Augmented Intelligent Space

Intelligent Systems Laboratory — Winter Term 2013/2014

Benjamin Errouane & Jan-Erik Platte & Robin Schiewer

Supervisors: Thies Pfeiffer

Bielefeld University, Faculty of Technology

Abstract

The Augmented Intelligent Space (AIS) project is all about representing a physically existent environment with a virtual model, thus enabling the creator to modify this model in any desired way. Our goal is to achieve an as immersive as possible but yet adjustable and controllable environment to enhance productivity and simplify workflows, for example by presenting the output of a sensor by simply looking at the specific sensor. We achieve this by utilizing the Oculus Rift head mounted display to depict our virtual version of the environment, which is simulated in the blender game engine. While head movement is tracked by the Oculus' internal accellerometer, the position of the subject is determined via an ordinary VR marker tracking approach. Both, position and rotation information, are integrated inside the blender game engine.

Introduction

When dealing for example with problems in a cognitive information technology context, there are often lots of sensors, actuators and 3rd party applications involved, all connected via a sufficient middleware. Hence keeping track of things can become a difficult task. To carry out simple tasks like e.g. verify the output of a sensor or, in general, a system component, the user often has to do the same steps, (addressing components, query data etc.) over and over again. The idea behind AIS is to provide additional interfaces to simplify and speed up the process of interaction by providing firsthand information inside the virtual representation of an environment. We thereby use a similar approach like [3], but with a more leightweight software architecture and a non-see-through display. Due to the simplified interaction principle, this approach also allows untrained users to work with complex systems.

Main Objectives

1. Display an arbitrary virtual environment to the user via a head mounted display.
2. Provide an intuitive interface for controlling the viewport (i.e. rotate your head to look around).
3. Track the user's real position in order to realize movement in the simulation.
4. Integrate a basic interaction scenario (e.g. move objects in the simulation by moving objects in the real world)
5. Recreate an existing place virtually to enable the user to move through the virtually modified version of this place.
6. Integrate additional, more complex types of interaction (e.g. tracking of the user's focus)

Materials, Methods and Architecture

While [4] utilize the concept of an augmented reality mainly for displaying data, visualize processes and planning tasks, we concentrated our work in this semester on the creation of an as intuitive as possible user interface and a basic infrastructure for interactions. To do so, we use the Oculus Rift head mounted display (HMD) to depict the virtual environment to the user and make use of the accellerometer integrated in the Oculus Rift to track head rotation. To determine the user's position, we utilize a simple marker tracker approach and mount a commen VR marker on the Rift. We currently also provide tracking mechanisms for additional markers, which can represent for example objects in the virtual world. Then, this objects can be moved by relocating the markers in the real world.

Whereas [3] use a see-throgh HMD to project virtual content into a real scene, we prefer to trade in the advantages of a see-through solution (e.g. realistically looking and reacting environment is already present) against a fully virtual scene representation. The reason is, that controlling and manipulating a completely virtual scene is far more easy, than trying to do so with a mixture of simulated and real contents, like they are present on a see-through HMD. Another point is, that we expect the Oculus Rift to represent the first prototype in an upcoming series of devices, which will probably be programmalbe and usable in a very similar way. That makes it especially interesting for us to work with this technology right now.

To provide proper ways of communication for our program components, we use the Robot Service Bus (RSB) framework and have additionally developed two plugins for blender. Those plugins are basically wrappers for various c++ functions of the RSB framework and the Oculus Rift drivers. We thereby make use of the cython package for python to address c and c++ code in python, since blender is programmable in python only. The basic architecture of our application is shown in figure 1 and gives a brief overview about the current state of development.

