A First Step Towards Cage-based Deformation in Virtual Reality

Andreas Scalas, Yuanju Zhu, Franca Giannini, Ruding Lou, Katia Lupinetti, Marina Monti, Michela Mortara, Michela Spagnuolo

A first step towards cage-based deformation in Virtual Reality

A. Scalas[1], Y. Zhu[1,2], F. Giannini[1], R. Lou[2], K. Lupinetti[1], M. Monti[1], M. Mortara[1] and M. Spagnuolo[1]
[1]IMATI, Consiglio Nazionale delle Ricerche, Genova, Italy
[2]LiSPEN, Arts et Metiers, Institut Image, Chalon-sur-Saone, France, France

October 19, 2020

Abstract

The advent of low cost technologies makes the use of immersive virtual environments more interesting for several application contexts. 3D models are largely used in such environments for providing feelings of immersion and presence in the virtual world. 3D models are normally defined in dedicated authoring tools and then adapted to be used in the virtual environments; thus, any change in the model requires to loop back to the authoring tool for performing the wished modification and the successive adaptation processes. The availability of shape modification capabilities within the virtual environment can avoid the above modification-adaptation loop. To this aim, we present our first step in the development of a 3D modelling system in Virtual Reality. The shape modification is achieved through a cage-based deformation approach, applied to semantically enriched meshes, carrying annotated meaningful regions, thus allowing the direct selection and editing of significant object parts.

1 Introduction

The advances in Virtual and Augmented Reality technologies, including the improvements in the capabilities to trace human movements and gestures at low cost easily affordable by small companies, professionals and even by amateurs pave the way in the development of more natural shape interaction and modification applications usable in various industrial and leisure contexts. Early, immersive Virtual Reality (VR) systems were highly expensive and only large automotive and aeronautic companies could afford their costs. Immersive VR systems were mainly used for product evaluation allowing simulations at 1:1 scale differently from a desktop screen that provides only a small size product visualisation. Anyhow, their usage within the product development process was and still is requiring additional efforts. The use of engineering models in VR environments involves an adaptation process which also includes format conversion and results in loss of information, being only the shape and few other attributes transferred. Moreover, any further shape modification resulting from the evaluation and simulation cannot be done in a VR environment but requires to go back to the original CAD (Computer-Aided Design System).

The advent of more affordable systems and less intrusive devices has enlarged the range of users. For instance, it opens the possibility for companies to include end-users in product evaluation and customisation. Other industries that strongly benefit from these new technologies are those addressing training, marketing, cultural and entertainment context. In these contexts, game engines are normally used to create interactive applications. These game engines present two modalities: the editing and the play modality.

In the edit mode, the application is created, requiring the specification of the arrangement of the included digital assets, of the user interaction functionalities and modalities together with the corresponding system's responses, while in play mode the final application is played to be used by the end-user. Also in this case, the creation of VR applications involving 3D environments and data requires the creation of 3D models, their translation and adaptation for their use in the VR platform.

Similarly to the industrial product development context, the modification of 3D digital shapes requires to iterate the whole creation and preparation process, i.e., to go back to the 3D authoring system to apply the required shape modification and then re-perform the adaptation process with the consequent loss of information (e.g., surface type or other geometry organisation in terms of the constituting semantic components). Therefore, it is clear that improving VR environments with capabilities to modify the shape would greatly shorten the overall application development cycle. Moreover, considering that the people developing VR application might be inexpert in 3D modelling it would be important to provide such functionalities with easy to use commands, possibly directly in the play mode such that both the end-user can apply modifications to the shape according to his/her desires and the application developer can immediately see his/her modification. The former opportunity is of interest also for the development of systems allowing the customisation of products by customers.

In this paper, we present our initial feasibility study to develop a 3D modelling system integrated within

the VR environment, based on voice and gesture commands. The allowed modifications of the 3D model can either be global or local, possibly applied to semantically meaningful sub-parts of the object. Methods for importing semantic data into the application are presented. The system has been developed using the well-known and widespread Unity 3D game engine; gesture tracking has been achieved using Leap Motion sensors. This choice has been motivated by the capability of tracing bare hand and fingers movements at limited costs and of integrating more immersive environments with the use of relatively low-cost head-mounted displays.

The rest of the paper is organised as follows: Section 2 provides an overview of shape modelling tools in VR focusing on those using Leap Motion Controller. Section 3 introduces the adopted modelling paradigm based on cage deformation and the aspects related to semantic annotation of 3D meshes. Section 4 presents a system to perform 3D model modifications in a virtual reality environment, while Section 5 introduces the developed prototype, providing some examples of achieved results. Conclusions in Section 6 end the paper summarising the achieved results, the ongoing work and the possible future extensions and applications.

2 Related Works

During the last decade, the interest in the development of immersive VR and AR systems for manipulating and modelling 3D environments through gestures has increased [9,29,35]. This is also due to the technological improvement of low-cost AR/VR technologies and gesture tracking acquisition devices.

More recent Head-Mounted Displays (HMD) have reduced the limitation of early HMDs related to the sense of sickness, which allowed only a short use, and at the same time the lowering of their cost has made wider access to immersive Virtual environments possible. Recently, most of the HMD producers have begun to offer 3D modelling applications.

In general, the commercial systems either address the generation of objects for games and animation or try to integrate the CAD user interface in VR environments using controllers replacing desktop mouse providing buttons and pointing capabilities.

Shape sculpting by push and pull operations together with the combination of primitives are the main capabilities of commercial applications, such as Sculpting [28], Oculus Medium [40], targeting the animation sector. Similarly, in the engineering field some new applications are being developed, such as MakeVR Pro [34], which provides a CAD-based fully immersive 3D content creation experience. It uses controllers as 3D mouse and widgets (e.g. rulers, grids) for positioning and combining objects, while for 3D manipulation it adopts a smartphone-like metaphor for 2D handling.

MindeskVR [36] allows editing 3D models in a fully immersive environment with 6 degrees of freedom. Models are edited using surface control points selected through controllers acting as a 3D mouse. Combination of free-hand sketching for shape generation and push-pull and control point editing is also provided by the Gravity Sketch [18] tool by means of controllers.

These systems are only focusing on the resulting shape, without any consideration on its associated semantics, and require the use of controllers for acting on the shape. Similarly to other researchers [17], [48], we argue that being able to operate on selected semantically meaningful areas while simply using our hand movements makes the interaction more natural and pleasant.

First attempts in gesture-driven shape manipulation were made using wearable and ad-hoc developed devices or markers to fine track the hand movements and positions, e.g. [14, 15, 26, 27, 43]. However, adding hand tracking devices to the virtual set-up was formerly problematic because of the relatively high costs of such devices.

The advent of Leap Motion Controller (LMC), able to detect and trace the full hand and the fingers' joints, allows capturing fine bare hand gestures when hands are not overlapping or movements are not performed in the orthogonal direction of the LMC. The additional benefit of being easily transportable, the simplicity of their use and the lack of interference with the users' gestures aligns well with the goal to achieve intuitive interaction. For these reasons, LMC inspired a large number of gestural interface studies in recent years.

Various works address the manipulation, e.g. rotation, translation and limited modification in size of 3D objects. Among them, in [21] the user can manipulate 3D holographic objects in AR environments using the Microsoft Hololens in the real environment's scale. The designed holographic objects can be assembled/disassembled interacting with the real environment surfaces. Then, the obtained objects can be exported into a proper file format to be automatically produced by a 3D printer.

