Gesture-Based Manipulation of Virtual Terrains on an Augmented Reality Environment

Allan Amaral Ribeiro*, Douglas C. B. Oliveira†, Rodrigo L. S. Silva‡

Computer Science Department
Federal University of Juiz de Fora
Email: *allanmaralr@gmail.com, †dcoelho@gmail.com, ‡rodrigoluis@gmail.com

Abstract—This paper presents a free hand interface for modeling virtual terrains in an augmented reality environment. A secondary contribution of this work is to present a study of a set of suitable gestures that allows the manipulation of virtual terrains interactively, using augmented reality markers and the Leap Motion Controller. To demonstrate the study, an application was developed to allow the user to interact with the virtual terrain directly with his/her bare hands. Virtual objects were augmented using fiducial markers and the detection is done through the AVRLib Library. In our test layout, both markers and controller are co-planar. The application was tested and evaluated by six subjects. None of the subjects had previous knowledge on how to use the Leap Motion controller or the nature of the application.

1. Introduction

Augmented Reality (AR) is largely used as a tool to visualize 3D data over real-world environments. Most of the AR applications rely on the use of fiducial markers. These markers can be used to create 3D User Interfaces (UI) that go beyond the traditional mouse and keyboard set.

Among the categories of the 3D UI research field, one to be noted is the spatial arrangement, which studies, by definition, the properties of an array of things that have space between them. This scenario can serve as a powerful tool for developers, for instance, when generating terrains, or deforming meshes in an augmented reality environment. In fact, when associating a mesh to a physical marker it is not only possible to visualize it in 3D, but the user can also rotate and translate it freely by moving the fiducial markers, providing a higher interaction level between user and model.

The main problem of using fiducial markers to visualize and interact with a virtual model is the limitation of what a user can do with a single marker. For instance, if the user is interested in performing several operations over a virtual terrain, more than one marker must be applied, resulting in an inconvenient interface. One way to solve this problem is using the user’s hands instead of a fiducial marker, but in this scenario, it is hard to obtain a suitable precision to interact with a virtual model.

A free hand interface supports 3D input of hand gestures and motion allowing the user to interact with the system using its bare hands, without the constraint of standards input devices.

The main advantage of a free hand interface is that, unlike typical input devices like mouse and keyboard, 3D tasks do not need to be mapped in 2D or 1D inputs, providing greater flexibility and fewer limitations. In addition to making the commands more natural and intuitive, it still provides additional degrees of freedom to increase spatial and geometric information from the hands.

The main contribution of this work is to present a free hand interface for modeling virtual terrains in an augmented reality environment. The hands of the user are captured with high precision by a Leap Motion Device. Fiducial markers are used only to register the virtual terrain, but the manipulation is entirely performed free hand. A secondary contribution of this work is a preliminary study of a set of suitable gestures to interact with the virtual terrain.

The remainder of this work is organized as follows: Section 2 presents a brief survey of some works related to the use of the Leap Motion Controller in the field of gesture recognition and the generation of virtual terrain through augmented reality. In Section 3 the materials used in this work are presented. Section 4 presents the system design, including the augmented reality setup and the manipulation gestures used. Finally, in Sections 5 and 6 the results and some concluding remarks are presented, respectively.

2. Related Work

In the past few years, several works presented the use of the Leap Motion Controller in the field of gesture recognition in virtual reality environments.

In [1], a prototype of Leap Motion supporting an AR interface was presented. In this work, the tracked upper parts of the fingers were converted into voxels to perform the occlusion and interaction. They showed that the number of rendered voxels is the factor that limits the performance of the video. This work did not address any gesture interaction, but they obtained a good degree of immersion using this rendering technique.

