Edit Propagation using Geometric Relationship Functions

PAUL GUERRERO
Vienna University of Technology, KAUST
and
STEFAN JESCHKE
IST Austria
and
MICHAEL WIMMER
Vienna University of Technology
and
PETER WONKA
KAUST

We propose a method for propagating edit operations in 2D vector graphics, based on geometric relationship functions. These functions quantify the geometric relationship of a point to a polygon, such as the distance to the boundary or the direction to the closest corner vertex. The level sets of the relationship functions describe points with the same relationship to a polygon. For a given query point, we first determine a set of relationships to local features, construct all level sets for these relationships and accumulate them. The maxima of the resulting distribution are points with similar geometric relationships. We show extensions to handle mirror symmetries, and discuss the use of relationship functions as local coordinate systems. Our method can be applied for example to interactive floor plan editing, and it is especially useful for large layouts, where individual edits would be cumbersome. We demonstrate populating 2D layouts with tens to hundreds of objects by propagating relatively few edit operations.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Modeling Packages*

General Terms: Edit propagation, Geometric Relationships

Additional Key Words and Phrases:

ACM Reference Format:

This research was partially funded by KAUST and the Austrian Science Fund (FWF) projects no. P 24352-N23 and P 24600-N23. Authors' addresses: P. Guerrero; email: paul@cg.tuwien.ac.at S. Jeschke; email: sjeschke@ist.ac.at M. Wimmer; email: wimmer@cg.tuwien.ac.at P. Wonka; email: pwonka@gmail.com

DOI 10.1145/XXXXXXX.YYYYYYY
http://doi.acm.org/10.1145/XXXXXXX.YYYYYYY

1. INTRODUCTION

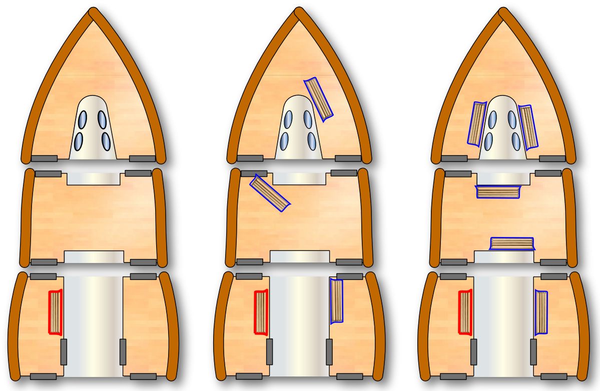
Fig. 1. To find good positions for a bench (red) on the deck of a ship, many geometric relationships to the deck and the objects on the deck have to be taken into account. A naive approach using a local coordinate system based on boundary distance and boundary arc length (center – for more information, see Section 8) does not result in acceptable positions. Our approach (right) is flexible in its use of relationships, giving better positions.

In recent years, flexible and user-assisted editing systems have become an important research direction in computer graphics. In particular, the question of example-based editing has gained interest: when a user applies a change to a given scene, how can other, similar changes be applied to the scene automatically? This touches on the question of *shape similarity* and *coordinate systems*: to propagate a change to other parts of a scene, we need to find those parts that are *similar* to the part that has been edited; furthermore, we need to find the exact *location* where the change should be applied in a particular similar part. For the latter problem, different "local" coordinate systems have been proposed in animation and geometric modeling. These coordinate systems can be used to transfer a position in one polygon to another polygon. Examples include mean-value coordinates [Hormann and Floater 2006] and harmonic coordinates [Joshi et al. 2007]. However, these systems work best in shape interpolation, where the basic topology of the shapes that

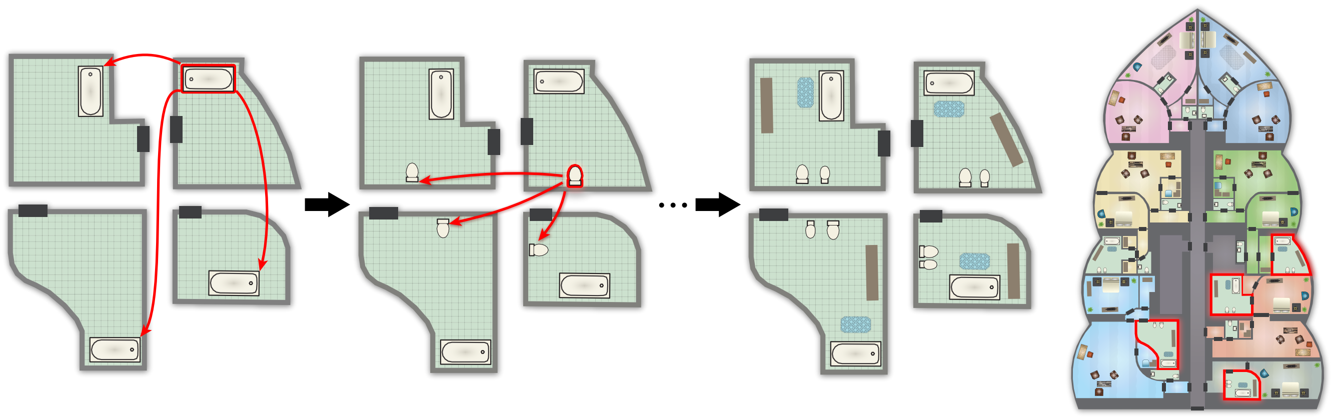

Fig. 2. We propose a method to propagate edit operations in 2D scenes. Objects can be propagated between general polygons of arbitrary shape while still giving plausible results. Our method can be used to place a large number of objects in 2D layouts using relatively few edit operations.

provide the reference frame varies only by a little. Also, coordinate systems typically encode the notion of *distance*, but are not flexible enough to represent other geometric relationships.

In this paper, we introduce a new method for edit propagation in 2D scenes that is more flexible and general than previous methods. It is based on the notion of *geometric relationships* between shapes and so-called *poses*, which we define as a location with an associated direction vector. While, for example, 2D Cartesian coordinates uniquely associate two values (coordinates) with every point in the plane, which represent the distance from one of the coordinate axes, our method can handle an arbitrary number of geometric relationships, and the mapping to points in the plane need not be unique. This allows geometric relationships that do not depend on distance alone (like the relative direction with respect to a feature in the scene) to be introduced, and abandons those concepts from coordinate systems that are not required in our setting of edit propagation (see Figure 1).

Our current setup focuses on object placement, which is why we choose poses as the main interaction element. A pose can represent an object (e.g., a furniture item) and its orientation in a room. Furthermore, we concentrate on geometric relationships as structuring elements rather than on similarity between the underlying polygons. This often gives plausible matches even if the degree of similarity is low, as different types of geometric relationships can be taken into account. Figure 2 presents an example of the proposed work flow.

2. RELATED WORK

This work is related to scene modeling by example, analyze-and-edit approaches as well as coordinates.

Scene modeling by example. The goal of this work is to reduce the manual design workload for populating complex environments with objects. It shares this goal with previous modeling-by-example approaches that synthesize mostly indoor scenes. In some related work [Fisher et al. 2012; Fisher et al. 2011; Yu et al. 2011], structural relations (such as spatial, hierarchical, or pairwise relationships; or ergonomic factors such as visibility and accessibility) and probabilistic models are learned from a database of example scenes. From these, new scenes are automatically synthesized. For specific applications such as interior design, synthesis can also build on special design guidelines [Merrell et al. 2011] including functional and visual criteria, some of them geometric in nature as in this work. Another way to populate virtual environments is to formulate a valid space of layouts via specialized probability distributions together with constraints [Yeh et al. 2012] and to sample this space appropriately to derive plausible scenes.

In contrast to the above approaches, which mostly synthesize complete scenes, in this paper, individual user edits are propagated to reduce the manual workload. In doing so, we rely on general geometric principles, object labelings and, optionally, a hierarchical scene decomposition. Our idea is still applicable if no specific design principles can be formulated or if no example scene databases are available. Instead, scene information is incrementally created and updated directly during modeling, i.e., when objects are successively placed.

Analyze-and-edit approaches. Analyze-and-edit approaches [Gal et al. 2009; Zheng et al. 2011; Zheng et al. 2012; Bokeloh et al. 2012; Bokeloh et al. 2011] derive some general geometric properties from a given object. These are reinforced after local edits by propagating the edit to similar parts of the entire object. The analysis phase can include the extraction of one-dimensional features [Gal et al. 2009] and learning their individual characteristics and mutual relations. It can also be a model decomposition into meaningful components together with certain control degrees of freedom for each component [Zheng et al. 2011]. During editing, structural relations are maintained after making local changes by propagation to other features or components.

This work also extracts local information around newly placed objects, and uses this information together with a set of geometric principles to propagate the edits. However, this work focuses on the placement of objects in complex scenes, rather than geometric deformation propagation within single objects.

Coordinates. This work also makes use of local coordinate systems to relate positions in space to the surrounding objects. Different local coordinates, such as mean value coordinates [Hormann and Floater 2006], harmonic coordinates [Joshi et al. 2007] and several others [Lipman et al. 2008; Weber et al. 2012], have been defined in animation and geometric modeling for similar purposes. However, the source and target locations we are considering here may have a varying local object count, type, and shape, thus rendering the above purely distance-based coordinates not flexible enough. To overcome this problem, we extend and enrich coordinates with semantic information as detailed further below.

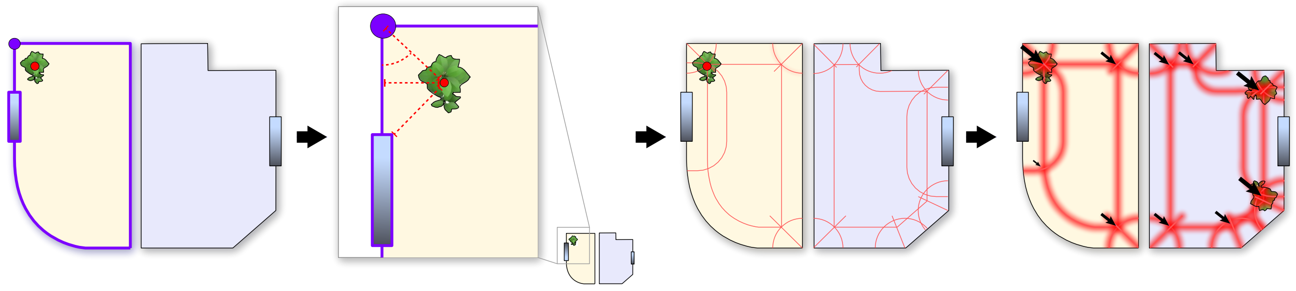

Fig. 3. The main steps of our method. First, a query point is placed (plant) and important features are identified (purple). Then the relationship functions to the important features are evaluated (red dotted lines). Level sets for each feature and relationship function are constructed (red lines) and accumulated in pose space. The resulting maxima correspond to poses that have similar relationship function values as the query point.

