EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

SimSelect: Similarity-based selection for 3D surfaces

Emilie Guy Jean-Marc Thiery Tamy Boubekeur

Telecom ParisTech
CNRS LTCI
Institut Mines Telecom

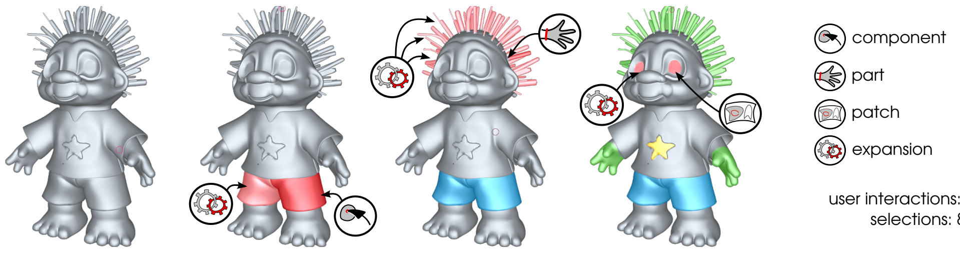


Figure 1: *Starting from the input mesh (left), the user clicks on connected components, cuts to define parts and brushes over patches. The expansion tool enriches the current selection by detecting similar ones, and greatly reduces the task repetitiveness.*

Abstract

Surface selection is one of the fundamental interactions in shape modeling. In the case of complex models, this task is often tedious for at least two reasons: firstly the local geometry of a given region may be hard to manually select and needs great accuracy; secondly the selection process may have to be repeated a large number of times for similar regions requiring similar subsequent editing. We propose SimSelect, a new system for interactive selection on 3D surfaces addressing these two issues. We cope with the accuracy issue by classifying selections in different types, namely components, parts and patches for which we independently optimize the selection process. Second, we address the repetitiveness issue by introducing an expansion process based on shape recognition which automatically retrieves potential selections similar to the user-defined one. As a result, our system provides the user with a compact set of simple interaction primitives providing a smooth select-and-edit workflow.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Shape.

1. Introduction

Current interactive modeling software allow users to create, capture and modify complex 3D meshes, with high resolution geometric details, for the many applications of computer graphics. In particular, high-end 3D packages – such as Maya, 3DSMax or ZBrush – are widely used for post-processing acquired meshes or creating and deforming new ones. Within such packages, many digital surface tools, such as *detail control* (e.g., normal mapping, procedural displacement), *repairing* (e.g., hole filling, cleaning), *uv-control* (e.g., constraints positioning, alignment), *geometric signal modulation* (e.g., smoothing, sharpening) or *copy-pasting* (e.g, region cloning, component merging) may either be globally applied or, more often, used on selected portions of the input.

While the automation of these tools has motivated a large portion of interactive geometry research over the last few years, the computational aspect of the selection process itself has emerged only recently as a key topic. This user-driven process is indeed one of the basic, fundamental operations in interactive 3D modeling, but remains a tedious and time-consuming task for any CG user, from casual graphics enthusiasts to highly skilled SFX designers. More specifically, there are at least two main challenges when interactively defining a selection on a surface: the *accuracy* required to hand-track the boundary of a particular region and the *repetitiveness* one endures when it comes to the selection of multiple similar regions on the surface, both problems becoming critical for captured geometry missing the high-level structure of e.g., CAD models.

For 2D image editing, the accuracy issue has been addressed by introducing new interaction metaphors, less demanding for the user and requiring only few approximate gestures (i.e., strokes) to control an automatic content-aware selection process running on-the-fly. At any time, the so-defined selection can be refined *progressively* – see for instance the *Paint Selection* system [LSS09]. These tools are usually based on heuristics which assume, for instance, that the selection should be bounded by a coherent high magnitude gradient structure or that the selected region should exhibit particular statistics in its texture. They often exploit popular combinatorial or variational methods for the optimization step such as the graph cuts or the k-means algorithms. The repetitiveness issue has been addressed only recently in particular for image collection editing where an automatic transfer of edits can be performed using non-rigid alignment [YJHS12].

When moving to 3D geometry, the problem becomes harder: firstly, depending on the scale, the notion of "coherent" selection may drastically change, from isolated components (e.g., a ring on a finger) which require that the surface topology be taken into account, to local bumpy patterns (e.g., a stamp on wax) which involve an analysis of the geometric signal. Secondly, as opposed to images, which decorrelate signal (color) from parameterization (pixels coordinates), 3D surfaces embed both notions in their positional field. Although intuitive semi-automatic methods have been introduced to separate a shape into several components (e.g., arms and legs from a body), we notice in practice that most scenarios still require the user to spend a significant amount of time selecting each and every region she wants to edit, ultimately selecting all polygons to be processed with a brush, a lasso [SS10] or simple combinatorial criteria (e.g., polygon strip selection [Aut]). However, we empirically observed that CG users often interact with a shape at three different levels by editing either (i) a single piece in a large multi-components model, (ii) a large part where only the low frequency structures of the shape matter or (iii) a local region presenting a strong on-surface structure. We base our approach on such a classification.

Contribution: We propose *SimSelect*, a new system for interactive selection on 3D surfaces which addresses the aforementioned issues. Our basic idea is to classify selections into three different types – *connected components*, *parts* and *patches* – providing the user with a specific interaction metaphor and a specific automatic selection optimization for each of them (Section 4). Beyond this type-aware process which improves the accuracy of the selection, we also introduce an *expansion* procedure (Section 5), which retrieves potential selections which are similar to the current one all over the shape. This process enables the simultaneous editing of multiple regions on the surface, thus reducing the repetitiveness of selection (see Fig. 1).

