Content-Aware Re-targeting of Discrete Element Layouts

Stefan Hartmann Björn Krüger Reinhard Klein
University of Bonn
Institute for Computer Science II
Friedrich-Ebert-Allee 144
Germany, 53113 Bonn
{hartmans,kruegerb,rk}@cs.uni-bonn.de

ABSTRACT

Example-based modeling is an active line of research within the computer graphics community. With this work we propose a re-targeting scheme for polygonal domains containing a layout composed of discrete elements. Re-targeting such domains is typically achieved employing a deformation to the initial domain. The goal of our approach is it to compute a novel layout within the deformed domain re-using the discrete elements from the initial layout. We show that the deformed interior of the initial domain can guide the layout algorithm that places the discrete elements. We evaluate our algorithm by re-targeting several challenging city blocks and the results confirm that generated layouts are visually similar to the original ones.

Keywords

example-based synthesis, data-driven re-targeting, layout algorithms.

1 INTRODUCTION

The automatic generation of high quality content is an important research topic in computer graphics. Especially with the emergence of a large variety of publicly available content repositories and community mapping services, modeling by example has increasingly become an active line of research in recent years. This simple and efficient modeling metaphor allows generating novel content either from a single exemplar (*e.g.* a small texture patch) or by first analyzing a larger set of input data (*e.g.* a model collection) to 'learn' structural information, and use the learned knowledge to guide automatic content generation algorithms in a second step.

With the work at hand we contribute a simple but efficient modeling metaphor for re-synthesizing existing layouts that contain sets of discrete element arrangements. Examples for this type of layouts are city blocks (Figure 1a), where building footprints and/or community places serve as discrete atomic elements that are primarily arranged along road segments. Other typical examples are floor plans composed by various types of connected discrete rooms (Figure 1b), interior rooms filled with discrete furniture instances or other types of objects composed of atomic substructures.

Modeling such content from scratch is a tedious task and typically much effort is invested to achieve high quality results. When existing content shall be re-used it shall be 'bended' or re-structured in a way to fit new demands. However, re-targeting and editing such structures usually involves continuous deformations to the initial layout, thus distorting the original arrangements and even their discrete elements. Instead of deforming and repairing distortions, our goal is to re-synthesize a plausible layout that resembles style of the original layout locally and even globally using discrete atomic building blocks. Our key insight for synthesizing such a new layout that mimics the style of the original one is to feed the synthesis technique with a guidance map, that encodes the style present in the initial layout. Such a guidance map can be retrieved from a modification of the initial domain, such as the appliance of a continuous deformation to the initial layout elements. Using this

(a) City block with discrete building footprints.

(b) Office floor plan with discrete room instances.

Figure 1: Real world examples of domains with discrete element arrangements. Both examples are taken from *OpenStreetMap*.

guidance has two major advantages: First, it allows our synthesis technique to resemble the global style of initial layout and it even allows introducing discrete copies of the original elements to maximize the layout compliance, *i.e.* the local style. Second, it guides the algorithm during the exploration of the layout space and allows for efficient pruning this typically large search space.

The main contributions of our work are in particular:

- We present a simple but efficient labeling algorithm for re-targeting polygonal structures incorporating a content-aware continuous guidance map for re-synthesizing arrangements groups.
- We show that the usage of such a guidance map transfers the style of the initial layout to the synthesized one globally and even locally.
- We present an in-depth evaluation on a large set complex layouts and structures containing a varying number of discrete element arrangements, namely urban city blocks.

2 RELATED WORK

City Block Synthesis: An important sub-step in urban modeling is computing interior layouts for city blocks or to be more specific generate parcels to provide space for additional urban sub-structures: buildings or parks. Usually these parcels are generated procedurally, by applying various techniques. In Parish and Müller [14], space for buildings is distributed, employing a subdivision scheme on the oriented bounding box of the city block outline. Aliaga *et al.* [1] compute a voronoi subdivision from perturbed points sampled along the principal direction of the city blocks' oriented bounding box. Vanegas *et al.* [16] generalize parcel generation, by introducing an adapted subdivision scheme on the oriented bounding box of the city block. Their second approach smartly merges regions resulting from the medial-axis to subdivide the city block into larger regions. These are then tessellated along the adjacent street edges.

In Yang *et al.* [18] an initial parcel layout is selected from a set of manually prepared template patterns. To achieve a tight packing of the city block, the template pattern is expanded in discrete steps and the final parcel layout is computed by a sophisticated warping approach. In contrast to these procedural approaches, our goal is to synthesize a new layout that resembles the style of an existing one. In terms of urban modeling, we want to compute a city block layout, that resembles the style of a real world city block *e.g.* taken from *OpenStreetMap*.

Model Re-targeting: Re-synthesis of novel 3D models from small pieces of exiting exemplars was early studied by Merrel *et al.* [8]. They use adjacency rules to guide their model synthesis technique to preserve the local look/style. They generalized their method, incorporating local object self-similarity and non-uniform grids in [9, 10]. The analysis of self-similarities to derive construction rules as context-free grammar that captures the larger object structure was introduced in [2]. In cases where the global structure of one or multiple methods is present *e.g.* provided as scene graph, conforming grammars can be 'learned', being able to reproduce mixtures of exemplar models as demonstrated by Talton *et al.* [15].

