Eurographics Workshop on 3D Object Retrieval (2017)
I. Pratikakis, F. Dupont, and M. Ovsjanikov (Editors)

Semantic correspondence across 3D models for example-based modeling

V. Léon[1], V. Itier[2], N. Bonneel[3], G. Lavoué[3] and J.-P. Vandeborre[4]

[1]Uraniom, Laval, France
[2]IMT Lille Douai, Univ. Lille, Département Informatique et Réseaux, F-59000 Lille, France
[3]Univ. Lyon, CNRS, LIRIS, France
[4]IMT Lille Douai, Univ. Lille, CNRS, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Abstract

Modeling 3D shapes is a specialized skill not affordable to most novice artists due to its complexity and tediousness. At the same time, databases of complex models ready for use are becoming widespread, and can help the modeling task in a process called example-based modeling. We introduce such an example-based mesh modeling approach which, contrary to prior work, allows for the replacement of any localized region of a mesh by a region of similar semantics (but different geometry) within a mesh database. For that, we introduce a selection tool in a space of semantic descriptors that co-selects areas of similar semantics within the database. Moreover, this tool can be used for part-based retrieval across the database. Then, we show how semantic information improves the assembly process. This allows for modeling complex meshes from a coarse geometry and a database of more detailed meshes, and makes modeling accessible to the novice user.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—Shape H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Search Process

1. Introduction

Modeling 3D shapes is a tedious task often performed by skilled artists. From the modeling of the global shape of a human to the sculpting of geometric details such as wrinkles or muscle bulges, technical skills, talent and time are necessary. These requirements make the task generally ill-suited to novice users.

Upon realizing that the space of possible shapes is often limited – for instance, a human is composed of a head, a torso, two arms and two legs – researchers have recently resorted to the use of mesh databases and example-based modeling. To date, this has remained limited to sketch-based interfaces to query entire meshes [XXM*13, FWX*13], or to rigidly replace segmented parts of an input mesh with other parts from the database [KJS07, CKGK11, CKGF13]. Even for experts, modeling often involves repetitive tasks that are more efficiently performed by re-using pieces of existing models in databases. These example-based approaches are thus expected to benefit the expert as much as the layman. Alternatively, an option is to resort to parameterized shapes (e.g., Poser [Smi15] for humans and animals) but limiting the diversity of the obtained results by a relatively small set of adjustable parameters.

We leverage recent advances in 3D mesh semantic description as well as the development of mesh databases to propose an example-based modeling framework suitable for the modeling of smooth shapes by novice users. To do so, we first propose an efficient selection tool that selects all regions similar in semantics to a brushed area within a mesh database. This makes use of a continuous semantic descriptor introduced by Léon et al. [LBLV16], that is, a high-dimensional descriptor able to discriminate between points sharing the same discrete label. Second, we show how semantic information also helps in the process of aligning and gluing shapes together. We demonstrate that our tool finally allows for the modeling of complex geometries by starting with a coarse 3D model and progressively refining it by replacing localized regions by regions of similar *semantic* (albeit geometrically different) within a database of more detailed meshes.

2. Related Work

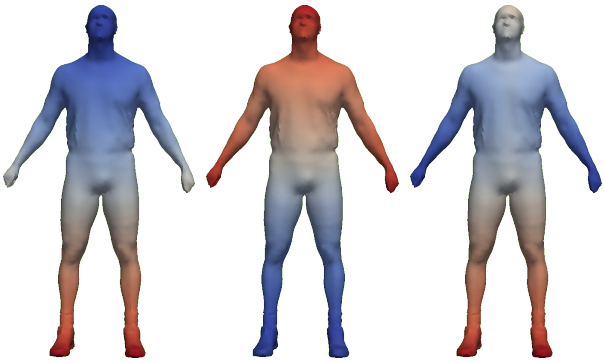
Example-based shape modeling is an important challenge in computer graphics. The objective is to ease the designers' work and speed up the production of 3D models. We can broadly classify existing techniques into: (1) manual editing tools and (2) part assembly based on consistent segmentation.

Manual editing. Example-based modeling was pioneered by

Funkhouser et al. [FKS*04]. In their approach, the user finds some parts of interest from an existing 3D shape database, cuts those desired parts using explicit scissoring, and glues them together to create a new shape. This approach introduces three main components: an interactive segmentation tool, a 3D local shape retrieval algorithm (to retrieve parts of interest) and a part composition tool. Several methods follow this framework, and enable a manual composition/modification of a 3D shape [SBSCO06, SS10, TSS*11].