2. Previous Work

Interactive mesh segmentation: With classical interactive surface segmentation systems, the user typically draws rough strokes over the surface to initialize an automatic segmentation process. Based on the mandatory interaction, we can classify these techniques into two main families: *boundary-based* and *region-based* methods. Boundary-based techniques ask the user to draw a coarse boundary of the selection by either tracing strokes along the desired boundary [MFL11, LLS*05, FKS*04] or across it [ZT10]. These strokes induce a small set of inside/outside constraints which are combined with a regularization prior – typically based on the surface curvature – to initialize an optimization procedure. In the optimization, all these relations can, for instance, be expressed as a linear system to solve or as weights on the edges of a graph to cut. Region-based techniques mainly differ in the user input, which is instead defined by brushing directly on the selection [FLL11], or by specifying explicitly inside/outside areas [JLCW06].

Similarity detection: Our *expansion* process requires a quick analysis of the input shape to detect potential similarities between the user's selection and other areas on the surface. This topic has been widely studied in several fields of shape analysis, including symmetry detection (see the survey by Mitra et al. [MPWC12]) and model-driven shape retrieval (see the survey by Tangelder et al. [TV08]). We refer the reader to these surveys for a complete overview, and we focus here on recent systems computing a global surface alteration from a local editing. In the context of mesh colorization, Leifman and Tal [LT12, LT13] proposed an approach where the user defines a small set of colored "scribbles" on the surface before running a colorization process which will assign a color to every polygon based on a similarity measure with the scribbled regions.

In the context of mesh processing, Maximo et al. proposed a system called SAMPLE [MPVF11] where, given a reference vertex and a processing operator to apply on it, a local heightmap descriptor is used to find the most similar vertices to the reference and apply the operator on them as well.

Beyond particular applications, the performance of such systems are linked to their underlying *descriptor*, which captures the essence of a shape or a region, and comes with a distance which models the notion of similarity. Ideally, this descriptor should be compact, fast to extract, translation/rotation-invariant, robust to small scale variations (e.g., noise) and to partial matching. This list of desirable properties often lead to statistical models (e.g., multi-dimensional histograms) which offer a good compromise between accuracy and scalability.

3. Overview

The SimSelect system takes as an input a 2 manifold triangle mesh (i.e, a list of polygons indexed over a list of vertices), which can have multiple components and boundaries, and provides the user with an interactive selection process which is composed of two main stages: the interactive construction of a reference selection and its expansion to similar ones.

During the first step, the user defines the reference selection by choosing among three different interactions: (i) *clicking* on a connected component, (ii) *cutting* (using multiple strokes) at the boundaries of a part or (iii) *brushing* (using multiple scribbles) over a patch. Depending on the interaction, we run a specific algorithm (see Section 4): (i) a flood-filling to find a connected component, (ii) the election of the best isoline within a harmonic field to cut a part or (iii) a normal-driven region growing to capture a patch.

In the second step, the user can enrich the reference selection by adding k similar regions which are automatically detected (see Section 5). This *expansion* process is based on a similarity measure to search and sort potential new selections, and exploits compact statistical shape descriptors, the conformal factor of the surface or machine learning depending on selection.

4. Reference Selection

The reference selection is a subset of the input polygons. Its interactive construction involves three different interaction metaphors: *clicking* components, *cutting* parts and *brushing* patches. In the following, we describe the algorithm we run for each case to define the subset of polygons belonging to the reference. All these algorithms are fast enough on dense polygon meshes to be run on-the-fly during the interaction.

4.1. Connected component selection

At loading time, we build the adjacency graph of the mesh based on the polygons edges. During interaction, when the user clicks on a polygon, we gather all faces belonging to the component and add them to the reference selection. Although trivial, this selection type turned to be quite important for meshes composed of numerous components in our early experiments.

4.2. Part selection

The part selection algorithm of SimSelect is largely inspired from the Cross-Boundary Brush [ZT10]. Isolines of a harmonic field, guided and progressively refined by user strokes, are used as cutting boundaries for the selection of a part. The field is obtained as the solution of the following linear system:

$$\left[\begin{array}{c} L \\ \hline W_0 P_0 \\ W_1 P_1 \end{array}\right] \Phi = \left[\begin{array}{c} 0 \\ \hline W_0 B_0 \\ W_1 B_1 \end{array}\right],$$

where L is the mesh Laplacian (cotan scheme [PJP93]), W_0 and W_1 are positional weighting matrices, P_0 and P_1 are positional constraints matrices, $B_1 = (1..1)^T$ and $B_0 = (0..0)^T$.

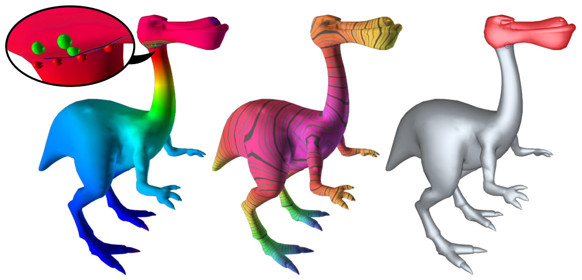
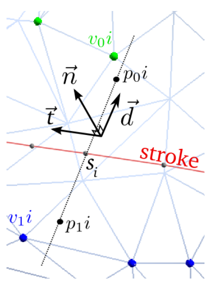


Figure 2: ***Part Selection:*** *the user stroke induces a set of constraints (close up) and a harmonic field (left); the selection boundary is defined as the best harmonic field isoline, while the conformal factor (middle) helps to decide which side of the cut the selection is on (right).*

We choose the best isoline of the harmonic field (as in [ZT10]) to be the part cutting boundary.