Apart from grammar based methods Lin *et al.* [6] suggest to re-target architectural models with dominating irregular structures. They manually annotate and construct an axis aligned bounding box hierarchy, used to automatically extract dominant/longest sequences. The model is then iteratively modified by repeating or scaling the geometry located inside the bounding boxes along these sequences. Very recently Wu *et al.* [17] presented a promising approach for interactive model re-targeting. They derive a set of principal re-targeting directions from descriptors measuring self-similarities. The model is replicated inside a 3D grid, and a multi-label graph cut optimization technique is used to compute the re-targeted result. Our approach in contrast does not rely on computation of self-similarities. It mainly relies on the existence of a guidance map, that captures the style present in the original layout in a higher level.

Orthogonal to our approach are discrete element layout algorithms that use stochastic optimization techniques, *e.g.* Markov Chain Monte Carlo (MCMC). These methods learn object relations from a set of examples [21], are feed with well-known design guidelines [11], or use object relations encoded in factor graphs [20, 19] in order to synthesize novel element layouts. However, this kind of algorithms rely on carefully designed probability distributions and typically are computationally expensive. Thus, they are usually not suited for interactive exploration of the solution space.

3 DOMAIN ANALYSIS

Herein we now explain our approach for re-targeting existing content. First, we briefly explain preliminaries to understand the basic data types. Second, we give a definition of the problem we want to solve. A mathematical formulation to tackle the problem is then introduced in Section 4.

3.1 Preliminaries

A large variety of real-world structures can be abstracted as a set of polygons enveloped by a closed boundary. Typical examples for such real-world structures are building footprints located inside a city

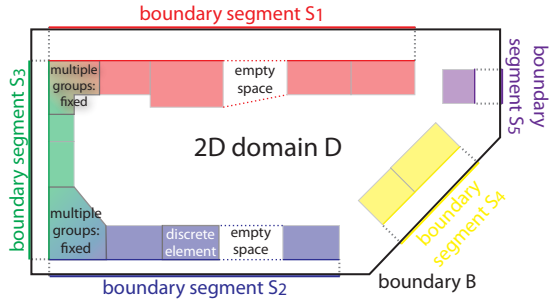


Figure 2: Abstraction of a complex scene as 2D domain D with a set of discrete elements. The domain is enveloped by a boundary B split into subregions S_i, each assigned with a set of discrete elements.

block or rooms located in a floor plan (Figure 1). We define $\mathscr{P} = \{P_1, \ldots, P_n\}$ as the set of n polygons, each representing a 2D outline of discrete elements, placed within a complex scene. Each of them may carry a set of application specific annotations. These elements located in a scene are typically enveloped by a closed curve denoting the boundary $B \in \mathbb{R}^2$ of the scene. The area enclosed by B represents the scene itself and is a planar domain $D \in \mathbb{R}^2$. Inside this area, we expect that the elements are arranged into a set of m groups $\mathscr{G} = \{G_1, \ldots, G_m\}$, each containing a subset of the discrete elements (Figure 2). However, the initial layout does not necessarily have to be a tightly packed layout, *i.e.* elements placed directly next to each other, there may be empty space in between the discrete elements

3.2 Detecting Arrangement Groups

Before we state the problem definition, we first need to analyze the atomic entities and merge them into a set of arrangement groups G. In our setting we expect that the discrete elements are located within an application specific distance near the boundary B. This curve might be further split into a set of boundary segments, separating the curve into l smaller entities $B = \{S_1, \ldots, S_l\}$. The segments $S_j \in B$ typically represent a road segment of a city block or a wall inside a room. In order to merge the elements into groups they need to be assigned to the segments S_j along the boundary B. In order to decide which element is assigned to a certain group we determine the Hausdorff distance $h(P_i, S_j) \forall S_j \in B$ between the outline of the discrete element P_i and the boundary segments $S_j \in B$. In case $h(P_i, S_j) \leq \tau$ falls below an application specific threshold ($\tau = 25.0$ meters in our city block application), the discrete element instance is assigned to that certain arrangement group G_j that corresponds to a boundary segment S_j. It typically happens that a few elements will be assigned to multiple boundary segments, this is an indicator that the element is located at the end of an arrangement group and might need special, application specific, treatment (Figure 2).

Most of the elements are typically assigned to at least one boundary segment S_j. The count of assignments of a discrete element P_i to different S_j is used to deduce three behavioral attributes, *i.e.* the element type, for the element instances:

- *fixed* (F): elements with two or more assignments, are not allowed to be repeated.
- *repeatable* (R): elements with a single assignment, allowed to be placed multiple times.
- *empty space* (E): element that serves as fill region, preferably placed in regions, where no discrete elements covers the space within the group G_j. Note: that element type is a virtually generated element instance and is automatically assigned to each group. In our examples we used a rectangular domain with a fixed dimension $w \times d$ with $w = 1.0$ *meter* and d being the average depths of the discrete elements assigned to a G_j.

3.3 Problem definition

Re-targeting existing models can be achieved by applying various operations to the initial object or domain. Our notion of re-targeting is to apply a deformation to the boundary B or one of its segments S_j. This implies, that the interior of the initial domain is affected by the deformation by applying a map $\phi : D_{in} \rightarrow D_d$, where D_{in} denotes the initial scene domain that shall be re-targeted and D_d denotes the deformed domain. Computing such mappings is well researched and can be achieved by employing different techniques *e.g.* the well known *Mean Value Coordinates* [4] or (quasi)-conformal mappings [7]. In the work at hand we used *Mean Value Coordinates* to realize the mapping $D_{in} \rightarrow D_d$. When ϕ is directly applied to the discrete elements and their corresponding geometry located within the scene, visually unappealing artifacts might be the consequence *i.e.* the elements get distorted and unnaturally stretched/warped. Typically, such effects might be reduced by applying smart deformation techniques, however, this claims for additional knowledge about the objects' structure, which is not the scope of this work.

