GPU Accelerated Gesture Detection for Real Time Interaction

Torsten Bierz
University of Kaiserslautern
International Research Training Group
P.O. Box 3049
67653 Kaiserslautern, Germany
Email: bierz@informatik.uni-kl.de

Achim Ebert
University of Kaiserslautern
P.O. Box 3049
67653 Kaiserslautern, Germany
Email: ebert@informatik.uni-kl.de

Jörg Meyer
University of California Irvine
644E Engineering Tower
Irvine, CA 92697-2625, USA
Email: jmeyer@uci.edu

Abstract: Over the past years, the interaction between humans and computers (HCI) evolved to one of the most important research topics in computer science. Therefore, finding a way for an intuitive, easy and affordable interaction is the main challenge. Optical markerless tracking using consumer hardware can satisfy these problems. However, in order to be able to interact in an efficient way, the tracking and furthermore the interaction must be handled in real time. This leads to efficient use of current GPUs in order to speed up the tracking and furthermore the gesture recognition. Involving these issues, an approach for an implementation and system will be presented in the following paper.

1 Introduction

Nowadays, humans are surrounded by a highly sophisticated environment and technology. From these circumstances, the interaction and in addition the human-computer interaction (HCI) is arising to a very important research topic among scientists. Interacting with computers can be handled in many different fashions and furthermore with a huge amount of different interaction devices. Consequently, the interaction should be as intuitive and simple as possible in order to provide an efficient and satisfying way for interaction even for untrained persons .

In recent years, many different interaction types and sophisticated hardware input devices have been developed. In this regard, one of the intuitive options for interaction commands are poses and gestures, which can be detected in a satisfying manner by additional hardware, e.g. data gloves or markers. However, most untrained or novel users are not familar with wearing additional hardware devices and behave unnatural. Although, the devices

have become smaller and more lightweight, they might still be uncomfortable to wear. Therefore, the question arises whether it is possible and sufficient to track only the poses and gestures of the users and therefore their hands by means of skin detection in order to obtain information for interaction purposes.

Commercially available optical tracking systems are often quite expensive and usually require markers or other visual means for optical reference. Therefore, the question arises whether we really need those tracking devices or whether off-the-shelf components like webcams or digital cameras can also be sufficient for this type of interaction.

Based on these ideas, a system has been developed which facilitates optical tracking without optical markers for human gesture-controlled real-time interaction with a software system. The real-time capability is accomplished by using a GPU-accelerated computation method.

2 State of the Art

Gaining insight into current state of the art research in human-computer interaction (HCI), many different interaction methods and approaches as well as hardware devices exist. These methods can be categorized in different tracking methods:

- Pointing devices, which are tracked in context of the users, and allow a three dimensional interaction, e.g., wands [CB03], or the dragonfly device [SR03],
- Spot pointing devices, e.g. laser pointers [ATK+05, ON01], where the resulting spot is used for selecting or moving of objects or writing on the screen,
- Or trackable interaction devices, e.g. the Cubic Mouse [FPW+00], or data gloves [Imm07], where the position needs to be tracked in order to obtain interaction information for the applications.

However, the user is always dependent on the use of a specific device for the interaction, keeping his or her hands occupied, which may not always be feasible, for instance, in a sterile medical environment.

In order to be independent of additional devices, interaction should be kept as intuitive and natural as possible. One good solution is the tracking of the user and the use of eye movements to control navigation or detect the user's focus [BDDG03] (gaze tracking). However, a serious drawback to this method is the anthropogenic fact that humans tend to focus on different targets in very short time intervals, which results in difficulties for deciding whether a change of the eye position should result in a scene change or not [Zha03]. When considering which human bodily expressions can or should be tracked, usually the hands come to mind first, as they are certainly the most natural and intuitive interaction devices.