[33] adopted LMC to browse, inspect and deform the results of a 3D CAD assembly search engine in an immersive environment provided by HTC Vive HMD. Using a combination of voice and gesture commands, the user can easily browse and select the assemblies most similar to a query one; visualise the corresponding parts; select, move and resize assemblies or their constituent parts to better inspect their content.

Among the work devoted to effective shape modelling, Vinayak and Ramani [48] developed a system for modelling and shaping pottery objects using LMC. In this work, they address the problem of determining how the shape and motion of a user's hand and fingers geometrically relates to the user's intent of deforming a shape. Thus, they consider that the expression of user intent in the shaping processes can be derived from the geometry of contact between the hand and the manipulated object. The developed system supports the user in modelling the shape of a pot as a gradual and progressive convergence of the pot's profile to the shape of the user's hand represented as a point cloud; each point in the hand's point cloud attracts a local region

on the pot, hence deforming the pot's section. Thus, in addition to the problems due to occlusions resulting from the camera position and hand orientation, this approach is affected by the density of points in the manipulating object and the resolution of the pot mesh.

Cui et al [10] proposed a web-enabled shape modelling system with mid-air free hand interactions whose movements were captured with Leap Motion sensors. They included sketching and various global deformations like compressing, squeezing, enlarging, twisting and tapering. They allowed the use of either one or two hands. To make the system more natural they included constrained deformations and metaphors helping the user to understand how to act, as the visualisation of driving wheel for rotation.

Later, Cui and Sourin [11] exploited LMC for shape modelling. To overcome the Leap Motion problems of hand jitter, jump release, and occlusion, they proposed the use of bi-manual interaction and the functional separation of the hands: one hand (the dominant hand) controls 3D position and rotation, while the other hand controls grasping and releasing. Through comparison with commercial 3D modelling systems using mouse and keyboard, the results of the paper prove the usefulness of optical hand tracking in precise 3D modelling.

Applying the clay modelling approach, Park et al. [41] proposed a virtual figure model crafting system aimed at allowing users to freely generate the shape of a figure and decorate its exterior in VR using HMD. To provide realistic crafting user experience, they proposed motion grammar-based gestural input with Leap Motion and considered the support of multiple difficulty levels for user engagement.

More devoted to the engineering context, Cohen et al. [7] used LMC to capture human gestures corresponding to the pinch and pull actions for flexible manipulations on control points coordinates of NURBS surfaces. The continuous motion of the designers' hands is interpreted in terms of intention and position of the selected control points of the surfaces. Both single and dual hands movements are developed.

All these systems are only modifying the geometry, either as clay or by control elements, without any explicit constraint or support for more meaningful localisation of the deformation. Conversely, we argue, as expressed in [39] and [30], that having a more semantic component-based arrangement of the shape can better support end-users in the modelling activity. The semantic organisation of the virtual scene is well recognised as a mean to better support behaviour specification of VR elements in response to user actions [6] and [4]. [17] described a semantic model usable for modelling purposes, highlighting its advantages for shape modification and the potential of its integration in game engines.

In this work, we present a starting effort to apply cage-based deformations, which are largely used in the animation context, to 3D models immersed in VR, providing an interaction as natural as possible to the user; to this aim, we exploit gestures and voice commands.

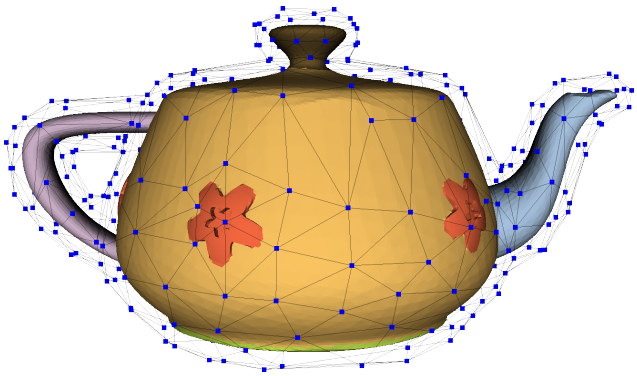


Figure 1: A teapot shape with some parts of interest annotated (colours correspond to tags) and surrounded by the associated cage, depicted in wireframe.

The deformation process is enhanced by a semantic component-based description of the shape to be manipulated, allowing easier selection and location of the modifications.

3 Background: deforming semantic shapes

In this work, we present our preliminary results in applying voice and gesture interactions to shape deformation directly in a Virtual Environment provided by the Unity 3D game engine. We represent 3D models as watertight triangle meshes defined as (V_M, T_M), where V_M and T_M are the set of mesh vertices and triangles, respectively. Furthermore, we expect models to carry semantic information in the form of Regions of Interest (RoIs), parts of the mesh having a meaning to the user, annotated with additional information, e.g., textual tags.

There are several paradigms to express mesh deformation, which differ according to the type of *handles* the user can interact with. Handles are typically a few points or line segments that can be manipulated and which propagate the deformation to the whole shape. Skeletons provide segment-like handles, and are the most popular deformation metaphor in animation because they are most suitable for pose modification of articulated shapes. In our work, we apply cage-based deformation because cages provide more flexibility for deforming generic shapes. We avoided the use of free-form deformation techniques (e.g., [44]), because they allow just limited control over the shape to be deformed (which depends on the initial shape of the control lattice), or other kinds of deformation techniques such as the *variational* ones [1], because of their higher computational complexity due to the minimisation of some cost function that is required in order to manipulate the shape.

In the following, we define annotations and describe how to properly manipulate annotated portions of the shape through cage-based deformation.

3.1 Semantic annotation

As anticipated in section 2, the possibility to operate on selected semantically meaningful areas could provide faster algorithms, better results and more pleasant interfaces. Just to give a few examples, the search for a specific shape could be a lot simpler introducing a filter to visualise only the shapes resembling a specific object or containing a specific part (e.g., search for "teapot" shapes containing a "flower" pattern); visualisation interfaces can be defined for highlighting certain parts (e.g., focus on the shape of the "pump" in a fridge assembly shape); and, of course, manipulation interfaces could greatly benefit from the introduction of semantic annotations.

There is a rich literature about annotation systems for helping users to define RoIs and associate them tags and, more in general, information. Some examples are: 3DSA [54], a system for crowd-sourcing textual annotations (tagging) of objects for inferring archaeological classification of findings; CHiSEL [47], an Information System based on the use of the 3D representation of an object as a sort of "blackboard" where different information is represented; Aïoli [12], a collaborative annotation framework based on photogrammetry and high-performance cloud computing; and CHER-Ob [46], a framework providing different tools for the evaluation and publication of the results of cultural heritage research, and the support for visualisation of different data formats.

In the engineering domain, the concept of feature is well known to identify parts associated with context specific semantic information and modelling behaviour [13, 45]. On the one hand, features are commonly adopted by commercial Mechanical Computer-Aided Design (CAD) systems, allowing an easy modification of parts through meaningful parameters, which drive the shape changes according to engineering objectives. On the other hand, feature recognition systems allow the annotation of CAD models in terms of features useful for production purposes, allowing the integration with manufacturing or assembling tools and operations [24].