Another possibility to reduce deformation artifacts is treating the object as a single point, represented by its centroid. This might avoid object deformations, however, will usually lead to different kinds of artifacts such as overlaps, penetration or getting objects dissolved from their initial grouping. In order to circumvent such artifacts described above our major goal is to re-use the undeformed building blocks of the initial scene layout and compute a novel layout which preserves the style of the initial layout. In order to compute a novel layout that mimics the style of the original one we utilize a

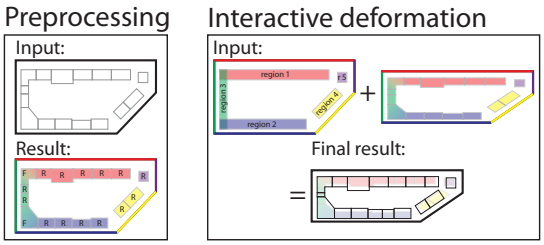


Figure 3: Illustration of our re-targeting concept: As preprocessing step a domain is analyzed and its discrete elements are assigned to arrangement groups (left). When interactively editing the domain, deformations are applied to the boundary. The interior of the domain is deformed and serves as prior to our iterative synthesis technique, that computes a novel layout guided by the deformed discrete elements.

guidance map. The guidance map is a set of deformed discrete elements $\mathscr{P}'$. It results from applying the map ϕ to the interior, *i.e.* the discrete elements $\mathscr{P}$ of the initial domain $\phi : \mathscr{P} \rightarrow \mathscr{P}'$. Although the elements are deformed the overall deformed layout still mimics the style of the initial layout. Therefore, we utilize $\mathscr{P}'$ to guide the algorithm in order to compute a novel layout.

4 RE-TARGETING ARRANGEMENTS

Herein, we present our solution for re-targeting domains that preserves the layout style of the initial domain. Before we present a mathematical formulation to tackle the problem, we comprehensible explain our notions of style that can be preserved by our algorithm. The notion of style covers a wide range of attributes, therefore, we focus on the following three properties in the work at hand:

1. element groupings present in the initial domain shall be preserved,
2. elements prefer to be located at similar relative positions, and
3. elements prefer their initial neighborhood.

In our setting, the term neighborhood does not express the local similarity of geometric attributes. Our goal is that we simply want to preserve the local element ordering found in the initial domain. Up to now we have discussed the input data and its segmentation into a set of discrete elements and their grouping into arrangement groups G_j. From now on, we focus on formulating the re-targeting problem as an optimization problem which can be solved efficiently, thus allowing to interactively explore solutions for different domain deformations.

4.1 Problem formulation

Re-targeting of arrangement groups: The key idea for re-synthesizing a deformed domain is the formulation as labeling problem. This type of problem formulations are well studied in the computer vision community and several efficient solution techniques exist [3]. While the formulation of a labeling problem is typically done for 2D problems *e.g.* image segmentation, we exploit the sub-structures found in our problem setting, namely the linear element arrangement groups. This allows us to solve the layout problem employing efficient algorithms. These considerations result in the following equation to be minimized:

$$L(f_1,\ldots,f_N) = \sum_{k=1}^{N} O(f_k) + \sum_{k=1}^{N-1} \delta(f_k, f_{k+1}) \quad (1)$$

The f_k's are representations of discrete elements that might get placed at different discrete positions located within the parameter domain of the boundary segments S_j. Thus, a $f_k = (P_i, m, d)$ is a tuple that consists of a reference to a discrete element P_i, its current discrete position m and d being the length of P_i projected onto the boundary segment S_j. In our setting the d's are rounded to be a multiple of 1.0 *meter*. The energy we want to minimize in order to find an optimal sequence of discrete elements is given in Equation 1. This energy is composed of two terms, that can be mapped to our notion of style preservation outlined above. The data term $O(f_k)$ measures the cost for assigning an element f_k to a specific location m. Thus, it penalizes elements that will be placed at positions, where they do not overlap with their corresponding distorted instance P_i' within the guidance map. From a different view it can also be seen as prior term pulling the undeformed elements towards the position, where the deformed instance of the particular discrete element is located. We designed $O(f_k)$ to be the area difference between the undeformed element P_i and the area of the intersection of both undeformed P_i and deformed P_i' defined as follows:

$$O(f_k) = A(P_i) - A(P_i \cap P_i')$$

Thus $O(f_k)$ penalizes space not covered between its discrete element P_i and its corresponding deformed P_i'. Further it depends on the current location m, where f_k might be positioned inside the result sequence and it changes every time the domain is modified. The transition between two consecutive discrete elements is penalized by $\delta(f_k, f_{k+1})$ that is defined as follows

$$\delta(f_k, f_{k+1}) = \begin{cases} \alpha & \text{if } f_k \in E \wedge f_{k+1} \in E \\ (1-\alpha) & \text{if } f_k = f_{k+1} \\ 0 & \text{if } f_k \neq f_{k+1} \end{cases}$$

This function encourages omitting two identical discrete elements P_i getting placed next to each other and instead prefers different instances. In order to achieve