In recent years, different approaches for using the hands for human-computer interaction have been studied and developed. Strickon and Paradiso [SP98] tried to track hands mov-

ing over a low-cost laser-scanning rangefinder. With this device, the user interacts with a laser in front of a screen. Varona et al. [VRL05] and Thayananthan et al. [TSTC06] presented other approaches for tracking of hands and faces trying to classify gestures by a hierarchical Bayesian filter, which consisted of different rotations of the gesture. They demonstrated that building up different rotated gestures in preprocessing can be avoided by using rotation invariant gestures.

Nowadays, quite complex and furthermore expensive camera solutions and systems exist, which are able to track a person or interaction devices. However, these devices often need a specific setup and may require additional hardware. Furthermore, they often require reflectors or other optical indicators, so called markers or targets, which must be attached to the desired tracking object [Gmb07]. In this article, we develop other tracking options, employing consumer hardware such as webcams or digital cameras and develop a markerless gesture-based input method. In addition, the computational power of the GPU is used to accelerate computations and to obtain real-time capability of the detection and tracking system. Modern available webcams or digital cameras have a sufficiently high resolution of at least 640x480 pixels and a frame rate of 25 to 30 frames per second. Furthermore, current computers are mostly equipped with satisfying graphic cards, that can take care of the necessary computations in order to match the real-time constraints, which comes along with the requirements for smooth interaction.

This article presents an efficient way of interacting with virtual environments by using consumer hardware. In Section 3, the basic concepts and ideas of the system will be presented. Within this Section, a short system overview will be given, followed by a description of the image capturing and the necessary image processing steps ofskin detection, noise filtering, and outline extraction. For acceleration purposes, these steps are computed on the graphics processing unit (GPU). In section 4, the pose and gesture detection is explained. Therefore, some basic operations on images must be explained first. These are the so-called image moments and their computation, which will be used for the comparison of the captured regions to poses, which are stored in a database. The paper concludes with results from the skin detection and gesture recognition algorithms and with an outlook on future work.

3 Markerless Interaction

In the following section first a short overview of the system architecture and the usage of the algorithms on the GPU is presented. Then, the different steps of capturing an image, applying skin detection, noise removal and silhouette reduction are described. The results are then used for pose and gesture detection, as described in Section 4.

3.1 System Architecture

In many applications, the real-time responce of the system for the interaction task is most important providing an immediate feedback to the user. Consequently, tracking and recognition must be handled as fast as possible, preferably at the same rate as images are displayed or obtained. Otherwise, the user could be easily confused or irritated and would not know whether he or she has already initiated an action. As a result, it is important to handle all necessary calculations within a given time frame. Although modern CPUs are becoming increasingly faster, it is sometimes difficult to process all incoming and outgoing data in real-time. Many researches now explore the potentials of the current graphic processing units (GPUs). Furthermore, by now the GPU is not only utilized for graphical purposes, but also for performing highly parallel tasks efficiently. The usage of graphic cards for such a purpose is defined as general purpose computation on the GPU (GPGPU for short) [Fun05, FM04]. Textures or in this case image data, as obtained from digital cameras or webcams, are well-suited for the computation and execution of multiple operations in parallel on the GPU.

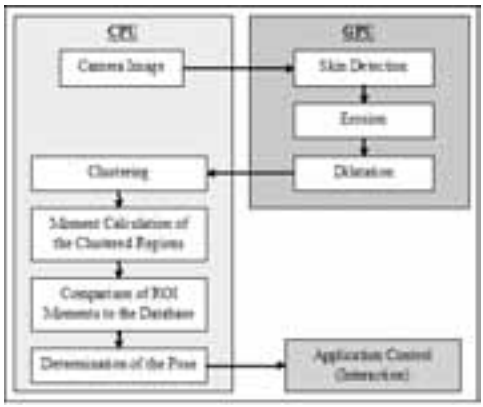


Figure 1: Scheme of the algorithm

The current implementation of the system uses advanced features and acceleration capabilities of the GPU. An outline of the system architecture is given in Figure 1. First, the camera image is captured and transmitted to the GPU as a texture. Next, the skin detection, which is presented in 3.2, is performed on the GPU implemented as a fragment shader. The opening of the image, which is separated in an erosion and a dilation part, reduces noise in the binary image. This algorithm can also be efficiently computed on the GPU. The results from the filter kernels are stored as alpha values in the camera image. All of these algorithms are implemented as fragment shaders. After their computation, the image is transmitted to the CPU where the remaining tasks are performed. The clustering as well as pose and gesture detection based on image moments, which will be handled in Section 4, cannot be implemented as a highly parallel GPU algorithm and therefore are carried out on the CPU.