Part assembly based on consistent segmentation. Techniques presented above have the drawback of requiring tedious manual mesh scissoring and composition. To fasten part retrieval and composition, most recent methods pre-process the database by a compatible segmentation and labeling. Such consistent semantic labeling of a whole 3D shape database may be obtained by data-driven techniques [KH10, vKTS*11], co-analysis [HKG11, SvKK*11, vKXZ*13] or by fitting a shape template [KLM*13]. After this offline pre-process, each shape of the database is cut into consistent meaningful parts, often associated to semantic labels (e.g, *harm*, *leg*, *torso* and *head* in the case of humanoid models). Such consistent semantic labeling is perfectly suited for modeling by assembling parts. Indeed, this allows to have a direct correspondence between the semantic parts of all objects of the database and thus allows to easily compose parts. Kreavoy et al. [KJS07] were the first to use consistent segmentation for model composition. With their system, the user just has to click on a part of a target model to exchange it with another part (with the same label) from a different model. This idea was extended by Chaudhuri and Koltun [CK10] who introduce an algorithm for automatic suggestions of novel components to augment a prototype shape. Chaudhuri et al. [CKGK11] improved this suggestion system by incorporating semantic relationships between parts, using a probabilistic model. Kalogerakis et al. [KCK12] considered a similar model for automatic component-assembly. Still based on a prior consistent shape labeling, Averkiou et al. [AKZM14] propose a low-dimensional parametrized space to explore easily the collection of 3D models and synthesize new arrangements of parts.

The main weakness of these techniques based on consistent segmentation/labeling is that their degree of freedom is limited by the pre-defined semantic domain. For instance it is impossible to glue an ear on the head of a 3D model, if these two components do not possess different labels in the database. On the other side of the spectrum, manual editing methods are purely geometric and thus lack semantic information that could help the editing process. In this context, from a manually selected region of interest, Kim et al. [KLM*12] proposed a method to compute fuzzy correspondences on meshes from a non-segmented database. This method relies only on corresponding geometric features across the database meshes. More recently, a continuous semantic representation has been introduced recently by Léon et al. [LBLV16]. Given a segmented and labeled 3D mesh, their descriptor consists of a set of geodesic distances to the different semantic labels. This local multidimensional signature effectively captures both the semantic information (and relationships between labels) and the underlying geometry and topology of the shape. Therefore, it provides more precise results than the Kim et al. [KLM*12] method. We rely on this representation to propose a new shape editing framework that bridges this gap between part assembly (which is too constrained by a prior segmentation) and manual editing (which requires tedious manual operations).



(a) Head (b) Foot (c) Hand

Figure 1: Illustration of the semantic descriptor of Léon et al. [LBLV16]. Here, a three-dimensional descriptor is considered, and each dimension represents the distance to points labeled as (a) *Head*, (b) *Foot* and (c) *Hand* respectively. Blue represents small distances while red represent larger distances.

3. Overview

Given a database of 3D shapes, our example-based modeling framework is illustrated in Figures 6 and 7. As offline preprocessing, all shapes are first segmented and semantically labeled, using one of the numerous existing mesh labeling methods [KH10, HKG11, SvKK*11, vKXZ*13]. We also compute, from this discrete labeling, the continuous semantic descriptor from Léon et al. [LBLV16] (see Section 4, Fig. 1). It computes, for each vertex, a multidimensional signature that describes its semantic context.

The user starts the modeling process with an arbitrary 3D shape from the dataset. Then, using a brush painting interface, the user can select an arbitrary region from this shape that needs to be replaced (see Fig. 7a). This input region is analyzed and approximated by an ellipsoid in the space of the semantic descriptor (Section 5). This fitted ellipsoid allows for a fast query of semantically similar target regions in the rest of the database (Fig. 7b). The user can thus replace the input region by any of the retrieved target regions. The geometric replacement is done by first aligning both regions (Section 6.1) and then smoothly merging the aligned mesh parts (Sec. 6.2 and Fig. 7c). Those two steps do not consider only geometric criteria but also rely on the semantic description to produce a realistic result.

4. Semantic description

Semantic labels are useful for defining the correspondence between segments, an operation commonly performed in example-

based modeling [KJS07, XXM*13]. However, defining a unique label for each segment restricts its use to correspondences between segments, and thus may only be useful to replace entire mesh segments. In practice, a user's selection will often lie in-between two segments or represent a small subset of a segment. The user could as well be interested in a different level of detail: for instance, a user may want to select a foot, when the segmentation and labeling has produced a unique segment for the entire leg. We thus resort to a continuous extension of this labeling process, akin to soft segmentation.

