b).*

We find that in most most urban environments the parcels face the most important and thus widest street; Therefore we use the *StreetWidth* scheme by default. There are situations in which this assumption is not suited. For example, residential street patterns in which parcels prefer to face the quieter, and longer, residential streets, rather than the wider access-streets. In our experiments it proved difficult to make this distinction automatically, so we allow the user to assign this parameter manually.

Given the parameter $T(v_i)$, we calculate the direction in which to reassign the region, (Figure 4b), creating the set of β-strips. The region is removed from strip s_x, where $s_x = s_i$ if $T(v_i) = Next$, or $s_x = s_{i+1}$ if $T(v_i) = Previous$. The region is removed by identifying the point on the boundary of both s_i and s_{i+1} that is furthest from $C(B)$, (Figure 4c, i), and cutting to the nearest point on $\psi(s_x)$, (Figure 4c, ii), which can be reached by a straight line interior to the strip (Figure 4d). The region is then unioned to the strip s_y where $s_y = s_{i+1}$ if $T(v_i) = Next$, or $s_y = s_i$ if $T(v_i) = Previous$. Finally we recompute the skeleton arcs to remain perpendicular to the local edges $\psi(s_y)$, (Figure 4e, iii), in order to ensure we are able to guide the subsequent splits in the following stage (Figure 4f). After processing each pair of adjacent strips in $LS(B)$, we are left with the list of β-strips. Any strips of zero area are removed (Figure 4a, cyan and brown).

The motivation behind this process is to (i) extend the lot's centerline such that it reaches $C(B)$ as orthogonally as possible, and (ii) allow a parameter controlling the direction (street access) of the parcel at intersections.

4.2.3. Splitting Strips into Parcels

To subdivide the β-strips into parcels, a set of points are sampled approximately equidistantly on $\psi(s_i)$ (Figure 5). Rays from these points, perpendicular to the nearest edges of $\psi(s_i)$, split the β-strip into parcels. The distance between the points is normally distributed around $(W_{min} + W_{max})/2$, with $\sigma^2 = 3\omega$, and clamped to the length of $\psi(s_i)$. This process adds a random displacement to the ray-origin points to create less uniform parcels. To prevent local perturbations in $\psi(s_i)$ adversely affecting the parcel geometry, we limit the rays to each skeleton face. If the ray crosses a skeleton edge, it follows the edge to the boundary of the strip (Figure 5b).

There are several special cases that are handled independently as post processing steps:

- There are situations in which the block is too shallow to accommodate the two rows of parcels assumed by the algorithm. In this case we group shallow parcels and replace them with parcels generated similarly to the skeleton subsequent split technique. An example occurs in the detail of Figure 1f, in an L-shaped lot in the bottom right corner.
- If a parcel is deep, the area may be unacceptably large. To mitigate this effect, blocks of area larger than A_{max} are split again, via a second ray cast from the street.
- Triangular parcels and parcels with small areas are repeatedly unioned with their neighbors until they are either larger than the minimum area (A_{min}), neither small nor triangular, or there is only one remaining parcel. We union such parcels with the adjacent parcel with which it shares the longest edge.

4.3. OBB-based Subdivision

This subdivision style creates parcels resembling the second prototypical parcel variety, i.e., quadrilateral parcels with and without street access, by using an adaptive spatial partitioning. To determine the initial split line, the minimum-area OBB of the block is computed. The pivot point of the split line is given by the midpoint of the largest edge of the OBB, translated by a random distance proportional to ω. The direction of the split line is given by the direction of the smallest edge of the OBB. It is worth noting, that if the block shape is near-rectangular, the middle axis of the OBB is roughly equivalent to the straight skeleton but will not be curved, or overly affected by reflex corners in the street graph. Thus, while we could use the straight skeleton to generate the initial split line, using the block's OBB turns out in practice to yield a more robust mechanism.

For subsequent recursive splits, we also found OBB-based partitioning to yield very robust and well-behaved subdivisions. At each step, each parcel is split into two smaller parcels, which are recursively split into more parcels until the area of the resulting parcels is within (A_{min}, A_{max}) or the width of the front side of the parcel is within (W_{min}, W_{max}). If a parcel resulting from a split has no street access, the direction of the split line is rotated 90 degrees about the normal vector of the plane containing the block, with probability ξ, where ξ is a user-specified attribute in $[0, 1]$ that indicates the preference for parcels with street access. Larger values of ξ result in more parcels having street access: If $\xi = 1$ street access is always guaranteed, while if $\xi = 0$ parcels with no street access are frequent. See (Algorithm 2 in Appendix A) for more details.

