Integrated Model/Scene Construction through Context-Based Search, Data-Driven Suggestion and Component Replacement

Po-An Chen and Chuan-Kai Yang
Department of Information Management
National Taiwan University of Science and Technology
No. 43, Sec. 4, Keelung Road
Taipei, 106, Taiwan, ROC
M10009101@mail.ntust.edu.tw,ckyang@cs.ntust.edu.tw

ABSTRACT

In recent years, 3D scene construction has become increasingly popular in movies and games. However, it is without doubt that the involved effort is significant, and therefore how to simplify such a process has drawn the attentions from many researchers. More specifically, the construction of a 3D scene consists of two parts: the creation of 3D objects, and their deployment. In general, one possible and popular solution is to *reuse* previous 3D scene construction results. In this regard, there are least two types of approaches. The first type of approaches places more emphasis on the spatial relationships. In particular, by placing a *query box* in the current scene and comparing its relationships with other objects under the current scene, a desired object in a previous scene can be retrieved if it shares a similar configuration. However, inappropriate representations of previous spatial relationships may lead to ambiguous or superfluous retrieval results. The second type of approaches focuses on the generation of a single object. A method of such could either start from an initial model and gradually evolve into a more complex/specific one by selecting a similar model in the database, or directly synthesize a new model by a combination of more than one model from the database. This paper proposes a framework that not only integrates the two types of approaches just mentioned, but also unifies the previous two different ways for model construction. In addition, the representation of spatial relationships is further refined so that more desired retrieval results can be obtained, together with a meaningful *object class* scheme to facilitate the involved interaction for model construction.

Categories and Subject Descriptors

I.4.8 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Modeling packages*

Keywords

Scene Construction, Context-Based Search, Data-Driven Suggestions, Component Replacement, Model Segmentation

1. INTRODUCTION

Due to the rapid advances in related technologies, the quantity and variety of 3D models have increased dramatically during recent years, and the popular use of *Google SketchUp* further confirms such a trend. With a powerful and user-friendly tool like this, a 3D model construction has become relatively easier than it was before. However, at least two issues remain. First, the purpose of constructing a model is often to place it in a scene, but how to find a proper location is still a problem. Second, in spite of the existence of powerful model construction tools such as Google SketchUp, oftentimes it is still difficult for a user to build a 3D model from scratch. Several approaches have been proposed, and based on their different perspectives, they can be categorized into two types. Nevertheless, these two types of approaches still share the essence of *reuse*, that is, they are developed to help a user search through existing 3D model/scene database to find a desired 3D model so that the design process could start or continue. The first type of approaches emphasizes on the spatial relationships among objects in a scene. For example, Fisher et al. [6] proposed to retrieve an object according to its spatial relationships with others, and the current scene and each scene in the database are compared to find the object sharing the most similar configuration. Though very useful, the involved spatial relationship only focuses on distance, while *angular* or *functional* relationship could also be considered. The second type of approaches pays more attention on the construction of a single object. For example, Chaudhuri et al. [2] proposed to construct a 3D model by starting from an initial model, and then gradually evolving into a target model through selecting a similar one in the database. Funkhouser et al. [9] adopted a different strategy to synthesize a new model by directly combining multiple models into one. It is interesting to observe that the two approaches just mentioned, though both focus on building a single model, provide different but non-conflicting ways.

Inspired by the aforementioned approaches, this paper proposes to search and synthesize a 3D model as shown in Figure 1. Our scene/model construction process consists of potentially three stages. First, given the current scene, whether it is newly or partially constructed, a user could place a *query box*, denoted as *QB* and shown as a *blue transparent bounding volume* in the figure in the current scene, and select a desired *neighborhood size*. Based on the spatial relationships in terms of distance, angle and relative size with respect to other objects, several *object classes* in the database are found and retrieved in the order according to their similarities in the corresponding scenes, thus completing the *context-based search*,

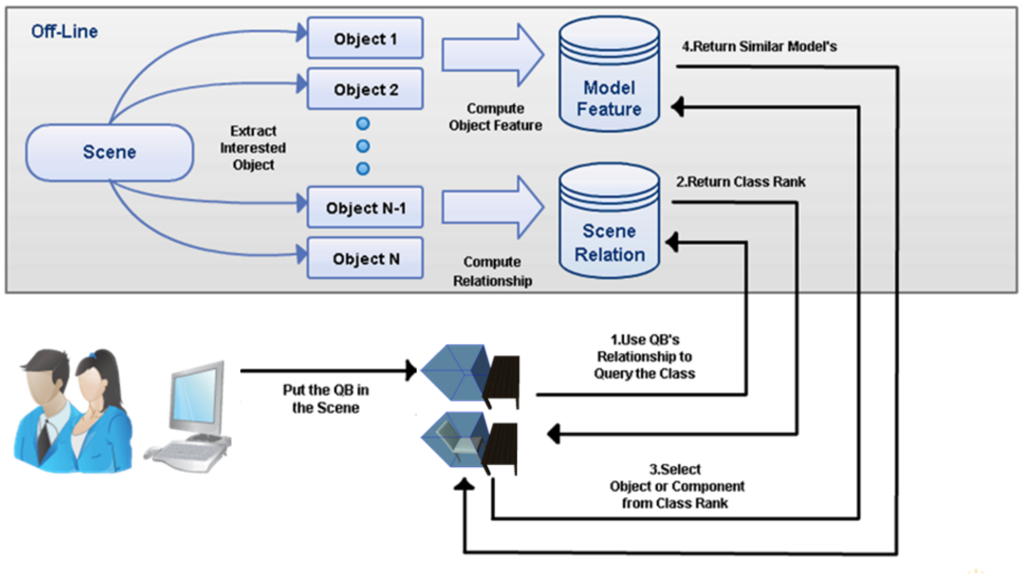


Figure 1: System overview.

similar to what is proposed by [6]. However, unlike [6] where *objects* are identified, we select the most possible *object classes* as our target for this stage for a smooth conjunction with the next stage, to be explained later. In addition, by taking angle and query-box size into consideration can further enhance the retrieval result, as will be shown in the Result Section and Section 4.4, respectively. Second, the user could choose one desired object class, and then a representative 3D model in that class is treated as an initial model to gradually refine its shape by repeatedly selecting a similar and wanted model from the database until no evolution is possible. Notice that to be able to perform the above *data-driven suggestion* as proposed by [2], a relative simple *template model* should be selected as the initial shape so that a desired evolution towards the target shape is more possible. This explains why in the first stage *object classes*, instead of simply *objects*, are chosen. As a result, in the corresponding *spatial relationship analysis* that makes *context-based* search possible, the information of objects belonging to the same object class should be aggregated and each object class should be represented by the one with a relatively simpler shape in that class to serve as the initial template for the ensuing evolution in the second stage. In particular, a *class scoring* scheme, as will be described in Section 4.3, is developed to properly aggregate the similarities of all of the objects in an object class regarding a context-based search. There is, however, yet another limitation for the model construction in the second stage, that is, the model evolution process can only terminate at an existing model in the database, which somehow still constrains the modeling capability or powerfulness. We therefore add the third stage, and in this final stage, the user could replace any part/component in the model by that from other models in the database, that is, perform a final model synthesis from multiple models. Moreover, according to the portions that remain fixed before and after the component replacement process, our system will rank the potential component replacements to facilitate the model refinement procedure. Though a final stage, we should remark that a model constructed in such a way is not final, that is, it can be added into our model database to enrich its capability for model construction in the future.