To obtain such a continuous semantic description, we use the multidimensional descriptor of Léon et al. [LBLV16]. As input, this descriptor takes a segmented and labeled mesh along with a random sampling of each segment. Then, each dimension of the descriptor corresponds to one label, and its value is computed as the geodesic distance to the closest sample bearing this label. To compute this descriptor, we use the exact geodesic distance of Surazhsky et al. [SSK*05], making it robust not only to rigid transformations, but also to changes in mesh resolution. Figure 1 illustrates different dimensions of the descriptor. This descriptor gives a continuous notion of semantics – for instance, determining that a point is within the *Head* region but close to the *Neck* – while also carrying geometric information.

5. Correspondence and selection

The first step is to determine regions within a mesh database that are similar to the user selection on the input mesh. This similarity measure is taken as a semantic similarity, that is, the semantic descriptors within these regions should be similar in term of Euclidean distance between descriptors. This process leads to a correspondence between an input region and a set of semantically similar regions in the mesh collection.

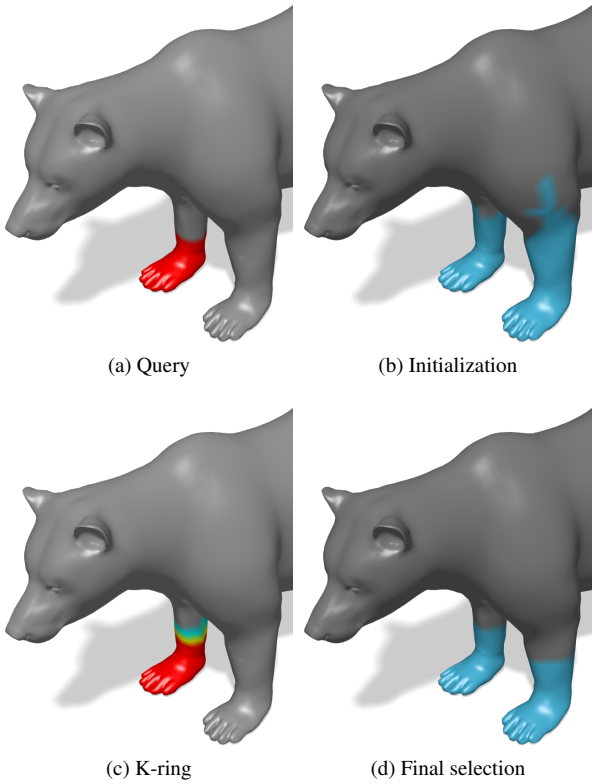
To avoid disconnected or highly distorted regions, we represent the user selection by a high-dimensional axis-aligned ellipsoid in the descriptor space. Each axis of the ellipsoid corresponds to a semantic label and the axis length models how that particular label varies within the selected area. The ellipsoid representation is compact, fast to compute and allows for the easy matching of selected areas. In fact, given an ellipsoid representing a selection on the input mesh, retrieving a similar selection on any other mesh merely amounts to selecting all vertices falling within the ellipsoid in the descriptor space. It is however important to note that these ellipsoid live in the high-dimensional space of the semantic descriptors (with as many dimensions as labels) and that user selections on the 3D mesh need not be elliptical.

To find the parameters of the ellipsoid representing a selected area, we first initialize it with a bounding hypersphere encompassing the descriptors of all vertices within the selected area (Sec. 5.1). We then refine it in an optimization step (Sec. 5.2).

5.1. Initialization

An axis-aligned ellipsoid in an n-dimensional space is defined by the following equation:

$$\sum_{i=0}^{n} \frac{(x_i - p_i)^2}{a_i^2} = 1 \qquad (1)$$



(a) Query (b) Initialization

(c) K-ring (d) Final selection

Figure 2: Overview of the selection and correspondence process. (a) Manual selection of a reference region on the paw. (b) Initialization with an hypersphere. (c) K-ring around the reference patch, for $k = 3$. (d) Final selection on the reference model with the ellipsoid. Vertices in blue are classified as part of the hypersphere or the ellipsoid. In (c), vertices in red are part of the original selection, and vertices in cyan and yellow are part of the 3-ring around it.

where $\{a_i\}_{i=1..n}$ are the parameters (axis) of the ellipsoid and $p = \{p_i\}_{i=1..n}$ its center.

Since we will rely on iterative optimization techniques, the initial values for the $\{a_i\}_{i=1..n}$ parameters have to be chosen carefully so that they are not too far from the expected final values. For the initialization, we use a bounding hypersphere of radius r in the descriptor space H, with r defined as follows:

$$r = \|\mathbf{d}(p) - \mathbf{d}(q)\|_2 \qquad (2)$$

where p is the center of the selected region in descriptor space, q is the selected vertex farthest from p in term of descriptors, and $\mathbf{d}$ is the semantic descriptor. Figure 2a illustrates this initialization. As we can see in figure 2b using this descriptor, the selection process accounts for symmetries.

