applied sciences

MDPI

Article

Interactive Modelling in Augmented Reality with Subdivision Surfaces and Advanced User Gesture Recognition

Alessio Cellupica [1], Marco Cirelli [1], Oliviero Giannini [2] and Pier Paolo Valentini [1,*]

[1] Department of Enterprise Engineering, University of Rome Tor Vergata, via del Politecnico 1, 00133 Rome, Italy; alessio.cellupica@uniroma2.it (A.C.); marco.cirelli@uniroma2.it (M.C.)
[2] Department of Engineering, University Niccolò Cusano, via Don Carlo Gnocchi 3, 00166 Rome, Italy; oliviero.giannini@unicusano.it
* Correspondence: valentini@ing.uniroma2.it

Abstract: The paper discusses an integrated methodology to implement an interactive augmented reality 3D modelling environment with natural interaction, empowered by real-time gesture recognition. The methodology is developed from a geometry-sculpting algorithm based on the use of the subdivision surfaces approach to combine the ease and versatility of interactive modelling even of complex shapes, while maintaining high geometric continuity and smoothness. The interaction with the deformable elements of the geometry's control cage to be divided uses an optimised version of the Grasp Active Feature/Object Active Feature algorithm developed from hand tracking and gesture recognition based on zero-invasive stereo-infrared techniques. Modelling, combined with an augmented reality environment, allows the modification of geometries having real objects as a reference and, in any case, a general spatial awareness during activities. The methodology was implemented and tested using an advanced mixed-reality headset, the Varjo XR-4, with hi-resolution pass-through and a second-generation Ultraleap for accurate and precise hand tracking.

Keywords: augmented reality; natural interaction; interactive modelling; Varjo; subdivision surfaces; Ultraleap

Citation: Cellupica, A.; Cirelli, M.; Giannini, O.; Valentini, P.P. Interactive Modelling in Augmented Reality with Subdivision Surfaces and Advanced User Gesture Recognition. *Appl. Sci.* **2024**, *14*, 11873. https://doi.org/10.3390/app142411873

Academic Editor: Chaman Sabharwal

Received: 13 November 2024
Revised: 16 December 2024
Accepted: 18 December 2024
Published: 19 December 2024

1. Introduction

In recent years, virtual reality (VR) and augmented reality (AR) have been established as revolutionary technologies capable of profoundly transforming a wide range of industries due to their innovative potential. They are now redefining the future of the academy and industry, becoming essential tools to improve education, production processes, design, and business operations. AR was introduced in Industry 4.0 as a key-enabling technology, and with the advent of the fifth industrial revolution (Industry 5.0), which aims to create a more human-centred work environment [1], promoting the collaboration between humans and machines [2,3], technologies like VR and AR are taking on a more strategic role. In this context, VR and AR serve as crucial tools to improve human–machine interaction [4,5], offering immersive experiences that support advanced training [6,7], assisted maintenance [8,9], real-time problem-solving [10,11], and interactive modelling [12].

One particularly promising area for the application of VR and AR is their integration with Computer-Aided Design (CAD), where these immersive technologies enhance **how** designers and engineers create, interact with, and refine complex 3D models. The desire to merge 3D modelling tools, especially with AR [13], derives from the numerous advantages these tools offer [14], particularly when multimodal interfaces are available to interact with the software [15]. Direct interaction with the real world provides users with immediate spatial awareness of the dimensions of the model they are creating. Additionally, enabling multiple users to access a single 3D model simultaneously supports collaboration among designers, engineers, and other stakeholders, allowing them to view and interact with the project in real time, even in different locations. Project meetings and reviews become

more efficient, as everyone can share a common vision of the model and suggest changes instantly. The ability to link real objects with the virtual model allows designers to see how products or structures integrate into the physical context. This helps to identify issues related to space, proportions, or interactions with the surrounding environment in a more intuitive way than simple on-screen visualisation, thereby reducing the likelihood of errors.

The current state of the art presents several cases of implementing 3D modelling in VR/AR, each differing in terms of the type of mathematics used for solid modelling, the modelling method, or the framework used to provide input to the program. The use of a tilted interactive high-resolution display with multi-touch and pen input has enabled the development of immersive 3D modelling, allowing users to modify a primitive geometry through standard operations or model objects using extrusion or revolution from a 2D sketch [16]. Due to significant advances in consumer-grade AR and VR hardware and software, it is now possible to sketch 3D curves directly in mid-air. Several software tools have been developed based on generating surfaces through 3D sketches, using controllers [17,18], motion-capture pens [19], hand tracking [20], or trackers and specialised VR gloves to monitor finger trajectories or hand movements [21]. Many of these conceptual design methods, such as sketch-based interfaces, require designers to construct all the surface edges before visualising the final surface as for a Boundary-Representation (B-Rep) approach. Other approaches use mesh sculpting [22], allowing users to model 3D objects from a rough geometry as if sculpting clay [23]. For these applications, techniques such as Octree [24], Voxel [25], and polygonal mesh [26] modelling have been used. In addition to these algorithms, other methods have been developed to facilitate sculpting, including those based on spring-mass systems using [27], mesh morphing [28], and Non-Uniform Rational B-Splines (NURBS) formulations [29]. Some of these applications have also included haptic devices to increase users' immersion in the virtual scene. Mesh sculpting is well-suited for creating complex and organic surfaces; however, it is not ideal for precision modelling. To address this, precision CAD models have been integrated with virtual and AR tools, using solid modelling algorithms such as B-Rep [30] and Constructive Solid Geometry (CSG) [31]. B-Rep defines a solid by its geometric boundaries using NURBS [32] curves and surfaces, while CSG builds complex solid objects by applying topological operations to primitive solids like cubes, spheres, and cylinders [33]. In Malik et al. [34], haptic gloves were introduced for force feedback, providing a much more immersive virtual experience, and deep learning was applied for more intuitive reverse engineering based on CSG algorithms. Fuge et al. [35] defined a procedure for conceptual design and freeform modelling using hand gestures based on NURBS surfaces. The use of hand gestures enhances the interactivity and intuitiveness of the application [35,36].

Despite the state-of-the-art applications for qualitative modelling through mesh sculpting and precision CAD modelling using NURBS, all these methodologies may be inadequate where there is a need to model complex organic shapes with geometric accuracy. To combine the advantages and versatility of mesh sculpting with the mathematical accuracy and generalisation of the B-Rep approach using NURBS, freeform modelling through subdivision surfaces using refinement algorithms can be employed [37]. Subdivision surface algorithms are able to create smooth, high-resolution surfaces by iteratively refining a coarse polygonal mesh (control cage). The user acts on these control cages, relocating its entities (points, edges, and faces) or adding/removing entities, and the final surface is changed accordingly. These approaches are increasingly widespread and included in computer-aided modelling environments, becoming an alternative to history-based parametric modelling. Subdivision surface modelling has the extreme advantage of simplifying the modelling of complex organic shapes, but it is limited in creating exact functional fit surfaces (such as cylindrical holes, cylindrical pins, and flat surfaces). However, the complete modelling procedure always involves a final phase of modifications and detail of the geometries built with feature-based approaches.

The great advantage compared to sculpting methodologies based on polygonal meshes is that a recursive subdivision scheme can tend to surface patches with high geometric

continuity up to G^2 (curvature continuity), allowing a final conversion into NURBS patches and a direct integration with geometries modelled with the standard B-Rep approach.

Another important aspect of modelling complex shapes is the ability to act on the geometry to be sculpted through direct gestures, without the use of wearable devices, controllers, or pointers, to directly accept the user's intent, and transform it into actions. This approach is often called natural interface and has gained relevance in both the scientific and industrial communities, especially for assisted assembling and interactive simulations [38,39].

From this background, the study aims to develop a novel integrated methodology, implementing an immersive modelling environment based on freeform modelling through subdivision surfaces in AR, and implementing an ad hoc natural interface based on hand tracking and advanced gesture recognition. The sculpting algorithm implemented in the methodology is based on adapting the well-known Catmull–Clark surfaces algorithm. It combines the effectiveness of the direct modelling approach with the capability of post-processing the outcome of the modelling with a B-Rep geometrical kernel to finalise the design into an exact NURBS patch. The interaction with the geometry to be sculpted is implemented as an extension of the Grasping Active Feature/Object Active Feature methodology (GAF/OAF) [40], adapted for the grasping and manipulating of subdivision control cage entities. Therefore, the user interacts naturally with these entities, and the subdivision algorithm updates the shape in real time.