Our method takes into account the following additional considerations during subdivision:

- edge alignment - to increase the performance of subdivisions under interactive editing operations, we find the best-fitting OBB by fitting OBBs that align with at least one of the parcel edges,
- random seeds - to increase the stability of subdivisions under interactive editing operations, the random seeds for

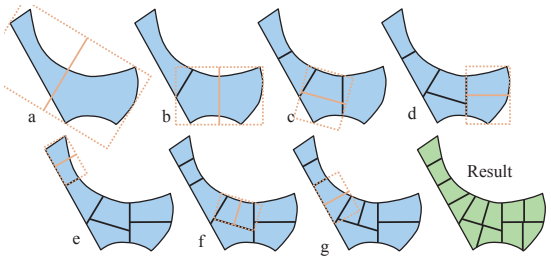

Figure 6: *Oriented bounding box subdivision. This adaptive algorithm recursively splits a parcel into two smaller parcels along the minor axis of the oriented bounding box of the original parcel. The subdivision continues until user-specified shape attributes are satisfied.*

the child parcels of a given parcel are computed before the recursive call to the subdivision function,

- snap to block contour vertices - if the split line is within a threshold distance from one of the vertices of the contour of the original block, the pivot point of the split line is set to that vertex.

This recursive approach considers only the geometry of the parent shape and road access, instead of the geometry of the entire block (Figure 6). As a result, recursive OBB subdivision is very stable in the sense that localized changes in the block geometry will typically only affect the parcels nearby the parts of the geometry that are modified.

Notice that the inner region subdivisions could alternatively be computed using the same *slice* operation as the skeleton subdivision, since the two regions obtained upon the initial OBB split conform to our definition of a strip. While the *slice* operation is suitable for the relatively thin strips generated by the initial split contours in skeleton subdivisions, applying this operation for subsequent splits in the OBB style will likely result in parcels with atypical aspect ratios. In practice, for inner regions, recursive use of OBB splits has proven to be more suitable than *slice* operations for obtaining subdivisions that comply with user-specified shape parameters.

5. Parcel Consistency under Live Editing

This section presents the problem of consistent relative location of any given parcel within a block and discusses its relevance to live interactive editing of urban models. It also proposes a solution to the parcel consistency problem that has been implemented in our parcel generation system.

5.1. Consistent Relative Location

Changes in the geometry or the topology of a road network require updating not only the road network itself (e.g., lanes, sidewalks, intersections), but also the shapes defined by the road network, specifically blocks and parcels. A system that follows the urban modeling pipeline sequentially typically proceeds by first creating the road network, then extracting the blocks as faces of a planar graph, and finally subdividing the extracted blocks into parcels (e.g., [PM01, VABW09b, WMWG09]). In this type of systems, when an urban model is regenerated, the connection between the entities of the previous model and the entities of the new model has little importance. In live editing, however, editing operations include large-scale modifications to the road network, small displacements of road network vertices or edges, and changes in the attributes of a subdivision. Many of these edits are sufficiently small and localized for a correspondence between the sets of parcels at two consecutive states to be expected and desired. For this reason, any urban modeling system that aims to support interactive editing of road networks, blocks and parcels, must support parcel location consistency. In spite of this requirement, existing urban modeling systems have not addressed or proposed solutions to the parcel consistency problem.

The consistent location of parcels under live editing operations is crucial to guarantee persistence in the relative location of specific objects in the urban models. Consider, for instance, a parcel with an associated set of parameterized rules that procedurally generates a building in the parcel. Assume that the rule parameters determine the geometry and textures of the building. If no matching between the parcels of two consecutive subdivisions of a same block was enforced, such parameters would either be lost or would be inherited at random from the parcels in the previous subdivision. By avoiding this randomness, the relative location of a building with a distinctive style is preserved.

5.2. Parcel Matching and Inheritance

Our live editing method (section 3.2) proposes a solution to the parcel consistency problem by computing a matching between the sets of the parcels of a block at two consecutive editing steps t and $t+1$ (Figure 7). The pairs of matched parcels are used to establish inheritance of parcel-specific attributes. Our matching approach uses the relative position of a parcel inside its block and a metric to compute the distance between instances of a single parcel at consecutive time steps of the editing. The chosen metric is robust to changes in the vertices of the block.

Let f be a user editing operation that modifies one or more of the following: the position of the vertices in the contour $C(B)$ of a block B; the number of vertices in $C(B)$; or the attributes used in the subdivision of B. Let B_t and B_{t+1} be two instances of a block, before and after the editing operation f, respectively. Let $L_t = \left\{l_1^t, l_2^t, \ldots, l_m^t\right\}$ be the set of m parcels that results from subdividing B_t, and $L_{t+1} = \left\{l_1^{t+1}, l_2^{t+1}, \ldots, l_n^{t+1}\right\}$ be the set of n parcels that results from subdividing B_{t+1}. The values of m and n may or may not be equal.