5.2. Energy minimization

The ellipsoid parameters completely describe the user selection, thus they are used as a region descriptor for part-based retrieval. In order to compute the parameters of the ellipsoid that fits best the

user selection S, we minimize the energy:

$$e = \sum_{s \in \hat{S}} \mathbb{1}_S \left(\sqrt{\sum_{i=0}^{n} \frac{(d_i(p) - d_i(s))^2}{{a_i}^2}} - 1 \right)^2 \quad (3)$$

where $d_i(s)$ is the i^{th} dimension of the n-dimensional semantic descriptor computed for vertex s and the center p. $\hat{S}$ is the selected region S extended with a k-ring.

The energy term is computed for every vertex s inside the reference selection and inside the k-ring around this region, as shown on Figure 2c. The k-ring region is a set of negative examples which don't belong to the selected region. They are used to evaluate the parameters of the ellipsoid. The indicator function $\mathbb{1}_s$ evaluates the consistency between the geometric selection and the semantic ellipsoid. If a vertex is inside the selected area and its descriptor is inside the ellipsoid, then $\mathbb{1}_s = 1$ (we refer to them as positive examples). Likewise, if the vertex is outside of the selected area (but within a k-ring – typically $k = 3$) and its descriptor is outside of the ellipsoid, then $\mathbb{1}_s = 1$ (we call them negative examples). In every other case, $\mathbb{1}_s = 0$ and the energy contribution for this vertex is not computed (eq. 4).

$$\mathbb{1}_s = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} \frac{(d_i(p)-d_i(s))^2}{{a_i}^2} < 1 \text{ and } s \in S \\ 1 \text{ if } \sum_{i=0}^{n} \frac{(d_i(p)-d_i(s))^2}{{a_i}^2} \geq 1 \text{ and } s \in \hat{S} - S \\ 0 \text{ otherwise} \end{cases} \quad (4)$$

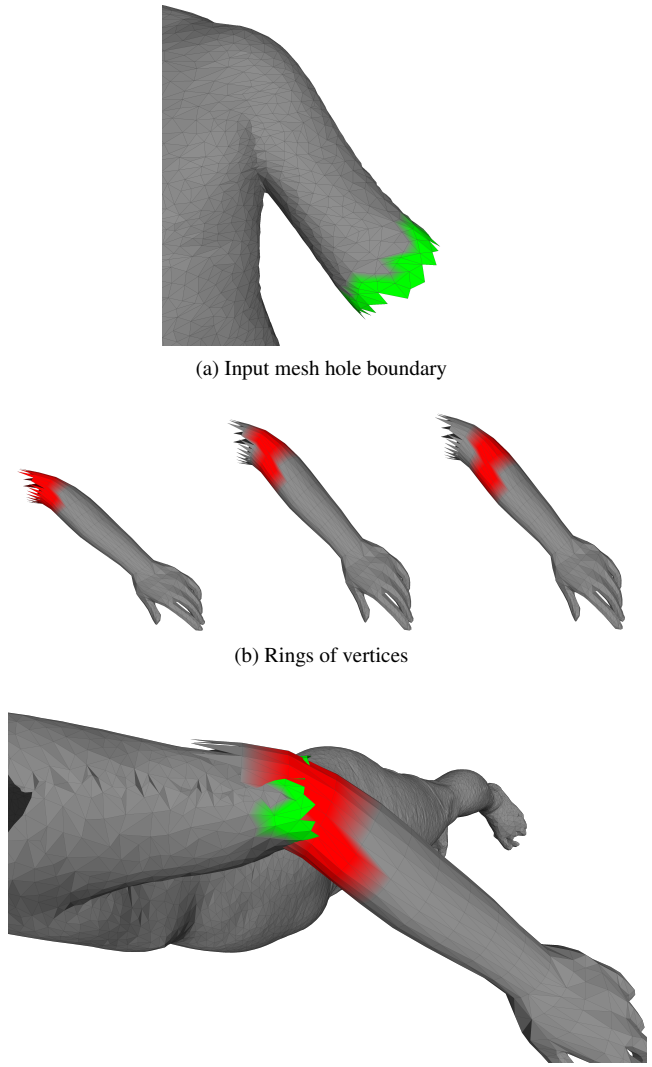
This method allows for a tight ellipsoid, and prevents the ellipsoid to enclose too many vertices that do not belong to the initial brushed area. Using the k-ring around the user selection brings a set of negative examples, that are not part of the selection, to evaluate the parameters of the ellipsoid. Figure 2c illustrate this k-ring for the previously considered selection.