The paper is structured as follows: the initial section outlines the general idea of the methodology, discussing the integration among different aspects. In the second part, the details of the modified subdivision surface algorithm and the adapted GAF/OAF methodology are presented. In the last part, a comprehensive case study is developed, discussing all the implementation aspects using a Varjo XR-4 mixed-reality headset, Ultraleap Motion Controller for hand tracking, and the integration with the proposed modelling algorithms.

2. Natural Interactions

In the modelling of complex shapes using subdivision surfaces, the user must manipulate virtual objects that identify the control cage to modify the resulting surface. The natural interaction between the user and these virtual objects is essential for achieving high interactivity. Effective interaction requires hand tracking, simple gestures, natural movements, and robust and easily recognisable actions. Implementing natural interfaces based on hand tracking and advanced gesture recognition enhances intuitiveness and accuracy [41]. Hand tracking is crucial for enabling this interaction in augmented and virtual reality applications. The human hand consists of 24 main joints: each finger has four joints (knuckle, proximal, intermediate, and distal) and five additional joints for the palm and wrist. Detecting the position and rotation of these joints enables recognition of the overall hand conration and allows for the interpretation of complex gestures. The precise recognition of hand movements relies on advanced techniques, including machine learning algorithms, computer vision, and depth sensors. These techniques capture three-dimensional information, accurately detecting and mapping the hand's position and movements in space.

2.1. The Adapted GAF/OAF Methodology

The interaction between the user and the entities of the control cage takes place through customised gestures. The control cage comprises three types of entities: points, edges, and faces; the latter can be identified by their centroid. For point-like or small-radius spherical entities, the pinch pose is the most suitable as it represents the natural gesture for picking them; for linear or cylindrical entities with a small base radius, the cylindrical pose is the most suitable.

To recognise poses, it is necessary to track the position and orientation of each joint and bone of the hand. A pose is recognised if the relative attitude between adjacent bones is within a tolerance range. Once a pose is recognised, it may be used for picking and moving objects by imposing virtual constraints between the real hand and the virtual

entities. In [40], the use of the GAF/OAF methodology is presented as a robust and straightforward approach able to be easily implemented in both modelling and simulation environments. In particular, a modified version of the methodology is proposed to better suit the type of entities to be picked and moved, by defining virtual constraints that enhance grip stability and robustness. According to the GAF/OAF methodology, for each type of pose, two reference frames are defined: one on the hand (representing the Grasping Active Feature—GAF) and the other on the object to be grasped (representing the Object Active Feature—OAF). For the cylindrical grip, the reference frame on the hand includes the origin $\mathbf{O}_{CG}$ and the direction $\mathbf{z}_{CG}$ of the grasping axis, while the reference frame on the cylindrical object includes the origin $\mathbf{O}_{Cyl}$ and the axis of the cylindrical surface $\mathbf{z}_{Cyl}$. As shown in Figure 1a, the cylindrical grip virtual constraint can be imposed if both the coincidence between the origins Equation (1) and the alignment between the axes Equation (2) are fulfilled within predefined tolerances ($t1$ linear tolerance and $t2$, angular tolerance):

$$\left\| \mathbf{O}_{CG} - \mathbf{O}_{Cyl} \right\| \leq t1 \tag{1}$$

$$1 - \left| \mathbf{z}_{CG} \cdot \mathbf{z}_{Cyl} \right| \leq t2 \tag{2}$$

Equation (1) describes a spherical zone of tolerance and Equation (2) a cylindrical one. These constraints are different from the original GAF/OAF implementation since they can explicitly express the angular tolerance of the alignment between the cylindrical axes, which is a condition easier to be checked and integrated for real-time monitoring. For the pinch grip, the fundamental elements of the two reference frames on the hand and on the virtual object are the origins $\mathbf{O}_{PG}$ e $\mathbf{O}_{Sph}$, respectively. Figure 1b shows the spherical tolerance, described by Equation (3), within which the constraint can be activated.

$$\left\| \mathbf{O}_{PG} - \mathbf{O}_{Sph} \right\| \leq t1 \tag{3}$$

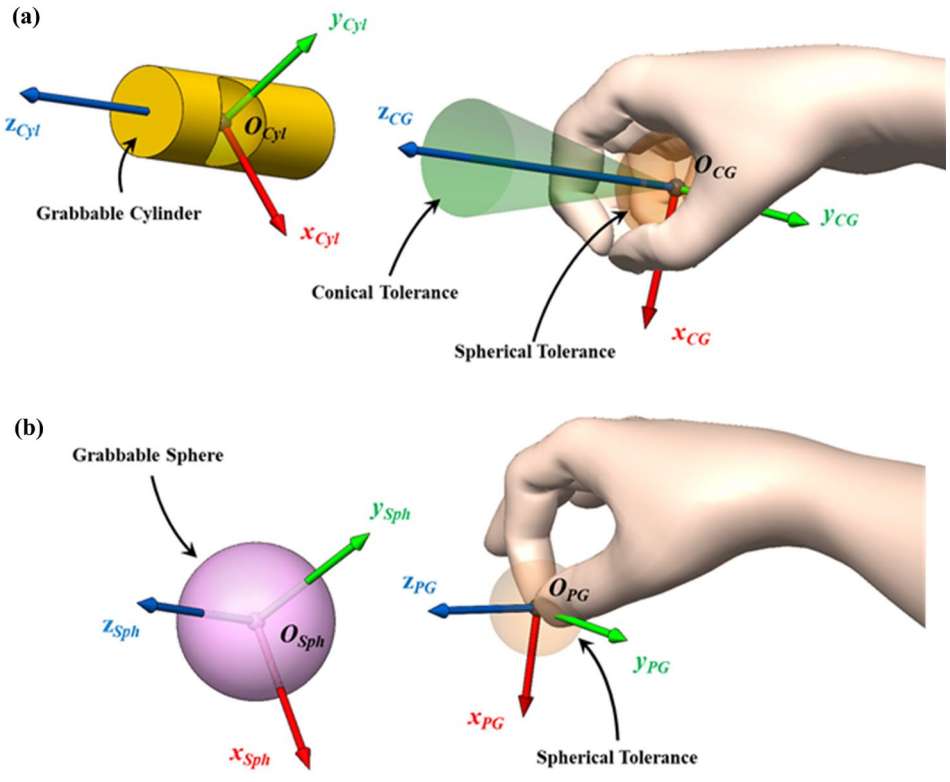

Figure 1. GAF/OAF methodology adapted for (**a**) cylindrical grasp and (**b**) pinch grasp.

The framework that manages the user's intention to grasp is structured as follows:

- The type of hand pose expresses the user's intention to grasp the object as well as the type of object. The corresponding reference frame (GAF) is activated.
- The relative position and orientation between the activated GAF and the OAFs of the objects that can be grasped are checked.
- If an OAF meets the corresponding tolerances, the grasp is confirmed, and a constraint condition between the two reference frames is applied.
- The object is released when the hand pose changes.

The strategy for implementing interactive manipulation of the various geometric entities is based on a procedural scheme in which the user first selects the entities and then modifies their geometric features. With reference to Figure 2, during the AR experience, the hand-tracking system is always active. When the reference pinch pose is recognised, and the proximity GAF/OAF constraints are satisfied for a geometrical grabbable entity (vertex, edge of face), the entity is added to the selection list. The user can continue to select other objects by repeating the pinch pose in proximity to other entities, satisfying the GAF/OAF constraints simultaneously. The selection ends when the thumb-up pose is recognised. If the user uses the thumb-down pose, the selection is cancelled. Once the entities are selected, the user can act on them, depending on another chosen pose. If the user chooses the pinch pose, the selected entities will be translated according to the pinch point displacement. If the user chooses the open palm pose, the entities will be extruded following the tracked displacement of the centre of the palm. The modification ends when the user switches the pose to thumb-up, accepting the change, or thumb-down, rejecting the change. In a very similar way, it is possible to proceed in the case of rotating the entities by first selecting them with the simultaneous fulfilment of the fist pose and the GAF/OAF cylindrical constraints and then changing the palm orientation which constrains the attitude of the selected entities. The modification ends when the user switches the pose to thumb-up, accepting the change, or thumb-down, rejecting the change. In general, the recognition of the thumb-down pose always cancels any active commands. With this logic, it is also possible to expand the sequence of operations while maintaining a completely natural and intuitive interaction with the geometry to be modelled.