3.2 Skin detection

Common skin detection methods classify pixels according to a specific color space, e.g., RGB, normalized RGB, HSV, or YCrCb. Other methods try to classify skin regions based on probability maps, which are generated using training sets. This process needs the manual selection of the regions containing skin, and is therefore expensive. Neural networks can be used to simplify it. A more detailed overview of current state-of-the-art approaches are given by Martinkauppi et al. [MSP03] and by Kakumanu et al. [KMB07].

For the detection of skin in form of a binary classification, Gomez and Moralez proposed an approach based on normalized RGB values [GM02]. The normalization of the RGB values reduces the ambient illuminance and offers the advantage of being more robust against varying light conditions. The decision whether a pixel is classified as "skin" or "not skin" is made by logically combining the following three thresholding equations:

$$\frac{r}{g} > 1.185 \quad and \quad \frac{rb}{(r+g+b)^2} > 0.107 \quad and \quad \frac{rg}{(r+g+b)^2} > 0.112 \qquad (1)$$

where $r, g, b \in [0, 1]$ are the normalized RGB values.

If the query succeeds in all three cases, the pixel is classified as "skin", otherwise as "not skin". Consequently, when applying this skin detection method to a captured image, the result will be a binary coded image. This separates the image into regions containing either "skin" or "no skin" (see Figure 2).

(a) Captured Image (reproduced in color on p. 192) (b) Image after skin detection

Figure 2: Applying the skin detection to a captured image

3.3 Noise reduction

When using consumer off-the-shelf cameras, a well known problem is their susceptibility to noise. In order to improve the captured image, one can smooth it using a discrete

Gaussian filter. For this, the image is convoluted with a Gaussian filter mask. However, when the noise reduction is applied to the captured image, it is already in a binary format (see Figure 2). Applying a Gaussian filter to a binary image would result in blurred edges and increased pixel errors. To avoid this, morphological operators are employed as they are more efficient on binary images. We use an opening operation, which is a combination of an erosion and a dilation filter.

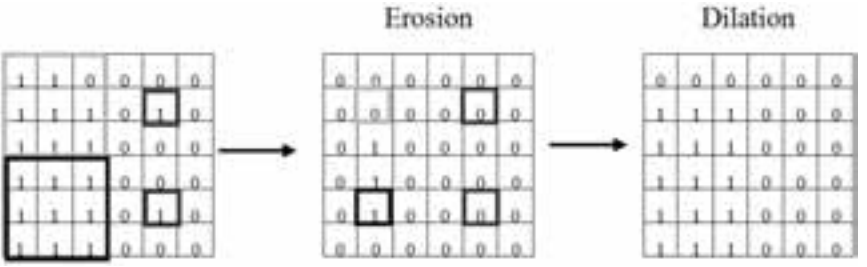


Figure 3: Opening of a binary image (reproduced in color on p. 192)

The erosion as well as the dilation steps are applied as filter kernels of a 3x3 matrix. An example is shown in Figure 3. Additional information on image processing techniques and morphological operations can be found in [Jäh97].

In order to receive the contour or silhouettes of the segmented image, algorithms and methods for edge detection must be applied. Multiple approaches have been described in the literature, e.g., gradient-based methods like Sobel or Canny operators [Jäh97]. However, for binary images, the computation can be done in a more efficient way by applying a modified hit-miss operator. The defined hit-miss operator operates as a search operator for a 4-connected neighborhood of the current pixel with the value ”1”. If no zero is found in the neighborhood of the pixel, the value of the pixel is set to zero. As a result, only the outline of an object remains visible (represented by ”1” pixels).