The energy function minimization is performed using the COBYLA algorithm [Pow94] which supports derivative estimation, as the derivative of the energy does not have a simple closed-form expression due to the indicator function $\mathbb{1}_s$.

Given the optimized parameters of the ellipsoid, vertices are classified according to their descriptors. Figure 2d illustrates this classification. We notice on Figure 2d that the descriptor accounts for symmetry. Compared to the hypersphere initialization, shown in Figure 2b, the ellipsoid brings the selection closer to the reference query, shown in Figure 2a.

5.3. Query on the database

Once optimal parameters of the ellipsoid are computed for the user selection, we obtain similar regions among other meshes by determining, for each vertex of these meshes, whether its semantic descriptor lies within the ellipsoid in the descriptor space. The vertex is selected if and only if its descriptor $\mathbf{d}$ is such that $\sum_{i=0}^{n} \frac{(d_i - p_i)^2}{a_i^2} < 1$. The results illustrated in figure 2d, 4b, 6b, and 7b are obtained using this method.



(a) Input mesh hole boundary

(b) Rings of vertices

(c) Semantic alignment of the target region with input mesh boundary

Figure 3: Semantic alignment of the corresponding rings of vertices: a) mesh hole boundary, b) example of rings of vertices, c) the second ring is chosen as corresponding set of vertices and the target region is aligned with a rigid ICIP.

6. Automatic geometry transfer

Once the user has chosen a target region to replace the selected region from the input mesh, our algorithm performs a semantic-aware geometry transfer. First, we align both regions, and then, the target region is smoothly and realistically blended onto the input mesh.

6.1. Automatic alignment

Merging the target region to the input mesh first requires the to-be-exchanged regions to be aligned. Hence, the target region is first translated and scaled to fit the selected input region. After this coarse alignment, the selected region of the input mesh is removed.

Due to the semantic nature of our selection tool, this may result in several holes in the input mesh – for instance, if a user brushes a foot, both feet will be selected (Fig. 2).

Then, the target region is associated with its closest hole in the input mesh and an iterative closest point (ICP) algorithm [BM92] is processed to stitch both meshes. Standard ICP uses geometric features such as the euclidean distance, curvature, and angle difference between normals to put vertices in correspondence and rigidly reorient the target mesh region to the input mesh hole. However, using only geometric information to compute these correspondences may result in unnatural stitching (e.g., see the cat's head in Fig. 4c). To resolve this issue, we integrate the semantic descriptor in the ICP algorithm, by replacing the geometric correspondence by a *semantic* correspondence between vertices from the input mesh hole boundary and vertices from the target region. Our algorithm is the following, as illustrated in Figure 3: we defines several *rings* of vertices on the target region, starting from the boundary and moving away from it. We then select from these rings, the one that is the closest to the input mesh boundary with respect to its semantic descriptor. In practice, we define the semantic distance $sd(v,\mathcal{S}')$ between a vertex v from the input mesh boundary and a ring of vertices $\mathcal{S}'$ from the target region as:

$$sd(v,\mathcal{S}') = \min_{v'\in\mathcal{S}'} \|\mathbf{d}(v)-\mathbf{d}(v')\|_2. \tag{5}$$

with $\mathbf{d}$ is the semantic descriptor. The distance between the input mesh boundary $\mathcal{S}$ and the i^{th} ring $\mathcal{S}'_i$ is then given by:

$$d(\mathcal{S},\mathcal{S}'_i) = \sum_{j=0}^{j=n} sd(v_j,\mathcal{S}'_i), \tag{6}$$

The *corresponding* ring $\mathcal{S}'_c$ is then selected as $\text{argmin}(d(\mathcal{S}),\mathcal{S}'_i)$. We can then find a correspondence between each vertex of $\mathcal{S}'_c$ and $\mathcal{S}$, as:

$$v' = \underset{x\in\mathcal{S}'_c}{\text{argmin}}\, wd(x,v), \tag{7}$$

where $wd(x,v)$ is a weighted sum taking in account the semantic descriptor and the geometry.