3. OVERVIEW

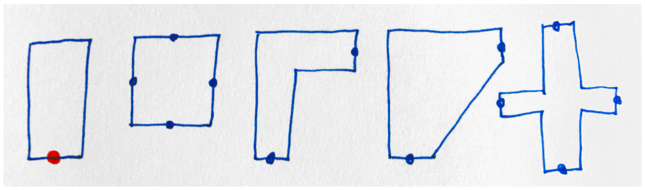

Fig. 4. A human intuitively finds the blue point correspondences in the output polygons for the red point specified in the input polygon.

The starting point for this work was our observation that most humans have an intuitive idea of how to transfer points between polygons. In Figure 4, for example, a human would be likely to pick the blue points as correspondences to the specified red point. Currently there is no computational tool available that could do the same. Generalizing this research problem to include directions and multiple polygons, we obtain the following research question:

Given a scene consisting of a set of polygons $\mathbf{A} = A_1, \ldots, A_n$, and a query pose $Q = (Q_p, Q_d)$, defined by a point and a direction, determine a number of poses $P_1, \ldots, P_n$ that have a local neighborhood that is "similar" to Q.

In Section 4, we introduce our notion of similarity. It is based on so-called *relationship functions*, which assign a numeric value to a certain *geometric relationship* between a pose and a feature in the scene. Features can be the polygons themselves, their individual corners, a certain segment, etc. Examples of relationship functions include the distance of query point Q_p to a polygon, or the angle by which Q_p is offset from the bisector of a polygon corner.

Given a query pose and a set of polygons, our algorithm has the following steps (see Figure 3):

(1) Identify features in the scene (Section 4.1).
(2) Evaluate the relationship functions (Section 4.2) for selected features, called *input features*.
(3) For each relationship function and each feature, find the locations/poses in the scene that would give a similar function value as the query point. Basically, this corresponds to constructing a level set in "pose space" (Section 6.1) for each relationship function around a feature that is "compatible" to the input feature – we call these the *output features*.
(4) Combine the level sets in pose space based on two weighting factors (Section 5):
    —the *local importance*, i.e., how relevant is the input feature to the query point for a given relationship function, and
    —the *matching accuracy*, i.e., how closely does the output feature match the input feature.
(5) Find local maxima in pose space (Section 6.2) – these correspond to the desired poses that share as many of the defined relationships with the query pose as possible.

We also describe an extension to handle mirror symmetries present in the scene geometry (Section 7), and we describe how specific geometric relationships can be selected to form local coordinate systems with respect to features in the scene (Section 8). The application case of floor plan editing is described in Section 9, which also includes useful extensions like probabilistic placement and feature selection, labels and hierarchies to restrict matching, and pose validation to avoid intersections in the result.

4. GEOMETRIC RELATIONSHIPS

In this section, we formally introduce the concept of geometric relationships through the definition of geometric features and relationship functions. As discussed in Section 3, we start with a scene $\mathbf{A}$ and a query pose Q in a pose space $\mathbf{P}$. The pose space is the set of poses of interest, usually encompassing the scene and all angles from 0 to 360 degrees.

4.1 Features

Geometric relationships operate on features; the first step is therefore to extract features from the scene. The main features are the polygons in the scene themselves and the elements of the boundary of the polygon. We assume that the polygon boundary consists of "smooth" segments, i.e., sequences of edges with non-sharp angles, connected by "sharp" corners. We therefore use the following features:

—a **polygon** A, defined by a sequence of vertices $v_k, k \in \{1, \ldots, n_A\}$,

—a polygon **segment** $s_j \in A$, defined as a sub-sequence of consecutive smooth vertices $s_j = v_a, \ldots, v_b, a, b \in \{1, \ldots, n_A\}$. A vertex is called smooth if the dot product of its adjacent edges is above a threshold ($\frac{v_{k+1}-v_k}{||v_{k+1}-v_k||} \cdot \frac{v_k-v_{k-1}}{||v_k-v_{k-1}||} > \epsilon$), otherwise it is called sharp.

—a **corner** c_j of a polygon, defined by a vertex v_{k_j} that is shared by two adjacent segments s_{j-1} and s_j. Corners correspond to sharp vertices. In practice, the *bisector* c^b of the corner and the

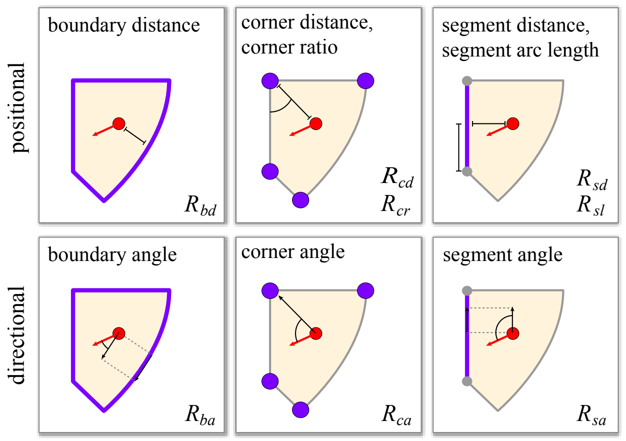

Fig. 5. Relationship functions between a polygon and a query pose (red). Features (purple) in column from left to right: polygon, corner and segment. The top row shows all positional relationships; the bottom row all directional relationships. One positional and one directional relationship of the same column are used to form pairs of relationships.

opening angle c^α will be relevant, as they are related to the direction vector of the query pose.

4.2 Relationship Functions

Given a set of features $\mathbf{F}$ and pose space $\mathbf{P}$, we define a relationship function as a functional on features and poses:

$$R : \mathbf{P} \times \mathbf{F} \rightarrow \mathbb{R}.$$

Each relationship function expresses a geometric relationship between a feature and a pose in a numeric way. Relationships include the distance of a pose from a feature, but also the angle between a pose and the feature. In the following, we list the geometric relationship functions that we use (see Figure 5). For convenience, we define the angle between vectors as $\angle(a, b) = \arccos(a/||a|| \cdot b/||b||)$.

—The **boundary distance** is defined as the minimum distance between the query point and a polygon boundary: $R_{bd}(Q, A) = \min_{x \in b(A)} d(Q_p, x)$, where $b(A)$ is the boundary of A.

—The **segment arc length** is the normalized arc length between the location of minimum distance to the query point and the start of the segment. If $x = \arg\min_{x \in b(s_j)} d(Q_p, x)$, then $R_{sl} = \frac{t_x - t_a}{t_b - t_a}$, where t_x is the arc length at x and t_a, t_b the arc length at the start and the end of the segment, respectively.

—The **segment distance** $R_{sd}(Q, e)$ is the same as the boundary distance, but considers only a segment $s \in A$.

—The **corner distance** is the distance between the query point and a polygon corner: $R_{cd}(Q, c_j) = d(Q_p, v_{k_j})$.

—The **corner ratio** is the angle that the direction from corner to query point makes with the start of the first polygon segment adjacent to the corner, normalized with the opening angle of the corner: $R_{cr}(Q, c_j) = \frac{\angle(Q_p - v_{k_j}, v_{k_j - 1} - v_{k_j})}{\angle(v_{k_j + 1} - v_{k_j}, v_{k_j - 1} - v_{k_j})}$.

—The **boundary angle** is the angle between the pose direction and the direction of the polygon boundary at the location of minimum distance to the query point. If $x = \arg\min_{x \in b(A)} d(Q_p, x)$, then $R_{ba}(Q, A) = \angle(\frac{d\ b(A)}{dt}, Q_d)$, where t is a parametrization of the polygon boundary by arc length.

—The **segment angle** $R_{sa}(Q, e)$ is the same as the boundary angle, but considers only a segment $s \in A$.

—The **corner angle** is the angle between the pose direction and the direction from corner to query point: $R_{ca}(Q, c_j) = \angle(Q_p - v_{k_j}, Q_d)$.

The first five relationship functions are called *positional*, because they depend only on the position of the query pose. The other three are called *directional*, as they also depend on the direction of the query pose. Which of those relationships are actually applied depends on the application scenario, and more general relationships can be used. Later, we will always combine a positional with a directional function to clearly define similar poses. For this purpose we assign one directional relation R_d to each positional relation R_p. For the relationships defined above, we assign each R_{fd}, R_{fr} or R_{fl} the corresponding R_{fa} where $f \in \{b, s, c\}$ according to the type of feature. This set of relationship functions works well in practice, but we can easily remove or add additional functions to the set. For a discussion of the influence of individual relationships on the final result and an example of an additional relationship function, see Section 10.1.

5. POSE MATCHING IMPORTANCE

In order to find poses in the scene whose neighborhood matches the query pose, we evaluate both a *local importance* and a *matching accuracy*.

5.1 Local Importance

The local importance determines how important features near the query pose (i.e., input features) are for determining matching poses. It is therefore again a functional on features and poses:

$$I_l : \mathbf{P} \times \mathbf{F} \rightarrow \mathbb{R}.$$

However, while a relationship function is used to determine the *location* of poses with similar geometric relations to the query pose, the importance is used to determine the *influence* each surrounding feature has with respect to a geometric relationship in the final selection of matching poses.

There are many possible importance functions, including semantic weightings assigned by the user, but in our current implementation, we simply use a weight based on the normalized distance to the feature, as defined already by the relationships functions involving distance:

$$I_l(Q, F) = 1 - \min(1, R_{fd}(Q, F)/n(A_F)),$$

where the normalization factor $n(A_F)$ is the radius of the largest circle inscribed in the polygon corresponding to feature F for Q inside the polygon, or twice the radius of the smallest circle enclosing the polygon for Q outside the polygon, and $f \in \{b, s, c\}$ according to the type of feature. This reflects the fact that features close to the query pose should have stronger influence on the matching than features farther away.

5.2 Matching Accuracy

The matching accuracy determines whether a “distant” candidate feature for matching (i.e., an output feature) resembles a local feature that has been determined to be important to the query pose. For example, if the query point is positioned near a window-shaped polygon, other window-shaped polygons in the scene will receive higher weighting in the subsequent matching algorithm. The matching accuracy is a functional on features:

$$I_m : \mathbf{F} \times \mathbf{F} \to \mathbb{R}.$$

Obviously, if the two features are not of the same type, a value of 0 should be assigned. In our implementation, we use the following matching accuracies:

—for **polygons**, the similarity of polygon areas: $I_m(A_1, A_2) = \min(a(A_1), a(A_2))/\max(a(A_1), a(A_2))$, where $a(A)$ denotes the area of polygon A.

—for **segments**, the similarity of segment lengths and curvatures: $I_m(s_1, s_2) = \frac{\min(t(s_1),t(s_2))}{\max(t(s_1),t(s_2))} * (\max(\tilde{\eta}_1, \tilde{\eta}_2) - \frac{|\tilde{\eta}_1 - \tilde{\eta}_2|}{\max(\tilde{\eta}_1, \tilde{\eta}_2)})$, where $t(s_i)$ is the arc length of segment s_i and $\tilde{\eta}_i$ is the average curvature of the segment.