As in [42], in this work we define a 3D part-based annotation A as a pair (S, I), where S is the geometric selection (or RoI) and I is some associated information. A geometric selection can identify a region, a line or a point of interest; in this study, we experiment annotated regions, with S specifying the set of connected triangles $T_A \subseteq T_M$ constituting a specific region of interest; vertices of the region $V_A \subseteq V_M$ can be derived from triangles efficiently, if needed. We associate to annotated regions a special tag from a controlled vocabulary, defined a priori, that is, a textual information specifying the semantic part of the object. For instance, parts of the teapot depicted in Figure 1 will be annotated with tags such as *spout*, *handle*, *flower*.

Annotations (annotated RoIs) can overlap (e.g., flowers overlap with the "body" part of the teapot) hence indicating a hierarchy among parts. We can arrange parts into a tree, having the whole object, annotated with the name of its class (e.g., "teapot"), as the root. Starting from the root node, we build the hierarchy of parts populating the tree, checking the containment of annotations - an annotation is contained inside another if all its associated triangles belong to the other one as well. Figure 2 shows an annotated hand and the corresponding hierarchy of parts.

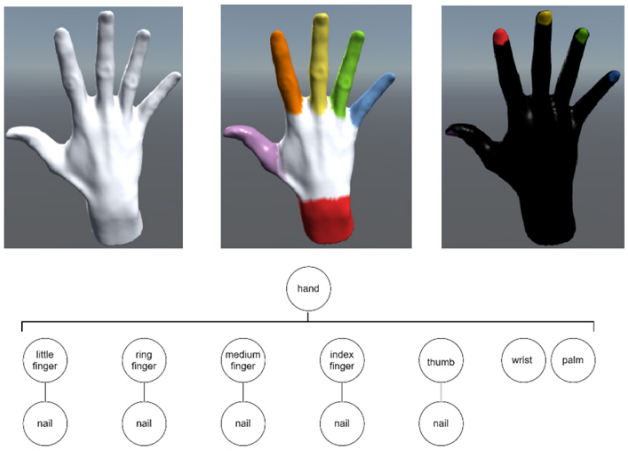


Figure 2: Top row: a hand model highlighting the semantic parts at the same containment level: the whole model (left); fingers, palm and wrist (middle); nails (right). Bottom row: the corresponding hierarchy of semantic parts.

In this work, we assume the object has already been annotated and we import the "semantised" shape into our system by loading geometry and tags from a structured file, as detailed in Section 4.2.1 (see Figure 1).

3.2 Cage-based deformation

Let us denote a cage C any closed mesh that envelops another mesh M to be deformed (see Figure 1). The cage is usually a simplified version of the mesh M and contains much less vertices. This tool finds several application contexts, such as the collision detection and in general everywhere a coarser version of a mesh could be useful e.g., for reducing the computational complexity. In this work, we will focus on the usage of cages for manipulating the shapes of objects in 3D authoring systems. Basically, the user selects and moves in space a set of cage vertices, that act as control points, and the object mesh will deform accordingly.

The optimal cage size in terms of vertex cardinality is given by a trade-off between the number of Degrees of Freedom (DoF) available to the user for the deformation and the deformation complexity and computation time. A fine cage (e.g., having the same number of vertices of M) will give the user a lot of control points allowing very detailed deformations, which, however, are more complex to perform: she/he should be skilled in order to manage many control points and obtain the desired shape. Moreover, this will increase considerably the computational time so that possibly a standard hardware will not suffice to achieve an interactive deformation. Conversely, a cage with few points

enables a coarser control on the deformation but allows the modification to be much more intuitive and a faster computation. Some possibilities to automatically construct cages can be found in [3, 5, 49–53].

Cage-based deformation techniques base on the concept of Generalized Barycentric Coordinates (GBC), which give means for defining the value of a certain function over a point in space (and so, for example, over a mesh vertex) in terms of a linear combination of some control points (cage vertices).

Barycentric coordinates were first introduced by Möbius in [37] for computing the value of a function over a point inside a triangle in terms of a linear combination of the triangle's vertices. Then, it has been generalised to closed triangular meshes simultaneously by Floater et al. [16] and Ju et al. [23] introducing issues that were partly solved in some successive works [19,22,31,32,55].

Apart from the framework defined in [32], where the cage triangles were used for obtaining a quasi-conformal deformation, the general formula for the computation of the value of a function f over a vertex v_i of M is:

$$f(\mathbf{v}_i) = \frac{\sum_{j=0}^{m} w_{ij} f(\mathbf{c}_j)}{\sum_{j=0}^{m} w_{ij}}, \tag{1}$$

where $\mathbf{c}_j$ is the j-th vertex of the cage and w_{ij} is the weight associated with $\mathbf{v}_i$ and $\mathbf{c}_j$. So, if we take $f(\mathbf{x}) = \mathbf{x}$, we can use the previous formula for computing the position of the vertices in V_M using the positions of the vertices in V_C, no matter their position being the original or a new one. For further detail, we refer the reader to any of the surveys existing on the topic (e.g., [38]).

We can rewrite the formula in matrix form as

$$V_M = B \cdot V_C, \tag{2}$$

where B is the matrix containing the normalised GBC $\lambda_{ij} = \frac{w_{ij}}{\sum_{k=0}^{m} w_{ik}}$.

4 The proposed 3D shape deformation system in VR

Cage-based deformation techniques are already quite common on desktop applications (such as the CageLab tool [2]) for the deformation of shapes, thanks to their flexibility, ease of implementation and speed. However, VR environments introduce a number of additional difficulties that require careful evaluation and treatment.

Here, objects are embedded in a 3D space that replicates the physical world, increasing the sense of realism; but how can we exploit the 3D setting to effectively communicate the information associated with the annotated geometry to the user? How can she/he interact with the 3D elements according to a natural and intuitive behaviour? These questions call for an efficient mechanism and a natural interface to: i) select the control points (CPs) related to the area to be deformed; ii) specify the desired deformation on the selected CPs; iii) apply the required deformation in real time. Furthermore, we want to exploit the semantics associated to the object.

In the following, we first describe the application setting in the virtual environment; then, we focus on the proposed interaction mechanisms and metaphors to address the above issues.

4.1 Application setting

The proposed system uses the graphical engine Unity 3D to visualise a single virtual scene accessible by the user through an HTC Vive head-mounted display (HMD). The interaction in the 3D space is ensured by the Leap Motion Controller (LMC), which is able to track fingers' joint positions and detect simple gestures.

Figure 3 shows the organisation of the proposed system. A 3D object and its cage are given as input, along with the list of annotations associated to the model (e.g., defined by a domain expert) and the generalised barycentric coordinates (GBC) values. The two 3D meshes (object and cage), the annotations and the GBC are processed in the *data manager* module, also responsible for managing cage and model modifications resulting from the user interactions. The *visualisation manager* module is responsible for rendering the scene, updating it during the user interaction, and communicating information, e.g., by adding extra virtual elements or exploiting colour variation of scene elements. Finally, the *interaction manager* module processes and interprets the data derived by the LMC and the HMD recognising different interaction techniques.

4.2 Data manager module

The Data Manager module loads the input data in the system and performs the geometric analysis and processing operations required by the deformation, as expressed by the user either through gestures or voice and interpreted by the Interaction Manager.