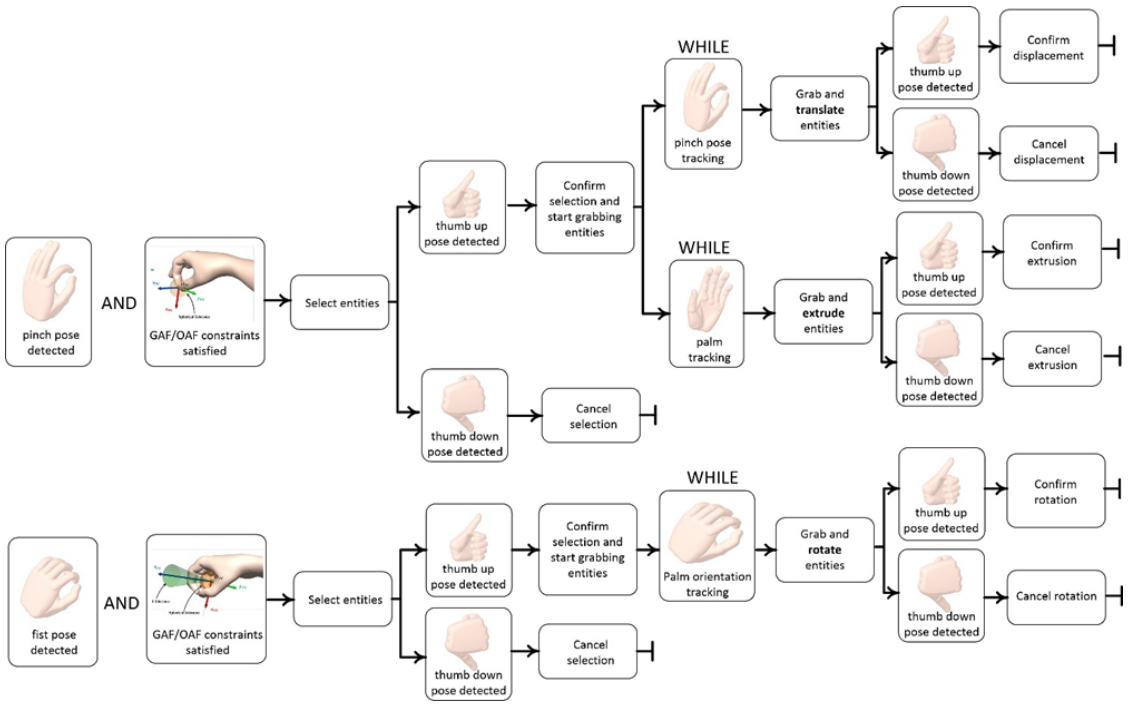

Figure 2. Pose recognition and interactive manipulation scheme.

The procedure is compatible with the use of different hand-tracking devices and implementations of different libraries and is to be considered completely general. However, in the description of the case study in the next section of the manuscript, hand tracking and pose recognition will be specialised for a specific device.

2.2. Catmull–Clark Refinement Algorithm Adapted for Triangular Meshes

The subdivision surface approach is a technique in Computer-Aided Design used to create smooth, high-resolution surfaces by iteratively refining a coarse polygonal mesh. At each subdivision step, new vertices are introduced, and existing vertices are repositioned according to specific rules to smooth out sharp edges and corners. This process produces a limit surface that appears smooth and organic, while the initial mesh provides control over the surface's general shape.

During the last decades, several refinement algorithms have been proposed. The most commonly used are the Catmull–Clark [42], Doo–Sabin [43], and Loop [44] algorithms. For the present study, the Catmull–Clark algorithm is chosen due to its flexibility, generalisation, and straightforward implementation. The Catmull–Clark algorithm is designed to work with polygonal meshes composed mainly of quadrilaterals, and it generates limited surfaces with G^2 continuity everywhere and G^1 at extraordinary vertices. The algorithm operates iteratively, with each iteration refining the surface by adding new vertices based on the positions of existing ones, as well as the surrounding edges and faces. The number of iterations directly influences the smoothness and detail of the resulting surface: as the iterations increase, the surface converges towards a smoother approximation of the desired geometry (Figure 3a).

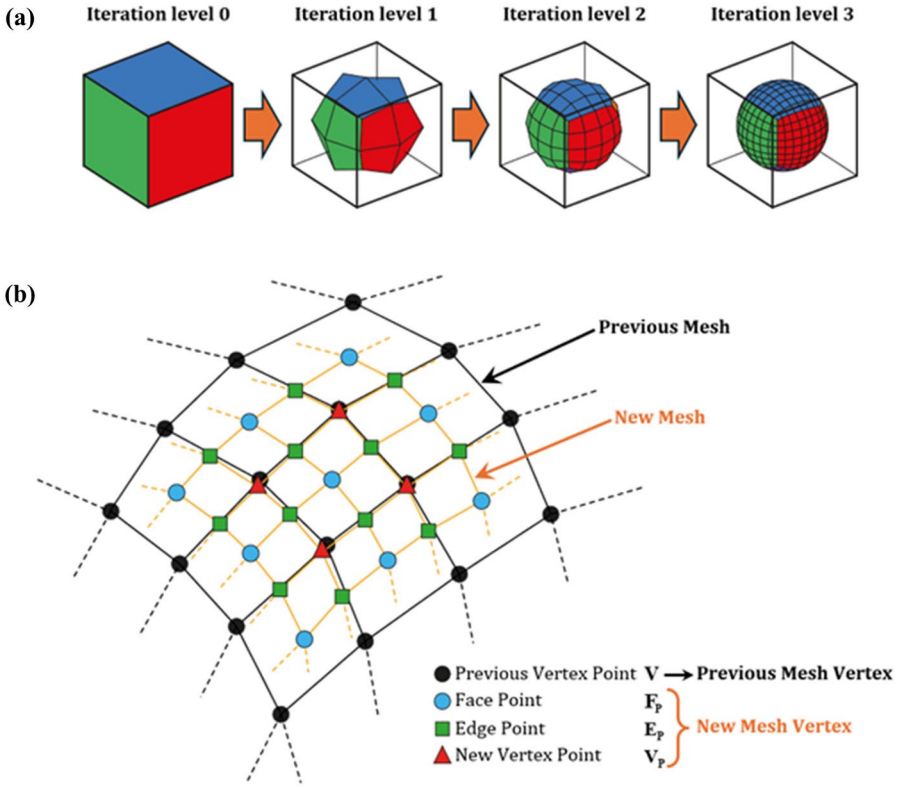

Figure 3. (**a**) Catmull–Clark iteration levels in the recursive procedure, (**b**) an iteration of the Catmull–Clark refinement algorithm.

According to the original algorithm, here summarised for the sake of completeness, at each iteration, the previous mesh is subdivided by calculating the positions of the new vertices based on the vertices of the previous mesh. For each face of the mesh, its midpoint (Face Point) $\mathbf{F_P}$ is calculated as the average of the coordinates of the vertices that form

the face. For each edge of the mesh, the Edge Point $\mathbf{E_P}$ is calculated as the average of the coordinates of the two vertices $\mathbf{V}_1$ and $\mathbf{V}_2$ that define it, along with the two Face Points $\mathbf{F_{P1}}$ and $\mathbf{F_{P2}}$ of the adjacent faces:

$$\mathbf{E_P} = \frac{1}{4}(\mathbf{V_1} + \mathbf{V_2} + \mathbf{F_{P1}} + \mathbf{F_{P2}}) \tag{4}$$

For each edge, the midpoint $\mathbf{E}$ is calculated as the average of the adjacent vertices. For each generic vertex $\mathbf{V}$ of the previous mesh, the average $\mathbf{E_M}$ of the m midpoints of the adjacent edges is calculated as follows:

$$\mathbf{E_M} = \frac{1}{m}\sum_{i=0}^{m} \mathbf{E}_i \tag{5}$$

and the average $\mathbf{F_{PM}}$ of the m Face Points $\mathbf{F_P}$ adjacent to the vertex is calculated as follows:

$$\mathbf{F_{PM}} = \frac{1}{m}\sum_{i=0}^{m} \mathbf{F_{P}}_i \tag{6}$$

For each vertex of the previous mesh, its new position $\mathbf{V_P}$ is calculated as the weighted average of the corresponding $\mathbf{F_{PM}}$, $\mathbf{E_M}$ and the vertex $\mathbf{V}$ itself:

$$\mathbf{V_P} = \frac{1}{m}(\mathbf{F_P} + 2 \cdot \mathbf{E_M} + (m-3) \cdot \mathbf{V}) \tag{7}$$

The new mesh is generated by connecting the Face Points with the Edge Points and linking the New Vertex Points to the Edge Points. Figure 3b shows one iteration level, highlighting the previous mesh, the Face Points, the Edge Points, the new positions of the Vertex Points, and the connections between them that generate the new mesh.

