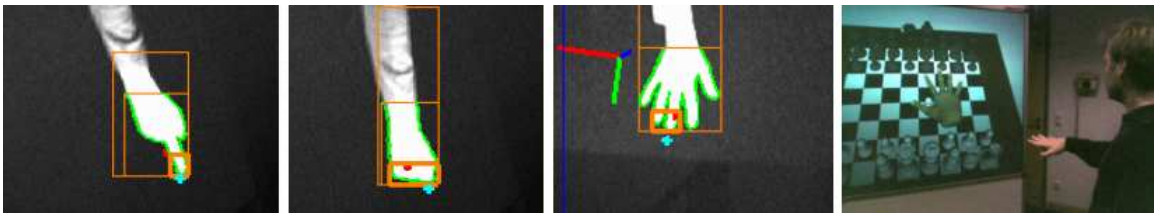
# Intuitive Interaction with VR Applications Using Video-based Gesture Recognition

Cornelius Malerczyk* Timo Engelke†

Fraunhofer Institute for Computer Graphics Research (IGD)
Department of Virtual and Augmented Reality
Fraunhofer Street 5, 64283 Darmstadt, Germany



## ABSTRACT

This paper describes the implementation of a deviceless interaction 'device' using hand gesture recognition within a calibrated stereo system. Video-based interaction is one of the most intuitive kinds of Human Computer Interaction with Virtual Reality applications due to the fact that users are not wired to a computer. If interaction with three-dimensional environments is considered, pointing, grabbing and releasing are the most intuitive gestures used by humans. This paper describes a video-based gesture recognition system that observes the user in front of a large displaying screen, identifying three different hand gestures in real time using 2D feature classification and determines 3D information like the 3D position of the user's hand or the pointing direction if performed. Different scenario applications like a virtual chess game against the computer or an industrial scenario have been developed and tested.

**Keywords:** Gesture Recognition, Human Computer Interaction, Computer Vision, Classification

**Index Terms:** H.5.2 [User Interfaces]: Input devices and strategies I.4.8 [Image Processing and Computer Vision]: Scene Analysis — Object Recognition

## 1 INTRODUCTION

Interaction with three-dimensional virtual worlds is today no longer unusual. Many computer users are familiar with special input devices such as 3D computer mice or even tracking systems used for 3D interaction. Nevertheless, if VR applications are to be used by a large number of different and even technically unversed users (e.g. because the system is to be used at a public place), the system has to meet some special requirements:

- The system has to be capable of operating in real-time. This includes both an image refresh rate of more than 20 frames per second to ensure a jitterfree interaction and a system delay between the performed status change of the gesture and the reaction of the system of less than 200 ms, which is not perceived as an unnatural system behaviour by most users [10].
- The system has to be usable by all users. Therefore, no or only a minimal training phase should be necessary to get the application to work.
- The system must not make use of any technical device, the user is forced to wear (like special gloves) or to hold in the hand (such as typical interaction devices used for e.g. infrared-light based tracking systems). Furthermore, for a video-based system there is the need of concealing all technical equipment such as the cameras observing the user. The output device should be the only technical piece of hardware visible for the user.

## 2 RELATED WORK

Hand gesture recognition in computer vision is an extensive area of research that encompasses anything from static pose estimation of the human hand to dynamic movements such as the recognition of sign languages. A primary goal of gesture recognition research for Human-Computer-Interaction is to create a system, which can identify specific human hand gestures and use them to convey information or for device control. Therefore, hand gesture recognition systems can be divided into different tasks: Hand tracking, dynamic gesture recognition, static gesture recognition, sign language recognition and pointing. Elaborate reviews on vision-based hand pose estimation can be found in [3] and [4]. Due to the high amount of different approaches and methods for hand pose estimation, we only consider approaches, that are capable of tracking the human hand in real-time and differentiating static hand postures. Exemplarily, [7] presented an approach for tracking the position and the orientation of four different postures of two hands in real-time by calculating the visual hulls of the hands on the GPU directly. Therefore, the results necessarily depend on the correct reconstruction of the 3D pose of the hand using at least three cameras. [5] uses a calibrated stereo system with two colour cameras to estimate the position and orientation of different hand postures in 3D. The approach is based on 2D feature extraction using various techniques based on geometric properties and template matching. Therefore, it is necessary to identify corresponding features in image pairs and to derive 3D features that are afterwards fitted to an underlying 3D hand model to classify different gestures. Due to the

*e-mail: Cornelius.Malerczyk@igd.fraunhofer.de
†e-mail: Timo.Engelke@igd.fraunhofer.de

fact that no orientation of the hand is determined in our approach, we are able to reduce the classification problem to pure 2D feature extraction and to calculate the 3D position of the hand for interaction purposes using the center of gravity of the segmented hand only.