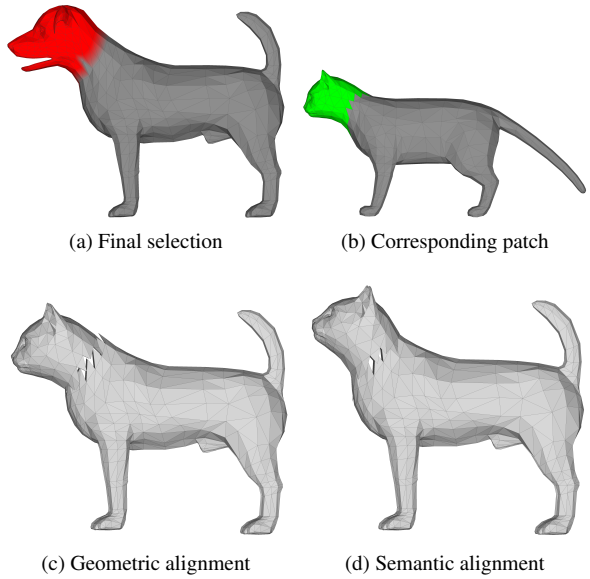
$$\begin{aligned} wd(x,v) = {} & \alpha_1\|v-x\|_2 + \alpha_2 \arccos(N_v - N_v) \\ & +\alpha_3(c_v - c_x) + \alpha_4\|\mathbf{d}(v)-\mathbf{d}(x)\|_2, \end{aligned} \tag{8}$$

where terms correspond respectively to the Euclidean distance between the vertices, the normal difference of the normals at the vertices, the difference in the local Gaussian curvatures at the vertices and the distance in the descriptor space at the vertices. We found the following weights to perform well in practice: $\alpha_1 = 0.2$, $\alpha_2 = 0.$, $\alpha_3 = 0.2$, $\alpha_4 = 0.4$.

This new semantic correspondence results in an improved ICP alignment, as compared to classical geometric ICP, as illustrated in Figure 4.

6.2. Mesh fusion

Now that the target region is well aligned with the input mesh, and that corresponding surfaces are overlapping around their opened border, it becomes possible to merge both meshes. For this step we consider the SnapPaste algorithm proposed by Sharf *et al.* [SBSCO06]. This algorithm is an iterative process that performs a *soft* ICP by calculating a partial transformation between the snapping regions of the two parts alternatively, until convergence. In our work, we propose to use the semantic descriptor to define the correspondence between vertices, similarly to the rigid ICP alignment performed in the previous step. We thus consider the vertex distance defined in Eq. 8. The method iteratively transforms the snapping regions to produce two overlayed meshes. Finally, we use a local re-meshing of the snapping regions in order to generate a single artifact-free mesh.
When the selection results in multiple connected components, both the alignment and mesh fusion are performed independently for each connected component.



(a) Final selection (b) Corresponding patch

(c) Geometric alignment (d) Semantic alignment

Figure 4: Alignment of the corresponding patch on the cropped mesh: a) semantic selection on a given mesh, b) corresponding patch on an other mesh, c) geometric alignment, d) semantic driven alignment.

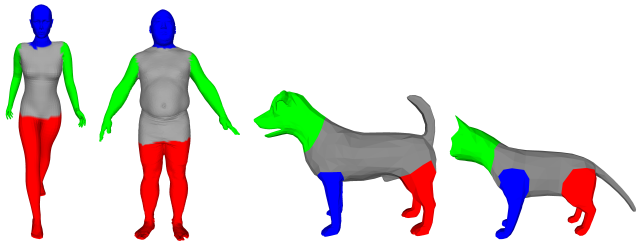
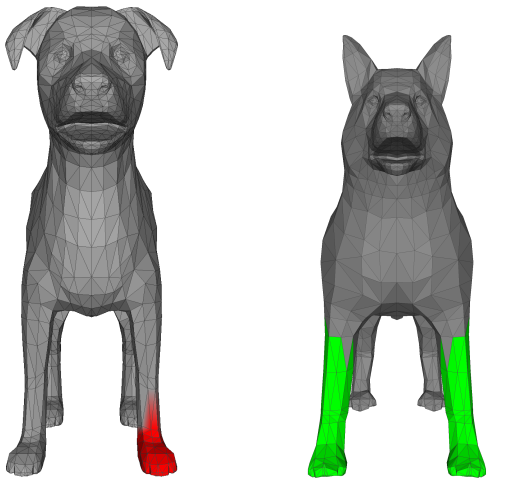
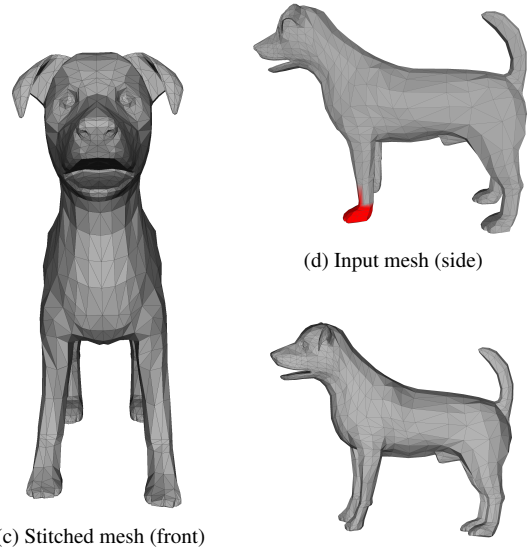


Figure 5: Database segmentation examples for bipeds and quadrupeds.