The Catmull–Clark algorithm is optimised for polygonal meshes composed mainly of quadrilaterals, but it has to be adapted to deal with triangular meshes to achieve general and robust results. In fact, in VR and AR applications, triangular meshes are preferred for real-time rendering efficiency [45]. Triangular meshes offer superior computational performance and are best suited for implementing models that need to be able to be viewed on standalone devices with limited hardware resources. For this purpose, the Catmull–Clark algorithm can be adapted for VR and AR applications by converting the triangular mesh into a quadrilateral mesh where possible. The subdivided quadrilateral mesh must then be reconverted into a triangular mesh, as triangular meshes are optimised for faster rendering and better performance in real-time applications. The conversion from triangular to quadrilateral mesh, as well as the subsequent reconversion from quadrilateral to triangular mesh after refinement, must be carefully controlled, with a specific focus on the diagonals of the quadrilateral faces. In transitioning from the initial triangular mesh to the initial quadrilateral mesh, each pair of triangles highlights a diagonal of the corresponding quadrilateral face. After the subdivision process, to achieve the desired result, the conversion from quadrilateral to triangular faces in the final stage should ensure the alignment of the directions between the diagonal identified by each pair of final triangles and the diagonal of the initial generator pair (Figure 4). In particular, the control of the direction of the diagonals of each pair of triangles, necessary for using the Catmull–Clark algorithm adapted to triangular meshes, is based on the use of base cubes with face indexing (Figure 5a). The face indexing F_{ID} allows for the identification of the reference frame RF of the faces and their types (front-face or back-face). Adjacent faces of different base cubes are defined by the same F_{ID}. Each face is described by an F_{Num}, which consists of a set of indices corresponding to the vertices that compose it. The order of these indices is determined by the corresponding RF. For faces indexed with F1 and F4, the order of the indices follows the x-axis first and then the y-axis; for faces indexed with F2 and F5, the order follows the y-axis first and then the z-axis; and for faces indexed with F3 and F6, the

order follows the z-axis first and then the x-axis. For each quadrilateral face, the pairs of triangles can be generated using the F_{Num} and the type. Two lists of information describe the mesh of a triangle: the first contains the spatial coordinates of the three vertices, while the second specifies the order of these vertices. This order is fundamental for managing back-face culling [46]. To extract the second list of information, two arrays of integer are used, which contain the positions of the vertices in the F_{Num}. The array in Equation (8) is used for front-faces, while the array in Equation (9) is used for back-faces.

$$\mathbf{v}_{Front} = \begin{bmatrix} 0 & 2 & 3 & 0 & 3 & 1 \end{bmatrix} \tag{8}$$

$$\mathbf{v}_{Back} = \begin{bmatrix} 0 & 3 & 2 & 0 & 1 & 3 \end{bmatrix} \tag{9}$$

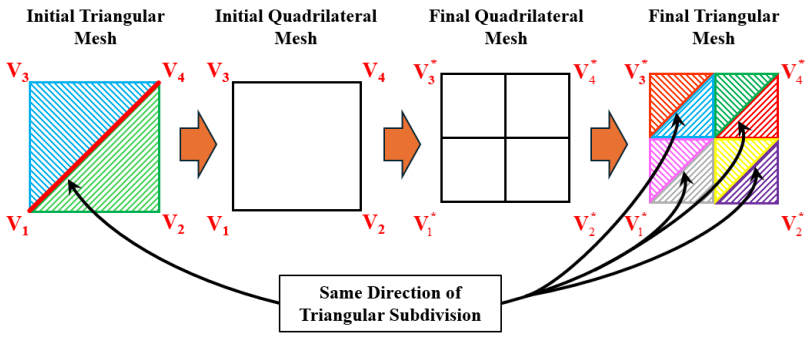

Figure 4. Catmull–Clark algorithm adapted for triangular meshes preserving the diagonal direction when transitioning between triangular and quadrilateral meshes.

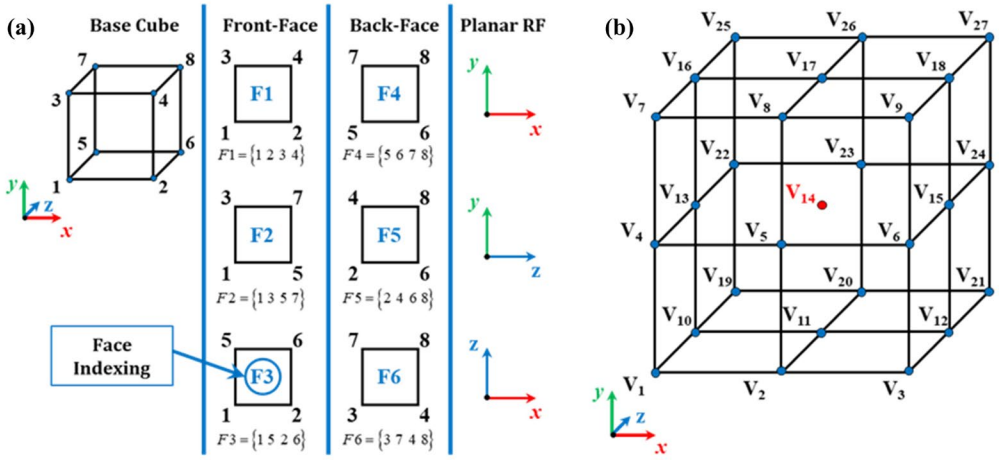

Figure 5. (**a**) Face indexing for each face of the base cube created in the application with the left-handed reference frame, and (**b**) the vertex naming order in the generic mesh, which allows the adaptation of the Catmull–Clark algorithm and the removal of not-showed vertices (red).

The first three indices identify the first triangle, and the last three identify the second triangle of the pair. These arrays are invariant under cyclic rotations for the two groups of three elements, as each cyclic permutation of each group preserves the geometric structure of the triangles defined by the specified vertices. By extracting from F_{Num} the indices at the positions specified by the vectors $\mathbf{v}_{Front}$ and $\mathbf{v}_{Back}$, a new array $\mathbf{T_{FV}}$ is obtained for each face if it is a front-face, or $\mathbf{T_{BV}}$ if it is a back-face. The arrays $\mathbf{T_{FV}}$ and $\mathbf{T_{BV}}$ contain the order of the vertices needed to generate each pair of front and back triangles, respectively. These arrays contain two pairs of repeating vertices: by using nested loops, it is possible to view all the combinations and generate the integer array that defines the quadrilateral face.

The conversion from a quadrilateral mesh to a triangular mesh is achieved by starting from new arrays of four integers that identify the indices of the new vertices describing

each quadrilateral face. By using the integer array in Equation (10), the quadrilateral face is decomposed into a pair of triangular faces.

$$\mathbf{v_t^q} = \begin{bmatrix} 0 & 1 & 2 & 0 & 2 & 3 \end{bmatrix} \quad (10)$$

The enumeration of the vertices of the initial mesh is carried out by iterating over all combinations of coordinates along the x, y, and z axes, using nested loops arranged in this order, from the innermost to the outermost, generating a three-dimensional grid. The vertices inside the control solid are eliminated (red vertex in Figure 5b).