(a) Original image and detected contours (reprduced in color on p. 192)

(b) Result from sample image

Figure 4: The modified hit-miss operator

4 Gesture Detection

In the following section the clustering of the resulting image is described. After that, the concept of image moments is explained. They are calculated for the regions previously found by the clustering and are compared to the moments in the stored database. Finally, the action assigned to the detected pose is performed.

4.1 Clustering

After the computation of the outlines of the skin region done on the GPU, these regions of interest are separated into clusters. Without region clustering a recognition of poses is not feasible. This can be seen in Figure 4. This image consists of several regions, e.g. the eyes, the head, and the hand. It should be noted that there is some noise present in the image. If the whole image was compared to a pose (e.g. as seen in Figure 5), there would be no success, because the image does not exclusively contain the pose. There are still some other objects present in the image, which lead to incorrect results. Therefore, the respective discrete regions must be found and clustered. In order to gather the regions, region growing [Ver91] is used. An 8-connected neighborhood is considered for determining connected regions. The boundary of each region is returned as a result.

Comparing gestures or poses is challenging. Several solutions exist, e.g. performing a pixel-wise comparison of two images or image regions. However, this method can only be used if the reference image and the searched region have the same scale, the same rotation, and the same position. Therefore, when dealing with poses or gestures, a pixel-based comparison approach is inefficient and will not provide sufficient results. The next section describes a more appropriate solution to this problem. The method is called "image moments".

4.2 Image Moments

The concept of moments is well known in physics and mathematics. Image moments are specific weighted averages of image pixels. These moments, or in this context the values of the moments, represent the feature which is supposed to be recognized. In our case, the feature is a gesture or a pose. Consequently, a definition of moments is required which is invariant to translation, rotation or scale. In order to be able to define and compute such moments, we define raw moments:

$$M_{ij} = \sum_{x=0}^{w-1} \sum_{y=0}^{h} x^i y^j I(x, y), \tag{2}$$

where $I(x, y)$ is the pixel intensity at location (x,y), w and h the width and the height of the image, and $i, j \in \mathbb{N}$. The raw moments are neither invariant to translation nor to

scale or rotation. In order to devise a translation invariant representation, the region is moved to the origin. These moments are called central moments. For this, the centroid $(x_c, y_c) \in \mathbb{R}^2$ can easily be calculated $(x_c, y_c) = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right)$. The central moments are calculated by:

$$\mu_{ij} = \sum_x \sum_y (x - x_c)^i (y - y_c)^j I(x, y). \tag{3}$$

Central moments are still not scale and rotation variant. For scale invariance, the central moments are dividedby the moment μ_{00}, which is the same as M_{00}. These moments are

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}} \quad for \quad i + j \geq 2. \tag{4}$$

In order to have a rotation invariant representation of these moments, Hu proposed a set of seven rotation invariant moments [Hu62]. Flusser proved in [Flu00] that a set of six moments is sufficient because of dependencies of two rotation invariant moments. These moments are

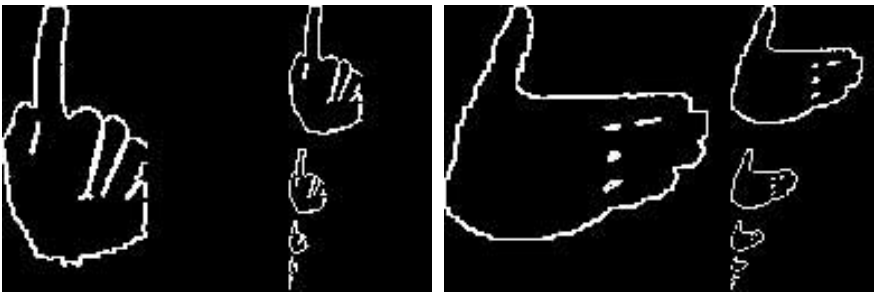
$$\begin{aligned}
\psi_1 &= \eta_{20} + \eta_{02}, \\
\psi_2 &= (\eta_{30} + \eta_{12})^2 + (\eta_{03} + \eta_{21})^2, \\
\psi_3 &= (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2) + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}), \\
\psi_4 &= \eta_{11}((\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2) - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}), \\
\psi_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{03} + \eta_{21})^2), \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{21})(3(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2), \\
\psi_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2), \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2).
\end{aligned}$$