The goal of the consistent relative position step is to find a matching between the elements of L_t and L_{t+1} (Figure 8). The matching aims to minimize the difference between the position of a parcel in L_{t+1} and its matched parcel in L_t, relative to the vertices of the block contour $C(B)$. More specifically, for each parcel $l_i^{t+1} \in L_{t+1}$, the matching function finds

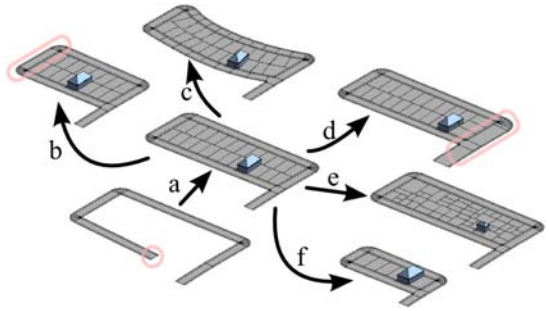

Figure 7: *A street network (bottom left) may be interactively modified in several ways (a-f). We use our matching function to compute the location of each parcel, and show the location of a given parcel (blue house) after each editing operation; moving a street graph vertex to enclose a block (a); moving an edge (b); editing the curve of two edges(c); changing the width of a street (d); changing the subdivision style (e) or scaling the street network (f).*

a parcel $l_j^t \in L_t$ such that

$$\forall_{l_h^t \in L_t} \left(d\left(l_i^{t+1}, l_j^t\right) \leq d\left(l_i^{t+1}, l_h^t\right) \right)$$

where $d : L_t \times L_{t+1} \rightarrow \mathbb{R}$ is some distance function.

We need to find an adequate distance function d robust to rigid and non-rigid transformations applied to all or some of the vertices $C(B) = \left\{b_1, b_2, \ldots, b_{|C(B)|}\right\}$ of the block. For instance, translations, rotations and scaling of $C(B)$, regardless of their magnitude, should not alter the matched pairs $\left\{\left(l_i^{t+1}, l_j^t\right)\right\}$. Relatively small translations, rotations and scaling of a subset of $C(B)$ should also not alter the pairs.

Our solution defines a distance function inspired on a generalization of barycentric coordinates for irregular, n-sided polygons, and is loosely based on [MLBD02]. This generalization exploits that the barycentric coordinates of a point on a triangle are invariable to rigid transformations, and that they exhibit a relatively small change when the positions of the triangle vertices are moved a small distance. The distance function is given by

$$d(l_i^{t+1}, l_j^t) = \sum_{k=1}^{|C(B)|} \left(\left| l_{i,k}^{t+1} - l_{j,k}^t \right| \right)$$

where $l_{i,k}^{t+1}$ is the Euclidean distance from the center of l_i^{t+1} to b_k, divided by the sum of the Euclidean distances from the center of l_i^{t+1} to all $b \in B$. It follows from this definition that $\sum_{k=1}^{|C(B)|} \left(l_{i,k}^{t+1}\right) = 1$. Our distance function allows uniquely identifying the position of a point relative to the positions of the vertices of a convex or non-convex polygon.

An assumption made by our consistency mapping solution is that both the order and the number of the vertices of B remain unchanged. One of the most common editing cases is adding a new road segment with one endpoint splitting one of the road segments enclosing B, forming a T-shaped junction and increasing the number of vertices in $C(B)$ by one. However, in our solution the new vertex is explicitly ignored

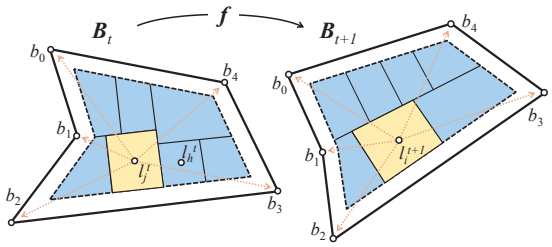

Figure 8: *Relative Parcel Position Consistency. Live urban layout editing requires computing a correspondence relation between the parcels in a block before and after an editing operation. Our approach uses a generalization of barycentric coordinates and a distance function to estimate the relative location of a parcel inside a block and a matching parcel.*

during consistency computation while it remains collinear to its adjacent vertices along the contour of B. While many other edits are possible (e.g., a non-collinear vertex is added or removed), it is often unclear, even to a human, how consistency is defined and established between a block before and after an editing operation. Thus, although arbitrary edits can be performed with our GUI, we have not explicitly addressed all cases. Similarly, certain sequences of geometric changes (e.g., scaling a block down and then back up) result in some parcels being removed and then added again. In order to support subdivision consistency throughout the entire editing sequence, comparison with several previous subdivision states would be required. Our user experience and system design decision is to support consistency only with respect to the previous state.

6. Results

Our approach has been used to generate parcels with different subdivision styles and attributes inside blocks of varying areas, aspect ratios and irregularity (Figure 9). Our algorithms have been implemented within CityEngine, a large software application for 3D city modeling [Cit]. All of the parcels in our results can be generated from scratch in under 3 seconds, and the subdivision of one block after local editing can be done at interactive rates of between 1 and 10 milliseconds per block.