—for **corners**, the similarity of opening angles: $I_m(c_1, c_2) = |c_1^\alpha - c_2^\alpha|/\pi$.

Note that there are many possible matching functions, including symmetry-based and semantics-based ones. For example, the matching could be further restricted according to labels assigned to polygons, e.g., matching only features that fulfill some conditions on the labels, as will be shown in Section 9.

6. MATCHING ALGORITHM

6.1 Theory

In this section, we discuss the algorithm that computes potential matching poses from the query pose. We assume a set $\mathbf{R}$ of relationship functions to be given, as well as a set $\mathbf{F}$ of features. The main idea is to determine for each triple of relationship function R, "local" feature F_l and candidate matching feature F_m all poses in the scene that have a similar function value as the query point. In other words, we want to find the **level set**

$$L(R, F_l, F_m) = \{P \in \mathbf{P} | R(P, F_m) = R(Q, F_l)\}.$$

In order to combine the potential candidate poses generated by different relationship functions, local and matching features, we accumulate their level sets in pose space according to local importance and matching accuracy. There are two ways to combine candidate poses: We can either *intersect* different level sets, or we can *accumulate* their importance.

Intersect:. We note that level sets of positional relationships $R_p \in \mathbf{R}_p$ are independent of the angle and therefore span the entire angular dimension at each position, while level sets of directional relationships $R_d \in \mathbf{R}_d$ span the entire positional domain. To obtain level sets of finite extent, we intersect the level set of each positional relationship R_p of a pair (F_l, F_m) with the level set of its associated directional relationship $\hat{R}_d$:

$$\hat{L}(R_p, F_l, F_m) = L(R_p, F_l, F_m) \cap L(\hat{R}_d, F_l, F_m).$$

Accumulate:. For each pose, we accumulate the importances of the intersected level sets to obtain a final importance I, where, for each relationship, we only take the best match:

$$\begin{aligned} I(P, Q) &= \sum_{F_l \in \mathbf{F}} I_l(Q, F_l) \\ &\quad \sum_{R \in \mathbf{R}_p} \max_{F_m \in \mathbf{F}} \left(I_m(F_l, F_m) \mathbf{1}_{\hat{L}(R, F_l, F_m)}(P) \right), \end{aligned}$$

where $\mathbf{1}_X$ is the characteristic function of set X. The candidate poses we are looking for are then the local maxima of I. Figure 6 illustrates this process.

6.2 Finding Maxima of I

In practice, the definition of I makes a computation of local maxima computationally unwieldy, since pose space is both continuous and three-dimensional (2D location and 1D direction). In order to solve this problem, we make two simplifications:

—We accumulate importance in 2D, regardless of the directional information of poses, and maintain a direction of maximum importance during the accumulation.

—We discretize the 2D location space.

For this, we first calculate the projection L_p of the level set onto 2D location space:

$$L_p(R, F_l, F_m) = \{P_p | P \in L(R, F_l, F_m)\}$$

and compute the positional importance I_p only in 2D space:

$$\begin{aligned} I_p(P_p, Q) &= \sum_{F_l \in \mathbf{F}} I_l(Q, F_l) \\ &\quad \sum_{R \in \mathbf{R}} \max_{F_m \in \mathbf{F}} \left(I_m(F_l, F_m) \mathbf{1}_{\hat{L}_p(R, F_l, F_m)}(P_p) \right). \end{aligned}$$

For the directional information, we have to employ an approximation. When accumulating in 2D location space, the directional information of the accumulation is lost and we cannot obtain the direction of maximum *accumulated* importance at each position. Instead of accumulating, we use the direction of the level set with maximum importance at any given position. To express this mathematically, we introduce a directional importance I_d defined in full 3D pose space, which, however, is only needed to formally express the result:

$$\begin{aligned} I_d(P, Q) &= \max_{F_l \in \mathbf{F}} I_l(Q, F_l) \\ &\quad \max_{R \in \mathbf{R}_p} \max_{F_m \in \mathbf{F}} \left(I_m(F_l, F_m) \mathbf{1}_{\hat{L}(R, F_l, F_m)}(P) \right), \\ \phi_{max}(P_p) &= \arg\max_{\phi \in [0, \ldots, 2\pi]} I_d((P_p, \phi), Q). \end{aligned}$$

Note that ϕ_{max} may not be the same direction found when accumulating in the full 3D pose space. However, there are only differences in special cases and the 2D approximation does not lead to unintuitive results. For a discussion, see Section 10.1.

6.3 Implementation

The evaluation of I_p and ϕ_{max} is carried out on a 2D regular grid. In practice, this corresponds to a *rasterization* of the level sets onto a grid. In all our examples, the grid has approximately $100k$ grid points in total, with the aspect ratio adapted to the extent of the scene. We generate the level sets by rasterizing simple primitives like circles for the corner distance or lines for the segment arc length. For the boundary distance relationship function, we compute a polygon offset by taking the Minkowski sum [Behar and Lien 2011] of the polygon boundary with a circle and then rasterize this polygon offset. In order to avoid aliasing and to increase robustness (e.g., to account for inaccuracies in the input geometry), we replace the binary characteristic function $\mathbf{1}_X$ of the level set by a filter kernel $\mathbf{k}_X$, which slightly dilates the level set. In our examples, we use a linear falloff with a small fixed radius.

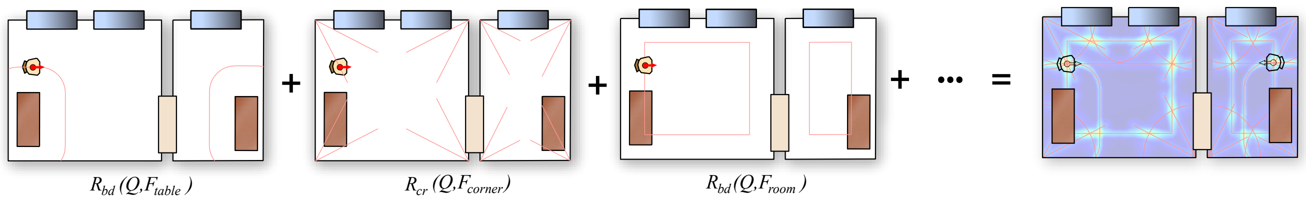

Fig. 6. Accumulation of level sets (red) of the relationship functions. The level sets of the relationships of the red query pose to local features are accumulated. The maxima of the resulting distribution are poses that have similar function values as the query point. The text below the figure indicates which relationship was used to create the level sets. The image on the right shows the result of the accumulation as heat map.

The result of this step is a 2D grid with an importance value I_p and a direction ϕ attached to each grid cell. The best candidate locations are then found by applying non-maximum suppression [Gil and Werman 1993] on the importance grid. The resulting maxima are then sorted according to their importance, giving an ordered list of those candidates that best match the query pose.

The level sets are rasterized using line or polygon rasterization in graphics hardware. Each type of relationship function requires a specific method for calculating the level set. For R_{bd}, the boundary distance, for example, the polygon boundary is eroded by that amount that makes the eroded polygon intersect Q_p.

7. MIRROR SYMMETRIES

Given a query pose Q, it is often desirable to find all objects that have similar relationship function values when *mirrored* along a symmetry axis of a feature. In other words, we want $R(s_{F_m}(P), F_m)$ to give the same result as $R(P, F_m)$, where s_{F_m} is a function that mirrors pose space along one (or more) *symmetry axes* of a feature.

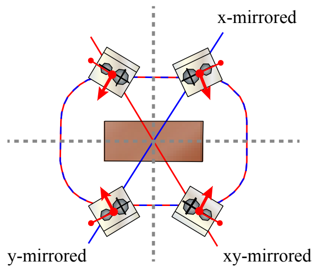

Fig. 7. Mirrored level sets of an object with two mirror symmetries. When mirroring an object on one of the axes, the resulting object is mirrored. Level sets that correspond to these mirrored poses are shown as blue lines. Non-mirrored level sets are shown in red. The blue/red dashed lines are two level sets with the same position (but different angle), one mirrored and one non-mirrored.

However, if a pose has been mirrored, it does not represent the same object anymore. Instead, it should represent an object that has been mirrored about P_d. In order to be able to represent such mirrored poses, we extend pose space $\mathbf{P}$ to $\mathbf{P}'$ by a binary flag P_m:

$$P' = (P_p, P_d, P_m) \in \mathbf{P}'.$$

To account for mirrored poses, the symmetry operator is extended to handle the mirror flag:

$$s_{F_m}((P_p, P_d, P_m)) = (P_p^m, P_d^m, 1 - P_m),$$

where (P_p^m, P_d^m) is the pose mirrored about a symmetry axis s of F_m. This properly accounts for the fact that after application of two symmetry axes, the pose is in unmirrored space again.

We then construct additional level sets, one for each symmetry axis or combination of symmetry axes s of the matched features (see Figure 7):

$$L^s(R, F_l, F_m) = \{P' \in \mathbf{P}' | R(s_{F_m}(P'), F_m) = R(Q, F_l)\}.$$

The remainder of the method is identical: the mirrored and non-mirrored level sets are accumulated in extended pose space $\mathbf{P}'$, and the maxima identify points that have relationship function values similar to the query point, up to symmetries. This is implemented by accumulating importance in two different grids, one for mirrored and one for regular space.

8. LOCAL COORDINATE SYSTEMS

So far, the algorithm works well for placing rigid objects, where considering the pose of the object is sufficient. However, it is difficult to generate a continuous mapping from source to target polygon directly from our method. For example, the map resulting from always taking the highest target maximum for every source pose is generally neither injective nor surjective. In floor plans, it can also be useful to transfer an object *non-rigidly*, i.e., transferring all points from a sample placement to another location in the scene, while preserving both the coherence of the points among themselves and the match of each point to its neighborhood. This requires a more strict definition of neighborhoods in the form of *local coordinate systems*. In this section, we clarify the connection between our relationship functions and local coordinate systems by providing a simple approach for constructing local coordinate systems using our relationship functions. For a more sophisticated approach to establishing a mapping between shapes using local coordinate systems, we refer to Solomon et al. [Solomon et al. 2012].

In a coordinate system, each coordinate can be interpreted as a relationship between the point being described and some geometric feature. Together, the coordinates uniquely specify the point in space. In Euclidean coordinates, for example, each coordinate measures the distance of the point to a plane orthogonal to the coordinate axis. In mean value coordinates, each coordinate is related to a generalized distance of the point to a polygon vertex.