The system is factorized once when the mesh is loaded, then we update it only from the few dynamic constraints stemming from the user strokes during interaction using the factorization downdating and updating techniques [XZCOX09], which allows us to solve it interactively. We handle meshes with multiple connected components by setting 0-constraints for all the components which are not touched by the strokes. On the contrary to Zheng and Tai [ZT10], who request the user to draw strokes *across* the desired cut, we argue that locating them *along* the cut is a more natural metaphor because it mimics the "slicing" of the shape better. To do so, we generate a set of inside/outside constraints (i.e., setting the lower part of the system) for few vertices based on the strokes. First, for each stroke point $\mathbf{s}_i$, we construct a right handed frame based on the normal $\mathbf{n}$ at $\mathbf{s}_i$ and on the stroke direction vector $\mathbf{t}$. Following the third vector of the frame $\mathbf{d}$, we define $\mathbf{p}_0^i = \mathbf{s}_i + \varepsilon\mathbf{d}$ (resp. $\mathbf{p}_1^i = \mathbf{s}_i - \varepsilon\mathbf{d}$)[†]. Then, we set a constraint in the system for $\mathbf{v}_0^i$ (resp. $\mathbf{v}_1^i$), the closest vertex to $\mathbf{p}_0^i$ (resp. $\mathbf{p}_1^i$); see the embedded figure for an illustration of this constraint positioning technique. Last, to define on which side we set positive constraints, we assume that the user is more likely to select a protuberant part than the rest of the shape and use the conformal factor (see Section 5.2) to decide. As a result, the constraints distribution remains both independent of the stroke direction and consistent along different strokes (see Fig. 2).



† $\varepsilon = 10^{-2}\%$ of the bounding box in our experiments

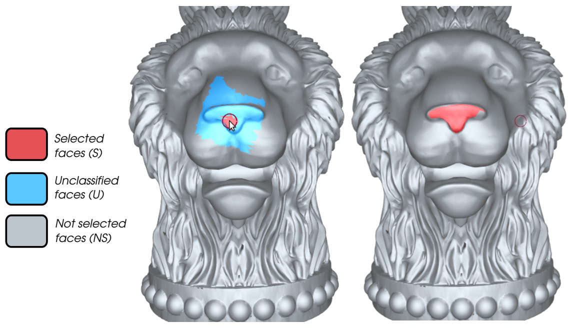


Figure 3: ***Patch algorithm:*** *given unclassified polygons (left) the final patch selection is delimited using a geometry-aware flood-filling (right).*

4.3. Patch selection

Inspired by 2D systems [Pho, LSS09], we let the user define patches by brushing them, while optimizing on-the-fly the selection boundary (see Fig. 3). More precisely, while brushing on the surface, every vertex touched by the brush is marked as *selected* (S). Starting from this set, we flood-fill the surface and mark every encountered vertex as *unclassified* (U) until reaching vertices located above a threshold distance f_c from the centroid of S that we mark as *non-selected* (NS). Last, we perform two competing flood-fillings on U, one starting from S, the other from NS. They compete based on a cost function to gather the elements of U: when they terminate, the patch selection is S. In practice, we found that using the (normal-based) $L^{2,1}$ [CSAD04] metric as a cost function gives the best results.

5. Expansion

Beyond the component, part and patch selection schemes, we propose an *expansion* process which gathers additional similar regions over the surface to enrich the reference selection. In this second stage, the user can navigate in the potential selections space by simply sliding a value k to augment the current selection with the k most-similar regions. This expansion functionality, which significantly diminishes the repetitiveness issue, can be expressed as a local shape matching problem and we use specific descriptor classes combined with fast retrieval algorithms, adapted to each selection type, to detect candidate regions efficiently.

5.1. Connected component expansion

If the reference selection is an entire connected component, the expansion process may propose other connected components to the user. To sort them and select the k most similar ones, we describe each of them using an approximation of the Shape Context descriptor [MBM05] or ASC. An ASC is a 2D histogram measuring the distribution of normalized distances and normal deviations w.r.t. to the component center (see Alg. 1). To ensure robustness against sampling variations, we start by computing the center **c** and a normalizing factor r_{max} using the MiniBall algorithm [G$\ddot{9}$9]. Then, we fill the histogram by accumulating, for each vertex **v** with normal **n**, the area of the dual face of **v** in $ASC[d,\alpha]$, with d the normalized distance between **v** and **c** and α the angle formed by **n** and $\vec{\mathbf{cv}}$. The ASC is rotation- and translation-invariant, has a linear computational complexity and, thanks to its typically low number of bins, is robust to small scale variations. In practice, at loading time, we compute an ASC for each component. Then, during an interactive selection expansion, we measure the similarity between the reference component and all the others using the L^2 distance between their ASC (see Fig. 4) and return the k closest.