In order to evaluate how well our method reproduces parcel configurations of real-world urban spaces, we present for each result a visual and statistical comparison to subdivisions in areas of selected cities. The evaluation process consists of (i) choosing a set of blocks from an existing city for which GIS parcel data is available, (ii) automatically extracting per-block descriptive statistics of the subdivision attributes of the *observed* parcels in each block, including the mean and standard deviation of the parcel area ($\bar{A}$, s_A) and minimum width ($\bar{W}$, s_W), (iii) loading the blocks into our application and assigning the extracted set of per-block parcel attributes (Section 4.1) and a subdivision style to each block, (iv) subdividing the blocks into parcels using the implementation of our method, and (v) automatically extract-

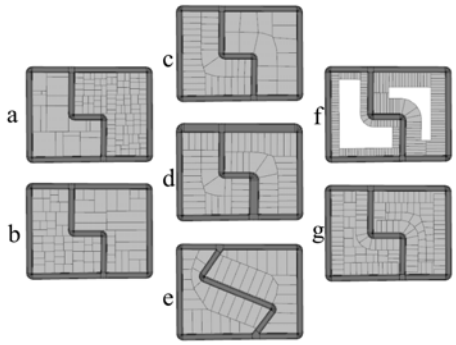

Figure 9: *Examples of varying different attributes of OBB (a,b) and skeleton, subdivision (c-g); a large and small difference between A_{min} and A_{max} (a); the effect of enforcing street access (b); low or high lot-width (c); editing the street widths, to change $T(v_i)$ (d); editing the criteria for minimum-area lot removal (e); low or high value of d_{offset} (f); a higher value of ω, and a variety of subdivision styles in the* patio *region (g).*

ing descriptive statistics of the attributes of the *procedural* parcels in each block, including the area, minimum width, aspect ratio and number of neighbors. The subdivision attributes used in (iii) are assigned automatically to each block and computed from the descriptive statistics extracted in (ii) as follows: $A_{min} = \bar{A} - ks_A$, $A_{max} = \bar{A} + ks_A$, $W_{min} = \bar{W} - ks_W$, $W_{max} = \bar{W} + ks_W$, where k is a positive constant that in our examples was set to 2. The subdivision style for each block or selected region of a city is determined by the user. The comparison between the *procedural* parcels and the *observed* parcels is made by visualizing the parcel configurations and the aforementioned statistics, either in color-coded maps of the parcels, or in superposed, non-normalized frequency histograms. We also compare our results to those obtained by [WMWG09] for the same set of input blocks (Figure 12 in Appendix C).

We have applied our subdivision method to historic cities with chaotic street and subdivision patterns (Figure 13 in Appendix D), and have achieved high visual similarity between real and procedural parcel configurations by manually adjusting attributes such as the split irregularity (ω). However, because of the stochastic component in the layouts of these cities, we limit our automatic evaluation to planned cities with street patterns of different geometric complexity.

Figure 1 shows the results of subdividing a set of blocks with OBB and skeleton styles. The observed blocks are located in a mixed use suburban area and are mostly rectangular with both straight and warped edges as a result of the geometry of the surrounding roads (Figure 1a,b). The set of blocks exhibits significant variability in both the area, the aspect ratio, and the minimum width of the parcels. This variability is visualized in the color-coded map (Figure 1b), and in the black-and-white map (Figure 1c) showing on top of each block four descriptive statistics of the observed parcels in that block. In the color-coded maps, the parcels with aspect ratios close to one are shaded red, and the shading becomes closer to green as the aspect ratio increases. A similar distribution of colors in the observed and the procedural parcels indicates high similarity. The parcels generated by our method are shown in Figure 1f.

At a small scale, the similarity between the parcels generated by our method and the parcels observed in the real world is evidenced by close visual inspection of the parcel configuration inside individual blocks, of either nearly-rectangular or warped and irregular shapes. At a larger scale, the similarity can be seen in the close resemblance between the spatial distribution of colors in (Figure 1f), and in the good match between the frequency histograms (Figure 1d,e) of aspect ratio and area for the *observed* parcels (red) and for the *procedurally generated* parcels (blue). All generated parcels have dimensions and aspect ratios that are adequate for containing buildings (Figure 1g). Following [MWH*06], we used a shape grammar to create procedural trees and buildings on the generated parcels (Figures 1i and 14 in Appendix E).