(a) Input mesh (front) (b) Corresponding patch



(c) Stitched mesh (front) (d) Input mesh (side) (e) Stitched mesh (side)

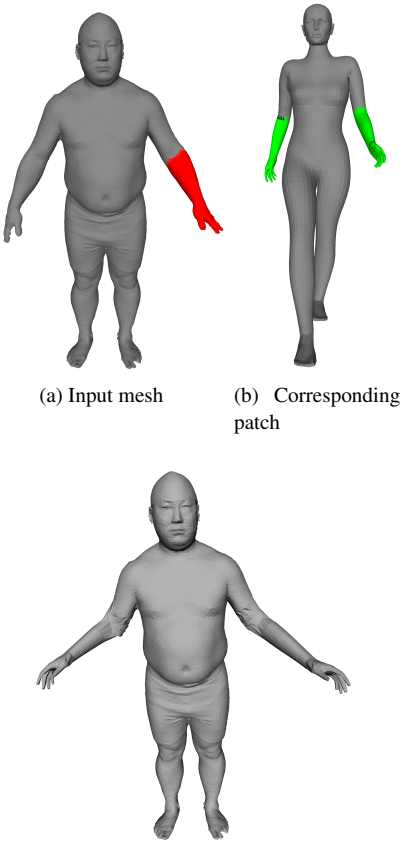
Figure 6: Geometry transfer example. a, d) semantic selection on a given mesh, b) corresponding patch on another mesh, c & e) resulting mesh after stitching (front and side views).

7. Experiments

Parameters from SnapPaste allow for additional control on the result. Specifically, the elasticity governs the rigidity of the transformation and the maximum number of iterations allows for a speed-quality trade-off. We also let the user fix an overlapping region size parameter which artificially increases the size of the ellipsoid on the target mesh. This parameter sometimes offers better alignment for snapping.

A result including multiple connected components due to the symmetry of labels can be seen in Fig. 6. Fig. 6a and 6d show the selected patch on the input mesh (front left leg, in red on the figure). Fig. 6b, shows the corresponding patch on another dog mesh. One can notice the symmetry in the semantic selection. Fig. 6c and 6e present geometry transfer results from the patch depicted in green on both front legs onto the input mesh. Due to the symmetry of the legs, both the left and right legs have been transfered at the same time.

Another example is provided in Fig. 7, which is more challenging because the junction between the parts to transfer is located on the elbow. Moreover, both meshes have different resolutions and poses. The definition of corresponding rings of vertices, described in Section 6.1, helps finding the correct alignment inside the elbow. We can see that, while the alignment is performed efficiently, the target mesh regions are correctly transfered. Nevertheless, the difference in quality between the two meshes (resolution, triangle size, *etc.*) produces some artifacts.



(a) Input mesh (b) Corresponding patch

(c) Result mesh front view

Figure 7: Geometry transfer example. a) semantic selection on a given mesh, b) corresponding patch on another mesh, c) resulting mesh after stitching.

Table 1 gives average times (in milliseconds) of the main steps of the proposed method, on a standard desktop computer without GPU optimization. Note that the ellipsoid selection, the correspondence and the semantic alignment are achieved in near real time. These three computation times only depend on the number of triangles

in the mesh, whereas the stitching time depends on the SnapPaste parameters and stopping conditions.

	Ellipsoid	Corresp.	Alignment	Stitching
Time	152.4	347.7	17.9	6.36×10^3

Table 1: Execution times in milliseconds

8. Perspectives and conclusion

We have extended example-based modeling approaches to allow for the replacement of more arbitrary mesh regions that extend over multiple segments, or that are contained within a unique segment. We believe this is an important step towards efficient artistic tools usable by hobbyists, as it allows for greater expressiveness. To achieve that, we have introduced a semantically based selection tool that makes use of a continuous semantic descriptor and ellipsoids. This tool retrieves all semantically similar mesh patches, and has potential for wider applications, from object retrieval to non-local mesh filtering operations. We have demonstrated that semantic information can be further used to improve the alignment and stitching of meshes, by adapting a (soft-)ICP procedure to handle semantics. Our approach runs in real time, and our results show that it is effective in creating new meshes by combining existing mesh pieces from a database.