Outside of the AR context but concerning the gesture recognition, [2] performed a qualitative study of potential in Auslan sign language recognition using the controller. They concluded that the API is not ready to recognize complex
A VRLib library [12], that is specialized in ARToolkit[59x163]A secondary contribution is to propose a set of suitable performed free hand in an augmented reality environment. approach in which the manipulation of a virtual terrain is and the interaction was performed using a second marker. multi-marker set to avoid occlusion during the manipulation, ToolKit framework. The terrain mesh was represented by a markers and the detection was accomplished by the AR- application the virtual terrain was augmented using fiducial generation of virtual terrains was presented. In the proposed simulation using a multi-touch screen to allow user interaction enormously dependent on menus. However, its gesture set is too simple, with rotate and zoom) making interaction, although limited, more intuitive. The authors of [8] presented a method to visualize and interact with large-scale 3D maps using light field display and free hand interaction. The authors opted to use gestures similar to the well-known multi-touch screens ones (pan, rotate and zoom) making interaction, although limited, more intuitive. In [9], the feasibility of hand-based interaction of AR glasses with basic manipulation of 3D virtual objects was evaluated. It uses a small set of gestures in favor of usability and intuition. However, its gesture set is too simple, with only a click and menu gesture, which makes manipulation overly dependent on menus.

Regarding virtual terrain modeling, but outside the Leap Motion context, [10] proposes an interactive terrain modeling framework focused on using geologically-inspired simulation using a multi-touch screen to allow user interaction with the system.

Finally in [11] an application that allows an interactive generation of virtual terrains was presented. In the proposed application the virtual terrain was augmented using fiducial markers and the detection was accomplished by the AR-ToolKit framework. The terrain mesh was represented by a multi-marker set to avoid occlusion during the manipulation, and the interaction was performed using a second marker.

The main contribution of our work is to present an approach in which the manipulation of a virtual terrain is performed free hand in an augmented reality environment. A secondary contribution is to propose a set of suitable gestures to perform this task. As far as we know, no other work presented a similar approach.

3. Materials In this paper, for hand gestures and motion detection we used the Leap Motion Controller, a small device with a tracking volume of about 8 cubic feet. For the visualization and augmentation of the virtual terrain we used the AVRLib Framework [12], which allows easy detection of multimarker, that deals with the possible partial occlusions of the fiducial markers caused by the movement of the user’s hands when dealing with the virtual terrain.

The discussed approach was built on a Windows system using the version 2.2.6 of the Leap Motion SDK and the MS Visual Studio 2013 framework.

The framework used in this paper was the AVRLib, an object oriented augmented reality library based on the well-known ARToolkit library [13], that is specialized in markers detection. The main reason for using this library despite the others available is due to its easy-to-use object-oriented interface and its complete documentation, making the integration of the library with the Leap Motion Device straightforward.

The Leap Motion Controller [14] is a small device, with about 6.2mm thick, 25mm wide and 75mm long, that allows hand, fingers and motion detection with a submillimeter precision (Figure 1).

The device has a large interaction space of 8 cubic feet, which takes the shape of an inverted pyramid, with a height of about sixty centimeter above the device.

Figure 1. Leap Motion Device.

The Leap Motion tracks the hands of the user at up to 200 frames per second using infrared cameras, giving a 150 degrees field of view with roughly 8 cubic feet of interactive 3D space (Figure 2).

It uses two cameras and three infrared LEDs to make the capture. The recorded data is streamed via USB to the provided detection software, and then the stereo grayscale image of the two cameras is analyzed to make a 3D reconstruction of what the device sees, a tracking algorithm is
applied to interpret the 3D data and infer the position of hands and fingers, including occluded parts.

The Leap Motion API provides a high-level information through a direct mapping of hands and fingers, with no need to process the raw data to infer hand position. The Microsoft Kineck SDK also provide hand positioning, but it is less accurate. Therefore, Leap Motion provides more accurate information regarding hand and finger position in comparison to Kinect.