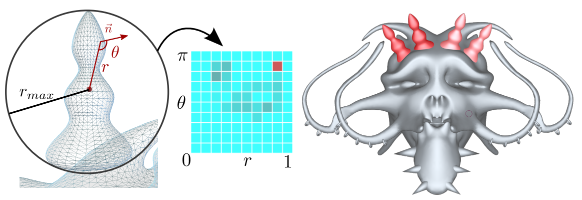


Figure 4: ***Connected component expansion:*** *using the center and radius of the miniball (left), we compute the ASC descriptor (2D histogram in blue) and find the most similar components (right)*

Algorithm 1 ASC Computation

```
for each connected component of the mesh do
    initialize its descriptor: $ASC$ ($10 * 10$ bins)
    compute its miniball (center: $c$, maximum radius: $r_{max}$)
    for all $v$ of the connected component do
        $\mathbf{d} = \frac{\|v-c\|}{r_{max}}$
        $\alpha = acos(< \mathbf{v}-\mathbf{c}, \mathbf{n} >)$
        $ASC[\mathbf{d},\alpha] += \sum_{t_i \in T_1(v)} \frac{area(t_i)}{3}$
    end for
end for
```

5.2. Part expansion

Finding regions similar to a reference part selection is more challenging: since the input shape does not come with a segmentation, we have to compute it before measuring similarity between the resulting regions. We define a pool of candidate part regions to compare with using the shape *conformal factor*.

The *uniformization theorem* demonstrates that any 2-manifold can be conformally mapped to a surface with the same topology having a constant Gaussian curvature. The conformal factor [BCG08] is a scalar function on the surface which describes this mapping and which is invariant to isometric transformations. We compute it once when

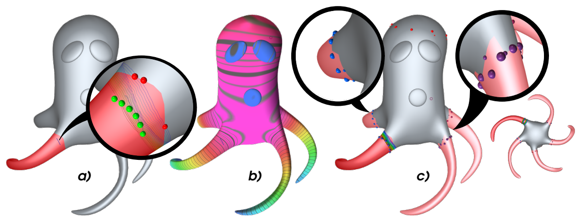


Figure 5: ***Similar part detection:*** *a) the reference part selection (close up: user constraints and user field) b) similarity map: conformal factor c) detected similar regions*

loading the mesh, by solving the following linear system: $L\Phi = K^T - K^{origin}$, where K^{origin} is the mesh Gaussian curvature and K^T is the average Gaussian curvature. Intuitively, the conformal factor represents the local stretch required to transform the model into a surface with constant Gaussian curvature. For instance, protuberant parts such as arms, legs or fingers have a large conformal factor value. We make the assumption that the user traces cuts roughly along isolines of the conformal factor. This allows us to populate our pool of potential part regions as follow: given the reference part selection, we compute the average conformal factor of its related stroke points, m_s and find all the isolines with the same conformal value on the shape. These isolines are similar to boundaries of potential similar selections. We use them as virtual strokes, repeat the part selection process (see Fig. 5) and add the resulting potential part region to our pool of candidates.

With this pool in hand, we can now search for similarity to the reference by computing a descriptor for each of its elements and returning the k closest parts. Rather than using the ASC descriptor, we exploit the previously computed conformal factor and use a 1D histogram representing the conformal factor distribution of each candidate part region. To sort them w.r.t. the reference, we use the χ^2 distance between their histograms (see Alg. 2). One strength of this algorithm is that it uses the locality of the part selection and the globality of the similarity map to find the best similar selections.

Algorithm 2 Part expansion

```
hypothesis: user strokes are roughly along a conformal isoline.
1) compute the average of the user strokes conformal factor : $m_s$
2) compute the descriptor of the reference selection $d_{ref}$
3) find all the isolines with the same conformal factor value $m_s$
for each isoline $i$ do
    set $l$ harmonic constraints on each side of the isoline
    find the potential part region $i$
    compute its descriptor $d_i$
end for
sort the potential part regions by their distance to $d_{ref}$
select the $k$ first part regions
```

5.3. Patch expansion

As in the case of parts, we need to generate a pool of candidate patch regions to expand a reference patch selection. We propose to compute a similarity map to find centers of potential similar patches (see Fig. 6b). We treat each vertex of the shape as the potential approximate center of a candidate patch selection. The similarity map captures the distance between the descriptor of the reference and the descriptors of all these potential patch regions (see Alg. 3). In the case of patches, we need to define a descriptor which is discriminative enough on the small scale high frequency signal which makes patches singular structures of interest on a surface (e.g., the relief of an ear).

In the following, S_i is the set of all the vertices inside the sphere of center $\mathbf{v}_i$ and radius r (the reference patch bounding sphere radius). We define a patch-based variant of the shape context (PSC) $PSC_{i,r}$ as a 2D histogram which first dimension captures the normalized distance d between a vertex $\mathbf{v}$ of S_i and its actual center $\mathbf{c}$, and the other dimension captures the angle θ between the normal of $\mathbf{v}$ and the normal of the center $\mathbf{n}_c$. Once the patch similarity map is computed,

Algorithm 3 Patch similarity map

```
find the patch center $c$ and the patch radius $r$
compute the reference descriptor $PSC_{ref,r}$
for every vertex $i$ of the mesh do
    find all vertices in $S_i$
    for every $v_j$ in $S_i$ do
        $d = \frac{\|\mathbf{v_j} - \mathbf{c}\|}{r}$
        $\alpha = acos(< \mathbf{n_v}, \mathbf{n_c} >)$
        $PSC_{i,r}(d,\alpha) += \sum_{t_k \in T_1(v_j)} \frac{area(t_k)}{3}$
    end for
    $similarityMap_i = \|PSC_{i,r} - PSC_{ref,r}\|$
end for
```

we know where the potential patch regions are located on the surface (i.e., the minimum of the map) but we still need to compute their exact extent. To do so, we *learn* a model of the reference patch selection w.r.t. its neighborhood using a support vector machine (SVM) to classify all faces in its vicinity.