## 3 System Setup

The equipment for the gesture recognition system consists of one single standard PC, which is used for both rendering of the scenario applications and gesture recognition and tracking. A standard video beamer or a large plasma display is connected to the PC, displaying the application scenario in front of the interacting user. Two Firewire (IEEE1394) cameras are connected to the computer feeding the system with grey-scaled images of the interaction volume in real time. Lenses with additional infrared light diodes (without infrared light filters) are used to ensure a bright reflection of the human skin (see figure 1). The purpose of the tracking

Figure 1: Firewire IEEE 1394 camera with additional IR diodes.

system is to recognize and to track three different static gestures of the user (pointing, opened hand and closed hand) to enable intuitive interaction with the scenario applications. The approach is based on the recognition of the position of the human hand in 3D space within a self-calibrated stereo system [1]. Therefore, position and orientation of the cameras are determined with respect to each other by swaying a small torch light for a few seconds in the designated interaction volume [8]. Afterwards only the world coordinate system has to be defined by marking the origin and the end of two axes of the world coordinate system. Within this coordinate system the corners of the displaying screen have to be declared to ensure a correct interpretation of especially the pointing gesture and its pointing direction. This calibration procedure has to be performed only once after setting up the cameras. During runtime of the system, difference images are used to detect moving objects, which then are analyzed and the 2D probabilities of a posture and its relevant parameters in 3D space is calculated. Smoothing of the tracking results like e.g. the 3D position of the hand using smoothing splines is used to reduce jittering effects [9], which leads to an immersing experience during the interaction without the need of any technical device.

## 4 Gesture Extraction

The segmentation of the user's hand is performed on 2D image basis. At the startup of the tracking system reference images of the empty interaction volume are taken from both cameras. Afterwards edge images are calculated. During runtime of the tracking system edge images are calculated from new images captured by the stereo system. These edge images are then compared with the edge reference images by calculating difference images. The resulting pair of images is segmented at a predefined threshold. Due to the camera

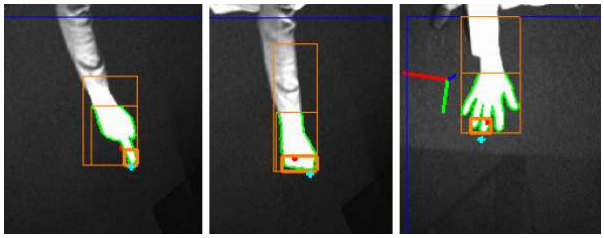


Figure 2: Three different gesture: Pointing, closed hand and open hand for grabbing and releasing events.

setup at the left and right hand side of the user and due to the fact that the user is always interacting with a screen in front of him/her, it can be easily assumed that the lowest extracted segment larger then an adequate segment size can be chosen as the user's hand. Nevertheless, often not only the user's hand but also his/her forearm is extracted into one segment. Therefore, an approximately square rectangle at the bottom of the segment is chosen containing the final hand segment (see figure 2). This segment is used afterwards for feature extraction. The lowest points of the segments are identified to be the finger tip if a pointing gesture is recognized or the used 3D position of the gesture. For each frame pair of the stereo camera system several two-dimensional features of the segmented human hand. Important examples of these parameters are:

- Ratio between boundary length and area
- Eccentricity, elongatedness and compactness
- Curvature of the segment boundary

Using two-dimensional feature extraction only has the advantage of a robust and fast parameter determination. However, parameters of one single object may differ in both camera images. Therefore, all parameter pairs are sorted by their size and stored as a feature vector for classification of the gestures.

## 5 Gesture Classification

Feature classification is a well known task for image understanding and object recognition. Often used classification methods are Hidden Markov Models (HMM) and Artificial Neural Networks (ANN) for the visual interpretation of hand gestures. For our system the Naïve Bayes Classifier[1] [11] is used for gesture identification, which leads to a sufficient balance for the model building process time of less then 5 seconds and an online classification rate of more than 50 Hz.

To ensure a robust and stable recognition of the different gestures (with a classification result of more than 95%) a short training procedure for each new user of the system is necessary. Basically, this training procedure can be skipped using a large predefined training data set consisting of the training data of several users, whereas the recognition result decreases by up to 10% and therefore sporadic misinterpretations of the system may occur. For the training procedure the user is asked to perform the different gestures for approximately ten seconds each at the startup of the system. This procedure is easy and short enough so that the user does not lose interest in starting to interact with the application itself. If desired the training result may be individually saved for a later personalized usage of the tracking system.