 ISBN 978-80-86943-65-7

continuous empty space regions in case if present, we penalize two consecutive empty space elements less severe, than two identical P_i. In our examples we used $\alpha = 0.05$. As described in Section 3.2 we exploit that, elements can be grouped to arrangements along segments of the boundary polygon. For each of them the optimization problem boils down to computing a sequence of elements along a curve, that in total fulfills a certain minimal length M along the parameter domain. In order to employ an efficient algorithm, that minimizes Equation 1 the f_k's are restricted to be placed at discrete positions along the curve. The discretization depends on the specific application (we used discrete steps of 1 meter in our examples). This allows us to reformulate the objective function in the following, recursive way:

$$\begin{aligned} L[f_1, m_{f_1}] &= O(f_1) \\ L[f_k, m_{f_k}] &= O(f_k) + \min(L[f_{k-1}, m_{f_{k-1}}] + \delta(f_{k-1}, f_k)) \end{aligned} \tag{2}$$

We solve this recursive formulation using a graph based dynamic programming approach adapted from Lefebvre *et al.* [5]. In our setting we need to incorporate both, node costs O, *i.e.* overlapping cost and edge costs δ, *i.e.* 'concatenation costs' when growing the graph implicitly. We can terminate the growing in case if the sequence length m including the current top node of the queue is $m \geq M$. As the graph nodes are managed by a priority queue sorted by current sequence cost, the current top f_k^* that satisfies $m \geq M$ node is part of the optimal sequence.

This optimal sequence

$$\underset{m^* \geq M}{f_k^*} = \operatorname{argmin}(L[f_k, m_{f_k}] + \delta(f_{k-1}^*, f_k)) \tag{3}$$

minimizing Equation 1 can then be found, by tracing back the optimal predecessors from f_k^* back to the start of the sequence.

4.2 Iterative Synthesis

While the proposed formulation in Section 4.1 only guarantees a global optimal solution in case the initial layout consists of a single arrangement group located within the initial domain, we employ a meaningful heuristic in case multiple arrangements groups exist. Instead of combining and linking them together and optimize for a globally optimal layout, we have decided to choose an iterative algorithm, that synthesizes each arrangement group separately. This has two major reasons: (1) one of our design goals is interactivity, (2) a global optimization scheme, might affect the performance severely. We decided to sort the groups by their cardinality of the discrete elements assigned to them. Thus, the synthesis technique starts with the most dominant group located inside the initial layout. In order to avoid overlaps with already synthesized sequences, we constrain the $j-th$ arrangement group by the previous $k \in [j-1, \ldots, 1]$ that potentially placed elements (fixed, repeatable) will not penetrate the convex hull of existing sequences.

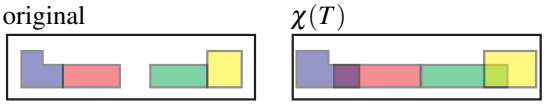


Figure 4: Concept modified guidance map. Left: the original map. Right: The guidance map is transformed by applying anisotropic scaling to the individual elements. Due to this deformation the elements in the guidance map may overlap partially. This allows for a less restrictive positioning of the elements in larger areas.

4.3 Modified guidance maps

As mentioned above the guidance map $\mathscr{P}'$ is the result of applying the map ϕ to the interior elements of the initial domain $\mathscr{P}$. Typically, the creation of such a guidance map is not restricted to a specific deformation method. It is possible to modify the size of the discrete elements in advance or apply a deformation scheme, where weights can be distributed using a simple brush metaphor e.g Möser *et al.* [12]. When using the guidance map $\mathscr{P}$, it might happen that the algorithm presents solutions, that may contain multiple consecutive occurrences of discrete elements. This results from the guidance map, which encourages the algorithm placing elements in regions, where they overlap with their corresponding distorted instance, is cheaper than placing them somewhere else. This may result in visually unappealing repetitions, *i.e.* the same elements is repeated multiple times (Figure 14(a)).

In order to overcome this unappealing side effect we use a modified version of the guidance map $\chi(T)$. Instead of applying the map to the initial discrete elements directly, we apply a scaling operator T to each P_i before applying the deformation map, thus $\chi(T) : \phi(T(P_i)) \rightarrow P_i''$. As a result of this scaling, the deformed discrete elements now partially overlap and, therefore, the algorithm is able to temporarily bypass the original ordering. In our examples we use anisotropic scaling along the direction of the boundary segment S_j by a factor of 1.5 and apply it to each discrete element present in the corresponding arrangement group G_j. An example for such a modified guidance map $\chi(T)$ is shown in Figure 4.

5 EXPERIMENTAL RESULTS

In order to showcase the versatility of our re-targeting approach, we evaluated our algorithm on a set of challenging test cases. Our primary application is the re-targeting of real-world city blocks. We extracted a set of

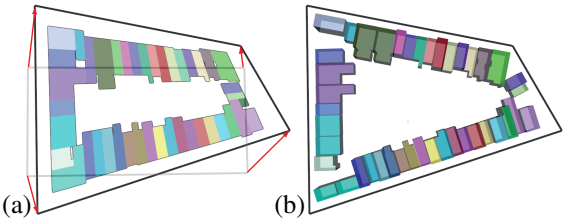


Figure 5: Re-targeting of a tightly packed city block. (a) Deformations (red arrows) applied to the initial domain (grey boundary) result in the edited domain (black boundary), distorted elements (various colors) serve as guidance map. (b) Re-synthesized block layout.