The directional relationships introduced in Section 4 can also be used to identify points in space with respect to a feature (i.e., locally). In 2D, exactly two such relationships are required. In order to form a local coordinate system, the selected relationships should provide a bijective mapping from the domain to coordinates. While it is evident that each of the directional relationship functions cover the domain, the uniqueness of coordinates can be described in terms

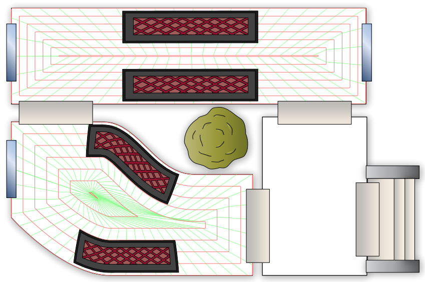

Fig. 8. Two objects propagated from the inside of the top room to the inside of the bottom room using a local coordinate system based on the boundary distance R_{bd} and a custom relationship R_{bf} based on the arc length from a fixed start point on the boundary to the position on the boundary closest to the query pose. Red lines are the iso-curves for different values of R_{bd} and green lines iso-curves for different values of R_{bf}.

of the level sets of the relationship functions, which, in the positional domain, correspond to iso-curves:

The mapping of points to coordinates is invertible if any two level sets (L_1, L_2) chosen from the relationship functions (R_1, R_2) intersect at exactly one point, and if R_1, R_2 themselves are injective (i.e., the corresponding iso-curves do not self-intersect).

The local coordinates of a point P_p with respect to a feature F are then simply $(R_1(P_p, F), R_2(P_p, F))$ (see Figure 8 for an example). From the relationships introduced in this paper, only the pair (R_{cd}, R_{cr}) fulfills this property. The drawbacks of using local coordinate systems are that the two relations R_1 and R_2 and the feature F have to be determined manually before the propagation and that only coordinates that fulfill the properties described above can be employed. Also, the resulting coordinate system does not necessarily provide a smooth mapping between polygons - there may be some C1 discontinuities. For example, in Figure 11, a pattern of flowers has been propagated from one lawn area to other lawn areas using local coordinates (for details refer to Section 10).

9. APPLICATION: FLOOR PLAN EDITING

For specific applications, we can extend our method in various ways. An interesting area of application is floor plan editing. Manually placing objects in these floor plans can be time-consuming but cannot be trivially automated, since the placed objects have to respect geometric relationships to existing objects. To apply our method to this problem, objects of the floor plan, such as rooms or tables, are modeled as polygons. To place an object, the user defines a pose. We can propagate this pose using geometric relationship functions as described in the previous sections. To specialize our method to this application, we introduce four simple extensions: polygon labels, a hierarchy over polygons that describes inclusion relations, a pose validation step that removes propagated poses that would result in unwanted object intersections, and placing arrays of objects. Additionally, we define probabilistic extensions for object placement.

Labels. Since the focus of our work does not lie in object recognition, we assume that all polygons in the floor plan are labeled according to their type, e.g., table, bathroom, and so on. All features F of a polygon have the same label as the polygon. The matching accuracy $I_m(F_1, F_2)$ of two features with different labels is set to zero, reflecting the fact that objects of different types should not be matched.

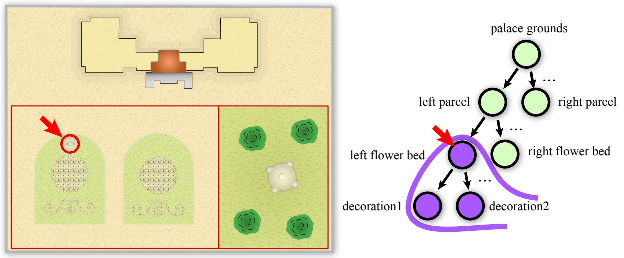

Fig. 9. The floor plan hierarchy for a small garden scene. When placing an object in the left flower bed (red arrow), only the siblings and the parent (purple) of the object are considered for I_l.

Hierarchy. A floor plan is usually divided into different areas like rooms or buildings. These areas are objects of the floor plan and correspond to polygons in our framework. When placing an object into one of these areas, only relationships to objects in the same area are important. To incorporate this fact into our framework in a general way, we define a hierarchy on all polygons. A polygon A_c is the child of a polygon A_p if A_c is contained in A_p, or if the polygons intersect and A_c has a smaller area. The local importance $I_l(Q, F)$ of a feature is set to zero if Q is not contained in the parent polygon of A_F. This allows, for example, first placing parcels in a larger area, then flower beds in each parcel, and then flowers in a flower bed. In each step, only the parent and siblings of an object will be considered for I_l (see Figure 9).

Pose validation. The placement algorithm does not take geometric properties of the object (apart from its pose) into account. Therefore, a placed object could have unwanted intersections with the existing geometry. To avoid this, we first collect all labels of polygons that are intersected by the object when placed at the query pose Q and at the candidate pose P. If the labels at P are not a subset of the labels at Q, then the pose is removed. Typically, this allows intersections with the parent polygons at the placed poses. For the intersection step, alternative geometry can be used. For doors, for example, the polygon can be extended to avoid objects being unintentionally placed too close, so that the door cannot be opened.

Arrays of objects. We can also facilitate the placement of arrays of objects. An array is a set of copies of the same object that are aligned along a path and have fixed spacing. The path is defined as the offset boundary of a nearby polygon. Arrays can be defined by a start point, an end point, a polygon, and a spacing. The spacing is user-defined and set before the array is created. First, the start point is propagated as usual and all resulting propagated points are marked as starting points. Then, a second point is propagated to mark the end points of the array. The polygon used to align the array is defined as the polygon closest to both start and end points. If the array start and end do not agree on the closest polygon, no array is created. If there are more than two end points on the same boundary, only the end point closest to the start point is used to form the array. For all valid pairs of start and end points, the end point is projected to the offset boundary of the polygon to mark the end of the array along the offset boundary, and objects are placed using the predefined spacing.

Probabilistic pose selection. In order to allow more variation, we introduce probabilistic placement of poses. First, we define a scope for probabilistic selection, typically by the parent polygon label (e.g., "room"). Several query poses are defined in the source scope. Then, in each target scope, one of the query poses is randomly selected and only matches corresponding to that query pose are accepted as candidates.

Probabilistic pose placement. In Section 6.3, we already introduced a kernel $\mathbf{k}_X$ to avoid aliasing and geometry errors when rasterizing the level sets. This idea can be extended by choosing a wider kernel and replacing the deterministic search for local maxima by a probabilistic one, to allow for more natural (i.e., irregular) placements. This can be done by interpreting the resulting regular grid as a probability distribution and sampling poses according to this distribution.

Parameters. In floor plan editing, we allow for two types of parameters: first, the maximum number of matches allowed for a polygon of a certain label (typically "room") per query pose, and second, a maximum placement density, which can be influenced by changing the radius of the non-maximum suppression.

We use the first three extensions (labels, hierarchy and pose validation) in every operation of our examples. See Section 10.1 for a discussion on the influence of labels on the final result. On the other hand, arrays of objects, probabilistic pose selection and probabilistic pose placement are optional operations that are not necessary for every scene, but give results that would be hard to achieve with the standard propagation operation or further facilitate object placement. All of the operations are used at some point in the examples. See the additional material for details.

10. RESULTS

We have implemented a simple prototype of our method in MATLAB. The prototype has a user interface that supports the operations and settings described in Section 9. A typical modeling session starts with a floor plan containing only a few objects, usually only the rooms. Note that it is necessary to have at least one object in the scene before an edit propagation is meaningful. A user of the interface can then place (position, scale and rotate) objects from a library of labeled objects and apply edit propagation. With a slider, the user can interactively set a threshold on the importance of placements that are displayed. The user can then select arbitrary objects and integrate them into the scene. Subsequent edit propagations take relationships to previously placed objects into account.

The computational complexity of an edit operation depends on the number of level sets, which is proportional to the number of features in the source room N_i^S and the total number of features N_i^F having the same label i. Each feature in the source room (or more generally parent object of the source pose) has to be matched with all the features having the same label in all target rooms, resulting in a computational complexity of $O(\sum_{i \in L} N_i^S N_i^F)$, where L is the set of labels in the scene. Each level set is rasterized on a 2D grid using standard rasterization techniques, as described in Section 6.3. In practice, N_i^S does not depend on the complexity of the scene, since, for example, larger floor plans usually have more rooms, but not more objects inside a room. N_i^S is also typically very small compared to N_i^F, so the complexity is approximately linear in the total number of features of a scene. In our implementation, one edit propagation typically takes between 1 and 6 seconds, up to approximately 15 seconds in the fully populated crown example (Figure 12). This example is close to worst-case for our method, since all features are placed in only two 'rooms' (i.e., the two crowns), resulting in a large N_i^S. Please note that our algorithm is an unoptimized proof-of-concept, and timings can be improved.

We demonstrate our method by populating two extensive scenes with a large number of objects. The goal is to create plausible object positions with few edit operations. We show only the result and the starting configuration for each of the two scenes. Unless noted otherwise, all relationship functions described in Section 4.2 were used for each edit operation. In a typical editing session, we placed the objects in no particular order, only following the semantic dependencies of objects (e.g., placing the night tables after the bed; see Section 10.1 for a discussion on the order of operations). A few operations required one or two retries with small adjustments of the input pose, e.g., if there was too little free space in one of the target rooms to fulfill the requested geometric relationships. For the results of each individual step, please see the additional material.

The first scene is one floor of an apartment building, as shown in Figure 10. The starting configuration for this example is a bare room layout. Rooms are labeled as bedrooms, living rooms, kitchens, closets, toilet rooms, and bathrooms. For illustration, different apartments are shown in different colors, although this has no effect on the algorithm. Several pieces of furniture are propagated successively, as shown by the red numbers. The manually placed query object for each operation is encircled. Note that even though the rooms have different shapes and sizes, and the furniture inside the rooms has different arrangements, our method ensures that object positions are always plausible and similar to the position of the query object. In this example, a total of 160 objects were placed with 26 edit operations.

The second scene is a palace garden, shown in Figure 11. Here, the domain is divided into several parcels, and the starting configuration only includes the boundaries of these parcels. Parcels are labeled as way, garden, and palace. These parcels only serve as a guide for the placement of the actual objects and are deleted at the end of the editing session. Placed objects include flower beds, flower decorations inside the flower beds, fountains, trees, statues and pagodas. In this scene, we use several different operations to propagate objects. In addition to the usual edit propagation, we use object array operations to place the objects that are aligned in straight lines or circles, such as the bushes around the fountains. In step 19, we use local coordinate systems defined with respect to the selected flower beds to propagate all flower decorations of the source flower bed in one operation. The local coordinate system is composed of the boundary distance relationship R_{bd} and a custom relationship R_{bf} based on the arc length from a fixed start point on the boundary to the position on the boundary closest to the query pose. To propagate the large trees, we applied the probabilistic pose placement described in Section 9. A total of 617 objects were placed with 27 edit operations.