Figure 10 shows a set of blocks subdivided using skeleton subdivision with offset. The observed blocks are located in a low-density residential suburban area and have highly irregular shapes as a result of the warped roads with frequent loops and cul-de-sacs (Figure 10a,b). The large inner regions in some blocks are covered by golf holes. The similarity between the observed subdivision and the procedural parcels can be seen in the color-coded maps (Figure 10c,f) and in the superposed histograms (Figure 10d,e). In this example, the area under the blue (procedural) histograms is greater than the area under the red (observed) histograms, which indicates that the number of generated procedural parcels is greater than the number of observed parcels. The reason for this mismatch between parcel counts is that the number of parcels is not directly specified by the user, but rather results from the chosen values for the subdivision attributes (in this case, parcel area and offset depth). Notice that one observed parcel close the center of the map is a statistical outlier in terms of area that is not captured by the extracted descriptive statistics, and is thus not reproduced by our method.

Figure 11 in Appendix B shows the results of skeleton subdivision applied to several blocks with nearly-rectangular and irregular contours (Figure 11a,b). In this result, the colors in the map indicate the parcel areas. In both the observed and procedural parcel configurations the parcels in the middle blocks are generally larger than those in the side blocks.

7. Conclusion

We have presented an interactive method for procedural generation of parcels inside city blocks. Our approach generates spatial configurations of parcels with high similarity to those observed in real-world cities, and supports the consistent location of parcels relative to their containing blocks under live editing operations.

Our method can be extended in several ways. One of them

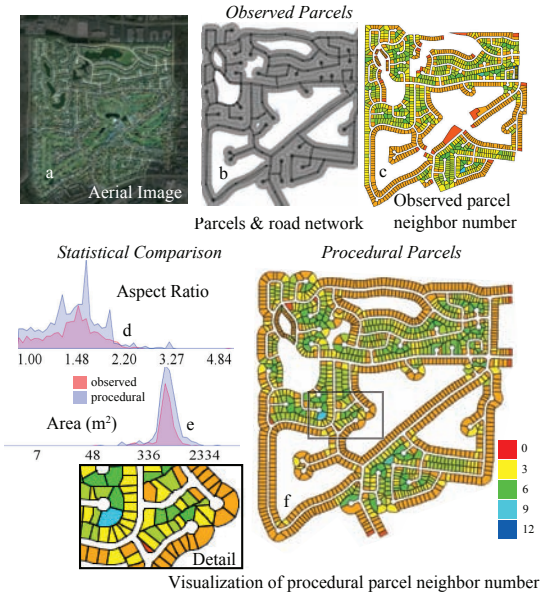

Figure 10: *Skeleton Subdivision with Offsets. Residential blocks are subdivided in offset style. The similarity between the observed parcels (a,b,c) and the procedural parcels is demonstrated in the color-coded maps showing the number of neighbors of each parcel (f), and in the histograms of other geometric attributes (d,e).*

is to use machine learning techniques to automatically capture the subdivision style of a block. A second line of work is to integrate our block subdivision algorithms with computer vision methods that extract blocks and parcels from orthographic and oblique-angle aerial imagery. The computed information could guide the parcel extraction process towards solutions that are statistically more likely to occur.

8. Acknowledgements

This research is partially supported by the European project V-City (ICT-231199-V-CITY) and by NSF IIS 0964302. We sincerely thank the reviewers for their insightful feedback.

References

[AAAG95] Aichholzer O., Aurenhammer F., Alberts D., Gärtner B.: A novel type of skeleton for polygons. *J. UCS 1*, 12 (1995), 752–761. 4, 5

[AVB08] Aliaga D. G., Vanegas C. A., Benes B.: Interactive example-based urban layout synthesis. *ACM Transactions on Graphics 27*, 5 (2008), 1–10. 3

[Car03] Carmona M.: *Public places, urban spaces: the dimensions of urban design*. Architectural Press, 2003. 2

[CEW*08] Chen G., Esch G., Wonka P., Müller P., Zhang E.: Interactive procedural street modeling. *ACM Transactions on Graphics 27*, 3 (2008), 1–10. 3

[cga] Cgal, Computational Geometry Algorithms Library. http://www.cgal.org. 5

[Cit] Esri CityEngine. http://www.esri.com/software/cityengine/index.html. 8

[Cur97] Curdes G.: *Stadtstruktur und Stadtgestaltung*. Kohlhammer, 1997. 2

[GPMG10] Galin E., Peytavie A., Maréchal N., Guérin E.: Procedural generation of roads. *Comput. Graph. Forum 29*, 2 (2010), 429–438. 3

[HKS08] Halatsch J., Kunze A., Schmitt G.: Using shape grammars for master planning. In *Design Computing and Cognition '08*, Gero J. S., Goel A. K., (Eds.). Springer Netherlands, 2008, pp. 655–673. 3

[LSWW11] Lipp M., Scherzer D., Wonka P., Wimmer M.: Interactive modeling of city layouts using layers of procedural content. *Comput. Graph. Forum 30*, 2 (2011), 345–354. 3

[LWW08] Lipp M., Wonka P., Wimmer M.: Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics 27*, 3 (2008), 1–10. 3

[Mar09] Marshall S.: *Cities design and evolution*. Urban design and planning. Routledge, 2009. 3