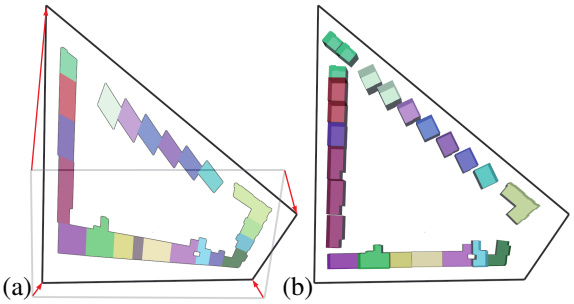


Figure 6: (a) Heavy enlargement of a city block. The applied deformation destroys the noticeable structure of diamond shaped buildings. (b) Re-synthesized block layout: the style of the diamond shaped buildings is preserved and additional elements are spawned in case of the heavily enlarged edge.

city blocks from the well-known community mapping service *OpenStreetMap* (OSM) [13]. When we speak of a city block we strictly speak of a simply connected and piecewise-linear closed boundary labeled as *street*. Inside, such a city block a set of polygons is located that are labeled as buildings in our examples.

Our first experiment, tackles the deformation of tightly packed city blocks, meaning that buildings and their 2-dimensional footprints are densely placed along the street segments.

In Figure 5(a) the deformations applied boundary intersections is highlighted by the red arrows. Applying this deformation to the initial layout results in distortions heavily shearing the elements and changing their size. The block re-layout computed by our algorithm (Figure 5(b)), resembles the style of the original block in the sense, that again a tightly packed layout is computed.

Figure 6 and 7 show different deformations applied to the same initial city block. Again the red arrows highlight the applied deformations. In Figure 6 the block was heavily enlarged, in Figure 7 the blocks was moderately shrunk.

In both cases, the noticeable structure and the orientation of the diamond shaped buildings is preserved.

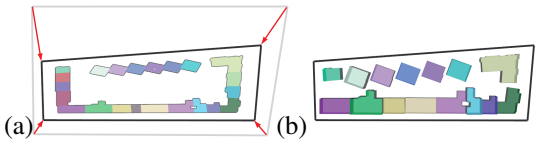


Figure 7: A city block moderately shrunk: notice, that elements get discarded form the layout in case the boundary regions reduced their size. However, the overall layout style is still recognizable.

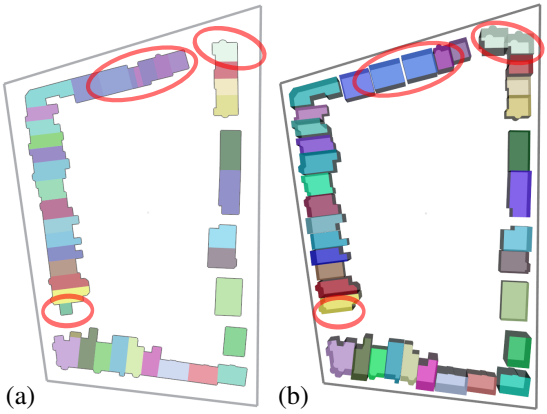


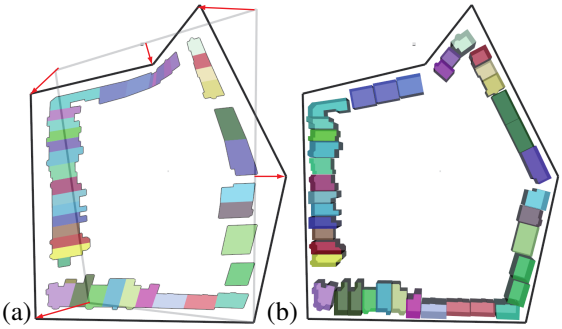
Figure 8: Re-synthesis of an undeformed domain. The original layout (a) is used to guide the algorithm in order to synthesize the novel layout (b) into the undeformed domain.

The result further illustrates that if boundary regions are heavily shrunk or enlarged the algorithm removes or adds discrete elements.

In Figure 8(b) shows a re-synthesis of the initial layout of a city block shown in Figure 8(a). Notice, the similarity between the initial layout shown in Figure 8(a) and the synthesized layout Figure 8(b). The algorithm is not able to reproduce the exact layout, due to the discrete nature of our approach. Figure 9 shows a result, where the outline of the original city block (Figure 8) was topologically modified by adding two additional intersections, leading to strongly visible shearing and bending of the elements within the guidance map. Using this guidance map as prior for the layout, our algorithm was able to produce a plausible new layout, although some footprints present in the initial layout were discarded.

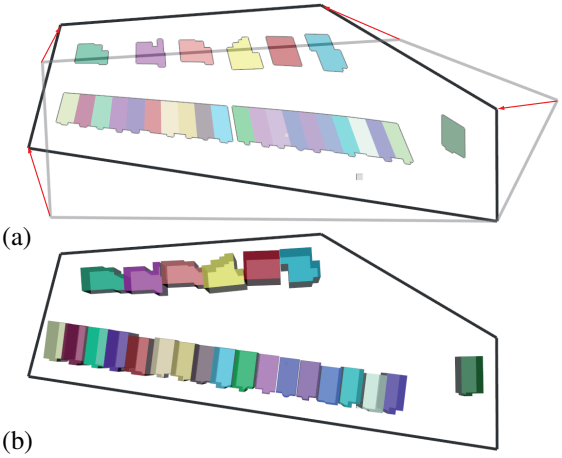
Other, examples demonstrating the strength our method are sparsely packed city blocks. Figure 10, illustrates that if empty space is present between buildings (five at the top street), the algorithm discards first empty space rather than discarding whole buildings since this would result in higher layout costs. For the result shown in Figure 11 we inserted two additional intersections along the edge with densely packed buildings. Notice that along the segment, where the split was introduced, the style (tightly packed footprints) is preserved, and even

 ISBN 978-80-86943-65-7



(a) (b)

Figure 9: Topological modification of the city block outline. Two additional points were inserted and the resulting outline was modified. (a) Applied transformations and warped interior of the layout shown in Figure 8 (a). (b) Resulting layout: Our algorithm splits up the sequences at the newly inserted points. Thus, the unappealing bending of the buildings is avoided.



