H. Reiterer & O. Deussen (Hrsg.): Mensch & Computer 2012
München: Oldenbourg Verlag, 2012, S. 293-302

3D Hand Gesture Recognition Based on Sensor Fusion of Commodity Hardware

Manuel Caputo, Klaus Denker, Benjamin Dums, Georg Umlauf

HTWG Konstanz, Germany

Abstract

With the advent of various video game consoles and tablet devices gesture recognition got quite popular to control computer systems. E.g. touch screens allow for an intuitive control of small 2d user interfaces with finger gestures. For interactive manipulation of 3d objects in a large 3d projection environment a similar intuitive 3d interaction method is required. In this paper, we present a dynamic 3d hand and arm gesture recognition system using commodity hardware. The input data is captured with low cost depth sensors (e.g. Microsoft Kinect) and HD color sensors (e.g. Logitech C910). Our method combines dynamic hand and arm gesture recognition based on the depth sensor with static hand gesture recognition based on the HD color sensor.

1 Introduction

User interfaces have changed a lot in recent years. Especially touch based interaction has become a common method to operate mobile phones and tablet PCs. However, touch screens have several problems. The finger touch leaves fingerprints on the screen, some on-screen content is occluded by the fingers, and for large displays (e.g. display size 2 × 3 meters) some display regions might be inaccessible. For 3d applications the biggest disadvantage is the limitation to two dimensions. Here, 3d gesture tracking allows a much more direct interaction with 3d objects. For a large stereoscopic projector system for 3d visualizations current interaction metaphors like mouse, keyboard, or Wii Remote are either unfeasible or cumbersome to use. A direct 3d gesture interaction for this system is much more convenient.

With the Microsoft Kinect a low cost 3d camera is available to capture data at interactive frame rates. However, its resolution is rather low at large distances. Thus, a large 3d interaction system should be based on several Microsoft Kinect devices and web-cams to increase the size of the operating space and the resolution.

2 Related work

Camera based 2d hand gesture recognition is used in human computer interaction (Sánchez-Nielsen et al., 2004) and robotics (Ghobadi et al., 2008). The silhouette of the hands is reconstructed and matched to a gesture database.

Interaction devices in virtual reality systems often use 3d hand gesture recognition. *Wands* are hand held devices using position tracking, rotation tracking, and buttons to interact with 3d objects (Wormell and Foxlin, 2003). More complex gestures are possible with *gloves* that also measure the movement of the fingers (Sturman and Zeltzer, 1994).

Time of flight cameras are used for motion capturing applications. Body parts are reconstructed from the depth data (Malassiotis et al., 2002; Plagemann et al., 2010) and their motion is tracked over time (Ganapathi et al., 2010). Hybrid approaches use intensity and depth information from time of flight cameras to achieve the recognition of static 3d hand gestures (Ghobadi et al., 2007, 2008). The depth information is used to create a silhouette that is refined using an intensity or color image.

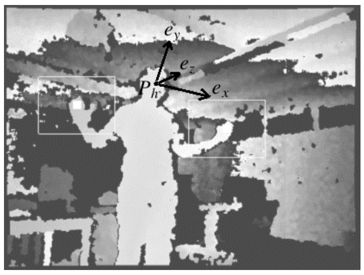
The depth quality of the Microsoft Kinect camera is not as good, as that of expensive time of flight cameras. Its main application is the recognition of full body gestures in games. Microsoft uses a per-pixel classification approach that needs huge amounts of training datasets (Shotton et al., 2011). Because of the popularity in research, several frameworks were created to simplify the use of the Kinect for research applications (OpenNI, 2012; PrimeSense Inc., 2010). For our work we use these frameworks.