Patch learning: An SVM is a popular binary supervised machine learning technique which projects data onto a *feature space* where its separability is easier to achieve i.e., classifying vertices as selected or not, in our case. We use an SVM with a radial basis function (RBF) kernel $\exp(-\gamma\|a - b\|^2)$ and a 3-dimensional feature space. The γ parameter of the RBF represents how much the RBF fits the training set (i.e., the reference selection). To have a reasonable value of γ for all selections, we set it to the median of the median distances between points and their closest neighbors (computed in the feature space). Ideally, going in the direction of example-based selection from large data sets, computing cross validation process for each selection could help to find

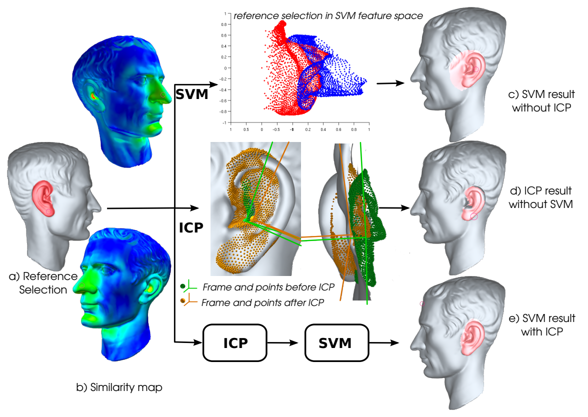

reference selection in SVM feature space
SVM
ICP
ICP
SVM
c) SVM result without ICP
d) ICP result without SVM
e) SVM result with ICP
a) Reference Selection
b) Similarity map
Frame and points before ICP
Frame and points after ICP

Figure 6: ***Patch expansion:*** *as shown on the right, both the SVM classification and the ICP registration are instrumental and complementary when expanding patch selections.*

the best parameters. For now, as we want our approach to work for all models and all selections, this is left as future work.

Feature space: We observe experimentally that patch regions typically represent on-surface structures where the geometry is mostly altered in the normal direction (i.e., displacement). Therefore, we express the elements of S in a more suitable feature coordinate system, reflecting this characteristic in the learning/classification stages. For each region (reference patch selection and candidate patch regions) we define a local frame $F_i = (c, \mathbf{u}, \mathbf{w}, \mathbf{n})$ where $\mathbf{n}$ is the average normal of the patch; $\mathbf{u}$ and $\mathbf{w}$ are the principal directions of curvature (see below). We use F_i to compute the feature coordinates (d, h, α) of every vertex $\mathbf{v}_j$ in S_i, with d the (rescaled) length of $\overrightarrow{\mathbf{cv}}$, h the length of the projection of $\overrightarrow{\mathbf{cv}}$ on $\mathbf{n}$ and α the angle between the projection of $\overrightarrow{\mathbf{cv}}$ in the $\{\mathbf{u}, \mathbf{w}\}$ plane and $\mathbf{u}$. The SVM is trained with all the vertices of S_{ref} and used to classify all the vertices of all S_i.

Registration: To guarantee that the whole approach is rotation invariant and to properly reproduce the boundary of the patch in the expansion, the local orientation (i.e., $\mathbf{u}$) of F and its center c are critical. This is indeed a *registration* problem, where we seek a (rigid) transformation from F_{ref} to F_i which aligns their neighborhoods as well as possible. This problem, classical in reconstruction from scans, can be efficiently addressed in two steps: firstly we estimate a global transformation from F_{ref} to F_i and secondly we perform a local adjustment.

For the global registration, we initialize F with a (normalized) principal component analysis performed on the *normal field* of S. With similar geometric structures in S_{ref} and S_i (as detected by our similarity map) the direction of $\mathbf{u_{ref}}$ and $\mathbf{u_i}$ is usually consistent. However, we also need to define the orientation of $\mathbf{u}$ in a consistent manner across both frames. To do so, we use differences between their first and second moments which are computed as follows: $m^1 = \sum_{v \in S_i} area_v h_v < \mathbf{v} - \mathbf{c}, \mathbf{u} >$ and $m^2 = \sum_{v \in S_i} area_v h_v < \mathbf{v} - \mathbf{c}, \mathbf{u} >^2$. If $\|(m^1_{loc}, m^2_{loc})^t - (m^1_{ref}, m^2_{ref})^t\| \leq \|-(m^1_{loc}, m^2_{loc})^t - (m^1_{ref}, m^2_{ref})^t\|$ then the $\mathbf{u}$ axis of F_i is flipped to be consistent with F_{ref}.

When F_i is found, we use the *Iterative Closest Point* (ICP) algorithm [BM92] to better adjust it. As our SVM feature space depends heavily on F_i, improving its orientation and center using the ICP significantly improves the results of the SVM classification (see Fig. 6).