To sum up, the contribution of this paper is twofold. First, we propose a single framework that integrates the concepts of *context-based search* and *data-driven suggestion* for constructing a single model or a scene by introducing an *object class concept* and a *class scoring scheme* in between. Note that the output from the *context-based search* work proposed by Fisher et al. [6] is in the form of *objects*, while the work proposed by Chaudhuri et al. [2] starts from a *template*, thus the necessity of some intermediate forms, called *object classes* in this work, to bridge the gap. Furthermore, the way we evaluate the appropriateness of an object for a *context-based search* also needs to be changed to become evaluating an *object class* as a whole. In addition, we also make further improvement on refining the relationships among objects in a scene, such as relative angles and sizes, so that the unified framework becomes more desired for model retrieval and scene construction. Second, even for constructing a single model, we unify two different existing methods including *data-driven suggestion* and *component replacement*, that is, evolving through similar models and synthesizing multiple models into one, and we believe adding the *component replacement* is very beneficial as the data-driven suggestion process may converge to a target object shape which is still not satisfactory to a user. *As a result, in addition the aforementioned modification and* *supplement, we are not claiming any contribution on other parts for their improvements over existing techniques. Instead, the main contribution really lies on the smooth integration of multiple existing techniques to facilitate the task of model retrieval and scene construction.*

The rest of papers is organized as follows. Section 2 reviews literatures related to this work. Section 3 describes the required preprocessing to make our framework possible. Section 4 details the algorithms regarding how to retrieve the most desired object class by placing a query box in a scene. Section 5 shows how to gradually evolve from an initial 3D model into a desired one through selecting a similar model in the database, together with a potential replacement of some components of the previously derived model. Section 6 demonstrates various results generated from our system. Section 7 concludes this work and hints for several potential future directions.

2. RELATED WORK

There are two research fields related to this study. The first one is *shape retrieval*. Tangelder et al. [24] wrote a survey and summarized the general architecture involved for shape retrieval. Usually we need to extract some *features* or *descriptors* from a 3D model database in advance or even construct some *indices* beforehand so that at the run time the search results can be returned quickly and accurately. They also classified shape comparison into the following three categories: *feature-based methods*, *graph-based methods*, and *others*. For example, the *MPEG-7 3D shape descriptor* proposed by Jeannin et al. [12], represents a 3D mesh with its *curvature histogram*. Fisher et al. [6] proposed a novel search method for 3D models. A *query box* is placed in the current scene, and based on its spatial relationships with other objects, the most similar object in the database is returned and placed at the location specified by the query box. To increase search accuracy, both *vertical displacement* and object *tags* are considered during the matching process. Later on Fisher et al. [7] further extended the query box to be the whole scene so that we could look for similar scenes from the database. Eitz [5] developed a model retrieval system based on 2D inputs. In general, these 2D inputs should be *projections* along particular directions. To support 2D inputs, the original models are converted to 2D *line renderings* and the features of which are extracted for run-time comparisons. Lian et al. [16] proposed to search through *bag-of-features*. Each model should be first placed as *axis-aligned* or *symmetric* with respect to the transformed coordinate system, and then the projections along various angles are calculated to extract related features, and the *histogram of visual words* is constructed so that it could be used later on to compare with others during the retrieval phase. By taking into account that similar 3D models may be associated with different colors, Liu et al. [18] proposed to combine color and geometry into account during model retrieval. And the histogram of the colors and *Gaussian curvatures* of feature points is used as the *shape signature* for shape comparison. Focusing only on the shape of a model, Mohamed et al. [19] proposed to convert a model into a *skeletonized topological Reeb graph* with a normalized *mixture distance function*, and a model retrieval problem is then a graph matching problem. Instead of processing a mesh in the spatial domain, Li et al. [15] proposed to convert it into a *wavelet domain* which allows a simultaneous localization in space and frequency. In their *multi-resolution wavelet framework*, the *cubic spline wavelet* is chosen for its *invariance*, efficiency, and *discriminative power*, which altogether make such a system suitable for model comparison or retrieval. More recently, Huang et al. [10] proposed to organize heterogeneous collections of shapes based the concept of *Quartet*, which defines a relation between two pairs of models, and this topological relation must be maintained by the *categorization tree*. This organization can then be used for *model retrieval* as it can dynamically reorder models according to their *distances* or similarities with respect to a given model.

The second related field is *3D modeling*. Though in general this is a very broad field, some closely related literatures must be mentioned. Funkhouser et al. [9] proposed to synthesize a new model from multiple models, and the key technique is, given a part of a model, how to locate a similar part in other models in the database. They also developed methods regarding how to smoothly combine two models along the desired cut. Kalogerakis [13] proposed the idea of *component-based shape*, and based on this structure, existing components can be combined to form new 3D objects. Jain et al. [11] proposed a similar but more complicated approach by first performing an *offline phase* of shape analysis of existing models, including *shape segmentation*, *contact analysis* and *symmetry detection*. And in the *online phase*, components from different models can be combined in various but reasonable ways. Xu et al. [25] developed a *Sketch2Scene* system that allows a 3D scene to be constructed according to the sketch drawn by a user, and an object size in the sketch is also taken into account during the construction process. In fact, the combination of parts from different models to form a model was also proposed by Shen et al. [23], but was used to recover a model that suffers from errors or incompleteness, such as the ones obtained from a Kinect-related scanning system. With a corresponding 2D color image and a small-scale shape repository, their system can recover a 3D model to desired quality. Chaudhuri et al. [2] developed a powerful framework that can make *data-driven suggestions* for 3D modeling. A user could input an initial model, then the system calculates its related features and tells the user which parts could be added or what the most similar models are in the database. The user could follow the directions to gradually evolve into the final desired model. To partition a mesh into parts, the measure of *shape diameter function* [22], or SDF for short, and *approximate convex decomposition* [17], or ACD for short, were used. A quite related work, proposed by Kim et al. [14], provides an interface that allows a user to browse similar models by specifying *regions of interest*, or *ROIs* for short. The core idea is to construct a graph for each model and make use the concept of *fuzzy correspondence* to align different models.