3 Sensor fusion

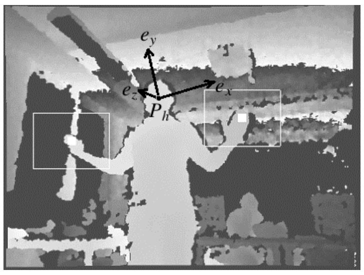
With a single Kinect it is not possible to recognize hand gestures over more than one meter distance. The resolution and the quality of the depth sensor are too low. Thus, for hand gesture recognition an additional sensor is necessary. We use a web-cam with a resolution of 1920×1080 pixels. At this resolution it is possible to recognize hand gestures over a distance of several meters. However, locating a hand in an image at this resolution is computationally expensive. Therefore, the Kinect is used to locate and track the hand. The Kinect and the web-cam are positioned at approximately the same angle. So, both sensors capture approximately the same spatial region resulting in similar images. Position and distance of the hand are extracted from the data of the Kinect and the size of the hand is estimated. Based on this information, an image region containing the hand is determined. This image region in the high resolution image is then used for the hand gesture recognition. These gestures are relatively slow, so no time synchronization of the cameras is necessary.

For this approach the Kinect and the web-cam have to be calibrated to minimize the size of the image region containing the hand. Minimizing the size of this image region speeds up the subsequent stages of the hand gesture recognition. Thus, the performance of the system depends on the calibration of the Kinect and the web-cam.

For this calibration we use the calibration of the depth sensor of the Kinect with its color sensor. Then, the color sensor of the Kinect is calibrated with the color sensor of the web-cam using standard calibration methods (OpenCV, 2011): First, identify two calibration objects in both images by their color and, second, determine the translational and scaling differences from the center points if these two objects in both images.



(a) Global coordinate system seen from the right Kinect.



(b) Global coordinate system seen from the left Kinect.

Figure 1: The common global coordinate system in a system of two Kinects. The white rectangles are the image regions for the hand gesture recognition using the HD color sensors.

4 Multiple sensor units

A gesture control system usually uses one depth sensor to detect persons. This sensor is placed in front of the user and as close to the virtual interaction surface as possible. However, for such a system some gestures might be occluded or undetectable due to the human operator's posture. Thus, we use multiple Kinects from different perspectives. Because multiple Kinects interfere with each other, we place the Kinects at least three meters apart at an angle of approximately 90°. In such a setup the interference is almost negligible. Furthermore, the setup is chosen such that a hand can be detected by at least one Kinect. This minimizes the probability of occlusions and enlarges the operating space.

If multiple Kinects are used, their depth sensors need to be calibrated. Each Kinect provides 3d position information relative to its own position of the body of the user. From this position information a common global coordinate system is computed. It is generated from three points which must be visible simultaneously for all Kinects. For example, we used the two hand positions P_l and P_r and the head position P_h of the user. The x-axis of the global coordinate system is defined as the vector connecting the two hands

$$e_x = \frac{P_r - P_l}{\|P_r - P_l\|}.$$

The z-axis is defined as the normal of the plane spanned by e_x and $P_h - P_l$

$$e_z = \frac{e_x \times (P_h - P_l)}{\|P_h - P_l\|}.$$

The y-axis is defined to yield a right-handed system of e_x, e_y, and e_z

$$e_y = e_z \times e_x.$$

For the origin of this coordinate system we chose for symmetry reasons P_h. Figure 1 shows the resulting common global coordinate system. Thus, the matrix transforming global coordinates to local coordinates of an individual Kinect is given by

$$M = \begin{pmatrix} e_x & e_y & e_z & P_h \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note, that for the calibration the head and both hands should not be collinear. Furthermore, the user should look at the virtual interaction interface to define the orientation of the global coordinate system such that e_z roughly points towards the projection screen.

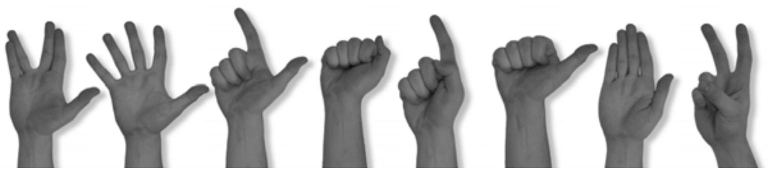


Figure 2: Set of static hand gestures.

5 Hand gestures

Typical static hand gestures are shown in Figure 2. Because the resolution of the depth sensor of the Kinect is too low to separate individual fingers at more than one meter distance, we use the image of a high resolution web-cam for the hand gesture recognition. The OpenNI framework (OpenNI, 2012) generates a skeleton of the tracked person. From this skeleton the positions of both hands are extracted. Using the hand position and depth its size can be computed. This information can be used to estimate a rectangular image region centered at the hand position containing the hand, see Figures 1 (a) and (b).