Robustness: Taken independently, SVM classification and ICP registration have their own weaknesses (see Fig. 6). For instance, when using ICP only, one can define virtual strokes (by expressing the original stroke of the reference in the local frame F_i) and run the patch selection algorithm. Unfortunately, this simple "template" strategy is sensitive to small variations in the local geometry and results in inadequate boundaries and holey selections (see Fig. 6-d). A better result is achieved when running our patch learning technique after the ICP (see Fig. 6-e), since the SVM classification accounts for the target geometry when retrieving the patch extent. Similarly, as the SVM depends on F_i, the classification is usually too rough without the local optimization performed by the ICP (see Fig. 6-c).

6. Results

Implementation and Performance: We implemented our SimSelect tool in C++ and report performances on an Intel Core 2 Quad/2.83GHz/8GB. We use the factorization downdating/updating techniques [XZCOX09] implemented in the CHOLMOD package [DH09] to solve linear systems. We use libSVM [CL11] and libICP [GLU12] for our patch learning and patch registration techniques. The interface is implemented in OpenGL with the Qt SDK.

The selection process is computed as the user is interacting with our tool and the three algorithms are efficient enough to run at an interactive rate on all models we tried (see Tab. 1 and 2).

The speed of our expansion step depends on the selection type: **(i)** For connected components, most of the work is done at loading time (4 seconds for the SeaMonster mesh, with 29 connected components and 110k faces). Then, the detection of similar components is done in real time (less than 2ms for the SeaMonster). **(ii)** To detect similar parts, we update a linear system for each potential selection, so the computation time depends on the number of vertices and grows linearly in the number of potential selections (see Tab. 1). Changing the number k of selected similar selections (moving the slider) is interactive because all the potential candidates are computed at the beginning of the expansion

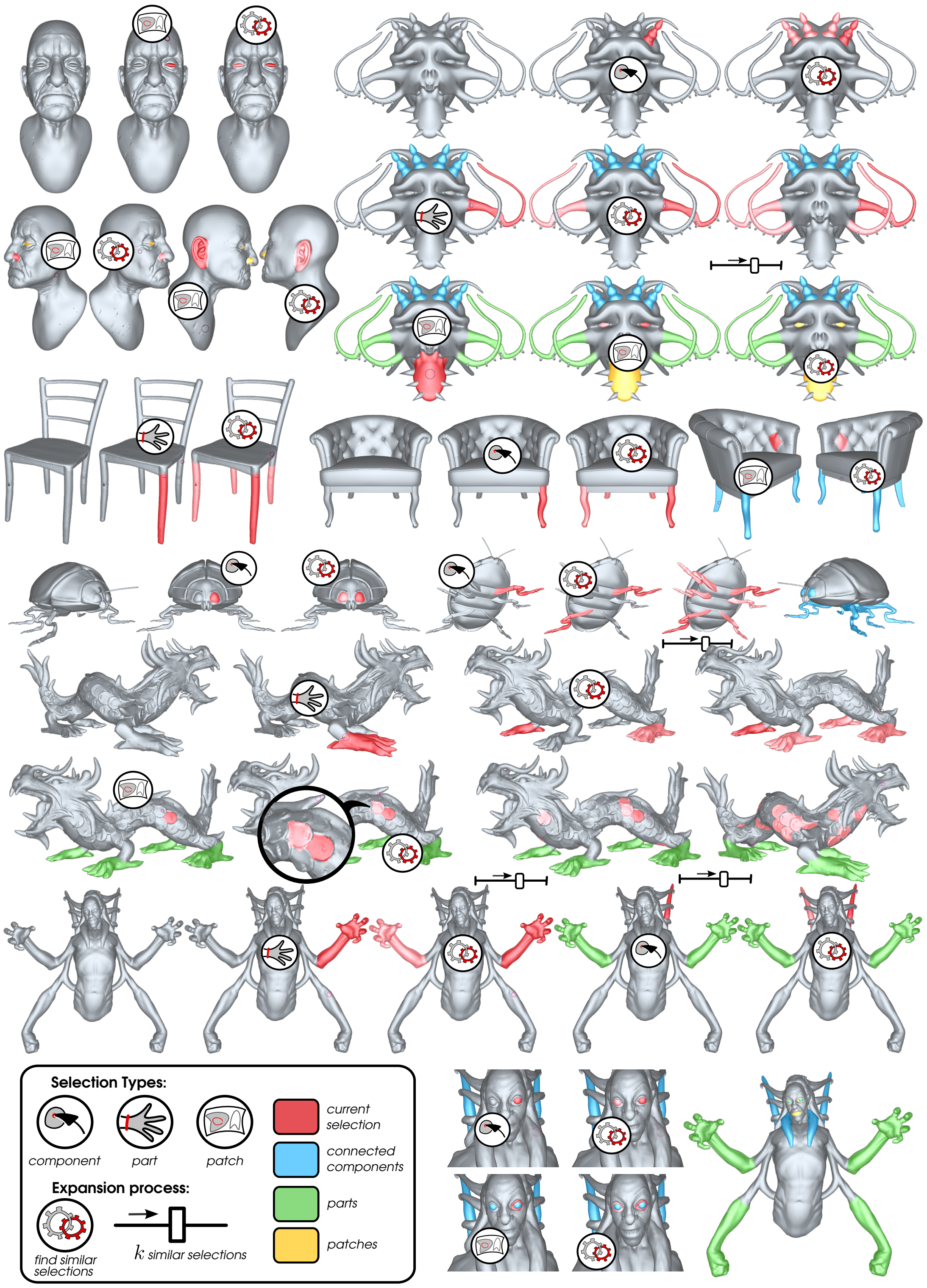


Figure 7: ***SimSelect Results:*** *the different models show results of the selection for patches (yellow), parts (green), and connected components (blue). The current selection is always in red, while automatic similar selections are displayed with color saturation proportional to similarity. User interactions are depicted by the circled logos and the slider.*