The moments of the poses are represented by this set of six values and are invariant to rotation, translation and scale. They can be computed as a prepossessing step for the stored poses in the gesture database.

4.3 Hierarchical Moments

In order to achieve higher recognition rates, the image moments are hierarchically build. The hierarchical moments can be described as moments of the poses in different resolutions. An example for the poses is presented in Figure 5.

The poses and their corresponding values are stored in the database. This reduces the computation time during gesture recognition.



(a) Pointing Gesture (b) Rotation Gesture

Figure 5: Two Sample Poses

4.4 Pose Detection

The resulting regions of the clustering, given by their bounding boxes, are used for the pose detection. For this purpose, hierarchical moments for the regions as previously defined are calculated and compared to the stored moments in the database. If the results are within a certain threshold range, the region is recognized as containing a pose. Two examples of the original image and the resulting poses are shown in Figures 6.

4.5 Following of Hands

The regions given by the clustering might contain poses. The detection based on the image moments provides the information, whether the region contains a pose. These regions usually move slowly. When comparing the resulting regions from one frame to another, the corresponding regions of the two frames usually overlap. Therefore, a simple overlapping test for axes-aligned bounding boxes can prove this issue. Having found two overlapping regions, the changing of their area usually remains in a certain threshold. In the current implementation, the threshold is set to 90%, which lead to sufficient results. So, if a pose was detected in a frame, and is not detected in the following frame, it is possible to keep track of the position of the hands.

5 Results and Future Work

Most cameras have a frame rate of either 25 or 30 frames per second, and the computation has to be done within a time frame of about 0.03 seconds to achieve interactive response of the visualization system. With the presented GPU accelerated approach the computation can be handled within this time interval, which is important for realtime interaction.

The recognition of poses and gestures has been tested under different lighting conditions.

The overall recognition rates depending on the normalized RGB values were sufficient. However, in dark rooms as well as under direct sunlight exposure, the pose and gesture recognition was insufficient and further research has to be done to calibrate the camera. In dark rooms it is quite difficult to detect skin. This may become an issue if the gestures are supposed to control an application projected onto a screen in a dark room. In this context simple fluorescent gloves in combination with an ultraviolet lamp or an additional infrared light source might lead to improved recognition results. For direct sunshine, it might be possible to compute the average of the region and an offset, which can be substracted in order to obtain higher recognition rates. Another option is an automated recalibration of the camera to compensate for different exposure situations.

In order to accelerate the process of detection, following of the hands can be used as a prediction model, where only a smaller or specific region of the image needs to be examined. In this case, the moments, which are currently computed for all occurring regions, can be reduced to the regions, where a gesture or pose was detected first.

Currently, interaction with the system is based on the first recognized pose or gesture. Two-handed interaction is currently not supported. One of the next steps will be the implementation of the two- instead of single-handed interaction, which will result in advanced possibilities for interaction. Furthermore, the current system is designed for single-user interaction. Another aspect is multiuser interaction. For this, suitable metaphors or mechanism must be developed to facilitate this type of interaction.

Acknowledgements The authors would like to thank the German Research Foundation (DFG) for supporting the International Research Training Group (IRTG 1131). In addition, the members of the Computer Graphics and Visualization Research Groups at the University of Kaiserslautern, Germany, and the University of California, Irvine, as well as the members of the IRTG 1131 for their cooperation and assistance.

References

[ATK+05] Benjamin A. Ahlborn, David Thompson, Oliver Kreylos, Bernd Hamann, and Oliver G. Staadt. A practical system for laser pointer interaction on large displays. In *VRST: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 106–109, 2005.