[MLBD02] Meyer M., Lee H., Barr A., Desbrun M.: Generalized barycentric coordinates on irregular polygons. *Journal of Graphics Tools 7* (2002), 13–22. 8

[MWH*06] Müller P., Wonka P., Haegler S., Ulmer A., Van Gool L.: Procedural modeling of buildings. In *ACM SIGGRAPH 2006* (2006), pp. 614–623. 3, 9

[PCDS04] Panerai P., Castex J., Depaule J., Samuels I.: *Urban forms: death and life of the urban block*. Architectural Press, 2004. 2

[PM01] Parish Y. I. H., Müller P.: Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH* (2001), pp. 301–308. 2, 3, 4, 7

[PPC08] Parolek D., Parolek K., Crawford P.: *Form-based codes: a guide for planners, urban designers, municipalities, and developers*. J. Wiley & Sons, 2008. 2

[TKD10] Thadani D., Krier L., Duany A.: *The language of towns & cities: a visual dictionary*. Rizzoli, 2010. 2

[VABW09a] Vanegas C. A., Aliaga D. G., Benes B., Waddell P.: Visualization of simulated urban spaces: Inferring parameterized generation of streets, parcels, and aerial imagery. *IEEE Transactions on Visualization and Computer Graphics 15*, 4 (2009). 3

[VABW09b] Vanegas C. A., Aliaga D. G., Beneš B., Waddell P. A.: Interactive design of urban spaces using geometrical and behavioral modeling. *ACM Trans. Graph. 28* (December 2009), 111:1–111:10. 3, 7

[VAW*10] Vanegas C. A., Aliaga D. G., Wonka P., Müller P., Waddell P., Watson B.: Modeling the appearance and behavior of urban spaces. *Computer Graphics Forum 29*, 1 (2010), 25–42. 2

[WCP*11] Wickramasuriya R., Chisholm L. A., Puotinen M., Gill N., Klepeis P.: An automated land subdivision tool for urban and regional planning: Concepts, implementation and testing. *Environmental Modelling & Software*, 0 (2011), –. 3

[WD11] Walker D., Daniels T.: *The Planners Guide to CommunityViz: The Essential Tool for a New Generation of Planning*. Orton Family Foundation Books. American Planning Association, 2011. 3

[WMWG09] Weber B., Müller P., Wonka P., Gross M.: Interactive geometric simulation of 4D cities. *Computer Graphics Forum (Proceedings of Eurographics) 28*, 2 (2009). 3, 7, 9, 13

[WWSR03] Wonka P., Wimmer M., Sillion F., Ribarsky W.: Instant architecture. *ACM Transactions on Graphics 22*, 3 (2003), 669–677. 3

Appendix A: Pseudocodes

Algorithm 1 Straight skeleton subdivision

subdivSkeleton(B)

$L \leftarrow \emptyset$
$SS \leftarrow$ *computeSkeletonOffset*$(C(B), d_{offset})$
$LS \leftarrow \emptyset$
for each face $f \in SS$ **do**
  Append convertToStrip (f) *to LS*
end for
$LS2 \leftarrow$ mergeOnLogicalStreets (LS)
$LS3 \leftarrow$ *fixDiagonalEdges* (LS2)
for each strip s in LS3 **do**
  slice (s)
end for
processSmallLargeOrTriangularLots(LS, A_{min}*,* A_{max}*)*

---

fixDiagonalEdges(LS)

for each strip $s_i \in LS$ **do**
  $v_i \leftarrow$ vertex between s_i and s_{i+1}
  $t \leftarrow$ triangular portion at v_i
  if $T(v_i) = Previous$ **then**
    assign t to s_i
  end if
  if $T(v_i) = Next$ **then**
    assign t to s_{i+1}
  end if
end for

---

slice(s)

$origins \leftarrow$ sample $\psi(s)$ by $n((W_{min} + Wmax)/2, 3\omega)$
$remainder = \bigcap$ offset faces of s
for each point $p \in origins$ **do**
  $normal \leftarrow$ average normal of B near p
  Create a ray, r, from p, in direction $normal$
  $[left|right] \leftarrow$ slice $remainder$ by r
  Append $left$ to L
  $remaining \leftarrow right$
  Append $remaining$ to L
end for

Algorithm 2 OBB subdivision

subdivOBB(B)

$L \leftarrow \emptyset$
recSubdivOBB$(C(B))$

---

recSubdivOBB(l)

if $area(l) \notin (A_{min}, A_{max})$ **and** $frontSideWidth(l) \notin (W_{min}, W_{max})$ **then**
  $s \leftarrow computeSplitLine(l)$
  $[l_A, l_B] \leftarrow split(B, s)$
  if l_A or l_B have no street access **then**
    Rotate s 90 degrees about the normal vector of the plane containing B, with probability ξ
    $[l_A, l_B] \leftarrow split(B, s)$
  end if
  $recSubdivOBB(l_A)$
  $recSubdivOBB(l_B)$
else
  Append l to L
end if