Model	(#V/#F)	PT (ms)	ST (ms)	PS	ET (ms)
Octopus	(12k/25k)	374	16	6	97
SeaMonster	(57k/114k)	1793	83	35	2385
Goro	(82k/165k)	2822	95	12	1158
Dragon	(100k/200k)	3588	162	47	6900
Chair	(14K/29K)	360	102	4	56
Hand	(55K/110K)	2870	102	8	1056

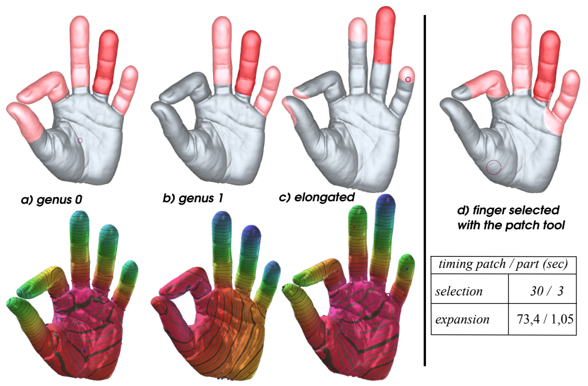
Table 1: ***Timing for parts selection.*** *PT: Precomputation time, ST: Selection time (average between each mouse motion), PS: number of potential strokes, ET: Expansion time*

Model	(#V/#F)	ST (ms)	ET(ms)			
			r	Map	SVMt	ICP
SeaMonster	(57k/114k)	8	0.07	3776	718	4
Armchair	(102k/201k)	27	0.16	14304	3829	142
Julius	(43k/85k)	11	0.2	11623	6470	24
Old Man	(84k/168k)	18	0.06	8119	1712	18
			0.05	7165	1731	13
			0.02	36689	5768	131
Dragon	(100k/200k)	25	0.06	10591	1175	10
Goro	(82k/165k)	7	0.02	5685	1471	56

Table 2: ***Timing for patches selection.*** *ST: Selection time (average between each mouse motion), ET: Expansion time, Map: Similarity map computation time, SVMt : SVM training time, ICP: ICP and SVM prediction computation time*

process. **(iii)** Finally the detection of similar patches depends on the number of vertices and on the patch size. The most expensive step is the similarity map creation, which requires the computation of a descriptor for each vertex of the mesh. Depending on the selection, the SVM training can also last few seconds. These two expensive steps depend only on the reference selection and are computed once, when the user starts the expansion process. Changing k is also interactive, as the only steps that remain are the ICP computation and the SVM prediction (see Tab. 2). The models shown in the tables are the ones used in the paper.

Discussion: To the best of our knowledge, our system is the first which intends to compute interactive selections of different kinds and to expand them automatically. Our selection step is mostly inspired by existing interactive segmentation tools; we aimed at finding a good tradeoff between user freedom and interactive feedback: the selection is inferred from few, rough interactions that the user can easily refine to explore the space of possible selections, both for local selection and global expansion. The expansion process is unified for the user who only has to control the number of similar selections she wants. The detection of similar selections greatly reduces the repetitiveness and the time consumed for selecting on surfaces in a select-and-edit workflow. We show some examples of results obtained using our tool in Fig. 7. Defining different expansion processes, depending on the reference selection type, is a key element in our approach. For instance, we illustrate in Fig. 8-d how expanding the selection of a part using the patch technique provides a less accurate result (not isometry-invariant) in a longer time (large selection radius). Similarly, defining this reference part selection using the patch tool is quite a tedious task for the user, requiring the navigation of the 3D camera to access hidden regions.



timing patch / part (sec)	
selection	30 / 3
expansion	73,4 / 1,05

Figure 8: ***Pathological cases:*** *(a-b) the part expansion depends on the mesh topology – here, the artifical tunnel between the two fingers prevents part similarity detection; c) the elongated finger results in inaccurate boundaries for similar parts; d) use of the patch tool to select a part*

The detection of similar connected components and parts is largely inspired from shape retrieval algorithms and partial shape matching techniques. However such algorithms aim at detecting similar models in huge databases and require long preprocessing, while we have developed specific solutions to retrieve surface portions on a single mesh *instantly*. Last, the SAMPLE [MPVF11] system detects similar patches, but requires an expensive precomputation step while limiting the online detection to disk regions with a pre-defined fixed radius. On the contrary, SimSelect detects patches based on a reference of any size and allows the interactive refinement of the expanded solution after its generation.

User feedback: Although a formal user study is beyond the scope of this paper, we gathered initial user feedback when using our tool to perform several selections on a variety of models. Over 16 user subjects, ranging from novice users to experienced CG designers, we gathered the following statistics: 44% of the users found the 3 selections equally important, 81% found the expansion very useful, and 56% would always use the tool if it were available in their favorite CG software (38% would often). We also notice that after an exploratory stage, most people naturally used the adequate tool to perform a selection. Details are provided as supplementary material.