In the rest of this section, after illustrating how input data are structured (sub-section 4.2.1), we describe the computational solution for the automatic selection of control points related to a semantic part (sub-section 4.2.2) and the process to apply the desired deformation to the model mesh (sub-section 4.2.3).

4.2.1 Data load

We assume the model has been previously annotated and comes ready for the manipulation; therefore, the data manager loads the following files: i) the model M, a 3D model represented as a watertight triangle mesh; ii) the cage, a triangle mesh, generally coarser than the model itself; iii) annotations, a structured file specifying geometry, tag, colour and other useful data to manage the region of interest; iv) the generalised barycentric coordinates (GBC), a list of pre-computed values depending on the vertices of M and C that specify the influence of the different cage vertices on each one of the object's vertices. In general, an object vertex might be influenced at a certain extent by all the

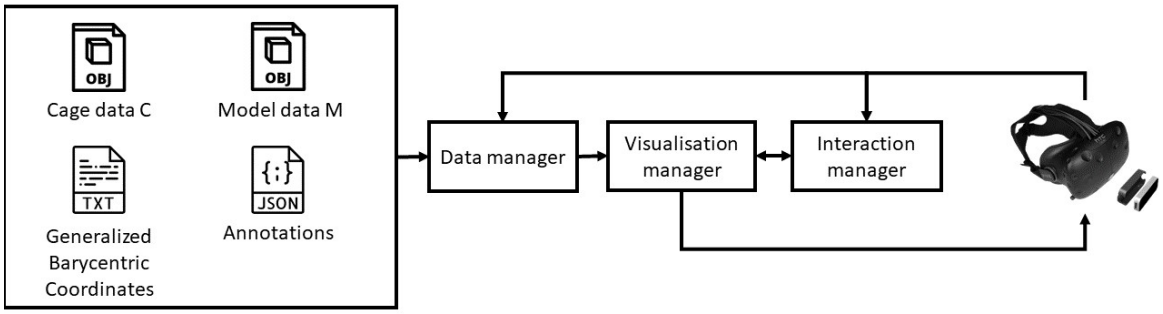


Figure 3: System overview. From the left, the input data, the modules involved in the system and the resulting VR environment.

cage vertices.

The input data are structured by the data manager module in appropriate data structures. Vertices of models M and C are stored in matrices V_M and V_C, where each row corresponds to the 3D coordinates of a vertex. As shown in Figure 4, V_M has size $n \times 3$ and V_C has size $m \times 3$, where n and m are the number of vertices in M and C respectively.

Then, a list of control points $\{CP_i\}_{i=1}^{m}$ is generated, where the CP_i is associated with the position of the i-th vertex of C (i.e. the i-th row in V_C). Finally, the values of the matrix B in equation 2 are extracted from the GBC file obtaining a matrix B of size $n \times m$, and annotations are arranged in a tree structure accommodating the hierarchy information among the different RoIs, as defined in the annotation file.

4.2.2 Automatic identification of CPs related to an annotation

As introduced in section 3, let A be an annotation associated with triangles T_A and vertices V_A and let the matrices V_M and V_C, whose rows correspond to the 3D coordinates of the vertices in the meshes M and C, respectively, be related by the formula (in matrix form) $V_M = B \cdot V_C$. From this relation, it follows that a certain cage vertex c_j impacts each of the model vertices at a certain extent, defined by the values specified in the corresponding column (the $j-th$) of B. Analogously, a mesh vertex v_i is influenced by each cage vertex at an extent defined by the elements in the $i-th$ row of B. However, some cage vertices will have a much higher influence than others on v_i. Thus, given a region of interest on the mesh, we will select the relevant control points as the cage vertices whose influence on the region vertices is greater than a certain threshold τ. So, given a vertex $v_i \in V_A$ associated with the RoI, we are interested in finding all the vertices $c_j \in V_C$ whose influence values b_{ij} are greater than τ (see Figure 4).

Applying this procedure for all the vertices in V_A, we end up having a set of cage vertices that can be used to manipulate the area corresponding to S, up to a certain precision that depends on the number of cage vertices around the interested area, their distance from the area, the presence of other annotations close to A and the locality of the GBC that have been used. Indeed, we remember that Mean Value Coordinates

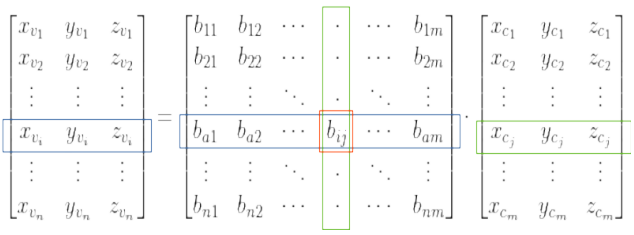


Figure 4: A detail of the matrix multiplication $V_M = B \cdot V_C$, where v_i is the vertex under analysis, b_{ij} is the highest value on row i and c_j is the corresponding cage vertex.

(MVC) [16, 23] have a global behaviour, meaning that every cage vertex always influences the shape almost in its entirety, while for example Local Barycentric Coordinates (LBC) [55] provides a high localisation of the influences.

Notice that, while in this work a single threshold τ has been applied to determinate the CPs associated with the different RoIs, different thresholds may be identified for obtaining better fittings of the single RoI.

Of course, the achieved selection can be subject of user validation, with the possibility of adding/removing single control points from the set of suggested ones (see Section 4.4). Note that this validation process does not affect the result of the previous step permanently: changing the CPs involved with a certain RoI requires the editing of the threshold value.

4.2.3 Computation of the deformed mesh

Once the control points have been selected, and the deformation parameters are interpreted from the user actions by the interaction manager, the actual deformation occurs; this computation is performed by the data manager. Firstly, the selected control points are transformed (applying the same transformation to the associated cage vertices), secondly the deformation is propagated to mesh vertices as explained in Section 3.

Cage-based deformation techniques give the user total freedom, so that he can move one or more control points in the space regardless of their mutual arrangement. Of course, extreme displacements of vertices can give bad results, mainly due to the nature of Linear Blend Skinning techniques, of which cage-based tech-

niques are a subset (details are given in [25] and [20]). In this work, we allow three main deformations:

- **Translation**. The control points are moved in the space in the same direction. The new control point position is defined as $\mathbf{c}'_i = \mathbf{c}_i + \mathbf{d}$, where $\mathbf{c}_i$ is the original position of the i-th cage vertex, $\mathbf{c}'_i$ its new position and $\mathbf{d}$ a translation vector. Note that the same transformation can be obtained in matrix form as $V'_{CP} = V_{CP} + D$, where V_{CP} is the matrix containing the cage vertices corresponding to the selected control points, V'_{CP} are their new positions and $D = I_3 \cdot d$.

- **Rotation**. The control points are rotated with respect to an arbitrary axis a of an angle θ. Given the two axes a_s and a_e returned by the interaction manager module and representing the start and end configuration, the rotation axis a is defined as the line passing through the centroid (R_c) of the selected CPs and with direction R_d equal to the vector product between a_s and a_e, while the rotation angle θ is defined as the variation between the starting and ending axes a_s and a_e. Notice that, if R_c does not correspond to the origin of the world reference frame, a combination of translation and rotation is required. So, the new control point positions can be obtained as $\mathbf{c}'_i = (R(\mathbf{R_d}, \theta) \cdot (\mathbf{c}_i - \mathbf{R_c})) + \mathbf{R_c}$, where $R(\mathbf{a}, \theta)$ is the matrix defining the rotation around the axis a of an angle θ. The corresponding matrix form is $V'_{CP} = R(\mathbf{R_d}, \theta) \cdot (V_{CP} - \mathbf{R_c}) + \mathbf{R_c}$.