Our method is not limited to architectural applications, but it can also handle other types of polygonal scenes as long as it is possible to find a clear set of relevant features (we discuss suitable types of scenes in Section 10.1 and Figure 20). In Figure 12, we demonstrate placing gems and ornaments on two crowns. Since we only handle two-dimensional domains, the crowns are unrolled using a cylindrical mapping before performing the edit operations (Figure 12, top row). The starting configuration is the outline of the two crowns shown in black. Objects labeled as ruby, emerald, pearl, diamond and ornament are propagated on the crowns. Note that each edit operation is propagated to both crowns. In each operation of this example, we use the *centroid direction* relationship function in addition to the other relationship functions. The centroid direction is the angle between the direction from pose to object centroid and the principal direction of the object. As mentioned earlier, our method can handle arbitrary relationship functions. In step 10, we use an object array operation to place both lines of pearls. A total of 122 objects were placed in 10 edit operations. Finally, we map the result back to the 3D crowns (Figure 12, bottom).

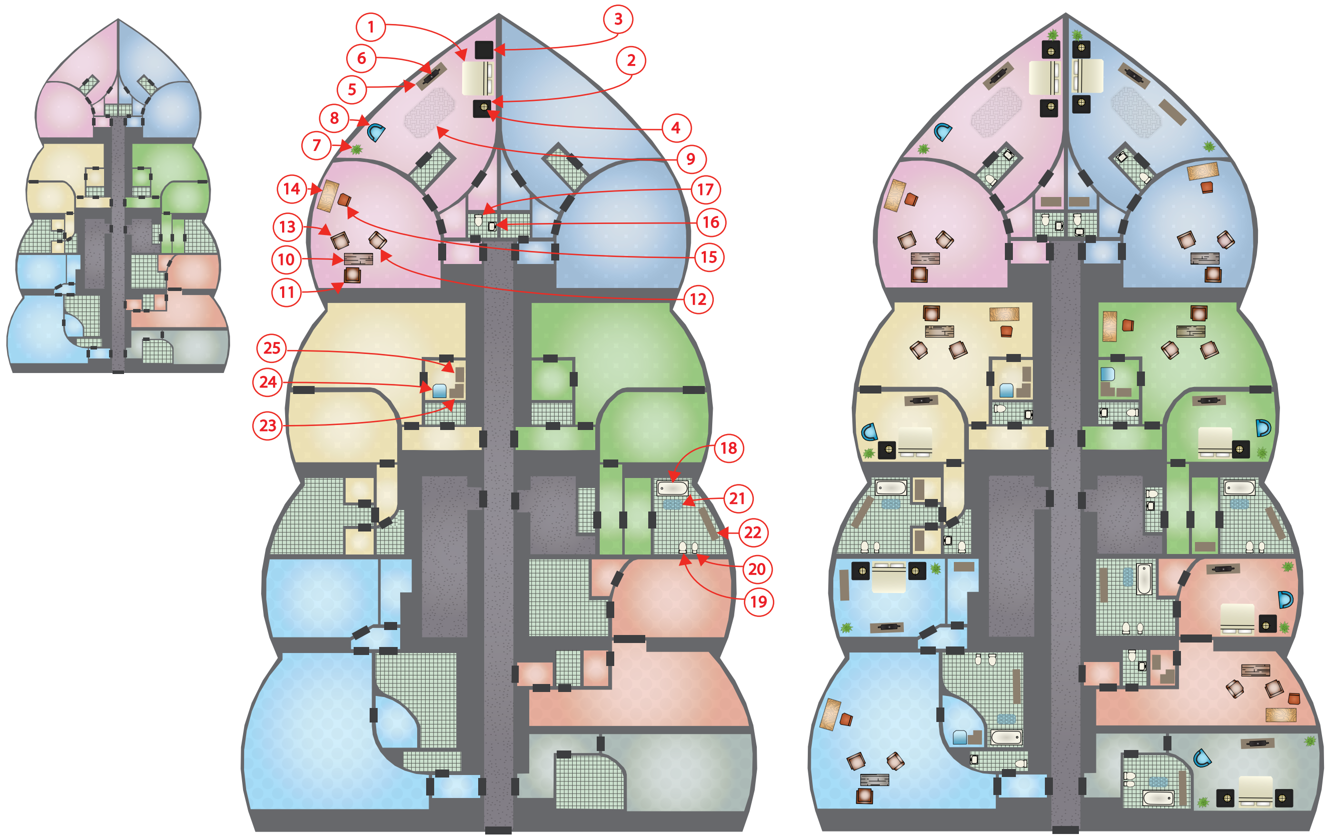

Fig. 10. In the apartment-building scene, 160 objects were placed with 26 edit operations. The left image shows the starting configuration, the middle image shows the individual edit operations – the numbers indicate the order of the operations – and the right image shows the final result. Zoom in to see details or refer to the additional material for a full-sized version and individual steps.

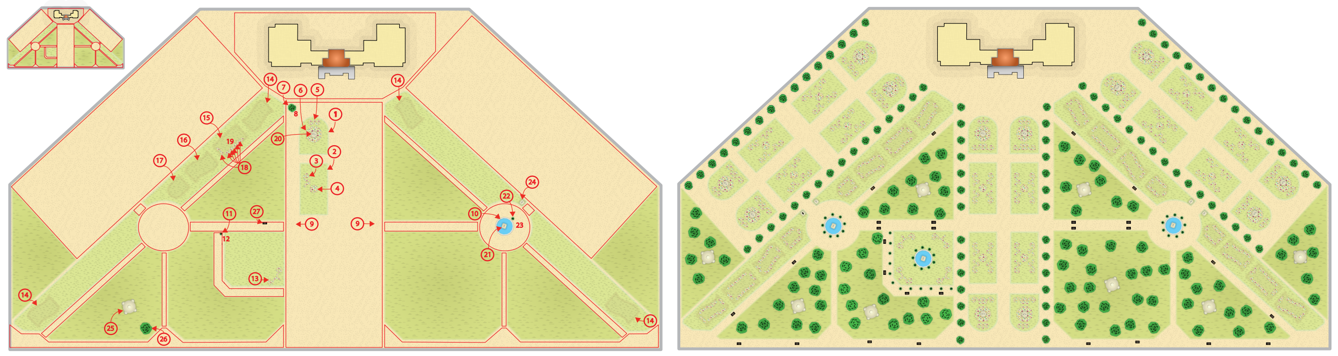

Fig. 11. In the palace-garden scene, 617 objects were placed with 27 edit operations. The left image shows the starting configuration, the middle image shows the individual edit operations – the numbers indicate the order of the operations – and the right image shows the final result. The red lines in the left and center image are polygons that guide the initial object placement and are removed in the end. Zoom in to see details or refer to the additional material for a full-sized version and individual steps.

10.1 Discussion and Limitations

In the following, we discuss several aspects of our method. We clarify the influence of the individual terms on the final result, examine approximations used in our approach, provide comparisons to shape matching with a local descriptor and discuss several limitations of our method.

Influence of Local Importance and Matching Accuracy. We use two terms to determine the relevance of a level set: the local importance I_l and the matching accuracy I_m. Both are needed to

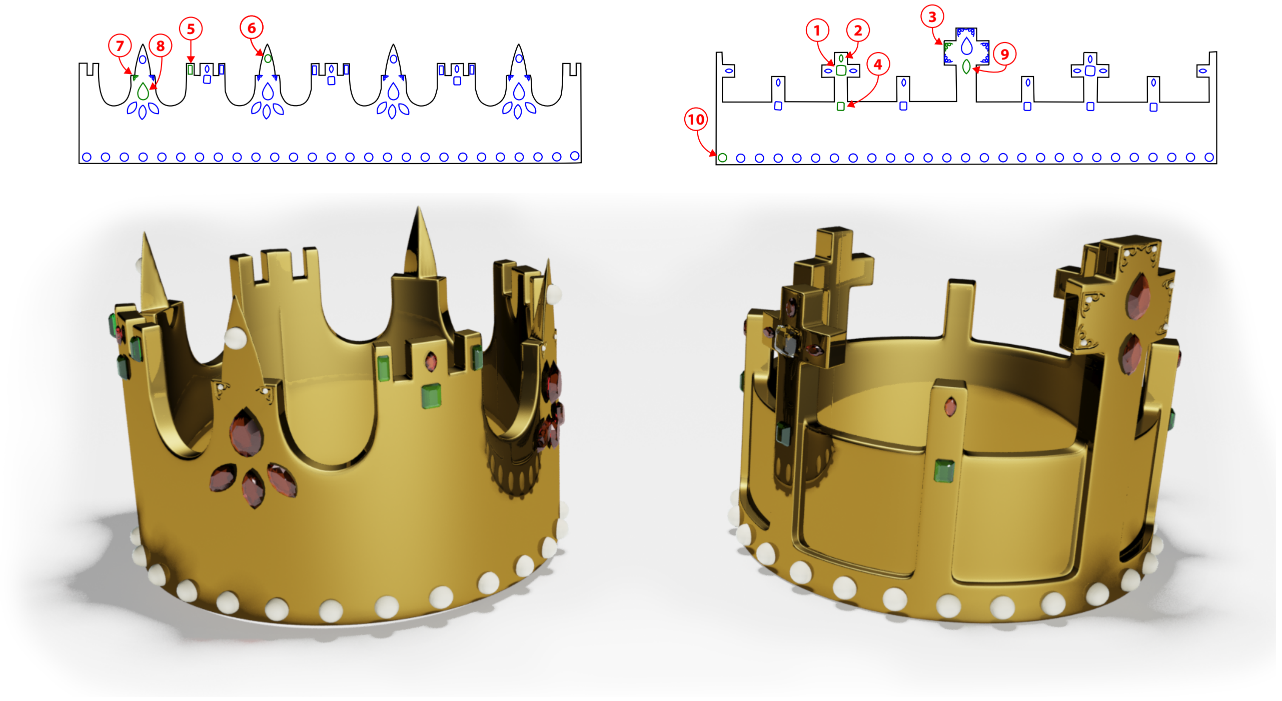

Fig. 12. Placing gems and ornaments on two crowns. In this example, we place all gems and ornaments for both crowns in ten edit operations. Each edit operation is propagated to both crowns. The crowns are unrolled using a cylindrical mapping before performing the edit operations (top row). Source objects are shown in green; propagated objects are shown in blue. The bottom row shows the final result mapped back to the crowns.