---

computeSplitLine(l)

$OBB \leftarrow computeOBB(l)$
Let the direction of l be the direction of the shortest side of OBB
Let the pivot point of l be the middle point of OBB
Apply a random translation of magnitude $\omega \cdot d_{OBB}$ to the pivot point of l, where d_{OBB} is the length of the shortest side of OBB

Appendix B:

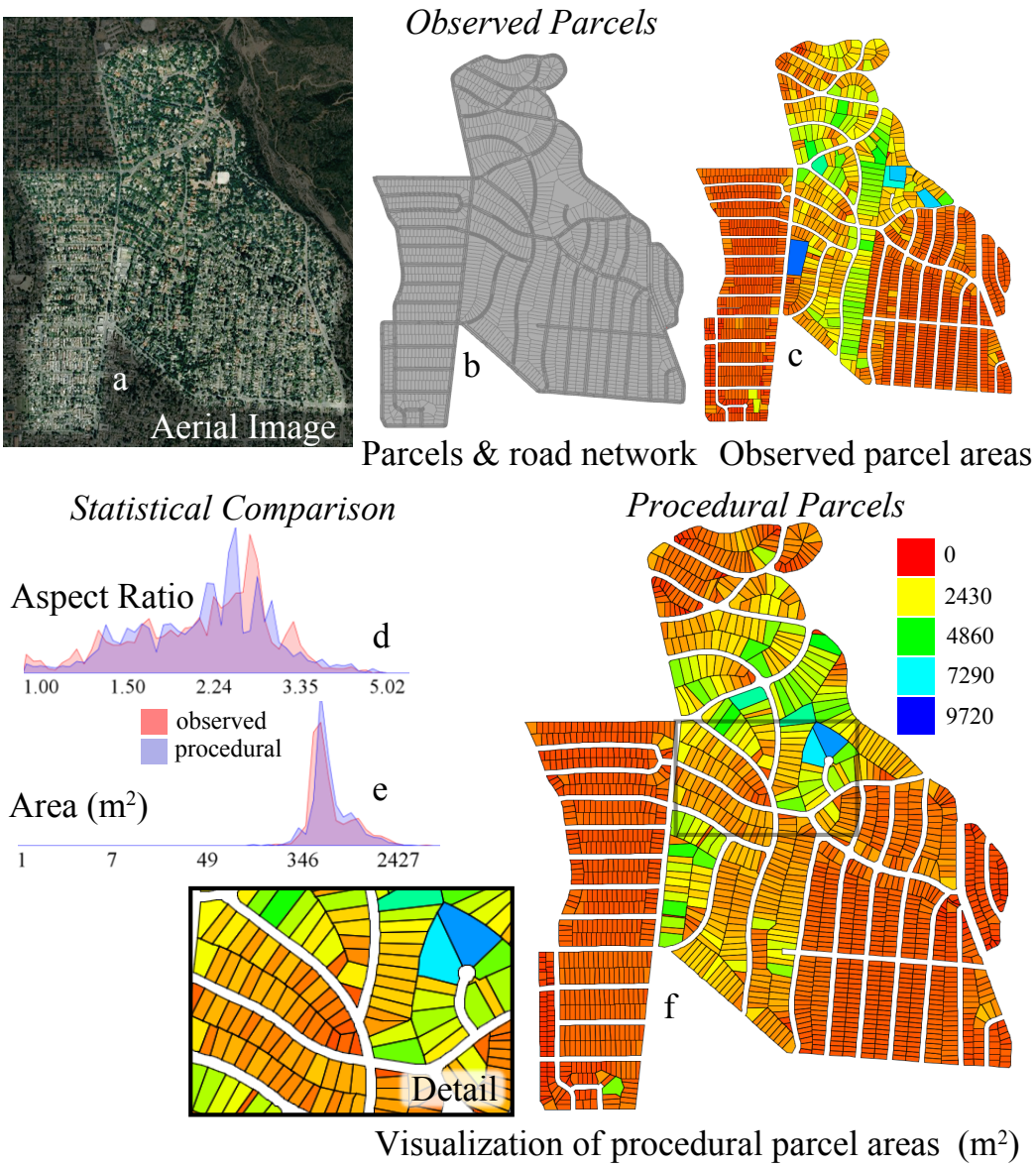

Figure 11: *Skeleton Subdivision. Residential blocks are subdivided in skeleton style. The color-coded maps (d,f) and the histograms (d,e) show the similarity between the areas of the observed parcels (a,b) and the procedural parcels (f). The two large blocks towards the middle of the map show a straight subdivision line that seems atypical when compared to those of the surrounding blocks, and which is likely to have originated in pre-existing city infrastructure or political divisions. These are also the two blocks where the similarity between observed and procedural parcels is the lowest. Nevertheless, the overall resemblance between the real-world parcels and the parcels generated by our approach is again visible at a large and at a small scale in the color maps (c,f) and in the histograms (d,e).*