- **Scaling**. Differently from the previous deformations, the proposed scaling operation acts on the entire 3D model changing the positions of all the control points. The new control point position is defined as $\mathbf{c}'_i = k(\mathbf{c}_i - \mathbf{o}) + \mathbf{o}$, where k is the scale factor, $\mathbf{c}_i$ is the original position of the i-th cage vertex and $\mathbf{o}$ is the centroid of the 3D model, computed as the average of the model's vertices position. The corresponding matrix form is $V'_C = K * (V_C - \mathbf{o}) + \mathbf{o}$, where $K = I_3 * k$. In this way, control points are moved toward or away from the scaling centre (i.e. the centroid of the 3D model) resulting in a uniform scaling.

4.3 Visualisation manager module

This module receives input from the data manager (see sub-section 4.2 and the interaction manager (see subsection 4.4 to visualise the "semantised" object, the virtual hands used to interact in the virtual environment and the effect of the user's actions on the 3D model.

Once the data manager module has processed the input data, the model mesh M is (always) visualised in the scene in a shaded mode, while the cage model mesh C is rendered in wire-frame mode to avoid hiding the model M. At the cage vertices positions, independent sphere elements are introduced to highlight the control points (CPs). The user can hide completely the cage and the CPs (pressing the "Hide cage" and "Hide CPs" button options with the virtual hand in the menu beside the model) to visualise only the 3D model with no obstruction of additional elements.

Annotations are highlighted in the virtual scene by assigning each a specific colour on the model M, as expressed by the data manager module. The latter is also responsible of identifying the CPs associated with each annotation (see Section 4.2.2); the visualisation manager receives this information and colours the CPs accordingly. If a CP influences multiple annotated parts, then the visualisation manager assigns it the average of the different colours involved. Figure 5a shows an example of this effect.

The visualisation manager presents also the data derived from the interaction manager module. First, it visualises the result of the hand tracking. Among the different avatar types of hands included in the Leap Motion asset, we adopt in our system the capsule hand type, a solution that allows to visualise flexible fingers improving the sense of reality in the virtual scene.

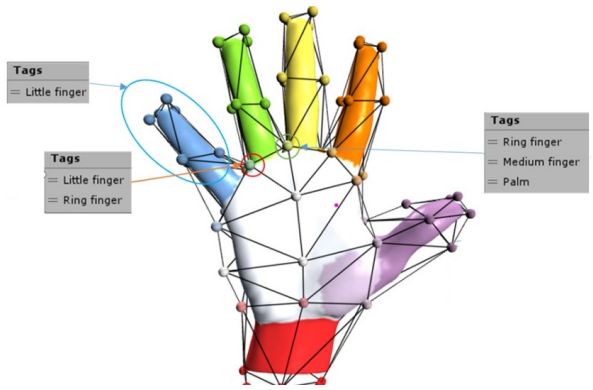
In addition, this module provides an echo to the user for the selection operations detected by the interaction manager module, i.e., when the user looks at a target it is highlighted. It is possible to select a single CP or a a set of CPs associated with an annotation; in the latter case, all the associated CPs are highlighted. Depending on the viewpoint and on the shape complexity, some CPs related to a semantic part may be hidden behind the model M itself; in this case, a yellow marker appears in correspondence of the covered CPs. An example of this behaviour is depicted in Figure 5b, where the user gazes at the palm.

Finally, to reduce the amount of information visualised simultaneously, this module allows the selective rendering of annotations according to their hierarchy level.

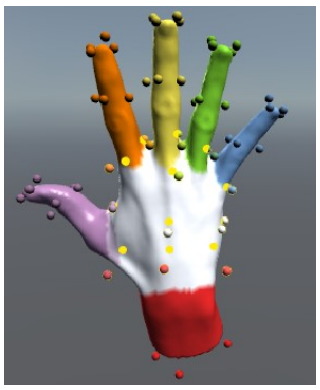
4.4 Interaction manager module

The interaction manager communicates with the data manager module every frame passing the parameters required for the deformations that are computed by analysing the user's commands. In this way, the deformation is computed and applied on the model every frame resulting real-time with the interaction. Then, to apply the developed shape deformation methods, we define a set of commands using hand gestures and voice keywords considering that standard users are non-expert designers of 3D shapes. Thus, we aim at defining interactions that are *easy to learn* and *easy to use*. Generally, to support these requirements, we define commands simple to be remembered and not similar among them to avoid getting mixed up. In addition, the user can apply mathematical transformations (translation, rotation and scaling) without the necessity of specifying axis, angles or others specialist concepts.

In accordance with these considerations, we designed different interaction techniques, summarised in Table 1 and described in the following.

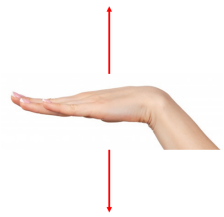


(a) CPs associated to multiple segments

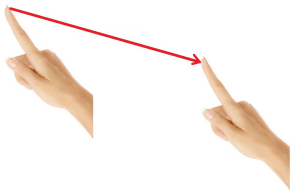


(b) Selected CPs not visible from the user point of view directly

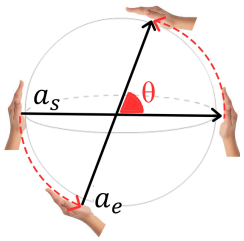
Figure 5: Example of the data represented by the visualisation manager module



(a) Hand-swipe



(b) Finger-drag



(c) Handling rotation

(d) Magnify

Figure 6: Interaction gestures

4.4.1 Selection

With the aim of providing *easy to use* interactions, we avoid a selection by using virtual hands. If, on the one hand this metaphor is extremely natural, on the other hand it requires an high precision level in picking exactly the desired target. Users' accuracy has to be even higher when target objects are small or surrounded by others possible targets.

For this reasons, we propose the use of a *selection gaze*, where the selection operation is performed in two steps. First, the user acquires the desired target by looking at the different selectable elements in the scene; then, she/he confirms the choice by using the voice command "select".

The selectable objects in the scene, i.e. the objects the user can interact with, are single control points, sets of control points associated with annotated parts of the object and all the control points.

Once a target has been acquired, it is highlighted to indicate the selectable element. Finally, once the target has been confirmed, the colour of the control points changes from the original colour (obtained as specified in subsection 4.3) to magenta. The same procedure is applied to deselect objects. In this case, the interaction is identical, the voice command to confirm the deselection is "discard" and the user's visual echo differs by changing colour from magenta to the original one.

As mentioned in section 3.1, annotations are organised into several levels of detail. To browse and change level of detail for the selection, we propose a *hand-swipe* command (see Figure 6a) allowing the user to increase or decrease the level of detail moving his/her open hand vertically. If the user aims at accessing the n-th level, then she/he can use voice commands saying "level n" to visualise and have access to the level of information she/he is interested in directly.