2.3. Modelling Functions

In addition to manipulating the initial mesh, a face extrusion function for one or more base cubes has been implemented. The main challenge lies in generating a triangular mesh after extruding quadrilateral faces, while preserving the nomenclature required for the adapted Catmull–Clark algorithm. Using the F_{ID} values and face types allows for a straightforward procedure to generate the triangular mesh following the extrusion function. When adding new base cubes that do not belong to the initial mesh, the ordered vertex enumeration using nested loops is no longer valid, thus requiring an algorithm that goes beyond ordered vertex enumeration. In this process, each F_{ID} must be considered individually (Figure 6a). The first step is to generate four new vertices which, combined with the four vertices of the initial face, form the new base cube. The face defined by these new four vertices retains the same F_{ID} and face type as the initial face. The $\mathbf{F_{Num}}$ of this face must be defined in accordance with the RF of the corresponding F_{ID}. The opposite face, with an opposite F_{ID}, is a not-shown face, as it coincides with the initial face. Consequently, the initial face also becomes a not-shown face (Figure 6c). The F_{ID} and the face type of the remaining faces of the new base cube depend on the adjacency to the initial base cube: adjacent faces of different base cubes share the same F_{ID} (Figure 6b). Regarding the $\mathbf{F_{Num}}$, they must also be defined in accordance with the RF of the corresponding F_{ID}. If there are multiple adjacent initial faces, the procedure is similar, with a difference concerning the last four faces. If there are two faces belonging to different new base cubes that share the same $\mathbf{F_{Num}}$ and opposite F_{ID}, these faces are not shown.

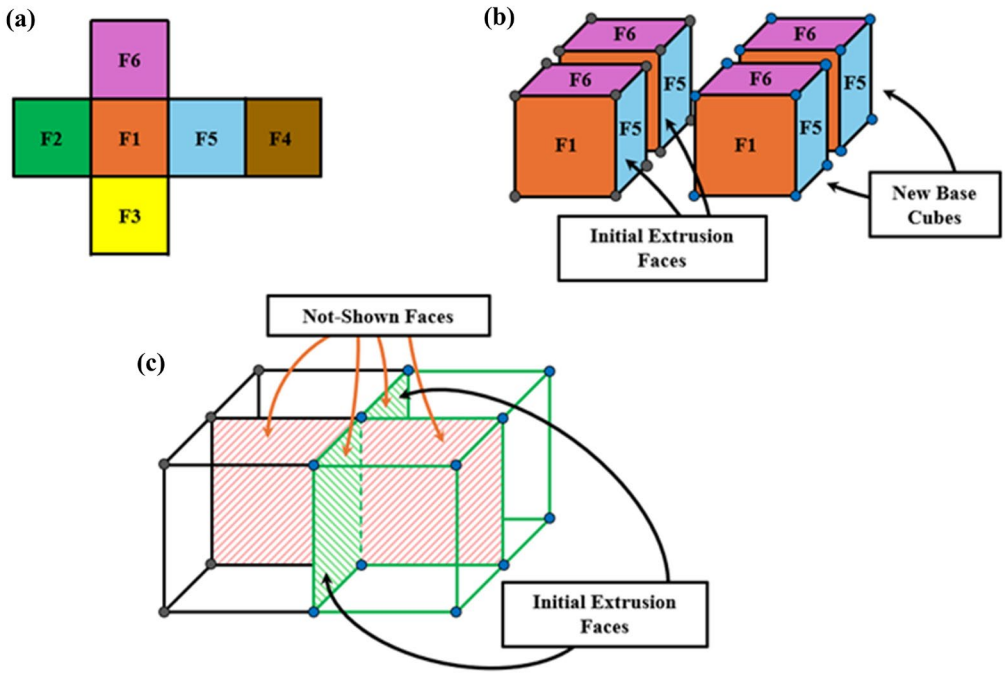

Figure 6. (**a**) Base cubes face indexing, (**b**) faces indexing of the new base cubes for an extrusion from F5 faces, and (**c**) not-shown faces after the extrusion operation.

3. Case Study: Interactive Modelling Implemented Using Varjo XR-4 and Ultraleap Motion Controller

The case study concerns an example of modelling an organic shape in AR mode developed using the Unity platform distributed by Unity Technologies, which allows you to directly implement the algorithms discussed in the previous sections through C# scripts. Unity was chosen over alternative platforms such as Unreal Engine by Epic Games and Nvidia Omniverse due to the availability of SDK and libraries for interfacing different devices and the straightforwardness and versatility in writing and integrating code scripts.

3.1. Implementation Details

The headset chosen for the implementation of the case study is the Varjo XR-4. It is a high-end mixed-reality helmet equipped with two mini-LED displays with a resolution of 3840 × 3744 at 51 PPD and a 90 Hz refresh rate. It is equipped with 20 MP pass-through cameras. The spatial awareness is achieved by a 300 KPix LiDAR with 7 m range. It does not have its own hand-tracking system, but it can be achieved by using 2nd generation Ultraleap's Leap Motion Controller to be integrated with a proprietary mounting kit. This device has two infrared cameras and multiple infrared LEDs. For each hand, the controller is able to track 24 joints, returning the spatial coordinates of 4 finger joints for each of the 5 fingers (knuckle, proximal, intermediate, and distal) and 5 additional joints for the palm and wrist. The 2nd generation device has improved performance with respect to its 1st generation that was extensively used and assessed by the scientific community [38,41]. The manufacturer provides development libraries that allow direct access to the tracking information that becomes necessary to be able to apply the algorithm and procedures described in the previous sections.

The great advantage in using the Varjo XR-4 is not only in the high technical specifications but above all in the availability of support libraries (Varjo XR Plugin and Varjo SDK) made available by the manufacturer that allow extensive and complete access. The libraries have several extremely useful features for customising mixed-reality environments and allow almost complete control of the potential offered by the instrumentation. More specifically, the Varjo XR Plugin gives access to the *VarjoMixedReality* class to manage the Varjo interface and to the *VarjoRendering* class to manage rendering options.

The AR functionality of the Varjo XR-4 can be enabled with the following two scripted commands:

VarjoMixedReality.StartRender();
VarjoRendering.SetOpaque(false);

The first line starts the rendering of the scene on the two displays and the second line enables the pass-through (rendering of the video frame acquired by the cameras). To ensure high visual realism, it is advisable to take into account depth occlusion [39] which allows the correct stacking between real objects and virtual objects, preventing the overlapping of virtual objects if they are at a greater distance from real objects than the observer's point of view. This verification can be managed directly from the libraries supplied with the Varjo by activating the calculation of the depth map calling the respective methods in the *VarjoMixedReality* class by using the script function following:

VarjoMixedReality.EnableDepthEstimation();

The calculation of depth occlusions is then performed using data from the lidar sensor. Although the depth occlusion is computationally demanding, it has a great contribution to the realism of the scene. However, note that how you enable mixed reality can vary from headset to headset, and not all headsets directly support depth map calculation, which is a quite time-consuming activity.

As for hand tracking, Ultraleap Motion Controller has a library that allows full access to both raw data (position and attitude of hand bones and joint) or elaborated data (pose and gesture estimation). In the case of the example, we extracted and processed raw data that are more suitable for implementing the adapted GAF/OAF methodology. The communication with the controller is managed by the *Leap Service Provider* that is able

to retrieve synchronous information that are stored in a *frame* data structure. For each acquisition frame, the *frame* object is updated and then the information can be extracted by the corresponding methods and properties. For example, it is possible to extract the vector of the recognised hands which contains 5 fingers which contain information on the connecting structures (bones and joints). Here are some of the C# commands that enable access to the data of the right and left recognised hands, using the Leap namespaces:

- to access the hand information from an acquired frame:

  Left hand → **Hand** leftHand = frame.**GetHand**(**Chirality.Left**);
  Right hand → **Hand** rightHand = frame.**GetHand**(**Chirality.Right**);

- to access the coordinates of the *j*-th finger of the right hand (0 = thumb, 1 = index, 2 = middle 3 = ring, 4 = pinky):

  Vector3 tipCoordinate = rightHand.**fingers**[j].**TipPosition**;

- to access the coordinates of the joint of the *k*-th bone of the *j*-th finger of the right hand (0 = metacarpal, 1 = proximal, 2 = intermediate, 3 = distal)

  Vector3 jointCoordinates = rightHand.**fingers**[j].**bones**[k].**NextJoint.ToVector3**();

- to access the angle between two adjacent bones (bone1 and bone2) on *j*-th finger of the right hand:

  Bone bone1 = rightHand.**fingers**[j].**bones**[k1];
  Bone bone2 = rightHand.**fingers**[j].**bones**[k2];
  float angle = **Vector3.Angle**(bone1.**Direction.ToVector3**(),bone2.**Direction.ToVector3**());

- to access the coordinates of the middle of the palm or the right hand:

  Vector3 palmPosition = rightHand.**PalmPosition**;

- to access the normal vector of the palm of the right hand:

  Vector3 palmAxis = rightHand.**PalmAxis**;

- to access the coordinates of the pinch point of the right hand:

  Vector3 pinchPoint = rightHand.**GetPinchPosition**();

The recognition of the hand pose is achieved by comparing the angle between adjacent bones with those of a reference pose, within a specific tolerance. For the case study, we used the *Pose Detector* object provided in the Ultraleap's library for the direct comparison with standard poses (pinch, fist, thumb-up, thumb-down).