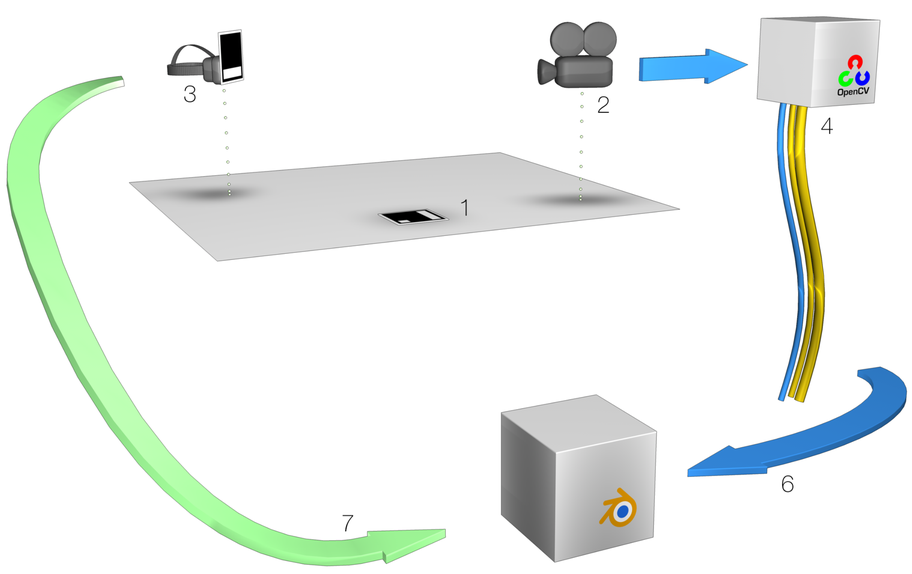

Figure 1: The basic program architecture is shown above: At (1) there is the marker located, which designates the origin of the global coordinate system. (2) marks the camera, which records video data, that is then sent to an openCV based image processing application (4). At (3) there is the Oculus Rift depicted, which serves as an accellerometer and an output display at the same time. Consider the marker attached to the Oculus Rift, which is tracked by the camera at (2). The processed results of (4) are sent via RSB, which should be illustrated through the two yellow and the one blue lead, and caught by blender via a small pugin (6). Blender then carries out all extracted movement information. At the same time, the Oculus Rift (3) is directly connected to blender via another plugin (7) and recieves data from the Rift's accellerometer in realtime. The transmission of the by blender produced images back to the Oculus Rift's display is omitted in this scheme.

Results

The representation of a given scene and the basic interaction with it work with overall satisfying results. First, the quality of the displayed scene depends on the level of detail of the underlying 3D model and the display technology, that is used. At the moment, we use low detail scenes intended for basic testing only, because creating a highly detailed 3D scenery by hand can be very time consuming and is moreover not decisive for our projet to work, so it was considered to be of minor importance at this point.

The display of the Oculus Rift provides a rather low solution of 640x800 pixels per eye, making it easy to spot single pixels. Whereas this on the one hand lowers the overall visual quality, on the other hand other factors like viewing angle, available drivers and the integrated position sensor provided decisive advantages.

The movement tracking works stable and fast, but there is only a small area covered, which lies directly in front of the camera. This area can be slightly increased by choosing bigger markers, but remains rather limited. As a result, the user experience is disturbed when the user moves too far away from the center of the camera's field of view, since in this case the camera will probably loose track of the marker attached to the HMD. This means, the user won't be able to move in the virtual world anymore, unless he or she retreats towards the center of the camera's field of view again. At this point, the importance of a high quality high resolution tracking camera became clear, since in contrast to for example [5], where mainly GPS is used, our navigation is based entirely on VR marker positions.

Another problem related to the limited camera viewing area is the tracking of an object-representing marker and the HMD marker at the same time. While there are no performance issues existent when tracking multiple markers at a time, the camera's field of view is simply to small, to allow excessive movement of both markers inside the stable trackable area. While the VR marker remains recognizable for the human eye when the user moves too far away from the camera, our approach tends to loose track after a few meters. This slightly reduces the freely traversable space further, but the Oculus Rift's cable is the more limiting factor here.

When it comes down to the tracking camera's resolution, it turned out that a too high resolution decreases the quality of the overall results. While the marker is moved, our camera delivers high resolution images with noticeable motion blur, which can interrupt the tracking. Moreover, the tracking range decreased as well. If the resolution is choosen too low, the marker is not tracked stable enough and thus it's position is flickering, causing the simulation to flicker as well.