4.4.2 Translation

Through the *finger-drag* gesture (see Figure 6b), the user can deform a 3D shape by moving the selected CPs in the virtual space in any direction. This technique allows to interact with single and multiple CPs. Selecting the unique segment belonging to the first level, it is possible to move the entire 3D model.

Selected elements can be moved according to the tip of the index finger position, whose displacement in the space defines the translation vector $\mathbf{d}$ (defined in Section 4.2.3), yielding the technique simple and intuitive. Indeed, it is sufficient for the user to point at elements she/he aims to relocate, extending the right index while closing the others, and wait for the acoustic signal confirming the gesture detection. Once the gesture is identified, the user can move the selected objects in the 3D virtual space with a whole freedom of movement.

Once the user has reached the desired modification, she/he can stop the interaction changing the hand posture, for instance opening the hand completely.

4.4.3 Rotation

To perform 3D rotation, the user can select the whole model or its portions, by selecting one or more control points or an annotated RoI.

Once the selection is accomplished, the user engages the *handling rotation* technique by maintaining both hands open, with palms facing each other, until an acoustic echo communicates the success of the command recognition. Maintaining the engaging posture, the user moves his/her hands defining the starting and the ending axes a_s and a_e in the space characterised by the hands posture (see Figure 6c). The a_s and a_e axes are used to compute the rotation parameters as defined in Section 4.2.3. To conclude the rotation, the user has to close his/her hands.

This technique is quite simple since the user does not need to combine several rotations to rotate along with a generic axis (i.e., not aligned with the coordinate system); then, also users non-expert in design can achieve the desired result. In addition, large rotations can be achieved by repeating the gesture several times.

4.4.4 Scaling

The *scaling* operation (see Figure 6d) increases and decreases the size of the model. To activate it, the user has simply to place the hand in front of herself/himself; if the hand presents all extended fingers then the model is enlarged, otherwise, if the hand is closed, the model size is reduced. To stop the interaction, it is sufficient to remove the hand from the leap motion field of view.

With this technique, the interaction is affected only by the hand posture (open/closed hand); this means that no matter if the user changes his/her hand position during the interaction. On one hand this ensures a quite simple interaction, since the user has to control only one action. On the other hand, the user cannot scale the 3D model along with a preferred direction, i.e. the scaling is uniform. In addition, it is not possible to apply the scaling operation to a portion of the model yet. A more in-depth study is required on local scaling. Indeed, increasing or decreasing the size of a portion of the model requires a careful analysis of what happens with respect to other parts of the object (self-intersection), of the possibility to introduce unacceptable distortions along the "boundary" of the part, but also of what the scaling centre would be (and so the direction of the scaling with respect to the original positions).

5 Preliminary results

The proposed system has been developed adopting Unity 2019.2.4f1 as graphical engine and programming in C# language. It has been designed for wearing a HTC Vive head mounted display and a Leap Motion Controller with the VR Headset set up. In this work, the threshold τ used to define the CPs associated with a certain ROI has been defined a priori and the user has no possibility to modify the default value.

(a) Level 1 (b) Level 2 (c) Level 3

Figure 7: The different levels of the annotation hierarchy; CPs are coloured according to their influence on annotations with threshold 0.5, as described in Section 4.2.2

.

(a) Initial stage of the 3D model (b) Final stage of the 3D model

Figure 8: Deformation example applying the same finger-drag translation on a set of CP.

In the results here proposed, we used a hand model and cage having 10K and 70 vertices respectively. The employed GBC are the MVC, which have been chosen because of their fast implementation and the closed form expression allowing better performances (parallel computation) in the computation phase. The threshold τ has been empirically estimated assuming the value 0.5 and the results of this setting are depicted in Figure 7. In particular, different annotation levels of the hierarchy are illustrated, where subfigure 7a represents the root of the hierarchy with the whole object annotated as "hand", subfigure 7b represents the segmentation in different semantic (and functional) parts of the hand and subfigure 7c represents the last level of the hierarchy, containing only the nails (here the black part means a not annotated surface).

A simple example of achievable deformation is illustrated in Figure 8. Here, the translation of a specific set of control points, by performing the finger-drag gesture introduced in subsection 4.4, produces a stretching effect on a portion of the 3D model.

Finger-drag interaction can be applied even to single control points translating the single CP in different directions. On the one hand, specifying the final positions of single CPs can seem a tedious operation; on the other hand, this allows to localise more deformation effects and, in some cases, to achieve more easily the desired deformation. In addition, this type of interaction allows to obtain deformations that have been not considered in principle. An example of this behaviour is illustrated in Figure 9, where the tip of the index finger undergoes a sort of "scaling" effect.

Table 1: Description of the interaction techniques

Interaction name	Interaction subjects			Interaction description			Effect
	Models	Control points	Segments	Engage	Perform	Disengage	
Selection gaze		✓	✓	Look at a selectable component	Stare at the target	Stop staring at the target or voice command "Select"	Select one or more control points, depending on whether the target is a control point or an annotated segment
Hand-swipe	✓			Keep one opened hand with the palm facing the floor	Move the hand vertically	Close the hand	Browse annotation levels from lower to upper and vice-versa
Finger-drag		✓	✓	Extend the right index and close others fingers	Move the hand in any direction	Open the hand	Move the selected objects in any direction
Handling rotation		✓	✓	Keep opened hands with palms facing each other	Rotate hands preserving their mutual orientation	Close hands	Rotate the selected objects in any direction
Scaling	✓			Place the hand in the middle of the scene	Close the fist to reduce the model, Open the hand to enlarge it	Remove the hand from the field of view	Scale the entire 3D model

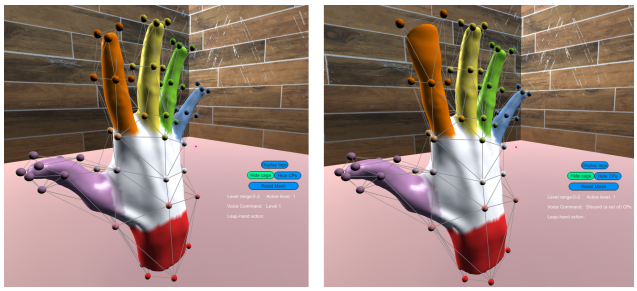


Figure 9: Deformation example applying several finger-drag translations on single CPs.

A more "complete" example of deformation is presented in Figure 10. Here, the user has asked for the visualisation of a certain level (level 1 - Figure10a) and then has selected a segment by gazing at it. This triggers the selection of the control points shown in Figure 10b (highlighted in magenta), which can be used to perform first a rotation of the control points (result shown in Figure 10c) and then a translation of the control points to reduce the distortion introduced by the operation shown in Figure 10c (result - with some minor adjustments of the single control points - shown in Figure 10d).

6 Conclusions

In this paper, we presented our approach and some preliminary results for the development of a 3D modelling system in VR, which exploits a semantic organisation of shapes to easily interact with 3D models and deform them. To our knowledge this is the first attempt to directly perform cage-based deformation in VR environments.