Acknowledgement

This work has been partially supported by the ANR (Agence Nationale de la Recherche, France) through the CrABEx project (ANR-13-CORD-0013), and also partially supported by the region "Hauts de France" and the European Union with the European Regional Development Fund, in the frame of the MATRICE Project.

References

[AKZM14] AVERKIOU M., KIM V. G., ZHENG Y., MITRA N. J.: ShapeSynth: Parameterizing model collections for coupled shape exploration and synthesis. *Computer Graphics Forum 33*, 2 (May 2014), 125–134. 2

[BM92] BESL P.-J., MCKAY N.-D.: A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256. 5

[CK10] CHAUDHURI S., KOLTUN V.: Data-driven suggestions for creativity support in 3d modeling. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 183:1–183:10. 2

[CKGF13] CHAUDHURI S., KALOGERAKIS E., GIGUERE S., FUNKHOUSER T.: Attribit: Content creation with semantic attributes. In *ACM Symposium on User Interface Software and Technology* (2013), pp. 193–202. 1

[CKGK11] CHAUDHURI S., KALOGERAKIS E., GUIBAS L., KOLTUN V.: Probabilistic reasoning for assembly-based 3d modeling. *ACM Transactions on Graphics 30*, 4 (July 2011), 35:1–35:10. 1, 2

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by Example. *ACM Siggraph* (2004). 2

[FWX*13] FAN L., WANG R., XU L., DENG J., LIU L.: Modeling by drawing with shadow guidance. *Computer Graphics Forum 32*, 7 (2013), 157–166. 1

[HKG11] HUANG Q., KOLTUN V., GUIBAS L.: Joint shape segmentation with linear programming. *ACM Transactions on Graphics 30*, 6 (2011). 2

[KCK12] KALOGERAKIS E., CHAUDHURI S., KOLLER D.: A Probabilistic Model for Component-Based Shape Synthesis. *ACM Transactions on Graphics* (2012). 2

[KH10] KALOGERAKIS E., HERTZMANN A.: Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics 29*, 4 (2010). 2

[KJS07] KREAVOY V., JULIUS D., SHEFFER A.: Model composition from interchangeable components. *Computer Graphics and Applications* (2007), 129 – 138. 1, 2, 3

[KLM*12] KIM V. G., LI W., MITRA N. J., DIVERDI S., FUNKHOUSER T.: Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph. 31*, 4 (July 2012), 54:1–54:11. 2

[KLM*13] KIM V. G., LI W., MITRA N. J., CHAUDHURI S., DIVERDI S., FUNKHOUSER T.: Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics 32*, 4 (July 2013). 2

[LBLV16] LÉON V., BONNEEL N., LAVOUÉ G., VANDEBORRE J.-P.: Continuous semantic description of 3d meshes. *Computer & Graphics 54*, C (Feb. 2016), 47–56. 1, 2, 3

[Pow94] POWELL M. J. D.: *Advances in Optimization and Numerical Analysis*. Springer Netherlands, Dordrecht, 1994, ch. A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation, pp. 51–67. 4

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snappaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9-11 (2006), 835–844. 2, 5

[Smi15] SMITHMICRO: Poser Pro 11. http://my.smithmicro.com/poser-3d-animation-software.html, 2015. Online; accessed Feb. 2017. 1

[SS10] SCHMIDT R., SINGH K.: Meshmixer: an interface for rapid mesh composition. In *SIGGRAPH Talks* (2010). 2

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.: Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics 24*, 3 (July 2005), 553–560. 3

[SvKK*11] SIDI O., VAN KAICK O., KLEIMAN Y., ZHANG H., COHEN-OR D.: Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Transactions on Graphics 30*, 6 (2011). 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: GeoBrush: Interactive Mesh Geometry Cloning. In *Computer Graphics Forum* (2011), vol. 30, pp. 613–622. 2

[vKTS*11] VAN KAICK O., TAGLIASACCHI A., SIDI O., ZHANG H., COHEN-OR D., WOLF L., HAMARNEH G.: Prior Knowledge for Part Correspondence. *Computer Graphics Forum 30*, 2 (2011). 2

[vKXZ*13] VAN KAICK O., XU K., ZHANG H., WANG Y., SUN S., SHAMIR A.: Co-Hierarchical Analysis of Shape Structures. *ACM Transactions on Graphics 32*, 4 (2013). 2

[XXM*13] XIE X., XU K., MITRA N. J., COHEN-OR D., GONG W., SU Q., CHEN B.: Sketch-to-design: Context-based part assembly. *Computer Graphics Forum 32*, 8 (2013), 233–245. 1, 3