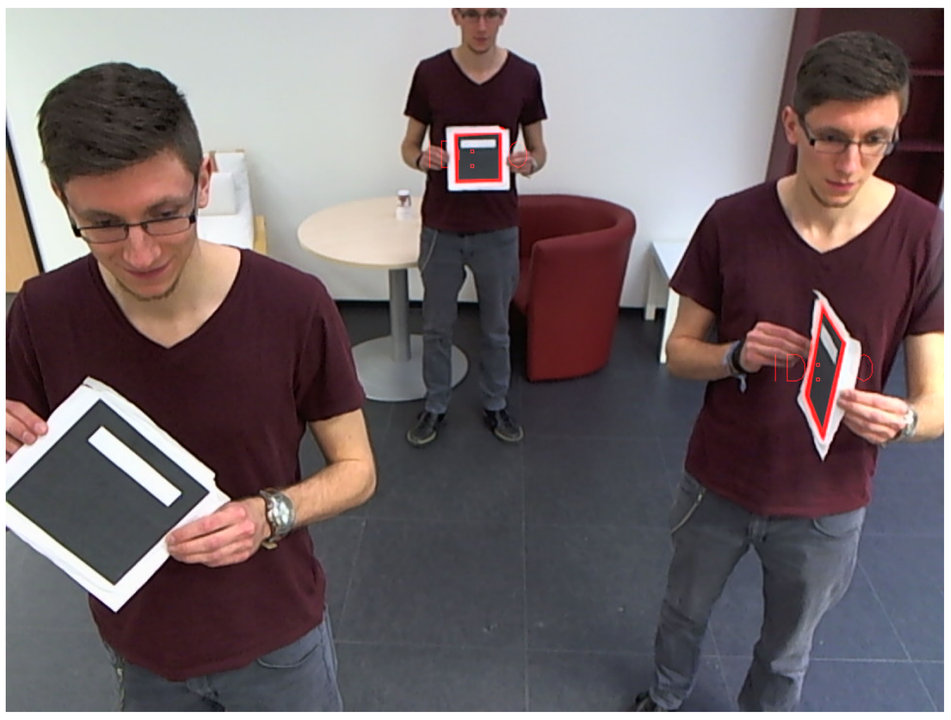

Figure 2: Various border cases of our marker tracking approach. Left: An arbitrary small part of the marker leaves the camera's viewport. Middle: The farthest point, where the marker can still be seen by out tracking application. Right: Even if the marker is orientated almost parallel to the camera's viewing axis, it can still be tracked.

Conclusions

- Due to the present navigation mechanisms integrated in our application, being in the simulation gives a really immersive feeling, regardless of the low quality 3D models used. Nevertheless, we aim to refine our tracking mechanisms and 3D models in order to further improve the user's experience.
- While the marker tracker approach gives good results in general, because of the limited traversable area we certainly consider the support of multiple tracking cameras to be our next step.
- To increase the range of our marker tracking approach, we also want to examine the performance of better cameras with a higher resolution. Another step will be to further examine the impact of the recording frame rates on the image sharpness in our scenario.
- With the manipulation of objects via trackable markers as a first way of interaction, we want to provide additional types of interaction to the user, for example looking at objects to cause a system reaction.
- Currently, we rely almost entirely on the blender game engine to depict our scene. For better flexibility and because of several blender-related limitations, we want to investigate other solutions.

References

[1] Rodney A. Brooks, editor. *The Intelligent Room Project*. MIT Artificial Intelligence Lab, 1997.

[2] Andrew Graham Daniel Brooker, Toby Collett, editor. *Improving Augmented Reality Visualisation for Mobile Robot Development*. School of Engineering, The University of Auckland, Auckland, New Zealand, 2009.

[3] Burkhard C. Wuensche Ian Yen-Hung Chen, Bruce MacDonald, editor. *Mixed Reality Simulation for Mobile Robots*. Dept. of Computer Science University of Auckland New Zealand, 2009.

[4] Jonathan Foote Sagar Gattepally Don Kimber Bee Liew Eleanor Rieffel Jun Shingu Jim Vaughan Maribeth Back, Anthony Dunnigan. The virtual chocolate factory; building a real world mixed - reality system for industrial collaboration and control. FX Palo Alto Laboratory.

[5] Shean White. Interaction with the environment: Sensor data visualization in outdoor augmented reality. Department of Computer Science, Columbia University, Department of Botany, Smithsonian Institution, 2009.

Acknowledgements

Special thanks go to Dr. Thies Pfeiffer for providing us a very good image tracking library and lots of ideas and help regarding our project. We also want to thank Florian Lier for assistance regarding ROS/RSB and a propererly organized course.