3.1. Virtual Terrains

The design and generation of virtual terrains cover many computer science disciplines as computer graphics, human-machine interaction, and simulation. The main problem, in the computer graphics perspective, is to find ways to represent, manipulate and visualize virtual terrains [15]. Virtual terrains dynamics deals with the modification of the terrain by an external system or directly by an user. Tools like the ones presented in [11] and [16] can be used to facilitate the manipulation of these terrains interactively, without the use of mouse and keyboard.

In this paper, the virtual terrain is represented by a triangle mesh, which defines its surface. Our initial terrain employs the classical Coxeter-Freudenthal space subdivision scheme [17]. The resolution of the space decomposition is a parameter of the system.

The focus of this work is not to propose a novel deformation model, but to use a model that is fast enough to enable the execution of the proposed system in real time. In order to attend this requirement, we developed a simplified deformation model based on a Gaussian function centered in the middle of the user’s hand and scaled by the distance from the hand to the Leap motion plane.

In this model, first we compute the projection of the hand position \( \vec{h} \) over the plane \( \pi_{xy} : z = 0 \) using the Equation 1.

\[
\vec{h}_{xy} = \text{proj}_{\pi_{xy}}(\vec{h})
\]  

Then, for each particle with position \( \vec{p} \) in the terrain, we calculate the translation in \( \vec{l} \) axis and sum to the particle position through the Equation 2.

\[
\vec{p}_{ij} = \vec{p}_{ij} + \vec{l} \times e^{-\left\| \vec{p}_{ij} - \vec{h}_{xy} \right\|^2 / \sigma^2}
\]

4. System Design

To perform free hand deformation of the virtual terrain and the augmentation of the objects, a setup based on fiducial markers and the Leap Motion Controller was used. A webcam was attached to the system to complete the system layout. Details of this layout and the set of gestures used to manipulate the virtual terrain will be explained in this section.

4.1. Augmented Reality Setup

For the augmented reality setup, a multimarker was used. That type of marker is suitable in a scenario where partial occlusion may occur during the system usage. The Leap Motion Controller was positioned at the center of the marker as shown in Figure 3 to increase the precision and range of the device. This particular position of the Leap Motion Controller was also chosen to avoid distortions at the borders of the virtual terrain. In our layout, seven fiducial markers can be used to perform the registration of the virtual terrain, and at least one of them must be visible to the camera.
by the Leap Motion Device to the AVRLib’s coordinate system we just need to perform a 90 degrees rotation in the X axis, because the up-vector of the Leap Motion is the Y axis and the AVRLib is the Z axis.

4.2. Manipulation Gestures

In order to perform the manipulation of the virtual terrain, a set of hand-based gestures showed in Figure 4 was proposed. We use the measure of confidence available in the Leap Motion library to evaluate if a gesture is suitable or not to be used. In our tests, gestures with confidence greater than or equal to 50% are considered suitable.

![Figure 4. Set of manipulation gestures used to manipulate the virtual terrain.](image)

(a) (b) (c) (d)

Figure 4. Set of manipulation gestures used to manipulate the virtual terrain. The gesture presented in (a) is used to change the state of the model editing mode. The pointing gesture (b) was used to activate or deactivate the influence radius of our deformation system. The gestures presented in (c) and (d) increase or decrease the radius, respectively.

The closed fist gesture (Figure 4.a) is used to change the state of the model editing mode. The available states are: attraction, repulsion and off. The Leap Motion confidence value for this gesture ranges from 60% to 90%. The pointing gesture (Figure 4.b) was used to activate or deactivate the influence radius of our deformation system. Once activated, the editing mode is turned off. This gesture has a confidence ranging from 70% to 100%. Once the influence radius function is activated, the gestures illustrated in Figures 4.c and 4.d can be used to increase or decrease the radius, respectively. In our tests, we use the value of 0.025 to modify this radius. We establish a minimum interval between the change of gesture of 0.8 seconds.