3. MODEL PREPROCESSING

Before a model can be used later on for spatial relationship retrieval and data-driven suggestions, it should be preprocessed. In general, starting from a scene, we first select the objects that we are interested, and then segment these objects one by one. Together with the segmentation, the spatial relationships among objects can thus be obtained. In addition, each object should be *labeled*, that is, its *frontal* face should be identified and its object type or *class*, e.g., a chair or a table, should be specified. Furthermore, its shape features as well as the segmentation of itself into smaller components should also be computed so that it could be retrieved or synthesized with other models. More details are to be given in the ensuing subsections.

3.1 Labeling

In order to accurately calculate the spatial relationships among objects, we have to label each object in a scene properly, and the labeling type includes *spatial labels* and *object labels*. For spatial labels, the main task is to label the *frontal* direction for each 3D object, as shown in Figure 2, as such a direction will be needed for more efficient computation of the spatial relationships among objects, to be discussed later on. Currently the frontal direction is identified manually. For object labels, we have to label its object type of *class* information and also its segmented parts properly for later use. Note that the class labeling is also done manually for the following two reasons. First, for all the objects that we have collected from multiple sources, it is not always the case that the object comes with a label. Second, even in the case where the label for an object does exist, we may still need to adjust the label to support the *object class* concept mentioned in the next Section. For example, an armchair, a bench, and a wing chair should all be labeled under the *chair* class. For segmenting the object, we start from an initial partition based on the features of the object and gradually form the final segmentation. However, the results may not always be satisfactory, and thus we provide tools that allow users to tune the obtained segmentation results manually.

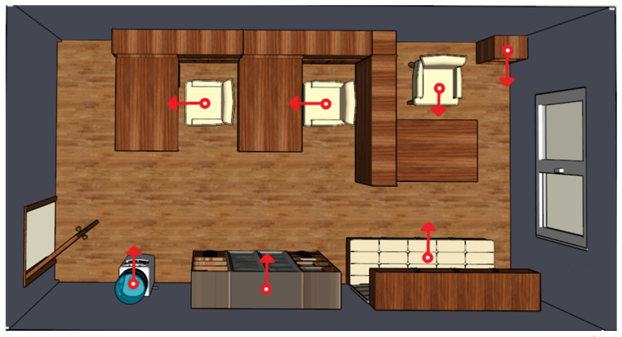


Figure 2: Frontal face labeling, where each white dot represents the mass center of each object, while the associated red arrow is the *normal vector* of the corresponding frontal face for that object.

3.2 Spatial Relationship Extraction

There are basically three types of spatial relationships considered in this study: *distance*, *angle*, and *vertical relationship*, while the last one refers to the case where one object is on top/bottom of the other.

The first one is distance. In order to compare the distance relationship across scenes, the distance used in each scene has to be *normalized*. Assume $D'(a, b)$ represents the normalized distance between object a and b, then:

$$D'(a, b) = \frac{D(a, b)}{Diag(Area(a \cup b))} \tag{1}$$

where $D(a, b)$ is the distance between objects a and b, as shown in Figure 3(a), and $Area(a \cup b)$ is the 3D region occupied by a and b together, while $Diag(Area(a \cup b))$ means the diagonal distance of the aforementioned region, as shown in Figure 3(b). To handle a vertical relationship, we modified the approach proposed by Fisher et al. [6], so that the vertical distance is also *normalized* for being able to compare across different scenes. The final distance feature between two objects a and b is represented as a 2D vector $D_{Feature}(a, b) = [D'(a, b), D'_h(a, b)]$, where $D'_h(a, b)$ is the vertical distance between a and b, or their distance along the height direction.

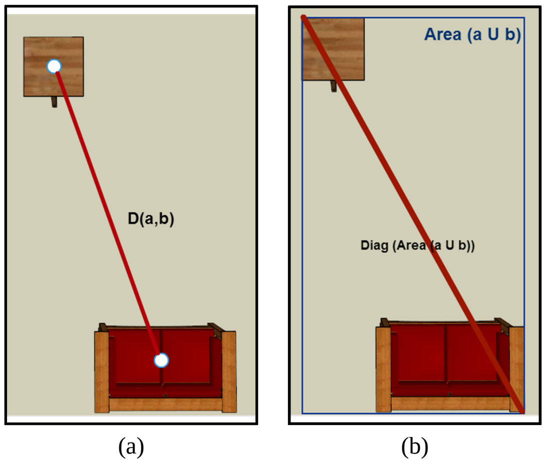


Figure 3: (a) The distance between two objects is defined to be the Euclidean distance between the their centers, determined by the marked red line. (b) The diagonal distance of $Area(a \cup b)$, denoted by $Diag(Area(a \cup b))$, is calculated from the marked red line.

The second one is angle. According to the mass centers of two objects and their corresponding frontal faces' normal vectors, their spatial relationship in terms of angle can be calculated, as shown in Figure 4. Note that, in this figure, the relative angle of object c_2, with respect to c_1, denoted by π_{c_1,c_2}, is θ, and it is evident that such a definition is *non-symmetric*, i.e., π_{c_1,c_2} is not necessarily equal to π_{c_2,c_1}.

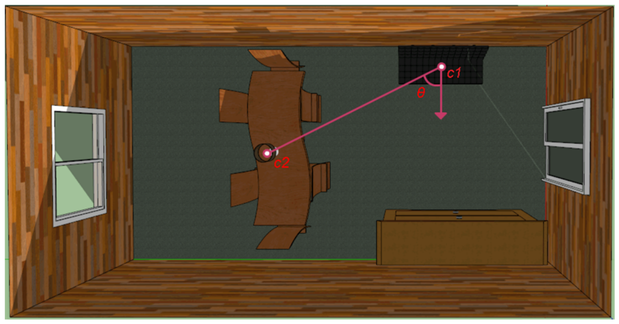


Figure 4: Angular relationship between two objects, where c_1 and c_2 are the mass centers for the two objects, while θ measures their angular relationship, from object c_1's perspective.

The final one is the vertical relationship. According to the bounding boxes of two objects, we could perform the calculation by the following:

$$Up(a, b) = \begin{cases} 1, & if\ \ min(a_h) \geq min(b_h) \\ 0, & otherwise \end{cases} \tag{2}$$

where $min(a_h)$ means the lowest height value of object a and similarly others. As a result, $Up(a, b)$ returns 1 if a is on top of b, and 0 otherwise. Note that in Equation 2, the objects a and b do not necessarily need to touch each other. For example, object a could be a *droplight*, while object b could be a *table*.