The image region in the high resolution color image is used to detect the hand using the skin color in HSV color space. Because hand detection based on skin color is not robust, we used colored gloves. Pink or neon green gloves are easier to detect than skin color. For the detection intervals for the hue $I_h = [h_{min}, h_{max}]$ and the saturation $I_s = [s_{min}, s_{max}]$ are defined. Using lookup tables the color (p_h, p_s, p_v) of each pixel p in the image region is set to $(\varepsilon, \varepsilon, 0)$ where

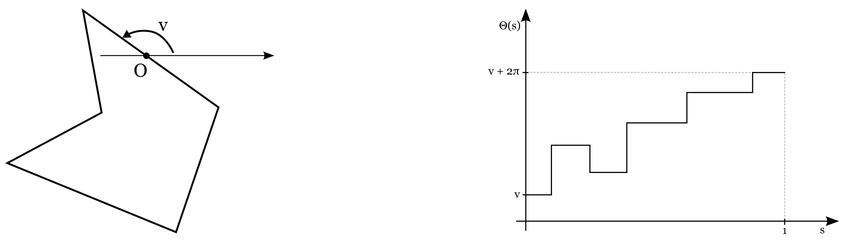
$$\varepsilon = \begin{cases} 1, & if\ p_h \in I_h \text{ and } p_s \in I_s \\ 0, & otherwise \end{cases}.$$

This yields a binary image region of the hand. The value channel is ignored, because it is too sensitive to environmental light. To get the components and contours of the hand in the binary image region a linear-time component-labeling algorithm using contour tracing is used (Chang and Chen, 2003). The resulting outer contour is extracted as polygon and simplified with the Douglas-Peucker algorithm (Douglas and Peucker, 1973). For the gesture recognition polygon matching is used. The polygon matching is based on a distance between two polygons. We tested a distance using Hu-moments (Hu, 1962) and the turning angle of a polygon (Arkin et al., 1991).

5.1 Hu-moment distance

The seven Hu-moments $h_i(A),\ i = 1, \ldots ,7$, for a polygon A are computed from the normalized central moments $\mu_{kl},\ k, l \leq 0$, of the image using Green's formula. For example the first Hu-moment is defined as $h_1(A) = \mu_{20} + \mu_{02}$. All other Hu-moments are defined similarly, see (Hu, 1962). They are invariant with respect to rotations, translations, and scalings. The distance d^H computed from these Hu-moments h_i is defined in (OpenCV, 2011) as

$$d^H(A, B) = \sum_{i=1}^{7} \left| \frac{\operatorname{sign}(h_i(A))}{\log(|h_i(A)|)} - \frac{\operatorname{sign}(h_i(B))}{\log(|h_i(B)|)} \right|.$$



(a) Polygon. *(b) Turning function.*

Figure 3: A polygon (a) and its turning function (b) (Arkin et al., 1991).

5.2 Turning-angle distance

To compute the *turning-angle distance* the turning function of both polygons are computed. The *turning function* $\theta_A(s)$ of a polygon A is defined on the arc length s and yields the counterclockwise tangent angle for each point on the boundary, see Figure 3. The computation starts at an arbitrary reference point O on the boundary of A (Arkin et al., 1991). The area between two turning functions of two polygons is the *turning-angle distance* d^T of these polygons. Since this area depends on the choice of the reference points O and the orientation of the polygons, the minimum distance between the two turning functions is used

$$d^T(A, B) = \min_{\theta \in \mathbb{R},\ t \in [0,1]} \left(\int_0^1 (\theta_A(s + t) - \theta_B(s) + \theta)^2 ds \right)^{1/2}.$$

Note for the implementation, which $\theta_A(s)$ is constant along the sides of the polygon.