Appendix C:

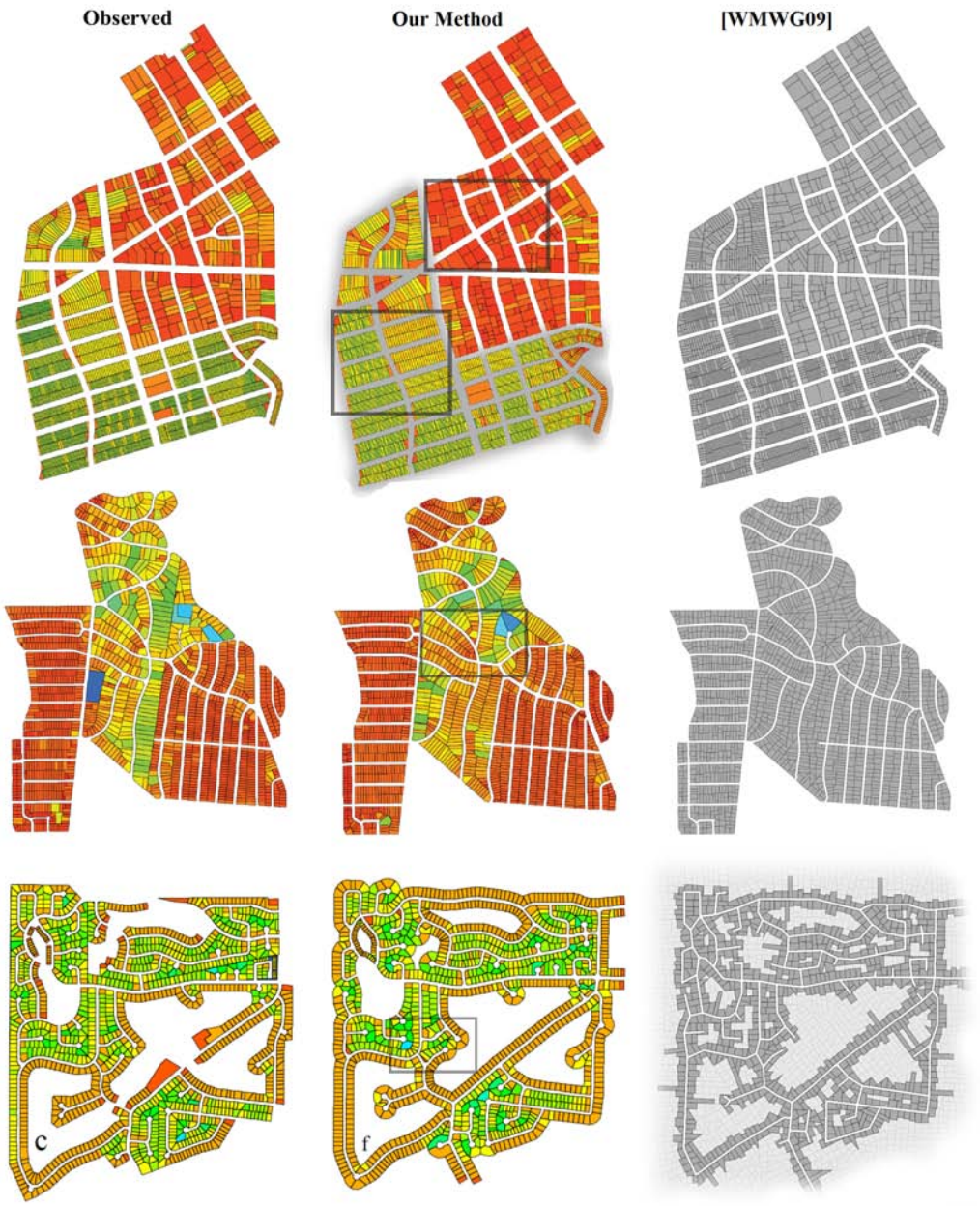

Figure 12: *Side by side comparison between real-world (left) and procedural (middle) parcels. The recursive split operations in [WMWG09] (right) often create unrealistic parcel "spikes" due to the rudimentary split orientation handling and their main drawback is the lack of user-friendly controllability. As a consequence, the geometry and layout of the resulting parcels are noticeably different from those observed in the real world data.*

Appendix D:

Figure 13: *Subdivision of blocks into parcels in the ancient Roman town of Pompeii. A comparison between the actual parcels of the city (green) and the procedural parcels generated by our OBB-based algorithm (blue) is shown (Top). Procedural 3D buildings and trees are created on top of these parcels (Bottom).*

Appendix E:

Figure 14: *Procedural houses, trees placed on our procedural parcels from Fig 1.*