3.3 Shape Feature Extraction

We adopt the *shape diameter function*, or *SDF* for short, proposed by Shapira et al. [22], to extract shape features of a mesh to be used later on. Starting from an arbitrary point on a mesh, we shoot rays forming a cone with the central axis to be the *inverse normal direction* of the point on the mesh. If this mesh is *closed*, then each ray must intersect with at least one point with the mesh, and thus the distance for this ray, measured from the starting point to the intersection point, can be determined. Note that this calculated SDF needs to be normalized so that SDFs at all other points could be merged altogether, and we adopt the normalization method proposed by Shapira et al. [21]. Finally we make a *histogram* of the SDFs from all the vertices on a mesh, and an example of such is shown in Figure 5.

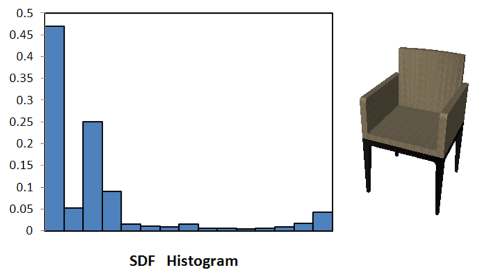


Figure 5: An example (a chair on the right) of a SDF histogram, where all the distance values have been normalized to be in the range from 0 to 1.

4. CONTEXT-BASED SEARCH

In a 3D scene, how an object is placed is often affected by its relationships with others. For this reason, we adopt the idea proposed by Fisher et al. [6], and further modify it to better fit our goal. In general, we consider the dimension of a *query box*, its relative distances and angles with respect to other objects. Figure 6 shows an example of such, where the object marked in blue is the query box, while the red arrow indicates the normal vector of its frontal face. And the results returned are ranked by their similarities, which are to be discussed in details in the ensuing subsections. Note that what are returned are in fact the potential *classes* that could be placed at where the query box is. And once a class, such as a mouse class is selected, we then replace the query box with the *initial shape* of that class, and then a user could gradually evolve the initial shape into a desired shape by repeatedly selecting similar models from the database. Here the *initial shape of a class refers to the object of the class that is the most often selected*. In case of a tie, the object with a simpler mesh segmentation is selected as the initial shape.

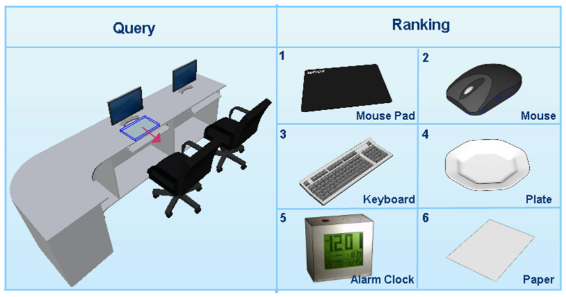


Figure 6: An example showing the results of a spatial query.

4.1 Relationship Retrieval

Using a spatial relationship for the retrieval of a single model has been discussed in [6, 7]. In particular, the spatial relationship in [6] is measured as follows:

$$K_{spatial}(f_{st}, f_{uv}) = G(z_{st}, z_{uv}, \sigma_z)G(r_{st}, r_{uv}, \sigma_r) \quad (3)$$

where s, t are any two objects in a scene, and u, v are two objects in another scene. f_{ab} represents the spatial relationship between a and b, r_{ab} the *absolute radial separation* of objects a, b in the X-Y plane, and z_{ab} the *absolute height displacement* of the objects a, b along the Z axis, respectively. Finally, $G(p, q, \sigma)$ is a Gaussian kernel defined as $G(p, q, \sigma) = e^{-||p-q||^2/\sigma^2}$. However, as our goal is to find an object class instead of an object for a query box, we modify the relationship measurement to become:

$$K_{spatial}(f_{ab}, f_{uv}) =$$
$$K_{up}(f_{ab}, f_{uv}) * (w_d G(d_{ab}, d_{uv}, \sigma_D) + w_\pi G(\pi_{ab}, \pi_{uv}, \sigma_\pi)) \quad (4)$$

Here a, b are objects in the current scene, and u, v are objects in another scene. d_{ab} is the *normalized distance* between a and b, that is, $D_{feature}(a, b)$, while π_{ab} is their relative angle, as discussed in Section 3.2. w_d and w_π are the corresponding weightings for the distance and angle terms, and currently are set to be 0.6 and 0.4, respectively. Note that how *normalized distance* and *angular relationship* are introduced in our *spatial relationship* consideration.

Finally, $K_{up}(f_{ab}, f_{uv})$ is defined as:

$$K_{up}(f_{ab}, f_{uv}) = \begin{cases} 1, & if\ \ Up(a,b) = Up(u,v) \\ 0, & otherwise \end{cases} \quad (5)$$

where $Up(a, b)$ has been previously defined in Equation 2.

Note that $K_{up}(f_{ab}, f_{uv})$ can help quickly filter out some unnecessary scene comparisons to speed up the corresponding spatial relationship calculation. In particular, the class information is also considered. For example, assuming we place a query box on a table, then what we are looking for in the scenes from the database is an object class that is put on top of a table (class).

4.2 Query Box Constraint

It is to our notice that, when searching for a 3D model to match the query box, not only its spatial location is important, but also its size along each dimension. Disregarding the dimension information of the query box may cause erroneous results since oftentimes when a user provides a query box, the size of the desired object is also roughly specified. However, as the size of each scene is different, we have to find a way to solve this issue.

Figure 7 shows the concept of our idea. Basically by making a *pairwise* comparison, we could solve this issue. For example, by treating the two objects in discussion as one, the *unified bounding box* can then be served as the base for computing the relative bounding box size or *proportion*. By considering this relative size, we could define object similarity, in terms of the bounding box size, as the following:

$$QB(a, b) = 1 - \frac{|b_{p_x} - a_{p_x}| + |b_{p_y} - a_{p_y}| + |b_{p_z} - a_{p_z}|}{3} \quad (6)$$

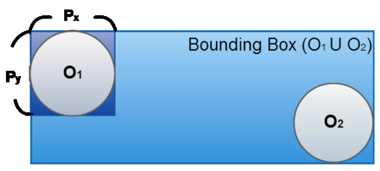


Figure 7: Example of computing the relative bounding box size, where P_x is the proportion along the x direction, while P_y the proportion along the y direction.

where the a and b in $QB(a, b)$ are two relative proportion just mentioned previously in Figure 7. With Equation 6 defined, we could change the original spatial relationship formula to become the following:

$$K'_{spatial}(f_{ab}, f_{uv}) = \\ w_{spatial}K_{spatial}(f_{ab}, f_{uv}) + w_{QB}QB(a_b, u_v) \quad (7)$$

where $w_{spatial} + w_{QB} = 1$, and they are the two weighting factors associated with the spatial relationship and the query box size, respectively, and can be tuned by users. In addition, a_b is a's relative proportion with respect to the bounding box formed by considering a and b together, as shown in Figure 7. Notice that, in the case where the returned object is larger than the query box, special care is needed. In order not to deform the object, we pick the largest dimension of the object and *uniformly scale down* the object to fit the query box.