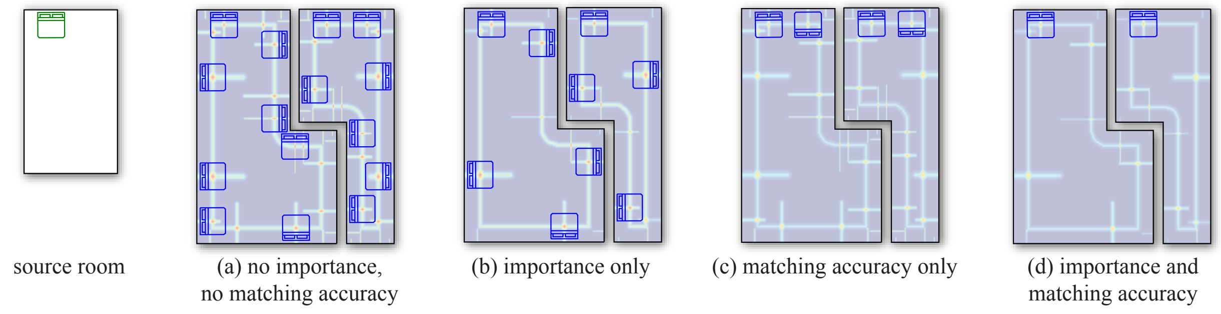
source room
(a) no importance, no matching accuracy
(b) importance only
(c) matching accuracy only
(d) importance and matching accuracy

Fig. 13. Influence of the local importance I_l and the matching accuracy I_m on object placement in a simple scene. By setting I_l to a constant (a and c), all features in the source room are weighted equally, independent of the distance to the source pose. When I_m is a constant (a and b), all features of the target room have the same matching quality. Low-quality poses in the target rooms are filtered out only when using both terms (d).

filter out level sets of low relevance. We show the effect of setting either term to a constant in Figure 13. For clarity, we use only the boundary distance and segment arclength relationship functions in this example. When setting I_l to a constant (Figure 13 a and c), all features in the source room have the same importance, e.g., the segment on the far side of the room has the same importance as the segments next to the bed. The level sets from these less-important features create maxima of low quality, such as the beds facing the wrong direction in Figure 13 c. By setting I_m to a constant, all features have the same matching quality, e.g., a match between a long and a short segment has the same score as a match between two short segments. The results are additional maxima along the badly matched segments. Undesirable poses are only filtered out when using both terms (Figure 13 d).

Influence of Relationship Functions. Another interesting aspect to discuss is the contribution of each relationship function to the final object placement. Each contribution varies from operation to operation and between output poses of the same operation. The relative contributions depend on the types of level sets that contribute to each maximum in pose space. Figure 14 shows the contribution of each relationship function in a typical sequence of operations. Level sets are shown in red. The composition of each maximum belonging to either the green source objects or the blue output objects is shown in the histograms next to the rooms. For example, the placement of the bath tub in the first operation depends mainly on boundary distance, segment arc length and segment distance. In many cases, not all relationships present in the source pose (green) are also present the output poses (blue). For instance, the bathtub in

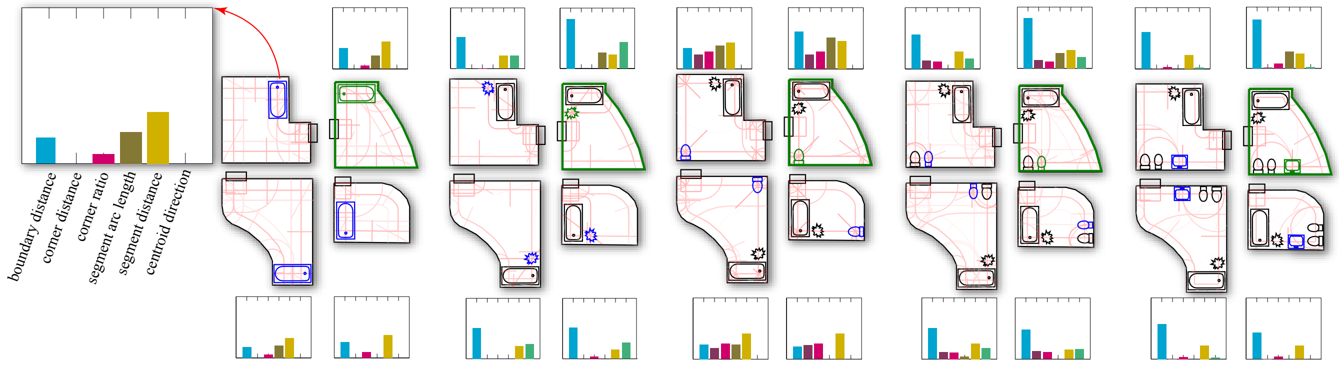

Fig. 14. Influence of the individual relationship functions on object placement. The histogram bars show the relative influence of each relationship function on the propagated poses, as well as the importance of each relationship type for the source pose in the green room. Level sets are shown as red lines in the background. Note that in general not all relationships are used to propagate a pose, but all relationships are used at some point in the sequence of operations.

the lower-right room does not have the same segment arc length as the bathtub in the source room. Although not all relationship types are used in every edit operation, none of them is superfluous. Each relationship type is used at some point in the whole sequence of operations. Note that as in the crowns example, in this example we also use the 'centroid direction' relationship type.

Order Dependency. When propagating multiple objects, the final result usually depends on the order of operations. Objects are propagated one at a time, and each propagation takes objects that have already been placed into account. Figure 15 shows the result of propagating five objects using the same operations, but in two different orders. Note how the shelf placed in the first operation of sequence e,a,c,b,d is propagated to a different pose in the lower right room than in the sequence a,b,c,d,e (last operation), since there are fewer relationships that constrain its placement. Both sequences result in different but plausible object placement. Often there is a natural dependency between objects that suggests a certain order of operations. For example, the placement of the bathmat depends on the placement of the bathtub and should therefore be placed after the bathtub. If there are no such constraints, it is generally advisable to place larger objects first, when there is still enough free space for them. In future work, we may consider transferring all objects in a room at once and automatically resolving all mutual constraints between the objects. However, in many cases, incremental furniture placement might still be preferable, since the amount of work is the same and the user has more feedback as well as more control over the final result.

Influence of Labels. In our floor plan editing application, we rely on labels to provide additional information about objects in a scene. In actual floor plans, relationships often depend on the semantics of objects in addition to their geometric relationship. For example, a night table is usually found next to a bed and not in the living room. Since the focus of our method does not lie in object recognition, we use labels to capture the semantics of objects. Figure 16 compares the results of propagating a night table in a typical floor plan without labels (left) and with labels (right). Note that without labels, there are additional maxima in the living room because our method has no way of distinguishing between beds, tables and chairs or between living rooms and bed rooms. However, the method does not break down completely. Even though the results are unintuitive for a floor plan, geometric relationships (e.g., close to the inner boundary of a polygon, close to other polygons from the outside, approximately at the center of a polygon segment) are met by all propagated night tables.

Approximation of 3D Accumulation. When accumulating level sets, we approximate the full 3D pose-space accumulation (position and direction) with a more efficient accumulation in 2D position space and take the direction with maximum weight at each position, as detailed in Section 6.2. For most cases, this results in the same output poses as in the full 3D pose-space accumulation, but there are some cases where the results are different. Figure 17 shows the same sequence of operations performed on the same scene with standard accumulation in 2D position space (left) and accumulation in full 3D pose space (right). All objects in the green source rooms are propagated to the remaining rooms in six edit operations. As in the crowns example, in this example we also use the 'centroid direction' relationship function. Note that most objects are propagated to the same poses (the small differences in direction are due to the discretization of the direction in the full 3D pose space). Two differences are introduced during the propagation of the two blue objects.

The first difference is the placement of the TV. In the source room, the TV is parallel to the wall and facing the bed. These two directional relationships cannot be satisfied in the far right room – the TV must either be parallel to the wall or facing the bed. In this room, there are two level sets with directions parallel to the wall (corner distance and boundary distance from wall) and two level sets with directions facing the bed (boundary distance and centroid direction from bed) meeting at one position. In the 2D case, all four level sets form one maximum and the direction with maximum weight is taken (facing the bed). In the 3D case, there are two maxima, shown as two versions of the same room in Figure 17. One maximum has a direction facing the bed; the other one is parallel to the wall. The four level sets do not actually meet in full 3D pose space, only the level sets shown in each of the two versions of the room meet. When using the full 3D pose space in practice, we would have to resolve such overlapping objects by either letting the user pick one of the objects or by automatically taking the object with maximum importance.

The second difference is the lavatory placement in the lower toilet room. Similar to the TV placement described above, two level sets with different directions (boundary distance to door and boundary distance to wall) are accumulated in the 2D case, whereas in the

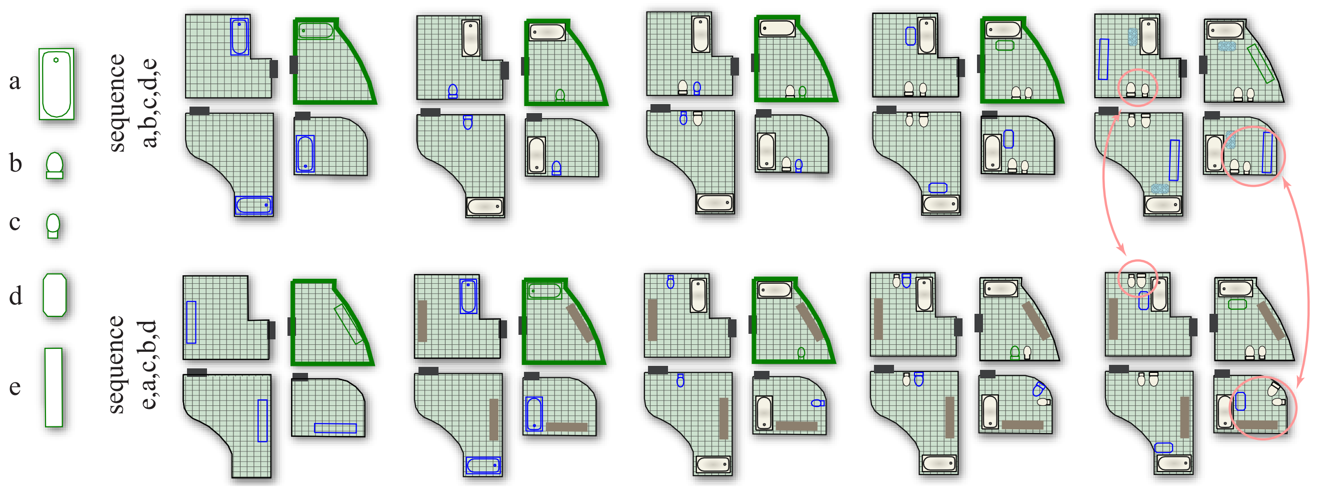

Fig. 15. Dependence on the order of operations. When propagating the five objects, bathtub (a), toilet (b), bidet (c), bathmat (d) and shelf (e), from the green source room to three target rooms, the final result depends on the order of operations. The top row shows each step of operation sequence a,b,c,d,e; the bottom row each step of sequence e,a,c,b,d. Note how object placement depends on the order of operations. Different placements are encircled in the right image.