The initial control cage is generated as a shape of a rectangular box of $0.5 \times 0.5 \times 0.5$ m dimensions. The parameters n_x, n_y, n_z specifying the number of cage subdivisions along the three main axes are chosen as 5, 5, 5. The number of vertices N_{Vertex} defining the initial control cage is calculated as the difference between the total vertices N_{Vertex}^{Tot} generated by the nested loops and the hidden vertices N_{Vertex}^{NS}, which are internal to the cage itself:

$$N_{Vertex} = N_{Vertex}^{Tot} - N_{Vertex}^{NS} = (n_x + 1)(n_y + 1)(n_z + 1) - (n_x - 1)(n_y - 1)(n_z - 1) \quad (11)$$

The number of edges N_{Edge} and the number of faces N_{Face} of the control cage are expressed by Equation (12) and Equation (13), respectively.

$$N_{Edge} = 2n_x(n_y - 1) + 2n_x(n_z + 1) + 2n_y(n_z - 1) + 2n_y(n_x + 1) + 2n_z(n_x - 1) + 2n_z(n_y + 1) \quad (12)$$

$$N_{Face} = 2n_x n_y + 2n_y n_z + 2n_x n_z \quad (13)$$

The vertices and the centroids of the faces are represented by spheres, while the edges are identified by cylinders. The assignment of the F_{Num} and F_{ID} to each face is managed using the parameters n_x, n_y, n_z, and the parameters δ_x, δ_y, δ_z, which indicate the variation in the indices of adjacent vertices along the x, y, and z directions, respectively.

Figure 7 shows an initial photo (a) and 6 frames captured by the Varjo XR-4 cameras during the AR experience of the case study. In the first step (b), the user explores the virtual

scene and locates the starting geometry collimating it to the real environment. In the first frame (b), it is possible to see how depth occlusion with the user's hands is active, which returns great realism. The control cage is depicted in blue, the control vertices in red, the edges of the cage in blue, and the subdivided surface in red. Points and lines are depicted as three-dimensional geometries in order to give a spatial awareness. Green spheres indicate the centres of the control cage faces to facilitate the selection and grabbing. In the second step, the user selects the various vertices of the control cage on which he imposes various movements to sculpt the final geometry. The frame (c) shows an instant when the user selects a vertex. There is also the persistence of depth occlusion which facilitates the selection of the correct entity. The third frame (d) captures the user performing an interactive movement of the vertex and it is clearly visible how the geometry described by subdivision surfaces follows the movement. The fourth frame (e) shows an example of extruding a face of the control cage with the tracking of the open palm. In the next frames, (f) and (g), the final geometry is visible, with and without the control cage, respectively.

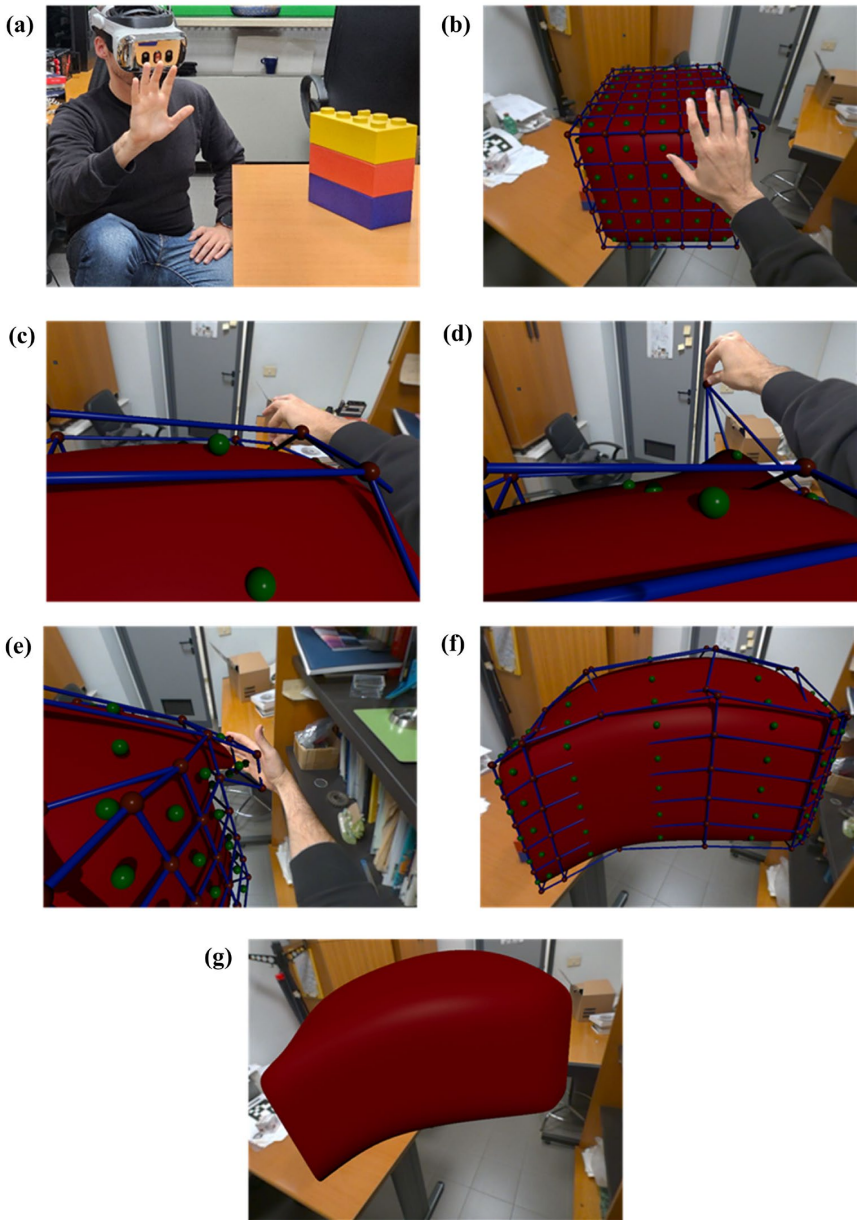

Figure 7. Snapshots from the interactive modelling experience using the adapted GAF/OAF methodology and the subdivision surfaces. Explanation of each snapshot is provided in the text.

After the modelling, the resulting geometry (mesh and subD cage) can be exported into a feature-based CAD environment to convert mesh polygons into NURBS surfaces and complete the design with functional fit features. Figure 8 shows the exported resulting geometry (on the left) into Rhinoceros (by McNeal Corp., San Jose, CA, USA) where it is converted to NURBS patches (in the middle) and a hole is added as a Boolean subtraction (on the right).

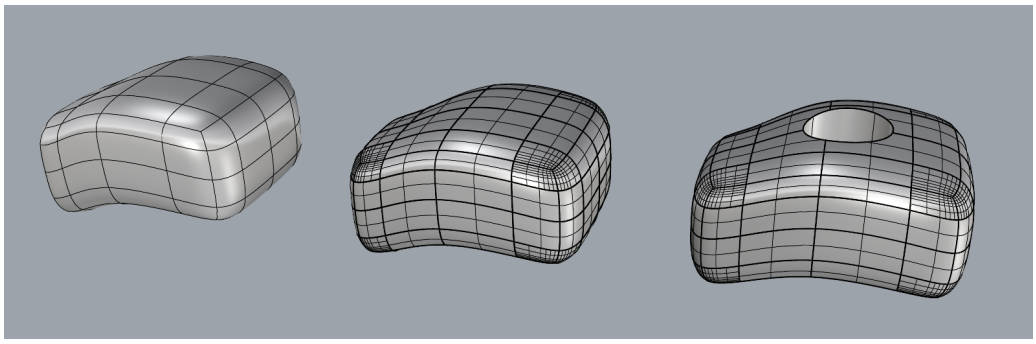

Figure 8. Finalising the modelling with the conversion of the SubD surface (on the **left**) into NURBS patches (in the **middle**) and then a perfect hole is subtracted from the resulting B-Rep solid (on the **right**). The final modelling is achieved into Rhinoceros.