(a)

(b)

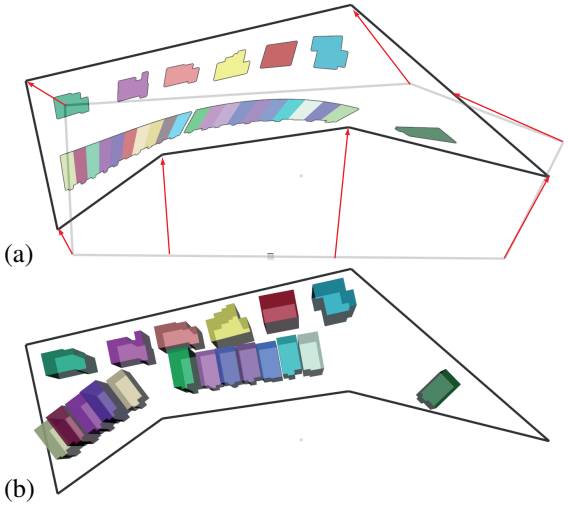
Figure 10: Sparsely packed city block, if edges get shrunk and empty spaces are available, the algorithm first discards empty space rather than discarding buildings.

the space between the five buildings (top segment) is preserved in the synthesized result.

Figure 13 presents an additional result, which combines changing the topology of the boundary and applying a heavy deformation to the initial domain. Even in this totally distorted domain, plausible building arrangements are synthesized.

In Figure 12 we present a result, where our approach was employed to a modeled city block populated with buildings taken from *Trimble Warehouse3d*. Please note, how additional buildings along the enlarged edges are introduced and even the single tree present in the original block is replicated.

Finally, Figure 14 illustrates the usage of the modified guidance map. Notice that when an unmodified guidance map is employed, multiple consecutive repetitions



(a)

(b)

Figure 11: Deformed block after two additional intersections where added to the boundary. It can be noticed, that the continuous tight building layout is split in two disjoint sequences, which are still tightly packed. Further note the space is preserved between the upper five buildings.

are present in the result Figure 14(a) and (c). Employing a simple anisotropic scaling to the individual elements present within the guidance map reduces, the consecutive repetitions (Figure 14(b) and (d)). Finally, we want to note that generating the results found inside the paper took no longer than 0.15*s*, using a desktop workstation with *Core i7 4930K (3.4 GHz)* and *32GB RAM*.

6 DISCUSSION AND LIMITATIONS

In the previous section we presented a large set of different re-targeting results. Although, all these results look plausible, we identified a few limitations that need to be discussed inhere. In our current implementation we rely on two conditions found in the input data (1) the atomic discrete elements blocks can be combined into larger groups, and (2) the elements are dominantly arranged along the boundary. Our algorithm is not restricted to element groups located near the boundary. In principle any group of elements that dominantly follow a curve can be synthesized with our algorithm. Only a few modifications need to realized to handle this case. (1) a method to fit a plausible curve, along the elements will be synthesized and (2) an additional data term, that handles, how 'well' the elements are oriented to the curve locally. Even in this case, we could again exploit the structure of the problem and still have an efficient algorithm to compute solutions. However, if no such groups are identifiable, our algorithm will produce failure cases. Further we may note, that we currently do not see a promising straightforward extension of our approach for 'real' 2D dimensional domains.