4.3 Class Scoring

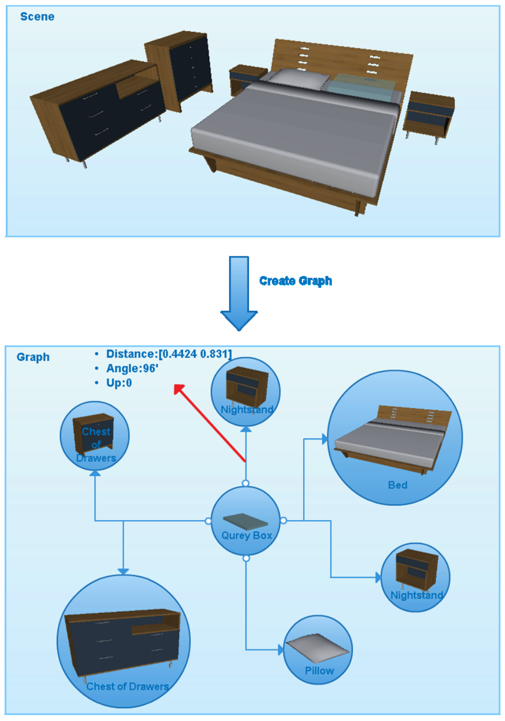


Figure 8: An example of converting a scene into a graph.

In order to know the most possible object class for the query box placed in the scene, we have to make use of the spatial relationships among objects, which can be converted into a *graph*, where each edge carries the information including distance, angle, and vertical relationship, as shown in Figure 8.

However, as it is possible that for a scene in the database, there is more than one object of the same class, then how to calculate the score for this class becomes an issue. In this study, we simply select the object that bears the most similar spatial relationship to the query box, and its score is returned for being the representative one for this object class, as shown in Figure 9.

Or more specifically, we perform the class score evaluation based on the following:

$$Score(q_{class} = c|S) = \\ \frac{1}{n_c}\sum_{c_i \in c_S}\frac{1}{n_u}\sum_{u \in NN(c_i, S_q)} \arg\max_{t \in S_q, t_{class} = u_{class}} K'_{spatial}(f_{qt}, f_{c_i u}) \quad (8)$$

Here S is a given scene, c_S is the set of objects belonging to class c in scene S, n_c the size of c_S, c_i an object in set c_S, u an object around c_i, S_q the scene where the query box q is located, respectively. $NN(c_i, S_q)$ is the set of objects near c_i but also present in S_q; in other words, we could use the scene S_q to filter out some unnecessary objects from further consideration. Finally, n_u is the size of $NN(c_i, S_q)$, while t_{class} is the class of object t.

Once we have the class score for an object class in a scene, we could then calculate its class scores for all the scenes in the database, and take the average of these scores for being the final score for this class:

$$Score(q_{class} = c) = \frac{1}{n_s}\sum_{S_i \in S_{DB}(C)} Score(q_{class} = c|S_i) \quad (9)$$

where $S_{DB}(C)$ is the set of scenes where the object class c appears, and n_s is the size of this set.

4.4 Class Ranking

When the scores for all object classes are obtained, we could rank and display the object classes accordingly, as shown in Figure 10. However, as mentioned previously, the size of a query box could potentially affect the score of a class. As is also shown in the same figure, once we change the size of the query box, the returned results change, both in object class and order. For example, the rank of the *lamp* class gets promoted while that of the *alarm clock* class becomes lowered.

5. DATA-DRIVEN SUGGESTION AND COMPONENT REPLACEMENT

A representative object from an object class is chosen as an *initial shape* [2], and the next step is to gradually select similar models in the database to evolve into the final model. To search for similar models, as in [2], we adopt SDF [22] for its easiness of computing, as well as its wide application occasions. In addition, we also apply the SDF retrieval approach proposed by Shalom et al. [20] as it can support retrievals from both local and global perspectives.

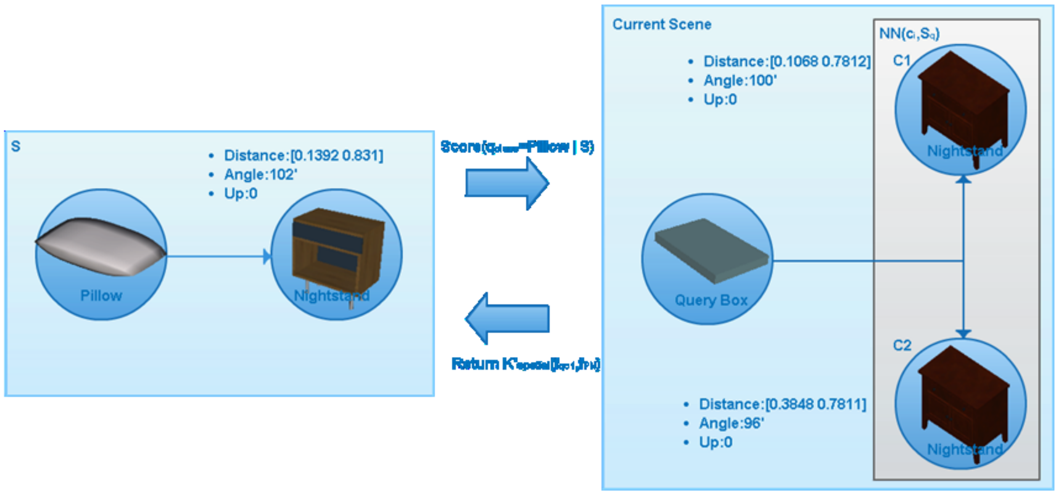


Figure 9: The calculation of an object class. As shown in the figure, when $K'_{spatial}(f_{qc_1}, f_{PN}) > K'_{spatial}(f_{qc_2}, f_{PN})$, we return $K'_{spatial}(f_{qc_1}, f_{PN})$. Here P denotes the *pillow*, while N the *nightstand*, respectively.

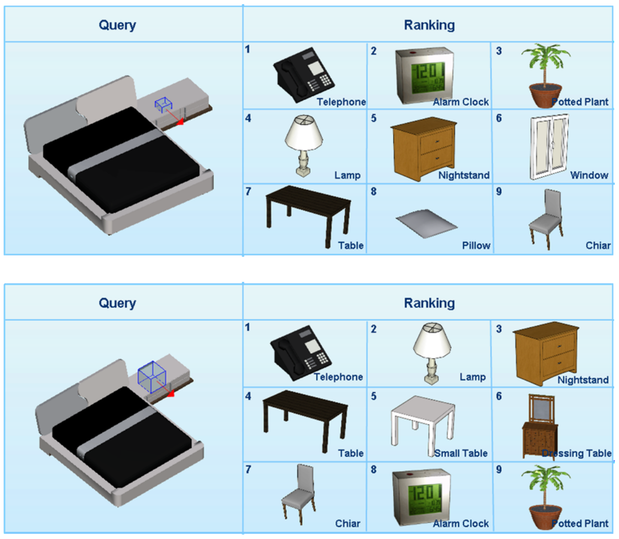


Figure 10: Two results for a query where different sizes of query box are used. Note that here we set both $w_{spatial}$ and w_{QB} to be 0.5 as mentioned in Equation 7.