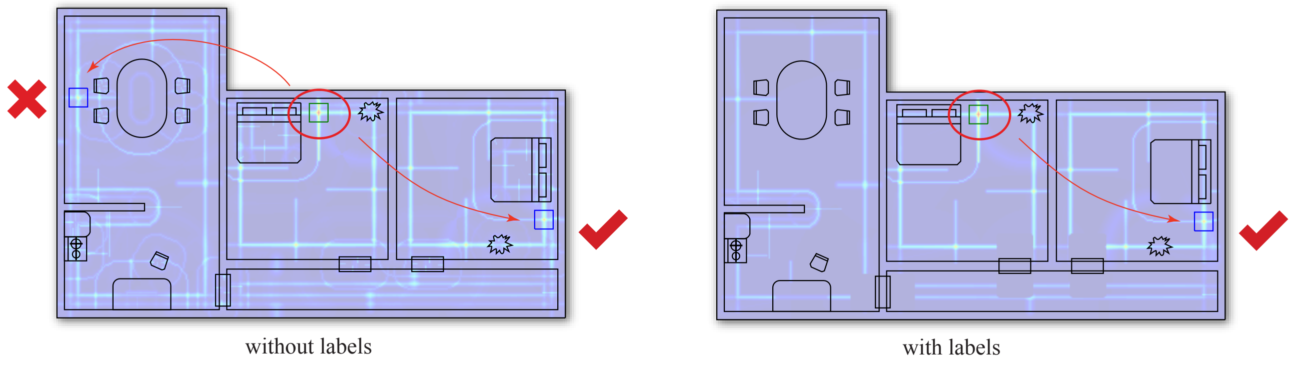

Fig. 16. The effect of labels on object placement. We propagate the single night table encircled in red. Note that without labels, there are additional maxima at positions that have relationship values similar to the source pose, but to object types that are not present in the source room. The accumulated level sets are shown in the background.

3D case, the two level sets do not meet. Unlike the TV propagation, there is no maximum at the same position (no level sets that meet at this position) and a different maximum is chosen instead. As a result of the different lavatory placement, the toilet is paced at a different position as well. Note that in most cases, the result of 2D position space and full 3D pose-space accumulation are the same and none of the differences result in unintuitive object placement. However, the speed gain and memory saving from accumulating in 2D position space are significant. Using 2D position space is more than one order of magnitude faster (1.4-4.3 seconds per operation in 2D position space versus 39-85 seconds per operation in full 3D pose space) and needs almost two orders of magnitude less memory (9.1 MB for discretizing 2D position space versus 762.9 MB for discretizing the full 3D pose space – in this example, directions are discretized into 100 bins).

Comparison to Shape Contexts. Local shape descriptors like the popular shape contexts [Belongie et al. 2002] can be used to establish a mapping between two polygons. In Figure 18, we compare our method to two methods based on shape contexts. A single object is propagated from the source room to three target rooms (rows 1-3) and from a room containing multiple objects to a target room containing a different arrangement of objects (row 4).

The first method (SC B-Spline), based on code by Dirk-Jan Kroon [Kroon 2011], establishes a one-to-one mapping between the source room and each target room. Shape contexts are computed at regularly sampled positions on the boundary of source and target rooms, and matching pairs of shape contexts provide constraints for a B-Spline deformation grid. The grid is refined iteratively to be as smooth as possible and to avoid fold-overs. In the case of multiple objects, shape contexts are placed on the boundaries of all objects. Affine transformations of the source objects (first column) are handled well by shape contexts, but stretching occurs for rooms that are not related by an affine transform (second column), resulting in bad object placement. Since the mapping from source to target room has to be kept diffeomorphic, the mapping cannot accommodate all matches between multiple objects in source and target rooms (last row) if they do not have the same spatial arrangement.

The second method (SC Inner) is a straightforward application of shape contexts to the area of the polygon (as opposed to the bound-

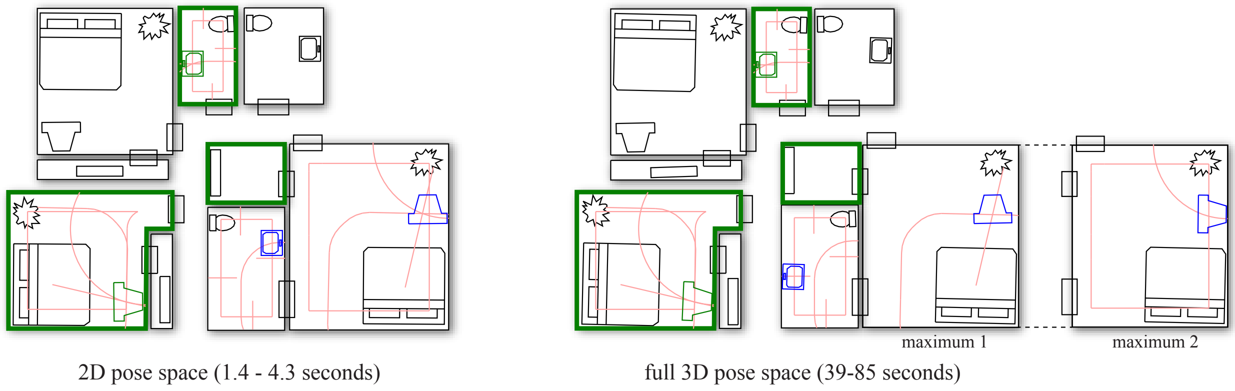
maximum 1
maximum 2
2D pose space (1.4 - 4.3 seconds)
full 3D pose space (39-85 seconds)

Fig. 17. Level set accumulation in the 2D space of positions (left) as described in Section 6.2 and in full 3D pose space (right). We propagated all objects in the rooms marked in green (bedroom, toilet and closet) in six edit operations to the remaining rooms. Note that the result is mostly identical except for small differences introduced by the propagation of the two objects shown in green (TV and lavatory). Relevant level sets of the two operations are shown as red lines in the background. For the TV propagation in full 3D pose space, there are two equivalent output poses at the same position, shown as two versions of the same room on the right.

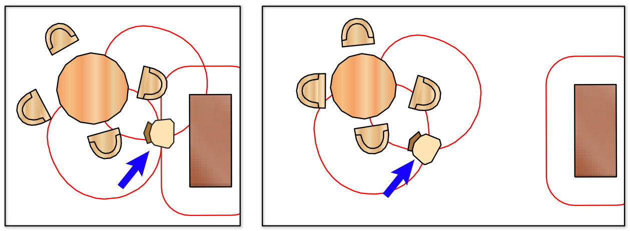

Fig. 19. Our method cannot handle cases where the desired local importance I_l does not depend on proximity. The red level sets show that the propagation uses the geometric relationships to the chairs instead of the desk to do the propagation. This produces an unintuitive result.

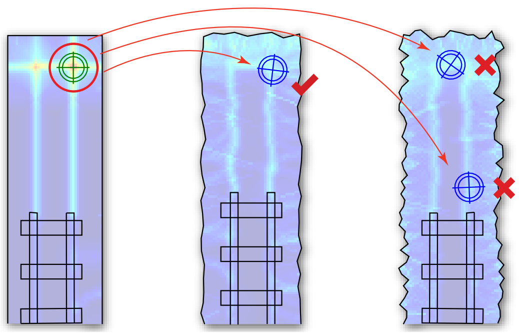

Fig. 21. We show three shapes with increasing noise from left to right. A single object encircled in red is propagated from the left shape to the two shapes on the right. Since we make no attempt to de-noise the shapes, noise can introduce false features like additional corners and segments that clutter pose space and reduce the propagation quality (middle shape) or even completely hinder propagation (right shape).

ary). We construct a three-dimensional regular grid over all poses (position and direction) inside the source room. A shape context is computed at each grid point and compared to the shape context at the source pose. The quality of the best match at each position is shown in the background of each target room. The bed is placed at the pose with highest matching quality. Since shape contexts cannot match geometry related by a non-affine transform, no clear maxima can be found in the second, third and fourth rows of Figure 18 for the 'SC Inner' method. Our method, on the other hand, is neither constrained by a one-to-one mapping, nor does it require the source and target geometry to be related by an affine transform. It can find clear maxima and good positions in all of the rooms.

Distance-Based Local Importance. Currently, the local importance I_l of relationships is determined by a predefined equation based on distance to the source pose. In some situations, however, it might be preferable to base the importance on other criteria. Consider Figure 19, where the chair marked by the arrow is propagated from the left to the right room. The only relevant relationship for the chair is to the writing desk. The relationships to the chairs around the other table are irrelevant. However, as is illustrated with red level sets, the relationship to the two chairs is stronger and produces a higher maximum, resulting in an unintuitive placement for the chair. Similarly, scenes might be constructed where distant features are more important than nearby features. These situations can currently not be handled by our method, but we plan to explore strategies for learning I_l from user interactions in future work.

Increasing Scene Difficulty. Since our approach focuses more on the relationships of a pose to features than on the features themselves, our method favors scenes in which the quality of a propagated pose is defined mainly by relationships rather than matching quality. The similarity of two features is determined only coarsely, using high-level information such as the polygon area or segment arc length (see Section 5.2). For this reason, scenes with many simple features are best suited for our method (see Figure 20). Decreasing the number of relevant features while increasing their complexity reduces the effectiveness of our approach, and would make it necessary to incorporate more selective feature-matching algorithms. Although we could use more sophisticated feature-matching methods to compute the matching accuracy I_m, e.g., methods based on the local neighbourhood of a feature, this is not the focus of our method and we opted to keep the computation of I_m simple.