4.3. Interface Widgets

The system interface has two main widgets. The first one is a wireframe sphere that controls when the influence radius mode is activated. The widget is red when the influence radius is on and gray otherwise (Figure 5.a and 5.b). A magnet widget was used to illustrate attraction. A gray magnet is used when the editing mode is off. When the attraction state is on, the widget is changed by a red magnet with an arrow pointing up (Figure 5.c). Finally, when the repulsion mode is on, a red magnet with an arrow pointing down is used (Figure 5.d).

![Figure 5. Interface widgets.](image)

(a) (b) (c) (d)

Figure 5. Interface widgets. In (a) both influence radius and editing mode are turned off. In (b) the influence radius is turned on. Attraction and repulsion mode is on in (c) and (d), respectively.

4.4. Modeling Algorithm

As our system was limited by the frame rate of the camera, the approach was to divide gesture recognition and display in two different threads to improve the application interactions.

One of the threads was responsible for Leap Motion data acquisition and processing, identifying any of the gestures and saving it in a thread-safe object that would be used in the terrain manipulation phase. The basic outline of the Leap Motion thread is presented in Figure 6.

The main thread was responsible for processing the camera input using the AVRLib framework and the terrain manipulation and display.

5. Results & Experiments

In this section, some results of the system proposed in this work will be presented. The visual effects were obtained using the graphical library OpenGL and its high-level shading language GLSL. All the aforementioned experiments were made using a Microsoft LifeCam Cinema Camera. The computer used in our experiments was equipped with...
Input: LeapMotionController

Initialization:
1: editingInfluence ← false
2: editingTerrain ← false
3: infRadius ← DefaultValue

Thread main loop:
4: while ApplicationIsRunning do
5:   frame ← AcquireLeapMotionData()
6:   hxy ← HandPosition(frame)
7:   gesture ← IdentifyGesture(frame)
8:   if editingInfluence = true then
9:     if gesture = OpenPalm then
10:        infRadius ← infRadius + InfluenceStep
11:     else if gesture = ClosedPalm then
12:        infRadius ← infRadius – InfluenceStep
13:     else if gesture = HandL then
14:        editingInfluence ← false
15:   end if
16:   else
17:     if gesture = CloseFist then
18:        editingTerrain ← ¬ editingTerrain
19:     else if gesture = HandL then
20:        editingInfluence ← true
21:        editingTerrain ← false
22:   end if
23: end if
24: end while

Figure 6. Leap Motion Thread

an Intel Core i7 2.93GHz, 8.0 GB of memory RAM and
a GeForce GTS 450 graphics card. The whole project was
implemented using the C++ language.

5.1. System Usage

In Figure 7 the standard deviation of the Gaussian func-
tion applied in this paper was used to decrease (Figure 7.a)
or increase (Figure 7.b) the size of the influence radius of
a sphere widget.

Figure 7. Influence radius based on the standard deviation of a Gaussian function.

In Figure 8, our method was applied on a virtual terrain
with 1280 x 720 resolution, and the modifications illustrated
in Figures 8.a to 8.d were achieved with a fixed standard
deviation in our Gaussian function. This method can be used
to create more detailed virtual terrains.

Figure 8. Virtual terrain created with a Gaussian function with fixed standard deviation in 0.2.

When using the proposed gestures to interactively
change the standard deviation of our Gaussian function we
generate the virtual terrains depicted in Figures 9.a to 9.d.

Figure 9. Virtual terrain created with a Gaussian function with variable standard deviation from 0.05 to 0.35.

5.2. User’s Evaluation

To evaluate our system we invited six post-graduate
computer graphics students. The main aspect we wished to
evaluate was the system’s easiness and usability.