5.1 Local Feature

Despite the merits of SDF, the lack of *locality* or spatial information could cause difficulty. We therefore adopt the approach in [20] to modify the shape feature extraction algorithm by *calculating SDF on each segmented component* after the mesh segmentation as proposed in [21]. The average SDF from all segmented components can be used to construct a *hierarchical graph*. From this graph, we could then determine the *edge value* between two nodes (components) x, and y:

$$Edge(x, y) = w_H(1 - \frac{\sum_{i=1}^{N_{bin}} |H_{x,i} - H_{y,i}|}{2}) + w_S QB(b_x, b_y) \tag{10}$$

Here N_{bin} is the number of bins in the SDF histogram, $H_{x,i}$ the value in bin i on the SDF histogram for the x component, w_H the weight for histogram difference, w_S the weight for the component size, and b_x the bounding box size for the x component, respectively. In our current implementation, w_H and w_S are set to be $2/3$ and $1/3$, respectively. Note that the bounding box should be normalized as similarly described in Section 4.2, while the formula of $QB(b_x, b_y)$ can also be found in the same Section.

With the edge values ready, we could compare between two components, as well as two 3D models, as to be discussed next. In particular, Equation 10 will be used in Section 5.3 for the component comparison through a bipartite graph, as discussed in [21].

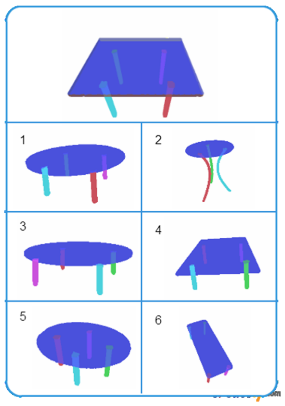
5.2 Feature Retrieval

We discuss how to compare two 3D models in this subsection, and defer the component comparison to the next one. And the technique described here is used to locate the *initial shape* to be ready for the next stage. Following previous discussion, once the object that best matches the query box is determined, according to its *SDF histogram*, we could search through the database to find the most similar object of the same class as follows:

$$S_{histogram}(q, p) = \begin{cases} 1 - \frac{\sum_{i=1}^{N_{bin}} |H_{q,i} - H_{p,i}|}{2}, & if \;\; q_{class} = p_{class} \\ 0, & otherwise \end{cases} \tag{11}$$

Each time an object in the database is selected this way, we could *update the current SDF histogram and use it to search through the database again*. This iterative process goes on until a user cannot find a newer model that suits his/her needs. Figure 11 shows two example results of such a process, together with their *similarity scores* listed in Table 1. From this point on, the user could perform *local feature retrieval*, to be described, to find desired components from the database, presumably from other models, so that a *new model synthesis* from multiple models can be achieved.

5.3 Local Feature Retrieval


(a): Table

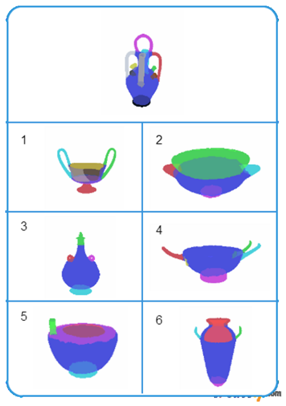

(b): Vase

Figure 11: Two results for an initial shape retrieval.

ranking	percentage	ranking	percentage
1	88.74%	1	82.05%
2	85.16%	2	77.80%
3	83.03%	3	72.72%
4	81.60%	4	72.57%
5	80.78%	5	70.81%
6	79.14%	6	69.93%

(a): Table (b): Vase

Table 1: The similarity scores for the initial shape retrieval in Figure 11.

Based on the *hierarchical graph* mentioned in Section 5.1, we could compare components from different models as follows. Similar to [21], we first build two new simpler graphs by connecting each component directly to its *root* or the whole model, and then the two new graphs are connected through the edges whose values have been discussed and calculated in the same Section. The resulting graph is a *bipartite graph*, with more details to be found from [21], and we then find the routes with the largest total edge weights to determine the similarity between the two components.

5.4 Model Synthesis

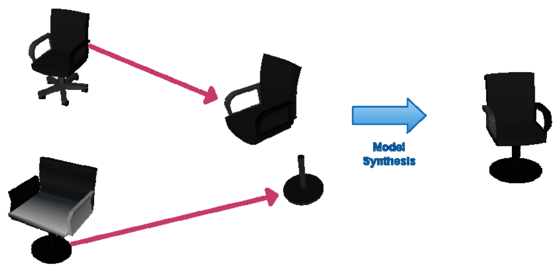


Figure 12: An example for model synthesis.

For model synthesis, as shown in Figure 12, we have to deal with the *model alignment* and *model stitching*. In general, when two components are to be synthesized together, the smaller component should be adjusted, through various geometric transformations such as scaling, rotations and translation, to fit the larger component. When one component is to be combined with two other ones, we also favor the larger component by transforming the smaller one first, and *patching the resulting holes* later if any.

To align two components, we adopt the *iterative closest point* approach [26], or ICP for short. Along the stitching *boundary contours* we randomly sample equal amount of points to create two *point clouds* to be aligned by ICP. And to speed up the correspondence finding in the ICP process, we apply a K-D tree [1] data structure.

6. RESULTS

We have developed our system on an Intel Core2 1.86 GHz machine with 4 GByte memory running on the Windows XP operating system, and using C# as the main programming language. The models used in this study are collected from the following three source: *Google 3D Warehouse* for about 150 objects, the datasets from the work by Chen et al. [3] for about 200 objects, and the datasets from the work by Chen et al. [4] for 60 objects, including 20 chairs, 20 tables and 20 vases, respectively.

Figure 13 shows a complete flow of this system through an example. Starting with the scene where only a table is present, a user could place a query box to find a desired object class. In this example, the *chair* class is chosen, then an initial shape of this class is used to "evolve" into a desired model by repeatedly selecting similar objects (of the same class) from the database until no more change can be performed. Now the user could turn his/her attention to the components of the object and continue the model change by replacing its component by that from other model, or equivalently performing a model synthesis to derive the final desired model. In fact, this newly synthesized model can also be added into the object pool in the original class to enrich its diversity.