6 3d gestures

With a system of depth and color sensors as described in Sections 3 and 4 true 3d gestures can be recognized: The 3d position of the hands and the hand gestures are known at any time. There are two types of 3d gestures: *static* and *dynamic* gestures. Static hand gestures are defined by the position and posture of the hand. An example is the *thumbs-up-gesture*, see sixth gesture in Figure 2. Such a gesture can be detected from a single image. For dynamic gestures also the motion of the hand and fingers matters. Thus, a set of consecutive images determines the gesture. An example is the *waving-hand-gesture*, which can only be recognized if the motion is detected.

For a 3d interaction system both types of gestures are necessary: static hand gestures to grab and release objects and dynamic hand gestures to transform objects. Furthermore, dynamic gestures need a start and end point, which can be triggered by static hand gestures. To grab an object we use the *closed-hand gesture* (fourth gesture in Figure 2) and to release it the *open-hand gesture* (seventh gesture in Figure 2). As long as the hand is open no interaction is triggered. As long as the hand is closed the dynamic gesture recognition runs until the hand is opened again. Static hand gestures are detected in the high resolution image using the techniques described in Section 5.

To detect dynamic gestures the difference between the current position P and the previous position P' of a hand is computed in each frame of the depth image. This difference yields a translation, rotation, or scaling depending on the interaction mode. For translations the user grabs the object with one hand to translated it along $T = P - P'$. For scalings and rotations the user grabs the object with both hands. This yields a difference for the left and the right hand or, equivalently,

$$d = P_r - P_l, \qquad d' = P_r' - P_l'.$$

to scaled the object along $S = d - d'$. For a rotation around the z-axis the angle between d and d' in the xy-plane is used. Dropping the third coordinate yields $\bar{d}$ and $\bar{d}'$. Then the angle between the x-axis and $\bar{d}$, $\bar{d}'$ are computed. The difference between these angles is the angle α_z for the rotation around the z-axis

$$\alpha_z = \mathrm{atan2}(\bar{d}) - \mathrm{atan2}(\bar{d}'), \quad \bar{d}, \bar{d}' \in \mathbb{R}^2 .$$

The rotations around the other two axes are computed similarly.

7 Implementation and results

For our prototype system we used two sensor units, each consisting of a Microsoft Kinect and a Logitech C910 web-cam, see Figure 4. Each device needs its own USB host controller to operate at maximal bandwidth. Thus, in addition to two on-board USB host controllers we used a PCI Express Card with two additional USB host controllers. This yields a resolution of 960 × 720 pixel for

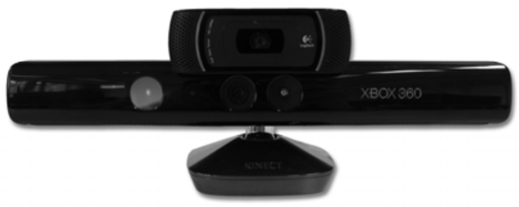


Figure 4: Sensor unit of a Kinect and a Logitech C910 web-cam.

the web-cams. The demo applications are implemented and tested with 8GB memory and an Intel i7-2600K CPU.

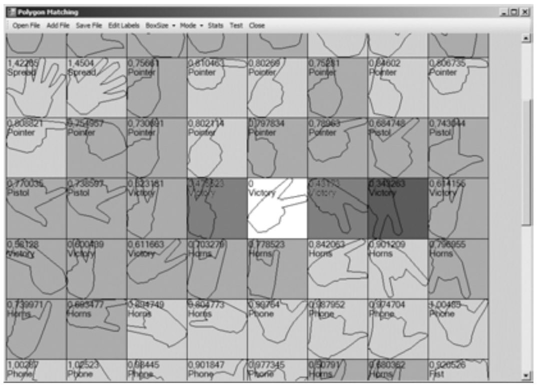
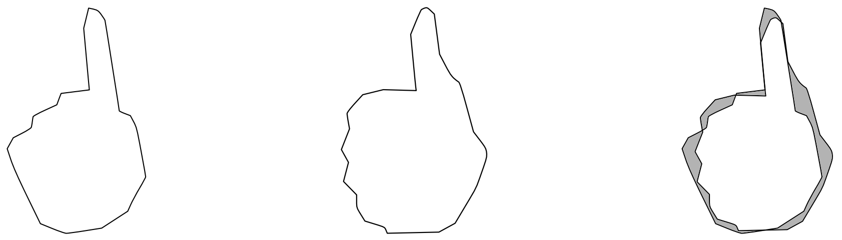


Figure 5: Demo application to compare distances of hand gestures showing the database of clean gesture prototypes. The color visualizes the polygon distance to a test gesture (white background), where background shades represent small, medium and large distances.