Limitations & future work: Our system has several limitations which could motivate future work. Firstly, the expansion could be speeded-up: using level-of-details meshes would accelerate the part expansion and the patch similarity map could be computed on the GPU. Secondly, the similar parts detection greatly relies on the conformal factor which depends on the mesh topology. Moreover, although the conformal factor is invariant to isometry, it depends on the shape elongation: if a part is much more elongated than an other, the cutting boundaries will be poorly positioned (see Fig. 8). Thirdly, the similar patches detection is not scale-invariant, as we use the radius of the patch to locate potential patch centers for the expansion. One potential direction for future work would be to extend our patch descriptor to affine and isometry invariance. One solution could be to express the geometry of the patch on a local normalized exponential map [TSS*11] and to use an affine-invariant descriptor in this space e.g., a geometric version of the ASIFT descriptor [MY09] for instance. However, this raises the question of the computational cost of such a solution.

7. Conclusion

We have proposed SimSelect, an interactive system to accurately define selections of different types on meshes and expand them on the whole surface based on automatic similarity detection. From a user perspective, only a small set of simple metaphors have to be carried out – namely *clicking* components, *cutting* parts, *brushing* patches and *expanding* the selection –, while from a technical point of view, we have improved existing local segmentation algorithms for parts and patches and combined them with new shape retrieval techniques, running all of them on-the-fly for instant feedback. We believe that SimSelect can be useful to a wide variety of modeling scenarios, including interactive texturing, deformation, cloning and remeshing.

Acknowledgements This work has been partially funded by the European Commission (contracts FP7-323567 *Harvest4D* and FP7-287723 *REVERIE*) and by the Chaire MODIM of Telecom ParisTech.

References

[Aut] AUTODESK:. 3D Studio Max. 2

[BCG08] BEN-CHEN M., GOTSMAN C.: Characterizing shape using conformal factors. In *EG 3DOR* (2008), pp. 1–8. 4

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *PAMI 14*, 2 (1992), 239–256. 6

[CL11] CHANG C.-C., LIN C.-J.: Libsvm: A library for support vector machines. *ACM TIST 2*, 3 (2011), 27. 6

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. In *TOG* (2004), vol. 23, pp. 905–914. 4

[DH09] DAVIS T. A., HAGER W. W.: Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *TOMS 35*, 4 (2009), 27:1–27:23. 6

[FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. In *TOG* (2004), vol. 23, pp. 652–663. 2

[FLL11] FAN L., LIU L., LIU K.: Paint mesh cutting. In *CGF* (2011), vol. 30, pp. 603–611. 2

[G9̈9] GÄRTNER B.: Fast and robust smallest enclosing balls. In *ESA, Lecture Notes in Comp. Sci. 1643* (1999), pp. 325–338. 4

[GLU12] GEIGER A., LENZ P., URTASUN R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR* (2012). 6

[JLCW06] JI Z., LIU L., CHEN Z., WANG G.: Easy mesh cutting. vol. 25, pp. 283–291. 2

[LLS*05] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design 22*, 5 (2005), 444–465. 2

[LSS09] LIU J., SUN J., SHUM H.-Y.: Paint selection. In *TOG* (2009), vol. 28, p. 69. 2, 4

[LT12] LEIFMAN G., TAL A.: Mesh colorization. In *CGF* (2012), vol. 31, pp. 421–430. 2

[LT13] LEIFMAN G., TAL A.: Pattern-driven colorization of 3d surfaces. In *CVPR* (2013), pp. 241–248. 2

[MBM05] MORI G., BELONGIE S., MALIK J.: Efficient shape matching using shape contexts. *PAMI 27*, 11 (2005), 1832–1837. 4

[MFL11] MENG M., FAN L., LIU L.: icutter: A direct cut out tool for 3d shapes. *Computer Animation & Virtual World 22*, 4 (2011), 335–342. 2

[MPVF11] MAXIMO A., PATRO R., VARSHNEY A., FARIAS R.: A robust and rotationally invariant local surface descriptor with applications to non-local mesh processing. *Graphical Models 73*, 5 (2011), 231 – 242. 2, 8

[MPWC12] MITRA N. J., PAULY M., WAND M., CEYLAN D.: Symmetry in 3d geometry: Extraction and applications. In *CGF* (2012). 2

[MY09] MOREL J.-M., YU G.: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences 2*, 2 (2009), 438–469. 9

[Pho] PHOTOSHOP A.:. http://www.adobe.com/support/photoshop/. 4

[PJP93] PINKALL U., JUNI S. D., POLTHIER K.: Computing discrete minimal surfaces and their conjugates, 1993. 3

[SS10] SCHMIDT R., SINGH K.: Meshmixer: an interface for rapid mesh composition. In *ACM SIGGRAPH 2010 Talks* (2010), p. 6. 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: Geobrush: Interactive mesh geometry cloning. In *CGF* (2011), vol. 30, pp. 613–622. 9

[TV08] TANGELDER J. W., VELTKAMP R. C.: A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl. 39*, 3 (2008), 441–471. 2

[XZCOX09] XU K., ZHANG H., COHEN-OR D., XIONG Y.: Dynamic harmonic fields for surface processing. *Computers & Graphics 33*, 3 (2009), 391–398. 3, 6

[YJHS12] YÜCER K., JACOBSON A., HORNUNG A., SORKINE O.: Transfusive image manipulation. *ToG 31*, 6 (2012), 176:1–176:9. 2

[ZT10] ZHENG Y., TAI C.-L.: Mesh Decomposition with Cross-Boundary Brushes. In *CGF* (2010), vol. 29, pp. 527–535. 2, 3