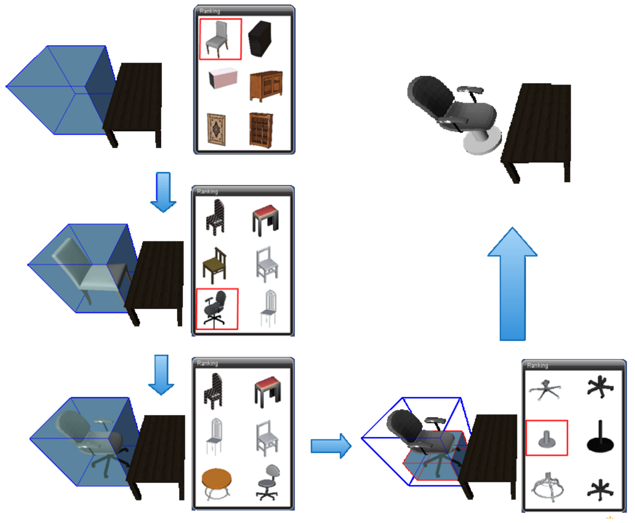


Figure 13: The system flow, where the red rectangle is the selected object.

To evaluate our system, we perform the following experiment. A query box with the same size is placed in two different scenes, as shown in Figure 14. Though different, these two scenes do share similar spatial relationships, and thus many common object classes are retrieved. Note that at least two object classes are different in these two scenes, that is, a nightstand in the first scene, and a dressing table in the second scene. The difference explains why the result sets are also a bit different. And also pay attention to the number one ranking object class, which does meet our expectation. Finally, as in previous examples, we set both $w_{spatial}$ and w_{QB} to be 0.5 in this query.

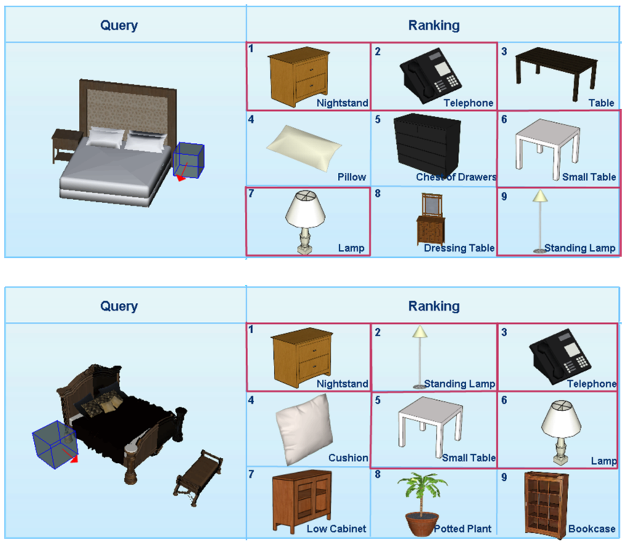


Figure 14: Query results for two different spatial relationships, where the ones marked in red are the common classes across two scenes.

As mentioned previously, our model retrieval algorithm is based on the concept of *SDF*, and to prove the correctness of our implementation and its relatively better efficiency, we compare our model retrieval approach with the *multiple view descriptor* approach Lian et al. [16] and *curvature histogram (shape 3D descriptor)* approach from Jeannin et al. [12] through *precision-recall curves*, as shown in Figure 15. There are totally 400 furniture models, belonging to 20 object classes, that have been tested during our model retrieval experiment. We have randomly selected 30, 60, 100 and 400 models from these 400 models to perform retrieval tests and therefore it is not necessary that all 20 object classes are selected for each time. Only the *top* 10 results are examined to determine the precision and recall ratios. To compare with the work from [16], we re-implement their algorithm, and to compare with the work from [12], we just adopt the existing source code, while both of them operate on the same datasets just described for being fair. As can be seen from the figure, our system offers better performance than the other two approaches and comparable performance with respect to the result shown in [6].

To show that angle plays an important role for retrieving objects, Figure 16 demonstrates a retrieval situation where a query box is placed besides a chair with its frontal direction given by the red arrow. As can be seen from the retrieval scores, there are at least two interesting things to observe. First, in the case where frontal labels are not marked, the *table* class ranks first while the *small table* class ranks third; however, when frontal labels are used, the *small table* class ranks first while the *table* class's rank becomes the second. This clearly shows how *frontal labels*, or equivalently, the *angular relationship* between two objects can influence the retrieval ranking result. Second, while the *lamp* class is not among the top 6 retrieved results if *frontal labels* are not considered, it becomes one of the top candidates when *angular relationship* counts, as it is quite often to put a lamp on top of a small table where a chair is placed aside.

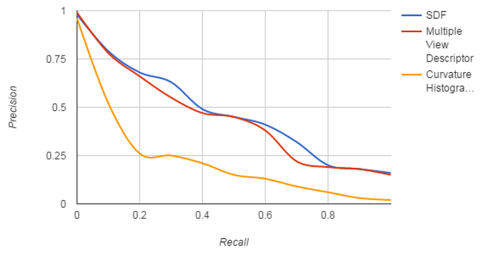
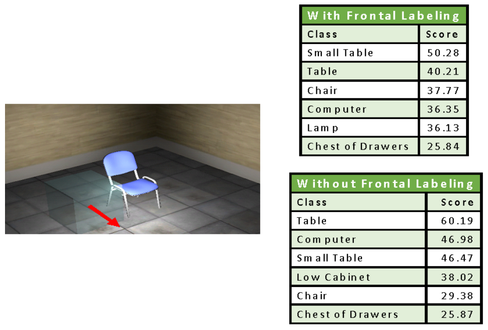


Figure 15: The precision recall results of the proposed system (SDF), marked in blue, the *multiple view descriptor*, marked in red, and the *curvature histogram*, marked in orange.



With Frontal Labeling	
Class	Score
Small Table	50.28
Table	40.21
Chair	37.77
Computer	36.35
Lamp	36.13
Chest of Drawers	25.84

Without Frontal Labeling	
Class	Score
Table	60.19
Computer	46.98
Small Table	46.47
Low Cabinet	38.02
Chair	29.38
Chest of Drawers	25.87

Figure 16: The corresponding retrieval scores for the retrieved objects with and without the frontal labels being marked in the database.

We also make use of this system to construct several simple scenes, as shown in Figure 17. More results can be seen from the video to be downloaded from http://140.118.9.45/ckyang/ContextBased.mpg.

To evaluate how users feel about our system, a user study has been conducted where 10 people are briefly instructed and asked to use our system to perform model retrievals and/or scene constructions. The result is shown in Figure 18. As demonstrated in this figure, most people feel satisfactory by the overall performance offered by our system. There is one exception though: one of the people does not think the functionality of *component replacement* is necessary.