7.1 Hand gestures

To test the matching algorithms for hand gesture recognition we generated a database of 120 polygons representing eleven different gestures. The gesture polygons in this database are manually enhanced to ensure clean gesture prototypes. A second test database contains 144 unedited gesture polygons from three different persons using all eleven gestures. Figure 5 shows the GUI of the demo application to manage these databases and visualize distances between hand gestures.



(a) Pointing-forefinger gesture. *(b) Thumbs-up gesture.* *(c) Overlay of both gestures.*

Figure 6: The gestures in (a) and (b) are difficult to distinguish. The white area in (c) is the intersection of both polygons and the gray area is the difference of both polygons.

The polygon matching algorithm using the turning-angle distance identifies 85% of the test polygons. The polygon matching algorithm using the Hu-moments identifies only 58%. Our tests show that the turning-angle distance is better suited for static hand gesture recognition. However, it is computationally more expensive, because most of the computations for the Hu-moment distance can be done a priori when the gesture database is loaded.

Hand gestures with similar polygons like the *pointing-forefinger gesture* (fifth gesture in Figure 2) and the *thumbs-up gesture* are difficult to distinguish for both distances, Figure 6.

Using the Logitech C910 web-cam we experienced a problem if the hand is in front of a dark background, because dark image regions contain large rectangular artifacts in the hue channel causing frayed hand boundaries.

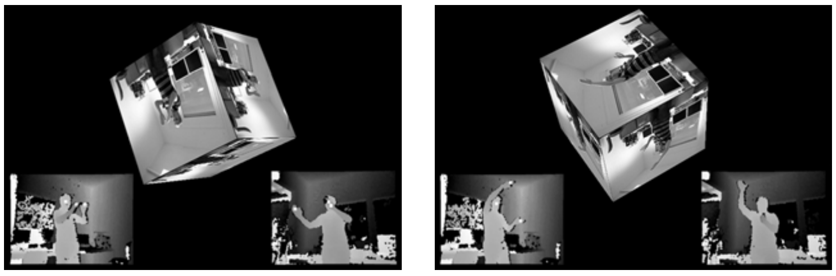


Figure 7: Rotation of a simple 3d scene using the demo application for 3d interaction.

7.2 3d gestures

For dynamic 3d gesture recognition we implemented a demo application, see Figure 7. At the bottom of the screen the depth-maps of both Kinect depth sensors are displayed. In the center of the screen a simple 3d scene consisting of one cube with the current color image mapped to its sides is displayed. This scene can be manipulated by dynamic 3d gestures. The demo application has four interaction modes for the two-handed manipulations: scaling and rotations around the three spatial axes. The rotation is mapped to three modes for the three axes to simplify the interaction. A translation of the scene can be done at any time since it is controlled by a one-handed gesture. The interaction modes are selected by pointing with one hand at specialized areas near the top of the window, not displayed in Figure 7.

7.3 Performance

Running our demo implementation on a computer with Intel i7 2600K processor (4 cores with hyper-threading) the static hand gesture recognition runs at 30 frames per second using one thread. The average load for the dynamic 3d gesture recognition is below 10%. So, most resources of the system are still available while using this input method. The complete interaction system has been tested by ten volunteers. Each volunteer had at first significant problems with the 3d control. However, after a short training of one to two minutes all volunteers got used to the 3d control and achieved their goals quickly and intuitively.

We experienced no problems due to misclassifications of gestures. However, there were contradicting classifications in some frames, due to the even number of sensor units used. We detected these contradicting classifications in the log of the system. These contradicting classifications did not jam the interaction, because of the high frame rate, i.e. in subsequent frames these contradictions were resolved.

8 Conclusion and outlook

We presented an effective real-time 3d gesture recognition system. Static hand gestures are used to start and stop dynamic 3d gestures. Our demo implementation allows for intuitive interaction with 3d objects.