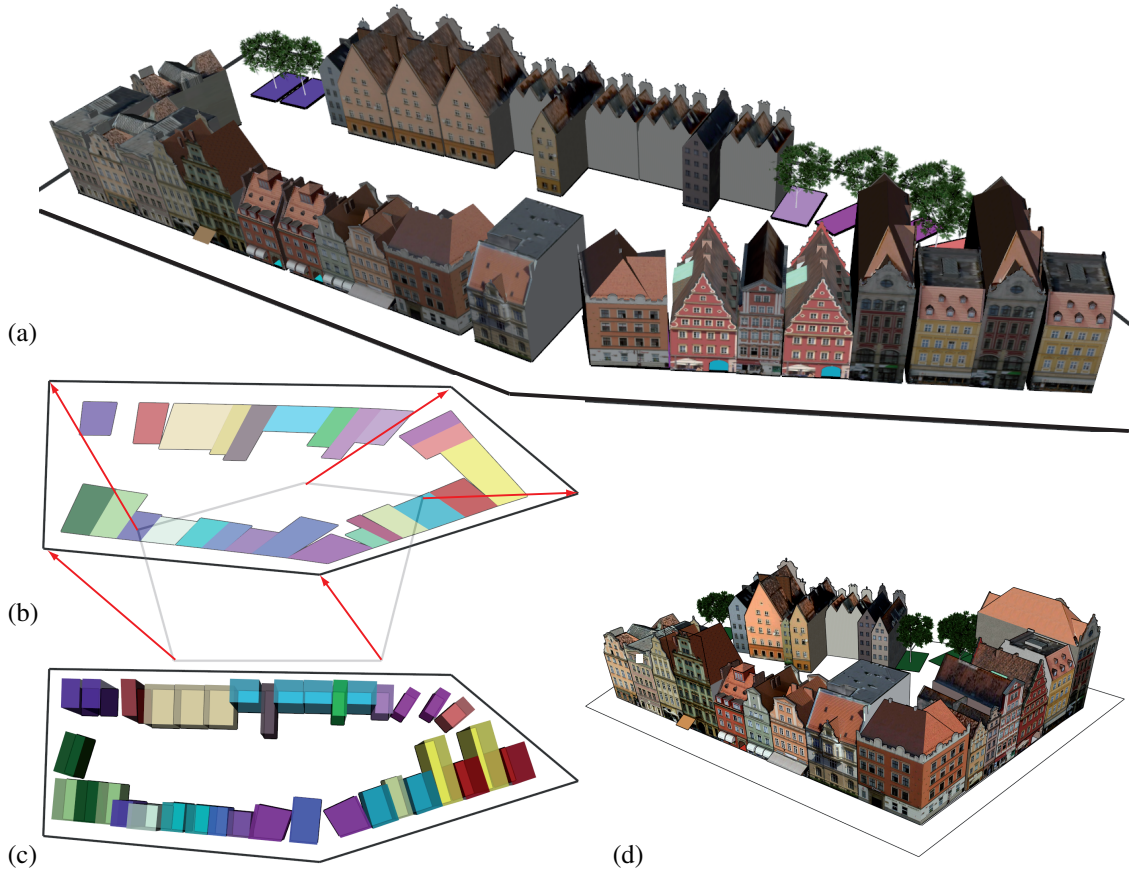


Figure 12: (a) A deformation of a city block populated with detailed modeled buildings. Notice, that in case of enlargement additional buildings and even trees are inserted. (b) We show the deformation and the guidance map. (c) Top-view of the resulting layout. (d) The original building block.

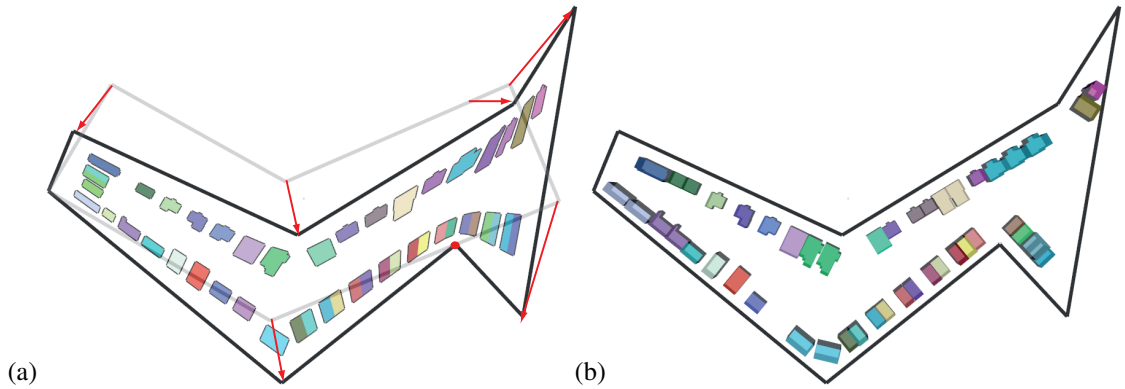


Figure 13: Heavily distorted block with additional intersections added to the boundary (a). Please note, that even applying such heavy deformations to the initial boundary, the element grouping (consecutive groups of two buildings) are present in the result (b).

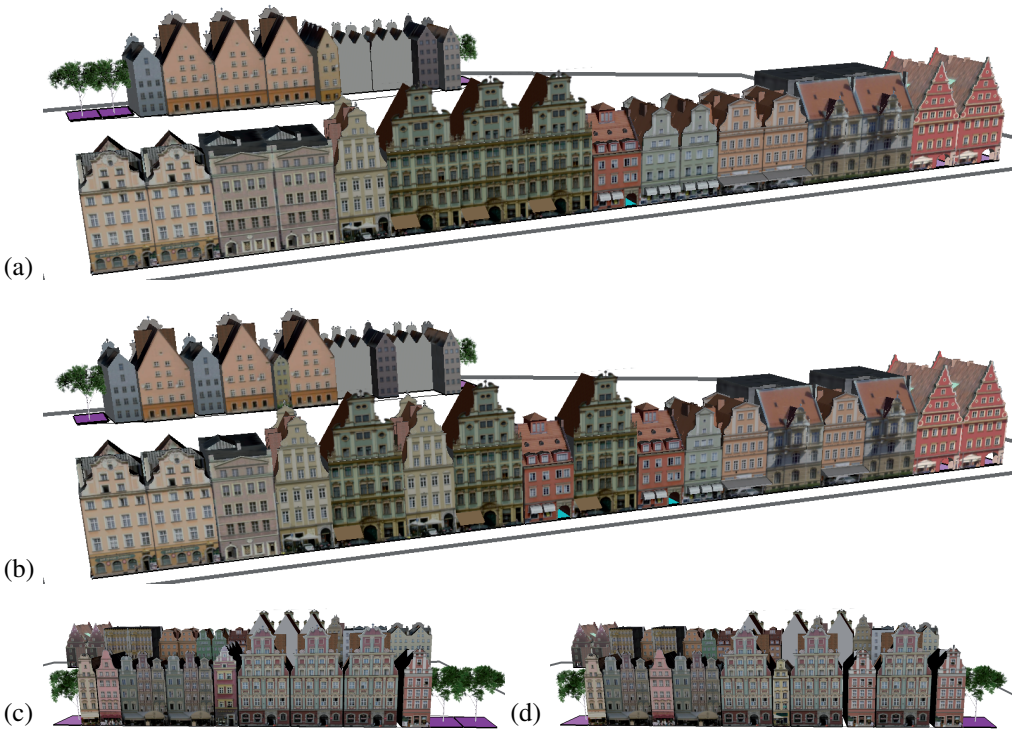


Figure 14: Illustrating the effect of using modified guidance maps. We show additional modifications to the block introduced in Figure 12. The results shown in (a) and (c) where obtained by simple warping of the guidance map. In this case multiple consecutive repetitions of the same building are visible. To create the results in (b) and (d), simple anisotropic scaling (referred to as blurring in Figure 4) was employed. This simple modification already reduced the number of consecutive repetitions significantly.