[1] http://www.cs.waikato.ac.nz/ml/weka/


Figure 3: Scatterplot of two selected features used for classification of pointing (blue), grabbing (red) and releasing (green).

Table 1: Classification matrix for three different gestures using a Naïve Bayes probabilistic model

| Gesture performed vs. classified | | | |
|---|---|---|---|
| | **Pointing** | **Grab** | **Release** |
| **Pointing** | 449 | 6 | 33 |
| **Grab** | 3 | 426 | 1 |
| **Release** | 24 | 2 | 379 |

## 6 Classifiaction Results

Using the hardware setup described in section 3 the system provides interaction feedback in real-time. After the calibration procedure performed only once at the system setup that needs approximately three minutes and after an initial training phase of less than two minutes (to perform the three different postures for ten seconds each and to build the classification model), the runtime recognition rates are at twenty frames per second and higher. The cameras deliver up to 30 frames per second with a delay shorter then 1/5 of a second. Even all image processing and computer vision tasks like segmentation, edge calculation and 3D reconstruction of corresponding image points last less then 50 ms. Overall a recognition rate of up to 25 Hz is achieved. Nevertheless, as important is the classification rate of the system. Figure 3 shows the classification results of one single user for three different gestures (pointing, grabbing/closed hand and releasing/open hand). The user performed each gesture 10 seconds during the training procedure, which leads to approximately 250 feature vectors for each gesture. Tables 1 and 2 show the recognition and classification results for 60 seconds of interaction with a recognition rate of approximately 95% for new single feature vectors. Due to the fact that outliers (single incorrect classified gestures) can be eliminated using a post-processing queue a completely correct visualization of the performed gestures is achieved for interaction with the application scenario.

Table 2: Recognition rates for three different gestures

| **Training data** | **Test data** | **Recognition rates** |
|---|---|---|
| Training session | Cross validation | 95.1% |
| Testing session | Cross validation | 97.6% |
| Training session | Testing session | 94.7% |

## 7 Applications

For the creation of new applications it is important to have standardized and easy to use authoring tools and rendering components at hand. We use the InstantReality Framework[2] [2] for the rendering part of the gesture recognition system. InstantReality is an open environment for AR/VR applications based on the scenegraph system for realtime graphics OpenSG[3] [6]. InstantReality uses X3D as the programming language for the virtual worlds the user interacts with. Like most traditional toolkits, InstantReality uses a scenegraph to organize the data, as well as spatial and logical relations. In addition to the scene description, VR applications

Figure 4: Playing virtual chess against the computer using video-based gesture recognition.

need to deal with dynamic behavior of objects and user interaction via non-standard input devices such as the gesture classifcation described in previous sections. Here, the 3D position of the gesture is routed into the virtual scene using low level sensor nodes such as a SFVec3fSensor (see also section 8). As a proof of concept

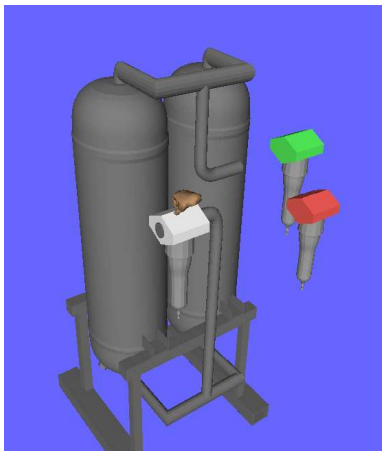


Figure 5: Learning assistance scenario: Placing filters to an industrial air pump system using grabbing and releasing gestures.

different scenario applications using virtual 3D environments for interaction have been developed. For a virtual chess application the user is standing in front of a large scaled screen rendering a three-dimensional chess board (see figure 4). Using the 3D position of the recognized gestures for grabbing and releasing the user is able to move chess pieces and therefore to play a game of chess against the computer. In a second scenario application the user is

[2] http://www.instantreality.org/
[3] http://opensg.vrsource.org/trac/

asked to place color labeled filters to virtual 3D industrial air pump system. An additional object snapping method (implemented as a JavaScript node within the application) is used to ensure a precise assembly of the three-dimensional objects, even if the user releases a filter object only roughly at the outlet of the air pump (see figure 5).

Figure 6: Interaction with a three-dimensional object using virtual buttons to point at for rotation (3D-Model by courtesy of the Picture Gallery of the Academy of Fine Arts Vienna).