8.1 Hand gestures

For the future we plan to avoid colored gloves using improved skin color detection or a gradient based approach. For the skin color detection also patches extracted from faces, using face localization, to generate dynamic filters can be used (Liu et al., 2011; Tan et al., 2012). Using two sensor units different gestures might be recognized. For a robust decision for the gesture recognition additional information is necessary, e.g. a third sensor unit.

8.2 3d gestures

In the demo implementation the interaction mode is selected manually. Multiple static hand gestures could be used for a gesture based selection. Also more complex interaction modes will be implemented. For example a rotation around an arbitrary axis could be initialized using a static two hand gesture defining the axis direction through the hand positions. For fast gestures, a synchronization of the multiple sensor units is necessary. The OpenNI framework (OpenNI, 2012) plans to support that in future versions.

References

Arkin, E., Chew, L., Huttenlocher, D., Kedem, K., Mitchell, J., 1991. *An efficiently computable metric for comparing polygonal shapes.* IEEE PAMI 13 (3), 209–216.

Chang, F., Chen, C.-J., 2003. *A component-labeling algorithm using contour tracing technique.* In: Seventh International Conf. on Document Analysis and Recognition. pp. 741–745.

Douglas, D. H., Peucker, T. K., 1973. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature.* Cartographica: Intl. J. for Geographic Information and Geovisualization 10 (2), 112–122.

Ganapathi, V., Plagemann, C., Thrun, S., Koller, D., 2010. *Real time motion capture using a single time-of-flight camera.* In: IEEE CVPR, 755–762.

Ghobadi, S. E., Loepprich, O. E., Ahmadov, F., Bernshausen, J., Hartmann, K., Loffeld, O., 2008. *Real time hand based robot control using multimodal images*. IAENG Intl. J. Comp. Sci. 35 (4), 500–505.

Ghobadi, S. E., Loepprich, O. E., Hartmann, K., Loffeld, O., 2007. *Hand segmentation using 2d/3d images.* In: Cree, M. J. (Ed.), IVCNZ, 64–69.

Hu, M.-K., 1962. *Visual pattern recognition by moment invariants.* IRE Trans. on Info. Theory 8 (2), 179–187.

Liu, L., Sang, N., Yang, S., Huang, R., 2011. *Real-time skin color detection under rapidly changing illumination conditions.* IEEE Trans. on Consumer Electronics 57 (3), 1295–1302.

Malassiotis, S., Aifanti, N., Strintzis, M., 2002. *A gesture recognition system using 3d data.* In: 1st Intl. Symp. on 3d Data Processing, Visualization, and Transmission, 190 – 193.

OpenCV, 2011. *OpenCV Documentation*, URL docs.opencv.org/.

OpenNI, 2012. *OpenNI Documentation*, URL www.openni.org/documentation.

Plagemann, C., Ganapathi, V., Koller, D., Thrun, S., 2010. *Real-time identification and localization of body parts from depth images.* In: IEEE ICRA, 3108–3113.

PrimeSense Inc., 2010. *Prime Sensor NITE 1.3 Framework Programmer's Guide.*

Sánchez-Nielsen, E., Antón-Canalís, L., Hernández-Tejera, M., 2004. *Hand gesture recognition for human-machine interaction.* In: WSCG, 395–402.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., 2011. *Real-time human pose recognition in parts from single depth images.* In: IEEE CVPR, 1297 –1304.

Sturman, D. J., Zeltzer, D., 1994. *A Survey of glove-based Input.* IEEE CG&A 14 (1), 30–39.

Tan,W. R., Chan, C. S., Yogarajah, P., Condell, J., 2012. *A fusion approach for effcient human skin detection.* IEEE Trans. on Industrial Informatics 8 (1), 138–147.

Wormell, D., Foxlin, E., 2003. *Advancements in 3d interactive devices for virtual environments.* In: Proceedings of the workshop on virtual environments, 47–56.

Contact information

Manuel Caputo, Klaus Denker, Benjamin Dums, Georg Umlauf[1]
macaputo|kdenker|bedums|umlauf@htwg-konstanz.de
Faculty of Computer Science, HTWG Konstanz, 78462 Konstanz, Germany

[1] corresponding author