3.2. Preliminary Usability Assessment

The case study has been shared with 20 users, between the ages of 22 and 50, 10 men and 10 women. The users were chosen from among the students of our university with a basic knowledge of geometric entities, feature-based geometric modelling, and 3D transformations. They also have experience in organic modelling based on subdivision surfaces, but using mouse and keyboard interfaces. They were asked to model an arch shape represented on a sheet of paper, wearing the AR headset and acting on the initial control cage through natural interaction based on gesture recognition. At the end of the experience of about 5 min in duration, they were asked to answer a questionnaire giving numerical score (from 0 to 10) to 7 characteristics of the experience. The results of the questionnaire were reported in Table 1 in the form of mean and standard deviation.

Table 1. Usability assessment through the evaluation of 7 features.

Feature	Score (0–10)
3D rendering quality (realism and graphics)	9.0 ± 0.5
Overall comfort	7.5 ± 1.2
Accuracy in pose recognition	7.2 ± 1.6
Easiness of selecting entities	7.2 ± 1.6
Easiness of moving entities	8.5 ± 0.9
Spatial awareness (depth occlusion)	8.6 ± 1.1
General judgement on natural interaction and modelling procedure	8.1 ± 1.0

The usability study results show that the quality scores of graphical renderings are very high. The scores are also very good with regard to the displacement of the virtual entity, the depth occlusion, and an overall judgement on the experience. On the other hand, the judgements regarding the recognition of gestures and the selection of entities that required a minimum of training are discreet. Also, the judgements given to comfort are more than discrete mainly due to the need to position the helmet correctly to benefit from the advanced eye-tracking features provided by the Varjo XR-4.

4. Discussion and Conclusions

A comprehensive methodology for creating an immersive environment for advanced modelling in AR has been presented and discussed in the paper. The methodology adopts freeform design through subdivision surfaces and incorporates a hand-tracking-based natural interface with sophisticated gesture recognition. The development of modelling software based on subdivision surfaces within an AR environment enhances the process of creating, interacting with, and optimising complex models.

The use of pass-through, which allows virtual objects to be inserted into the real world, provides real-time spatial awareness and helps reduce sizing errors. Additionally, linking real and virtual objects enables the visualisation of mating errors in the physical environment. Mixed reality, combined with the use of hand tracking that replaces controllers, enhances immersion by reducing the gap between the virtual and real worlds, which is more noticeable with traditional modelling software. Hand tracking is essential for ensuring natural interactions. The ability to control the position, direction, and angles of each finger's phalanges and joints allows for the development of customisable gestures that translate the user's intent into commands for the software. The GAF/OAF methodology has been adapted to manage interactions between the user and virtual objects. This approach has been particularly employed to facilitate selection and grasping actions, ensuring more stable connections through the definition of virtual constraints that improve grip control. The approach has once again proved to be not only easy and general to use, but also highly capable of integration with tracking, visualisation, and modelling algorithms.

The algorithm used for sculpting with subdivision surfaces is the Catmull–Clark algorithm, which has been adapted for use with triangular meshes in software designed for developing mixed-reality environments. These tools efficiently represent 3D objects and surfaces by leveraging the flexibility of triangular meshes. The choice of this algorithm is based on its general flexibility, straightforward implementation, and its ease of converting models from mesh to NURBS, enabling the models to be imported into traditional modelling software. This functionality also allows for the addition of further traditional operations to the model. The use of indices to identify the faces of the base cubes enables control over the transition between quadrilateral and triangular meshes, and vice versa. Also, the extrusion function is presented in this case, but the methodology can be generalised to other modelling functions.

The entire proposed procedure has proven to be easy to learn even for users who are not very experienced in AR due to the ease of natural interaction with hand gestures. The use of high-end devices such as the Varjo XR-4 and Ultraleap Motion Controller v2 has made it possible to obtain excellent performance in terms of visual feedback and immersivity of the scene, in accordance with the usability study. The usability study also suggested working on an easier and more robust system for recognising hand poses which, despite having received discrete scores, is still the most critical aspect detected by users. At the moment, we have limited the evaluations to 5 min tests and, therefore, the usability study must be considered preliminary. Test will be extended in the continuation of the research.

We believe that the proposed methodology can be generalised for the development of software for advanced modelling using subdivision surfaces, comprising all modelling functions and the ability to import the final design into traditional feature-based CAD software to convert meshes into NURBS. The interactive modelling phases should be intended as initial and not detailed design. This means that the user models initial three-dimensional shapes benefiting from the support of AR and then the laborious and time-consuming detailing of the geometry takes place on standard feature-based CAD platforms. Additionally, other customised hand gestures could be incorporated to enhance natural interactions and expand the user's modelling options.

Possible applications of the proposed methodology may concern the housing modelling of existing structures, to be performed under continuous reference of existing physical parts including obstacles, development of wearable devices, and clothes, to be modelled directly on the human body as a three-dimensional reference or the modifications of portions

of existing objects to be performed under the continuous persistence of the original parts. The great advantage of the use of subdivision surfaces allows the modelling of complex three-dimensional shapes using polygonal cages with a limited number of control points, favouring the possibility of building large complex shapes with relatively simple cages. So potentially, the user can also model entire organic or otherwise very complex shapes. A possible line of development can certainly be the inclusion of haptic devices to return force feedback to the user's hand, to improve the feeling of selection and movement of geometric entities, as well as to implement possible training applications. This integration could improve the score given by users in the usability study with regard to hand pose recognition and especially the selection of virtual entities.

Author Contributions: Conceptualization, P.P.V., A.C., M.C. and O.G.; methodology, P.P.V., A.C. and M.C.; software, P.P.V. and A.C.; validation, M.C. and O.G.; resources, P.P.V. and O.G.; writing—original draft preparation, A.C. and P.P.V.; writing—review and editing, M.C. and O.G.; supervision, P.P.V.; funding acquisition, P.P.V. and O.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Italian Ministry for Education and Research MIUR, PRIN 2022 program "Augmented Reality and Natural Interface for Computer-Aided Simulations", CUP E53D23003850006 and CUP: F53D23002000001.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Adel, A. Future of industry 5.0 in society: Human-centric solutions, challenges and prospective research areas. *J. Cloud Comput.* **2022**, *11*, 40. [CrossRef] [PubMed]
2. Pizoń, J.; Gola, A. Human–machine relationship—Perspective and future roadmap for industry 5.0 solutions. *Machines* **2023**, *11*, 203. [CrossRef]
3. Xu, X.; Lu, Y.; Vogel-Heuser, B. Industry 4.0 and Industry 5.0—Inception, conception and perception. *J. Manuf. Syst.* **2021**, *61*, 530–535. [CrossRef]
4. Choi, S.H.; Kim, M.; Lee, J.Y. Smart and user-centric manufacturing information recommendation using multimodal learning to support hu-man-robot collaboration in mixed reality environments. *Robot. Comput. Integr. Manuf.* **2025**, *91*, 102836. [CrossRef]
5. Liu, C.; Zhang, Z.; Tang, D.; Nie, Q.; Zhang, L.; Song, J. A mixed perception based human-robot collaborative maintenance approach driven by augmented reality and online deep reinforcement learning. *Robot. Comput. Integr. Manuf.* **2023**, *83*, 102568. [CrossRef]
6. Cellupica, A.; Cirelli, M.; Saggio, G.; Gruppioni, E.; Valentini, P.P. An Interactive Digital-Twin Model for Virtual Reality Environments to Train in the Use of a Sensorized Upper-Limb Prosthesis. *Algorithms* **2024**, *17*, 35. [CrossRef]
7. Gualtieri, L.; Öhler, M.; Revolti, A.; Dallasega, P. A visual management and augmented-reality-based training module for the enhancement of short and long-term procedural knowledge retention in complex machinery setup. *Comput. Ind. Eng.* **2024**, *196*, 110478. [CrossRef]
8. Palmarini, R.; Erkoyuncu, J.A.; Roy, R.; Torabmostaedi, H. A systematic review of augmented reality applications in maintenance. *Robot. Comput. Integr. Manuf.* **2018**, *49*, 215–228. [CrossRef]
9. Martins, A.C.P.; Castellano, I.R.; Júnior, K.M.L.C.; de Carvalho, J.M.F.; Bellon, F.G.; de Oliveira, D.S.; Ribeiro, J.C.L. BIM-based mixed reality application for bridge inspection. *Autom. Constr.* **2024**, *168*, 105775. [CrossRef]
10. Cirelli, M.; Cellupica, A.; Canonico, P.; Valentini, P.P. Impulse dynamics and augmented reality for real-time interactive digital twin exploration and interrogation. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2024**, *18*, 929–941. [CrossRef]
11. Chen, C.J.; Hong, J.; Wang, S.F. Automated positioning of 3D virtual scene in AR-based assembly and disassembly guiding system. *Int. J. Adv. Manuf. Technol.* **2014**, *76*, 753–764. [CrossRef]
12. Rosati, R.; Senesi, P.; Lonzi, B.; Mancini, A.; Mandolini, M. An auto-mated CAD-to-XR framework based on generative AI and Shrinkwrap modelling for a User-Centred design approach. *Adv. Eng. Inform.* **2024**, *62*, 102848. [CrossRef]
13. Kim, D.; Park, J.; Ko, K.H. Development of an AR based method for augmentation of 3D CAD data onto a real ship block image. *Comput. Des.* **2018**, *98*, 1–11. [CrossRef]