In addition, we identified another limitation introduced by our continuous deformation method acting globally across the polygon: It happens that even if the boundary edges do not change their length, the layout changes along these edges, because the underlying guidance map has changed. This effect, might be avoided by choosing a different deformation method, that can locally control the allowed deformation. We plan to investigate the approach presented by Möser *et al.* [12] for further evaluation. Furthermore, we see a promising set of different interesting research directions. First, we are interested in finding an efficient way to extend the algorithm to 2D or even 3D structures. In the 2D case similar deformations could be employed, however, presenting an at least near optimal solution at interactive rates, seems to be an important research direction. An orthogonal direction would be exploring the possibilities to combine the guidance map with existing forward procedural modelers, in order to guide the construction of the derivation tree. Finally, we would like to integrate our approach into an interactive data driven urban modeling framework.

7 CONCLUSION

We presented a simple but efficient way for re-synthesis of polygonal domains with groups of discrete building blocks. We employed our method to a large variety of different of city blocks and showed how the style of the initial layout can be preserved even under complex deformations. Our synthesis technique allows for exploring the layout space of different deformations at interactive rates, thus allowing further integration into an urban modeling and editing system.

8 ACKNOWLEDGMENTS

We thank the AiF Project GmbH for partial funding the work at hand. In addition the authors would link to thank the reviewers for their valuable comments and kindly thank Max Hermann for inspiring discussions.

9 REFERENCES

[1] Daniel G. Aliaga, Carlos A. Vanegas, and Bedřich Beneš. Interactive example-based urban layout synthesis. In *ACM SIGGRAPH Asia 2008 Papers*, pages 160:1–160:10, 2008.

[2] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. In *ACM SIGGRAPH 2010 papers*, SIGGRAPH '10, pages 104:1–104:10. ACM, 2010.

[3] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[4] Kai Hormann and Michael S. Floater. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*, 25(4):1424–1441, October 2006.

[5] Sylvain Lefebvre, Samuel Hornus, and Anass Lasram. By-example synthesis of architectural textures. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 84:1–84:8, New York, NY, USA, 2010. ACM.

[6] Jinjie Lin, Daniel Cohen-Or, Hao (Richard) Zhang, Cheng Liang, Andrei Sharf, Oliver Deussen, and Baoquan Chen. Structure-preserving retargeting of irregular 3d architecture. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2011)*, 30(6):Article 183, dec 2011.

[7] Yaron Lipman, Vladimir G. Kim, and Thomas A. Funkhouser. Simple formulas for quasiconformal plane deformations. *ACM Trans. Graph.*, 31(5):124:1–124:13, September 2012.

[8] Paul Merrell. Example-based model synthesis. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 105–112, 2007.

[9] Paul Merrell and Dinesh Manocha. Continuous model synthesis. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 158:1–158:7, 2008.

[10] Paul Merrell and Dinesh Manocha. Model synthesis: A general procedural modeling algorithm. *IEEE Trans. Vis. Comput. Graph.*, 17(6):715–728, 2011.

[11] Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph.*, 30(4):87:1–87:10, July 2011.

[12] Sebastian Möser, Patrick Degener, Roland Wahl, and Reinhard Klein. Context aware terrain visualization for wayfinding and navigation. *Computer Graphics Forum*, 27(7):1853–1860, October 2008.

[13] Foundation OpenStreetMap. Openstreetmap, 2015.

[14] Yoav I. H. Parish and Pascal Müller. Procedural modeling of cities. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA, 2001.

[15] Jerry Talton, Lingfeng Yang, Ranjitha Kumar, Maxine Lim, Noah D. Goodman, and Radomír Měch. In *Learning Design Patterns with Bayesian Grammar Induction*, 2012.

[16] Carlos A Vanegas, Tom Kelly, Basil Weber, Jan Halatsch, Daniel G Aliaga, and Pascal Müller. Procedural generation of parcels in urban modeling. In *Computer graphics forum*, volume 31, pages 681–690. Wiley Online Library, 2012.

[17] Xiaokun Wu, Chuan Li, Michael Wand, Klaus Hildebrandt, Silke Jansen, and Hans-Peter Seidel. 3d model retargeting using offset statistics. *International Conference on 3D Vision*, -(-):–, 2010.

[18] Yong-Liang Yang, Jun Wang, Etienne Vouga, and Peter Wonka. Urban pattern: Layout design by hierarchical domain splitting. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2013)*, 32:Article No. xx, 2013.

[19] Yi-Ting Yeh, Katherine Breeden, Lingfeng Yang, Matthew Fisher, and Pat Hanrahan. Synthesis of tiled patterns using factor graphs. *ACM Trans. Graph.*, 32(1):3:1–3:13, February 2013.

[20] Yi-Ting Yeh, Lingfeng Yang, Matthew Watson, Noah D. Goodman, and Pat Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM Trans. Graph.*, 31(4):56:1–56:11, July 2012.

[21] Lap-Fai Yu, Sai-Kit Yeung, Chi-Keung Tang, Demetri Terzopoulos, Tony F. Chan, and Stanley J. Osher. Make it home: automatic optimization of furniture arrangement. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11, pages 86:1–86:12. ACM, 2011.