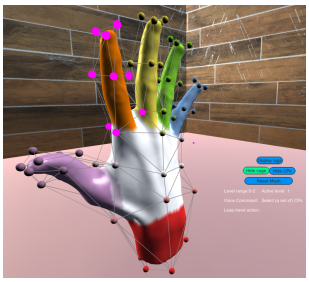
Cage-based deformation is recently achieving larger interest for authoring 3D shapes in computer graphics and animation contexts and it is here applied to semantically annotated objects. Their use in VR may allow VR application developers to perform quick shape modifications while testing their applications, without looping back to external 3D authoring tools and consequent model adaptation to VR systems. However, while cage-based deformation techniques give a lot of freedom for the manipulation of 3D shapes, they are not the best tool for dealing with articulated objects. For this reason, we are planning to introduce a dual approach in the framework for working simultaneously with cages and skeletons, as the one defined in [8].

The inclusion and exploitation of semantic annotation of meaningful object sub-parts in VR supports a more straightforward selection of the regions of interest and allows their modification in a more natural way. To this aim, gestures and voice commands have been combined to gaze selection mechanism to select and move cage control points to modify shapes possibly submitted to constraints. Even if limited in the developed functionalities, results are promising and allow us to identify possible future extensions.

The current system allows to consider only single objects, which can be deformed only within a defined space. A future extension will include the possibility of dealing with multiple objects, which can in turn be used to constrain the object deformation.

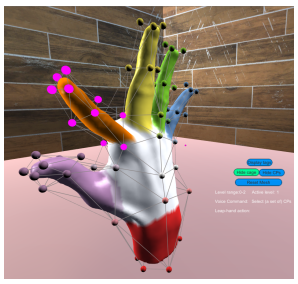
Modalities to easily specify important parameters,

(a) Initial stage of the 3D model



(b) Index finger selected

(c) Result of simple rotation



(d) Result after translation

Figure 10: Deformation result using selection of the control points associated with the "index" segment, rotation of the selected control points and translation of the selected control points, with some minor adjustments.

such as the threshold to identify the cage vertices which most influence the segments and reference elements for the translation and scaling, have to be studied .

From the modelling point of view, the main limitation is related to the use of the MVC, which provides only global deformations and can give anti-intuitive results due to the well known negativity issue (see Figure 10). For better quality results additional coordinates types will be integrated. In addition, a hierarchical specification of the cage corresponding to the annotation levels will be studied; this should provide more precise and localised deformation on the related level. The number of allowed modifications needs to be further extended. Among them, we plan to consider non-uniform scaling.

Finally, a user evaluation for assessing the usability of the system is currently in planning.

References

[1] Botsch, M., and Sorkine, O. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230.

[2] Casti, S., Corda, F., Livesu, M., and Scateni, R. Cagelab: an interactive tool for cage-based deformations. In *STAG* (2018), pp. 65–74.

[3] Casti, S., Livesu, M., Mellado, N., Abu Rumman, N., Scateni, R., Barthe, L., and Puppo, E. Skeleton based cage generation guided by harmonic fields. *Computers & Graphics 81* (2019), 140 – 151. http://www.sciencedirect.com/science/article/pii/S0097849319300457.

[4] Catalano, C., Mortara, M., Spagnuolo, M., and Falcidieno, B. Semantics and 3D media: Current issues and perspectives. *Computers & Graphics 35*, 4 (2011), 869 – 877. http://www.sciencedirect.com/science/article/pii/S0097849311000793.

[5] Chen, X., and Feng, J. Adaptive skeleton-driven cages for mesh sequences. *Comput. Animat. Virtual Worlds 25*, 3-4 (May 2014), 447–455. http://dx.doi.org/10.1002/cav.1577.

[6] Chevaillier, P., Trinh, T., Barange, M., De Loor, P., Devillers, F., Soler, J., and Querrec, R. Semantic modeling of virtual environments using mascaret. In *2012 5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS)* (2012), pp. 1–8.

[7] Cohen, M., Regazzoni, D., and Vrubel, C. 3d virtual sketching system using nurbs surfaces and leap motion controller. *Computer-Aided Design and Applications 17*, 1 (May 2020), 167–177. http://cad-journal.net/files/vol_17/Vol17No1.html.

[8] Corda, F., Thiery, J. M., Livesu, M., Puppo, E., Boubekeur, T., and Scateni, R. Real-time deformation with coupled cages and skeletons. *Computer Graphics Forum (to appear)* (2020).

[9] Cordeiro, E., Giannini, F., and Monti, M. A Survey of Immersive Systems for Shape Manipulation. *Computer-Aided Design and Applications 16*, 6 (2019), 1146 – 1157. http://cad-journal.net/files/vol_16/Vol16No6.html.

[10] Cui, J., Kuijper, A., and Sourin, A. Exploration of natural free-hand interaction for shape modeling using leap motion controller. In *2016 International Conference on Cyberworlds (CW)* (2016), pp. 41–48.

[11] Cui, J., and Sourin, A. Interactive shape modeling using leap motion controller. In *SIGGRAPH Asia 2017 Technical Briefs* (New York, NY, USA, 2017), SA '17, Association for Computing Machinery. https://doi.org/10.1145/3145749.3149437.

[12] De Luca, L., Pierrot-Deseilligny, M., Manuel, A., Chevrier, C., Lollier, B., Benistant, P., Pamart, A., Peteler, F.,

Abergel, V., and Alaoui, A. Aïoli – a reality-based 3D annotation platform for the collaborative documentation of heritage artefacts, 2018. http://www.aioli.cloud/.

[13] De Martino, T., Falcidieno, B., Giannini, F., Hassinger, S., and Ovtcharova, J. Feature-based modelling by integrating design and recognition approaches. *Computer-Aided Design 26*, 8 (1994), 646 – 653. http://www.sciencedirect.com/science/article/pii/0010448594901074.

[14] Fiorentino, M., de Amicis, R., Monno, G., and Stork, A. Spacedesign: a mixed reality workspace for aesthetic industrial design. In *Proceedings. International Symposium on Mixed and Augmented Reality* (2002), pp. 86–318.

[15] Fleisch, T., Brunetti, G., Santos, P., and Stork, A. Stroke-input methods for immersive styling environments. In *Proceedings Shape Modeling Applications, 2004.* (2004), pp. 275–283.

[16] Floater, M. S., Kós, G., and Reimers, M. Mean value coordinates in 3D. *Computer Aided Geometric Design 22*, 7 (2005), 623 – 631. http://www.sciencedirect.com/science/article/pii/S0167839605000725.

[17] Flotyński, J., and Walczak, K. Semantic multi-layered design of interactive 3D presentations. In *2013 Federated Conference on Computer Science and Information Systems* (Sep. 2013), pp. 541–548.

[18] Gravity Sketch.

[19] Jacobson, A., Baran, I., Popović, J., and Sorkine, O. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4 (July 2011). https://doi.org/10.1145/2010324.1964973.

[20] Jacobson, A., Deng, Z., Kavan, L., and Lewis, J. Skinning: Real-time shape deformation. In *ACM SIGGRAPH 2014 Courses* (2014).

[21] Jailungka, P., and Charoenseang, S. Intuitive 3D model prototyping with leap motion and microsoft hololens. In *Human-Computer Interaction. Interaction Technologies* (Cham, 2018), M. Kurosu, Ed., Springer International Publishing, pp. 269–284.

[22] Joshi, P., Meyer, M., DeRose, T., Green, B., and Sanocki, T. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3 (July 2007). http://doi.acm.org/10.1145/1276377.1276466.

[23] Ju, T., Schaefer, S., and Warren, J. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3 (July 2005), 561–566. http://doi.acm.org/10.1145/1073204.1073229.