The evaluation was performed in two steps. The first one
was a warm-up, in which the users have three minutes to
try the system freely, learning the gestures and building a
random virtual terrain. In the second step, a reference virtual
terrain (Figure 10) was presented to the user, and the idea
was to build a virtual terrain that looks similar. The idea was
not to replicate the reference virtual terrain in fine details
but to create a new one with the same characteristics with
a coarse similarity. The generated terrains should present
a portion of water surrounded by mountains. The height
of the mountains would define their composition. Higher mountains are made of stone, and lower mountains made of grass and both of them were required. The virtual terrains created by the students are presented in the Figures 11.a to 11.f. The students had five minutes to complete the task. The students that create the models presented in 11.a to 11.d completed the task successfully. The student of the model presented in Figure 11.e did not achieve the level of detail required and the student of the last model (Figure 11.f) was not able to create the stone mountains.

Despite the subjectivity of our evaluation, the virtual models created by the students presented interesting results with a very short training and no interference during the modeling process.

5.3. Performance Evaluation

The performance evaluation of the system was divided into rendering performance (Table 1) and terrain manipulation performance (Table 2). The rendering performance was evaluated when no Leap Motion manipulation was detected. The terrain manipulation was evaluated when the terrain was changed by using the Leap Motion Controller. In some scenarios, we can observe a slight difference between these two tests, where the terrain manipulation performance was around 20% slower than the rendering performance, especially when the number of triangles of the virtual terrain was high (in our tests, more than 20000 triangles).

<table>
<thead>
<tr>
<th>Resolution / Num. Triangles</th>
<th>3000</th>
<th>20000</th>
<th>45000</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280 x 720</td>
<td>24 fps</td>
<td>19 fps</td>
<td>14 fps</td>
</tr>
<tr>
<td>800 x 448</td>
<td>30 fps</td>
<td>30 fps</td>
<td>20 fps</td>
</tr>
<tr>
<td>640 x 360</td>
<td>30 fps</td>
<td>30 fps</td>
<td>22 fps</td>
</tr>
</tbody>
</table>

The performance evaluation shows that with the hardware used in our tests, the virtual terrain manipulation can be performed in real time (up to 30 fps) with up to 20,000 triangles with a max video resolution of 640 x 360 pixels and up to 5,000 triangles with a max resolution of 800 x 448 pixels. When using HD resolution (1280 x 720) even with 5,000 triangles, the average fps was only 24 fps.

It is worth noting that our system is limited by the camera frame rate of 30 frames per second.

6. Conclusion

In this work, we propose a free hand interface for modeling virtual terrains in an augmented reality environment. A set of suitable gestures to perform this task was also proposed to allow the manipulation of a virtual terrain. An augmented reality setup was presented, and issues regarding the positioning of the Leap Motion Device were addressed. The use of user’s hands instead of any other device or fiducial markers provides an intuitive method to create forms and patterns across the terrain. The proposed interface allows an easy surface modeling without any skills on computer programming or rendering knowledge.

A clear limitation of this work is regarding the field of view of the Leap Motion Device. During our tests, a common issue was positioning our hands outside that
viewing volume. One way to overcome this limitation is to render the virtual volume created by the device.

Another problem was the occlusion caused by the hands of the user over the virtual terrain. As the augmentation of the virtual terrain occurs over the video stream, the users can see their hands covered by virtual terrain in some points of view, even when the hand is supposed to be above the terrain. One way to overcome this issue is to apply some approach to create an occlusion of a real object (in this case, the hand of the user) over the virtual terrain as the one presented in [18].

An evaluation was carried out with users in order to verify if the proposed system was indeed intuitive. The evaluation was carried out in two steps: a training step followed by an exercise in which the user should build a coarse version of a reference terrain using the system. The evaluation presented in this article was preliminary with the purpose of evaluating the usability of the system. Further evaluations with a larger number of users will be conducted in the future.

We believe the approach presented in this work can be easily adapted to be used in applications where the lack of precision of the Leap Motion Controller will not impact the quality of the output.

Acknowledgments

The authors would like to thank CAPES, FAPEMIG and the GETComp/UFJF for their financial support.

References