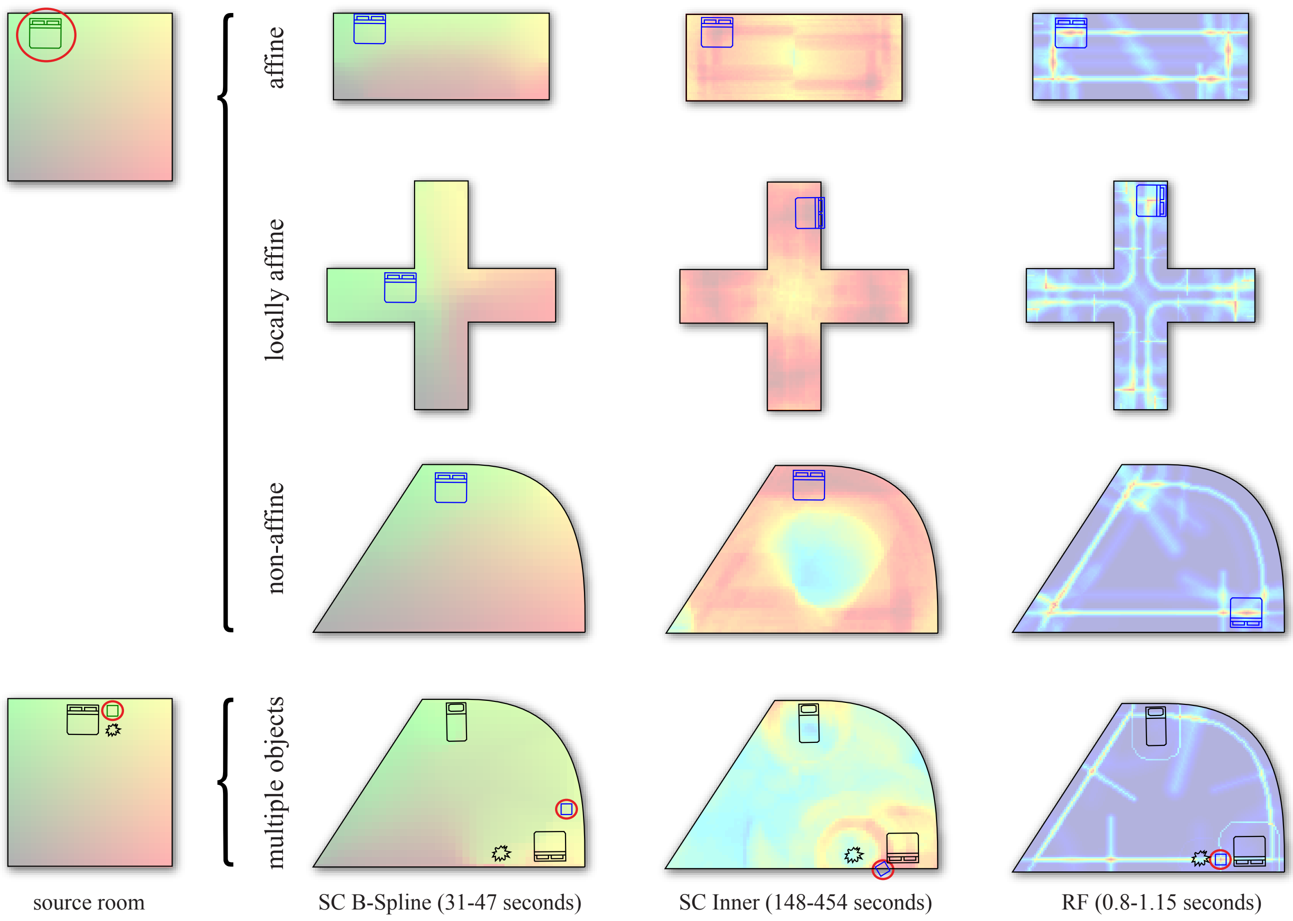

Fig. 18. Comparison to Shape Contexts. We propagate a single object from the source room to three target rooms using Shape Contexts with a B-Spline deformation grid (SC B-Spline), Shape Contexts computed at grid points inside the polygons (SC Inner) and our Relationship Functions (RF). The first room (top row) is an affine transformation of the source room; the second room (second row) is locally affine to the source room; while the third room (third row) is neither. The last row shows the result of propagating a night table from a source room containing multiple objects. The SC B-Spline column is a one-to-one matching, so we show the deformed coordinates of the source room color coded in red and green. In the other columns, we show the accumulated level sets as heat maps. Note that in the affine row, all methods give good results. As the transformation from source to target shapes becomes less affine, the maxima of the Shape Context become less pronounced. Our method finds good poses and clear maxima in all four cases, since it does not rely on partial shape matching where the matches have to be related by an affine transform.

Noise. Similar to overly complex local structures, noise in the input shape influences the usefulness of our method. In our current implementation, we make no attempt to remove it. Consequently, adding noise to the boundary of shapes effectively increases their complexity and introduces many false features like additional corners and edges (see Figure 21). As explained in the last example, increasing the complexity of features decreases the efficiency of our method. Small amounts of noise can be handled (Figure 21, center), but at higher noise levels, false features become more pronounced and clutter pose space, thereby obstructing good maxima (Figure 21, right). In future work, we plan to remove the noise in a pre-processing step, e.g., by adapting an L1 reconstruction method [Avron et al. 2010] or using bilateral mesh denoising [Fleishman et al. 2003] to preserve sharp features.

11. CONCLUSION

In this paper, we have introduced the notion of geometric relationships for edit propagation in polygonal scenes. We have exploited the fact that for propagating object placement, mostly the object *pose* with respect to neighboring objects is relevant. Geometric relationships are more flexible and general than local coordinate systems, and we have shown that a local coordinate system can be constructed from certain relationships as a special case. The method is suitable for propagating object placements in general polygonal environments, and we have shown an application to floor plan editing, where it is especially useful when populating large layouts. The advantage over knowledge-based algorithms is that no semantic knowledge about the environment is necessary, and since placements are specified by example, they can easily adapt to many different styles of floor plans.

There are several avenues of future work. It would be interesting to investigate relationships that relate to more than one fea-

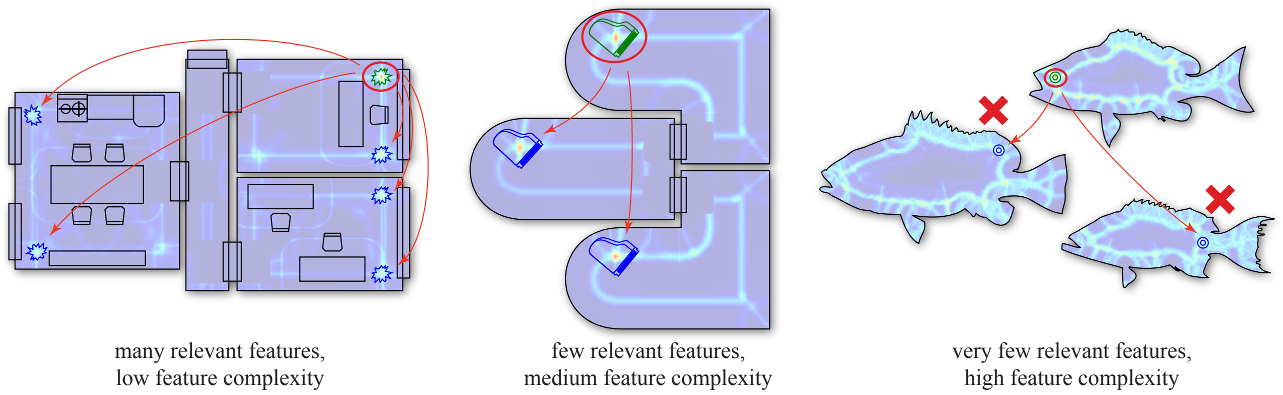

Fig. 20. Suitable types of geometry. We show three scenes with geometries that are increasingly difficult to handle for our approach. The left scene is well suited to our method; in the middle scene, we can still find good poses, while the right scene is not suitable. Since our method is based on relationships to features, not on feature matching, decreasing the amount of relevant features while increasing their complexity reduces the effectiveness of our approach.

ture, for example to encode relationships like "between feature A and feature B". Furthermore, relationships could be extended to incorporate more information about the source object than its pose. This could be realized by additional relationships between points of the source object. Thus, "extended" source objects could be propagated, such as a longer carpet that could adapt to the shape of the target room. Another interesting direction would be to transfer all objects in a source room at once and automatically handle mutual constraints. Finally, the weights I_m and I_l could be made more adaptable by extending them with a weight learned from previous edits or explicit user interactions.

ACKNOWLEDGMENTS

We would like to thank Jyh-Ming Lien and Evan Behar for making the source code of their method to compute Minkowski Sums available to us and Dirk Jan-Kroon for releasing his code for shape context matching on the Matlab File Exchange.

REFERENCES

AVRON, H., SHARF, A., GREIF, C., AND COHEN-OR, D. 2010. L1-sparse reconstruction of sharp point set surfaces. *ACM Trans. Graph. 29,* 5 (Nov.), 135:1–135:12.

BEHAR, E. AND LIEN, J.-M. 2011. Fast and robust 2d minkowski sum using reduced convolution. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*. San Francisco, CA.

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24,* 4, 509–522.

BOKELOH, M., WAND, M., KOLTUN, V., AND SEIDEL, H.-P. 2011. Pattern-aware shape deformation using sliding dockers. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. SA '11. 123:1–123:10.

BOKELOH, M., WAND, M., SEIDEL, H.-P., AND KOLTUN, V. 2012. An algebraic model for parameterized shape editing. *ACM Trans. Graph. 31,* 4 (July), 78:1–78:10.

FISHER, M., RITCHIE, D., SAVVA, M., FUNKHOUSER, T., AND HANRAHAN, P. 2012. Example-based synthesis of 3d object arrangements. *ACM Trans. Graph. 31,* 6 (Nov.), 135:1–135:11.

FISHER, M., SAVVA, M., AND HANRAHAN, P. 2011. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph. 30,* 4 (July), 34:1–34:12.

FLEISHMAN, S., DRORI, I., AND COHEN-OR, D. 2003. Bilateral mesh denoising. In *ACM SIGGRAPH 2003 Papers*. SIGGRAPH '03. ACM, New York, NY, USA, 950–953.

GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. 2009. iwires: an analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28,* 3 (July), 33:1–33:10.

GIL, J. AND WERMAN, M. 1993. Computing 2-d min, median, and max filters. *PAMI, IEEE Transactions on 15,* 5 (may), 504 –507.

HORMANN, K. AND FLOATER, M. S. 2006. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph. 25,* 4 (Oct.), 1424–1441.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26,* 3 (July).

KROON, D. 2011. Shape context based corresponding point models. Matlab File Exchange.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph. 27,* 3 (Aug.), 78:1–78:10.

MERRELL, P., SCHKUFZA, E., LI, Z., AGRAWALA, M., AND KOLTUN, V. 2011. Interactive furniture layout using interior design guidelines. *ACM Trans. Graph. 30,* 4 (July), 87:1–87:10.

SOLOMON, J., NGUYEN, A., BUTSCHER, A., BEN-CHEN, M., AND GUIBAS, L. 2012. Soft maps between surfaces. *Comp. Graph. Forum 31,* 5 (Aug.), 1617–1626.

WEBER, O., PORANNE, R., AND GOTSMAN, C. 2012. Biharmonic coordinates. *Computer Graphics Forum 31*, 2409 2422.

YEH, Y.-T., YANG, L., WATSON, M., GOODMAN, N. D., AND HANRAHAN, P. 2012. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM TOG 31,* 4 (July), 56:1–56:11.

YU, L.-F., YEUNG, S.-K., TANG, C.-K., TERZOPOULOS, D., CHAN, T. F., AND OSHER, S. J. 2011. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph. 30,* 4 (July), 86:1–86:12.

ZHENG, Y., CHEN, X., CHENG, M.-M., ZHOU, K., HU, S.-M., AND MITRA, N. J. 2012. Interactive images: cuboid proxies for smart image manipulation. *ACM Trans. Graph. 31,* 4 (July), 99:1–99:11.

ZHENG, Y., FU, H., COHEN-OR, D., AU, O. K.-C., AND TAI, C.-L. 2011. Component-wise controllers for structure-preserving shape manipulation. *Computer Graphics Forum 30,* 2, 563–572.

Received January 2013; accepted November 2013