[24] JungHyun Han, Pratt, M., and Regli, W. C. Manufacturing feature recognition from solid models: a status report. *IEEE Transactions on Robotics and Automation 16*, 6 (2000), 782–796.

[25] Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. Skinning with dual quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2007), I3D '07, Association for Computing Machinery, p. 39–46. https://doi.org/10.1145/1230100.1230107.

[26] Kil, Y. J., Renzulli, P., Kreylos, O., Hamann, B., Monno, G., and Staadt, O. G. 3D warp brush modeling. *Computers & Graphics 30*, 4 (2006), 610 – 618. http://www.sciencedirect.com/science/article/pii/S0097849306000781.

[27] Kim, H., Albuquerque, G., Havemann, S., and Fellner, D. W. Tangible 3D: Hand Gesture Interaction for Immersive 3D Modeling. In *Eurographics Symposium on Virtual Environments* (2005), E. Kjems and R. Blach, Eds., The Eurographics Association.

[28] Leap Sculpturing. https://gallery.leapmotion.com/sculpting/.

[29] LI, Y., HUANG, J., TIAN, F., WANG, H.-A., and DAI, G.-Z. Gesture interaction in virtual reality. *Virtual Reality & Intelligent Hardware 1*, 1 (2019), 84 – 112. http://www.sciencedirect.com/science/article/pii/S2096579619300075.

[30] Li, Z., Giannini, F., J.P. Pernot, Véron, P., and Falcidieno, B. Reusing heterogeneous data for the conceptual design of shapes in virtual environments. *Virtual Reality 21* (2017), 127 – 144. https://doi.org/10.1007/s10055-016-0302-z.

[31] Lipman, Y., Kopf, J., Cohen-Or, D., and Levin, D. Gpu-assisted positive mean value coordinates for mesh deformations. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), SGP '07, Eurographics Association, pp. 117–123. http://dl.acm.org/citation.cfm?id=1281991.1282007.

[32] Lipman, Y., Levin, D., and Cohen-Or, D. Green coordinates. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 78:1–78:10. http://doi.acm.org/10.1145/1360612.1360677.

[33] Lupinetti, K., Bonino, B., Giannini, F., and Monti, M. Exploring the benefits of the virtual reality technologies for assembly retrieval applications. In *Augmented Reality, Virtual Reality, and*

Computer Graphics (Cham, 2019), L. T. De Paolis and P. Bourdot, Eds., Springer International Publishing, pp. 43–59.

[34] MakeVR Pro, 2017. https://www.viveport.com/9e94a10f-51d9-4b6f-92e4-6e4fe9383fe9.

[35] Mendes, D., Caputo, F. M., Giachetti, A., Ferreira, A., and Jorge, J. A survey on 3D virtual object manipulation: From the desktop to immersive virtual environments. *Computers & Graphics Forum 38* (2019), 21 – 45. https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13390.

[36] MindeskVR, 2015. https://mindeskvr.com/.

[37] Möbius, A. F. *Der barycentrische Calcul: ein neues Hülfsmittel zur analytischen Behandlung der Geometrie.* Barth, Johann Ambrosius, 1827.

[38] Nieto, J. R., and Susín, A. *Cage Based Deformations: A Survey.* Springer Netherlands, Dordrecht, 2013, pp. 75–99. https://doi.org/10.1007/978-94-007-5446-1_3.

[39] O., D. T., F., K., B., P., and W., B. Conceptual modeling for virtual reality. In *ER '07: Tutorials, posters, panels and industrial contributions at the 26th international conference on Conceptual modelling, Darlinghurst, Australia* (Cham, 2007), Australian Computer Society, Inc., pp. 3–185.

[40] Oculus Medium. https://www.oculus.com/medium/.

[41] Park, G., Choi, H., Lee, U., and Chin, S. Virtual figure model crafting with vr hmd and leap motion. *The Imaging Science Journal 65*, 6 (2017), 358–370. https://doi.org/10.1080/13682199.2017.1355090.

[42] Scalas, A., Mortara, M., and Spagnuolo, M. A pipeline for the preparation of artefacts that provides annotations persistence. *Journal of Cultural Heritage 41* (2020), 113 – 124. http://www.sciencedirect.com/science/article/pii/S1296207418306125.

[43] Schkolne, S., Pruett, M., and Schröder, P. Surface drawing: creating organic 3D shapes with the hand and tangible tools. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI'01, ACM, NY* (March 2001), pp. 261–268.

[44] Sederberg, T. W., and Parry, S. R. Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1986), SIGGRAPH '86, Association for Computing Machinery, p. 151–160. https://doi.org/10.1145/15922.15903.

[45] Shah, J., and Mäntylä, M. *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications.* A Wiley-Interscience publication. Wiley, 1995.

[46] Shi, W., Kotoula, E., Akoglu, K., Yang, Y., and Rushmeier, H. CHER-Ob: A Tool for Shared Analysis in Cultural Heritage. In *Eurographics Workshop on Graphics and Cultural Heritage* (2016), C. E. Catalano and L. De Luca, Eds., The Eurographics Association.

[47] Soler, F., Torres, J. C., León, A. J., and Luzón, M. V. Design of cultural heritage information systems based on information layers. *J. Comput. Cult. Herit. 6*, 4 (Dec. 2013), 15:1–15:17. http://doi.acm.org/10.1145/2532630.2532631.

[48] Vinayak, and Ramani, K. A gesture-free geometric approach for mid-air expression of design intent in 3D virtual pottery. *Computer-Aided Design 69* (2015), 11 – 24. http://www.sciencedirect.com/science/article/pii/S001044851500086X.

[49] Xian, C., Li, G., and Xiong, Y. Efficient and effective cage generation by region decomposition. *Computer Animation and Virtual Worlds 26*, 2 (2014), 173–184. https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1571.

[50] Xian, C., Lin, H., and Gao, S. Automatic generation of coarse bounding cages from dense meshes. *2009 IEEE International Conference on Shape Modeling and Applications, SMI 2009* (2009), 21–27.

[51] Xian, C., Lin, H., and Gao, S. Automatic cage generation by improved OBBs for mesh deformation. *Visual Computer 28*, 1 (2012), 21–33.

[52] Xian, C., Zhang, T., and Gao, S. Semantic Cage Generation for FE Mesh Editing. *2013 International Conference on Computer-Aided Design and Computer Graphics* (2013), 220–227. http://ieeexplore.ieee.org/document/6814999/.

[53] Yang, X., Chang, J., Southern, R., and Zhang, J. J. Automatic cage construction for retargeted muscle fitting. *Visual Computer 29*, 5 (2013), 369–380.

[54] Yu, C.-H., Groza, T., and Hunter, J. Reasoning on Crowd-Sourced Semantic Annotations to Facilitate Cataloguing of 3D Artefacts in the Cultural Heritage Domain. In *The Semantic Web – ISWC 2013* (Berlin, Heidelberg, 2013), H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. Noy, C. Welty, and K. Janowicz, Eds., Springer Berlin Heidelberg, pp. 228–243.

[55] Zhang, J., Deng, B., Liu, Z., Patanè, G., Bouaziz, S., Hormann, K., and Liu, L. Local barycentric coordinates. *ACM Trans. Graph. 33*, 6 (Nov. 2014). https://doi.org/10.1145/2661229.2661255.