A typical task for the interaction with virtual worlds is the exploration of three-dimensional objects. But due to the fact that the gesture recognition described in this paper is not able to determine the 3D orientation of the user's hand, the rotation of a selected object can not be achieved using the standard SphereSensor node. Nevertheless, the task of an object rotation can be achieved by using four virtual buttons indicating the rotation to left, right, up and down. The buttons displayed on the edge of the scene (see figure 6) emit events if the user is pointing at them. These buttons are attached to touch sensors that are routed to a JavaScript node which is triggering the rotation of the 3D object.

## 8 Implementation

For applications such as the virtual chess game or the learning assistance scenario described in the previous section, the VRML and X3D standard does not provide input sensors, which are sufficient to handle the information provided by the gesture recognition system correctly. Already Behr et al. stated in [2] that the X3D specification documents are not specific how to deal with 3D or even 6D devices connected to a rendering system. Therefore, the InstantReality framework and its InstantPlayer provide a bunch of additional low level sensor types that are generic enough to be used for connecting the video-based tracking system to VR applications. Within the InstantReality framework low level sensors are provided for all supported X3D data types. This of course also includes sensors for the data types SFVec3f and SFInt32, which are used for routing the 3d position of the users hand and the currently recognized gesture into the virtual scene. After the post processing steps of the tracking pipeline (such as smoothing of the position of the users hand) for example the 3d position is used to drive the translation field of the `Transform` node of the virtual representation of the hand:

```
DEF hand3d SFVec3fSensor { label "Hand/Position3d" }
ROUTE hand3d.value_changed TO hand.set_translation
```

Furthermore, the currently detected gesture is coded as a simple integer value, which is used to display one of *n* different representations of the virtual hand model:

```
DEF handSwitch Switch {
    whichChoice -1
    choice [
        DEF openedHand   Transform {
          translation 0 0.08 0
          children Inline { url "release.wrl" }
        }
        DEF closedHand   Transform {
          translation 0 0.02 0
          children Inline { url "grab.wrl" }
        }
        DEF pointingHand Transform {
          translation 0 0 0
          children Inline { url "pointing.wrl" }
        }
    ]
}

DEF gesture SFInt32Sensor { label "Hand/Gesture" }
ROUTE gesture.value_changed TO handSwitch.whichChoice
```

Here, the value of the sensor is connected to the attribute `whichChoice` of a standard `Switch` node. Although the same geometrical model is used for all gestures, different poses lead to slightly changing centers of gravity and therefore also to a slight jumping of the model during the change of the gesture status. Therefore, an adjusting translation is added to each of the gestures to ensure a smooth transition between different gestures. Although some parameters (such as the scaling of the hand model) depends on the application the tracking system is interacting with and therefore some parameters has to be adapted for each new application, this method of connecting tracking and rendering application is simple enough to be implemented even by developers, which are new to VRML/X3D programming.

## References


[1] A. Azarbayejani and A. Pentland. Camera self-calibration from one point correspondence. 1995. MIT media laboratory, perceptual computing technical report 341.

[2] J. Behr, P. Daehne, and M. Roth. Utilizing x3d for immersive environments. *ACM SIGGRAPH: Web3D 2004 - Ninth International Conference on 3D Web Technology*.

[3] A. Erol, G. Bebis, M. Nicolescu, R. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Comput. Vis. Image Underst.*, 108(1-2):52–73, 2007.

[4] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man and Cybernetics - Part C*, 37:311–324, 2007.

[5] R. O'Hagan and A. Zelinsky. Visual gesture interfaces for virtual environments. *Australasian User Interface Conference*, 0:73, 2000.

[6] D. Reiners, G. Vo, and J. Behr. Opensg: Basic concepts. In *In 1. OpenSG Symposium OpenSG*, pages 200–2, 2002.

[7] M. Schlattman and R. Klein. Simultaneous 4 gestures 6 dof real-time two-hand tracking without any markers. In *VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 39–42, New York, NY, USA, 2007. ACM.

[8] B. Schwald and C. Malerczyk. Controlling virtual worlds using interaction spheres. In C. A. Vidal, editor, *Proceedings of the 5th Symposium on Virtual Reality (SVR)*, pages 3–14, Fortaleza, CE, Brazil, 2002. Brazilian Computer Society (SBC).

[9] S. Sun, M. Egerstedt, and C. F. Martin. Control theoretic smoothing splines. *IEEE Transactions on Automatic Control*, 45:2271–2279, 2000.

[10] B. Watson, N. Walker, P. Woytiuk, and W. Ribarsky. Maintaining usability during 3d placement despite delay. In *VR '03: Proceedings of the IEEE Virtual Reality 2003*, page 133, Washington, DC, USA, 2003. IEEE Computer Society.

[11] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, second edition, 2005. ISBN 0-12-088407-0.