14. Freeman, S.; Wright, L.; Salmon, J. Exploration and evaluation of CAD modeling in virtual reality. *Comput.-Aided Des. Appl.* **2018**, *15*, 892–904. [CrossRef]
15. Hecht, D.; Reiner, M.; Halevy, G. Multimodal virtual environments: Response times, attention, and presence. *Presence Teleoperators Virtual Environ.* **2006**, *15*, 515–523. [CrossRef]
16. Reipschläger, P.; Dachselt, R. Designar: Immersive 3D-modeling combining augmented reality with interactive displays. In Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces, Daejeon, Republic of Korea, 10–13 November 2019.
17. Bonneau, G.-P.; Hahmann, S. 3D sketching in immersive environments: Shape from disordered ribbon strokes. *Comput. Graph.* **2024**, *18*, 103978. [CrossRef]
18. Rosales, E.; Rodriguez, J.; Sheffer, A. SurfaceBrush: From virtual reality drawings to manifold surfaces. *arXiv* **2019**, arXiv:1904.12297. [CrossRef]
19. Arora, R.; Kazi, R.H.; Anderson, F.; Grossman, T.; Singh, K.; Fitzmaurice, G. Experimental Evaluation of Sketching on Surfaces in VR. In Proceedings of the CHI'17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, 6–11 May 2017.
20. Kang, J.; Zhong, K.; Qin, S.; Wang, H.; Wright, D. Instant 3D design concept generation and visualization by real-time hand gesture recognition. *Comput. Ind.* **2013**, *64*, 785–797. [CrossRef]
21. Jiang, Y.; Zhang, C.; Fu, H.; Cannavò, A.; Lamberti, F.; Lau, H.Y.K.; Wang, W. Handpainter-3D sketching in VR with hand-based physical proxy. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Online Virtual, 8–13 May 2021.
22. Galyean, T.A.; Hughes, J.F. Sculpting: An interactive volumetric modeling technique. *ACM SIGGRAPH Comput. Graph.* **1991**, *25*, 267–274. [CrossRef]
23. Dorman, J.; Rockwood, A. Surface design using hand motion with smoothing. *Comput. Des.* **2001**, *33*, 389–402. [CrossRef]
24. Zhu, X.; Yang, Y. Interactive Mesh Sculpting with Arbitrary Topologies in Head-Mounted VR Environments. *Mathematics* **2024**, *12*, 2428. [CrossRef]
25. Jang, S.-A.; Kim, H.-I.; Woo, W.; Wakefield, G. Airsculpt: A wearable augmented reality 3D sculpting system. In *Distributed, Ambient, and Pervasive Interactions, Proceedings of the Second International Conference, DAPI 2014, Held as Part of HCI Interational 2014, Heraklion, Crete, Greece, 22–27 June 2014*; Proceedings 2; Springer: Berlin/Heidelberg, Germany, 2014.
26. Attene, M.; Campen, M.; Kobbelt, L. Polygon mesh repairing: An application perspective. *ACM Comput. Surv. (CSUR)* **2013**, *45*, 1–33. [CrossRef]
27. Ix, F.D.; Qin, H.; Kaufman, A. A novel haptics-based interface and sculpting system for physics-based geometric design. *Comput. Des.* **2001**, *33*, 403–420. [CrossRef]
28. Valentini, P.P.; Biancolini, M.E. Interactive Sculpting Using Augmented-Reality, Mesh Morphing, and Force Feedback: Force-Feedback Capabilities in an Augmented Reality Environment. *IEEE Consum. Electron. Mag.* **2018**, *7*, 83–90. [CrossRef]
29. Perles, B.P.; Vance, J.M. Interactive virtual tools for manipulating NURBS surfaces in a virtual environment. *J. Mech. Des.* **2002**, *124*, 158–163. [CrossRef]
30. Bourdot, P.; Convard, T.; Picon, F.; Ammi, M.; Touraine, D.; Vézien, J.-M. VR–CAD integration: Multimodal immersive interaction and advanced haptic paradigms for implicit edition of CAD models. *Comput. Des.* **2010**, *42*, 445–461. [CrossRef]
31. Butterworth, J.; Davidson, A.; Hench, S.; Olano, M.T. 3DM: A three dimensional modeler using a head-mounted display. In Proceedings of the 1992 Symposium on Interactive 3D Graphics, Cambridge, MA, USA, 29 March–1 April 1992.
32. Benkő, P.; Martin, R.R.; Várady, T. Algorithms for reverse engineering boundary representation models. *Comput.-Aided Des.* **2001**, *33*, 839–851. [CrossRef]
33. Ghali, S. Constructive solid geometry. In *Introduction to Geometric Computing*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 277–283. [CrossRef]
34. Malik, A.; Lhachemi, H.; Shorten, R. A cyber-physical system to design 3D models using mixed reality technologies and deep learning for additive manufacturing. *PLoS ONE* **2023**, *18*, e0289207. [CrossRef] [PubMed]
35. Pavlovic, V.; Sharma, R.; Huang, T. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell.* **1997**, *19*, 677–695. [CrossRef]
36. Rautaray, S.S.; Agrawal, A. Vision based hand gesture recognition for human computer interaction: A survey. *Artif. Intell. Rev.* **2012**, *43*, 1–54. [CrossRef]
37. Ma, W. Subdivision surfaces for CAD—An overview. *Comput.-Aided Des.* **2005**, *37*, 693–709. [CrossRef]
38. Valentini, P.P. Natural interface for interactive virtual assembly in augmented reality using Leap Motion Controller. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2018**, *12*, 1157–1165. [CrossRef]
39. Valentini, P.P. Natural interface in augmented reality interactive simulations. *Virtual Phys. Prototyp.* **2012**, *7*, 137–151. [CrossRef]
40. Valentini, P.P. Interactive virtual assembling in augmented reality. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2009**, *3*, 109–119. [CrossRef]
41. Valentini, P.P.; Pezzuti, E. Accuracy in fingertip tracking using Leap Motion Controller for interactive virtual applications. *Int. J. Interact. Des. Manuf. (IJIDeM)* **2016**, *11*, 641–650. [CrossRef]
42. Catmull, E.; Clark, J. Recursively generated B-spline surfaces on arbitrary topological meshes. In *Seminal Graphics: Pioneering Efforts That Shaped the Field*; ACM Digital Library: New York, NY, USA, 1998; pp. 183–188.

43. Doo, D.; Sabin, M. Behaviour of recursive division surfaces near extraordinary points. In *Seminal Graphics: Pioneering Efforts That Shaped the Field*; ACM Digital Library: New York, NY, USA, 1998; pp. 177–181.
44. Loop, C. Smooth Subdivision Surfaces Based on Triangles. Master's Thesis, University of Utah, Salt Lake City, UT, USA, 1987.
45. Akenine-Moller, T.; Haines, E.; Hoffman, N. *Real-Time Rendering*; AK Peters/CRC Press: Boca Raton, FL, USA, 2019.
46. Zhang, H.; Hoff, K.E., III. Fast backface culling using normal masks. In Proceedings of the 1997 Symposium on Interactive 3D Graphics, Providence, RI, USA, 27–30 April 1997.