[BDDG03] Patrick Baudisch, Doug DeCarlo, Andrew T. Duchowski, and Wilson S. Geisler. Focusing on the essential: considering attention in display design. *Communications of the ACM ACM*, 46(3):60–66, 2003.

[CB03] Xiang Cao and Ravin Balakrishnan. VisionWand: Interaction techniques for large displays using a passive wand tracked in 3D. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 173–182. ACM Press, 2003.

[Flu00] Jan Flusser. On the independence of rotation moment invariants. *Pattern Recognition*, 33(9):1405–1410, 2000.

[FM04] James Fung and Steve Mann. Using Multiple Graphics Cards as a General Purpose Parallel Computer: Applications to Computer Vision. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, pages 805–808, Washington, DC, USA, 2004. IEEE Computer Society.

[FPW+00] Bernd Fröhlich, John Plate, Jürgen Wind, Gerold Wesche, and Martin Göbel. Cubic-Mouse-Based Interaction in Virtual Environments. *IEEE Computer Graphics and Applications*, 20(4):12–15, 2000.

[Fun05] James Fung. *GPU Gems 2 : Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems) (Hardcover)*. Addison-Wesley Professional, March 2005.

[GM02] G. Gomez and E. Morales. Automatic feature construction and a simple rule induction algorithm for skin detection. In *Proceedings of Workshop on Machine Learning in Computer Vision*, pages 31–38, 2002.

[Gmb07] Advanced Realtime Tracking GmbH. Advanced Realtime Tracking GmbH: Finger Tracking and Markers, 2007. http://www.ar-tracking.de.

[Hu62] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, 8(2):179–187, Feb 1962.

[Imm07] Immersion. Wireless Data Glove: The CyberGlove®II System by Immersion, 2007.

[Jäh97] Bernd Jähne. *Digital Image Processing: Concepts, Algorithms, and Scientific Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 6. edition edition, 1997.

[KMB07] P. Kakumanu, S. Makrogiannis, and N. Bourbakis. A survey of skin-color modeling and detection methods. *Pattern Recogn.*, 40(3):1106–1122, 2007.

[MSP03] B. Martinkauppi, M. Soriano, and M. Pietikainen. Detection of skin color under changing illumination: a comparative study. In *Image Analysis and Processing, 2003.Proceedings. 12th International Conference on*, pages 652–657, 17-19 Sept. 2003.

[ON01] Dan R. Olsen and Travis Nielsen. Laser pointer interaction. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22. ACM Press, 2001.

[SP98] Joshua Strickon and Joseph Paradiso. Tracking hands above large interactive surfaces with a low-cost scanning laser rangefinder. In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 231–232, New York, NY, USA, 1998. ACM Press.

[SR03] Oliver Stefani and Jörg Rauschenbach. 3D input devices and interaction concepts for optical tracking in immersive environments. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, pages 317–318, New York, NY, USA, 2003. ACM Press.

[TSTC06] Arasanathan Thayananthan, Bjorn Stenger, Philip H. S. Torr, and Roberto Cipolla. Model-Based Hand Tracking Using a Hierarchical Bayesian Filter. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(9):1372–1384, 2006.

[Ver91] David Vernon. *Machine vision: automated visual inspection and robot vision*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991.

[VRL05] Xavier Varona, Jose Maria Buades Rubio, and Francisco J. Perales López. Hands and face tracking for VR applications. *Computers & Graphics*, 29(2):179–187, 2005.

[Zha03] Shumin Zhai. What's in the eyes for attentive input. *Communications of the ACM*, 46(3):34–39, 2003.

(a) Captured Image of a pointing gesture (reproduced in color on p. 192)

(b) Resulting image with recognized pointing gesture

(c) Captured Image of a grabbing gesture (reproduced in color on p. 192)

(d) Resulting image with recognized grabbing gesture

Figure 6: Examples of recognized poses