7. CONCLUSIONS AND FUTURE WORK

We have proposed a powerful modeling tool that basically integrates three great modeling works [6, 2, 9]. Such a seamlessly merging makes our system not only enjoy almost all the merits from these works, but also render a more user-friendly and function-complete framework for end users.

However, there are still limitations in this work. First, when comparing object relationships between the current scene and a scene in the database, we do not take the difference in scenes into account. For example, a bedroom and a living room are in fact quite

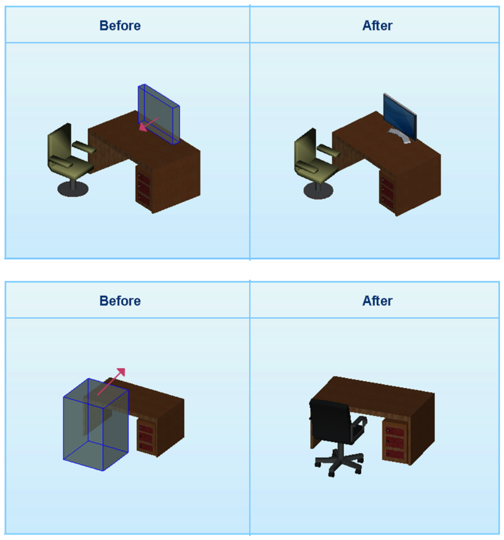


Figure 17: Some simple scenes constructed by this system.

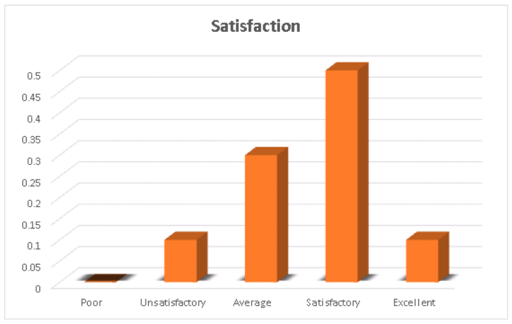


Figure 18: The result for a user study. The horizontal axis enumerates the five satisfactory categories while vertical axis represents the corresponding ratio for each category.

different, and placing them in one scene pool may lead to some erroneous object retrieval results. One solution is to classify scenes in advance, as proposed by Fisher et al. [7], so that the associated comparisons across scenes are more reasonable. Second, currently the labeling for the *frontal face* of an object is still manually done, and one interesting future direction is to automate this process by adopting a *principle component analysis*, or PCA for short, to find the desired frontal face, or resort to a similar technique proposed by Fu et al. [8] for finding the *upright orientation* of a model. Third, as the number of scenes and number of objects in a scene grow larger and larger, the total involved effort will also increase significantly. One idea is to resort to *machine learning* approaches: given a scene, a system could quickly and efficiently know what objects and where these objects should be placed in the scene. Finally, currently only spatial relationships such as distance, angle, and size are considered for retrieving a model, it would be interesting to add the concerns of color and texture so that more customized and stylish object arrangement can be achieved.

Acknowledgment

This work was supported in part by the National Science Council of Taiwan under the grant NSC 101-2221-E-011-150-MY3.

8. REFERENCES

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, Sep. 1975.

[2] S. Chaudhuri and V. Koltun. Data-driven suggestions for creativity support in 3d modeling. *ACM Trans. Graph.*, 29(6):183:1–183:10, Dec. 2010.

[3] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum (EUROGRAPHICS)*, 22(3):223–232, Sep. 2003.

[4] X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009.

[5] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa. Sketch-based shape retrieval. *ACM Trans. Graph.*, 31(4):31:1–31:10, Jul. 2012.

[6] M. Fisher and P. Hanrahan. Context-based search for 3d models. *ACM Trans. Graph.*, 29(6):182:1–182:10, Dec. 2010.

[7] M. Fisher, M. Savva, and P. Hanrahan. Characterizing structural relationships in scenes using graph kernels. *ACM Trans. Graph.*, 30(4):34:1–34:12, Jul. 2011.

[8] H. Fu, D. Cohen-Or, G. Dror, and A. Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3):42:1–42:7, Aug. 2008.

[9] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. *ACM Trans. Graph.*, 23(3):652–663, Aug. 2004.

[10] S.-S. Huang, A. Shamir, C.-H. Shen, H. Zhang, A. Sheffer, S.-M. Hu, and D. Cohen-Or. Qualitative organization of collections of shapes via quartet analysis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2013)*, 32(4):accepted, 2013.

[11] A. Jain, T. Thormählen, T. Ritschel, and H.-P. Seidel. Exploring shape variations by 3d-model decomposition and part-based recombination. *Comp. Graph. Forum*, 31(2pt3):631–640, May 2012.

[12] S. Jeannin, L. Cieplinski, J. R. Ohim, and M. Kim. Mpeg-7 visual part of experimentation model version 7.0. In *ISO/IEC JTC1/SC29/WG11/N3521*, Beijing, China, July 2000.

[13] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Trans. Graph.*, 31(4):55:1–55:11, July 2012.

[14] V. G. Kim, W. Li, N. J. Mitra, S. DiVerdi, and T. Funkhouser. Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11, Jul. 2012.

[15] C. Li and A. Ben Hamza. A multiresolution descriptor for deformable 3d shape retrieval. *Vis. Comput.*, 29(6-8):513–524, Jun. 2013.

[16] Z. Lian, A. Godil, and X. Sun. Visual similarity based 3d shape retrieval using bag-of-features. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI ’10, pages 25–36, Washington, DC, USA, 2010. IEEE Computer Society.

[17] J.-M. Lien and N. M. Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, SPM ’07, pages 121–131, New York, NY, USA, 2007. ACM.

[18] Y.-J. Liu, Y.-F. Zheng, L. Lv, Y.-M. Xuan, and X.-L. Fu. 3d model retrieval based on color + geometry signatures. *Vis. Comput.*, 28(1):75–86, Jan. 2012.

[19] W. Mohamed and A. B. Hamza. Reeb graph path dissimilarity for 3d object matching and retrieval. *Vis. Comput.*, 28(3):305–318, Mar. 2012.

[20] S. Shalom, L. Shapira, A. Shamir, and D. Cohen-Or. Part analogies in sets of objects. In *Proceedings of the 1st Eurographics conference on 3D Object Retrieval*, EG 3DOR’08, pages 33–40, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[21] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3d objects. *Int. J. Comput. Vision*, 89(2-3):309–326, Sep. 2010.

[22] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, Mar. 2008.

[23] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11, Nov. 2012.

[24] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39(3):441–471, Sep. 2008.

[25] K. Xu, K. Chen, H. Fu, W.-L. Sun, and S.-M. Hu. Sketch2scene: sketch-based co-retrieval and co-placement of 3d models. *ACM Trans. Graph.*, 32(4):123:1–123:15, Jul. 2013.

[26] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13(2):119–152, Oct. 1994.