Open access • Proceedings Article • DOI:10.1109/WHC.2011.5945505

Using Kinect for hand tracking and rendering in wearable haptics — Source link

Valentino Frati, Domenico Prattichizzo

Institutions: University of Siena

Published on: 21 Jun 2011 - World Haptics Conference

Topics: Wearable computer, Haptic technology and Rendering (computer graphics)

Using Kinect for hand tracking and rendering in wearable haptics

Valentino Frati *
SIRSLab, Dipartimento di Ingegneria dell'Informazione
Università di Siena, Siena, Italia

Domenico Prattichizzo †
SIRSLab, Dipartimento di Ingegneria dell'Informazione
Università di Siena, Siena, Italia
Istituto Italiano di Tecnologia, Genova, Italia

ABSTRACT

Wearable haptic devices with poor position sensing are combined with the Kinect depth sensor by Microsoft. A heuristic hand tracker has been developed. It allows for the animation of the hand avatar in the virtual reality and the implementation of the force rendering algorithm: the position of the fingertips is measured by the hand tracker designed and optimized for Kinect, and the rendering algorithm computes the contact forces for wearable haptic display. Preliminary experiments with qualitative results show the effectiveness of the idea of combining Kinect and wearable haptics.

1 INTRODUCTION

Kinect [1] is a new game controller technology introduced by Microsoft in November 2010. Since its launch date it was evident that Microsoft's device is transforming not only computer gaming but also many other applications like robotics and virtual reality, thanks to its ability to track movements and voices, and even identify faces, all without the need for any additional devices [2]. Kinect (Fig. 1) interprets 3D scenes from a continuously-projected infrared structure. It has a webcam–like structure and allows users to control and interact with a virtual world through a natural user interface, using gestures, spoken commands or presented objects and images. The device includes a RGB color camera, a depth sensor and a multi-array microphone. It provides full-body 3D motion capture, facial and gesture recognition. The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, and allows the Kinect sensor to process 3D scenes in any ambient light condition [1]. The depth sensor technology was developed by Israeli PrimeSense [3]. It interprets 3D scene information from a continuously-projected infrared structured light. A variant of image-based 3D reconstruction was used to recover the depth of the observed points in the 3D scene [4]. Kinect is able to simultaneously recognize up to six people, including two moving players, for motion analysis with a feature extraction of 20 joints per active player.

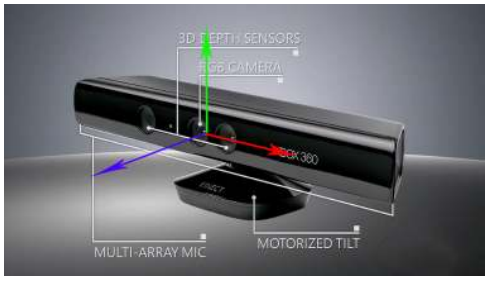

Figure 1: The Kinect device by Microsoft with the Kinect or camera reference frame. The z–axis is pointing out of the camera (courtesy of Microsoft).

To the best of our knowledge, the first attempt to use Kinect in haptics was made in a demo developed at the BioRobotics Laboratory of the University of Washington [5]. The demo consists of using Kinect to build a depth image of 3D scenes with objects which can be virtually touched using haptic devices like the PHANTOM or other devices developed by the same research group.

The approach proposed in this paper is different. We want to use the Kinect technology not to build a digital map of the environment but as an inherent component of the haptic display itself. In particular, we want to use the Kinect sensor to identify the position of the avatar in the virtual environment. In other words, we are not focused on the object but on the organ involved in the touch experience: the human hand.

The main objective of this study is to develop a wearable haptic device which we consider one of the promising research areas in haptics. To cite an example of wearable haptic device, consider the *gravity grabber* developed in [6]. It is a wearable device with two motors for force feedback at fingerpad contact point. The main difference of this kind of device with respect to the more sophisticated and involved exoskeletons [7] is that the kinesthetic feedback is practically missed and the force feedback is mainly tactile [8]. The other disadvantage is that the sensing part of the human hand or arm is poor.

Solving the problem of very poor sensing in wearable haptics is what we want to do with the use of Kinect technology.

Sensing or tracking the human hand is paramount in haptics for locating the avatar in the virtual environment and for implementing the force rendering algorithm in haptic devices of the impedance type as the one we are using in this work. Many solutions are possible in order to mitigate this problem, such as the use of sensorized gloves or markers or cameras. However all these approaches were found to be poor in terms of performance or very expensive, as in the case of markers and motion trackers. Moreover, these solutions make use of additional devices to be worn by the users thus making the overall system cumbersome.

This paper presents one of the first attempts to use Kinect as a sensor for wearable haptics. The example we focus on is the virtual pinch grasp of an object with two contact points. It is worth underlining that the advantages of wearable haptic devices are even more evident when multi–point interaction is considered [9].

2 KINECT CALIBRATION PARAMETERS

Kinect has been developed for Xbox 360 and currently Microsoft has not released the SDK, or an official driver. However, independent developers offer solutions for using Kinect separate from the game console and for the most common operating systems. Using the CLNUI Platform [10] or the OpenNI Platform [11] (by PrimeSense), it is possible to use Kinect under Windows OS with all libraries needed for data processing. In this paper, we used the CLNUI Platform which has a simple set of functions to retrieve data from the device (depth raw data, color depth data, color RGB

*e-mail: frati3@student.unisi.it
†e-mail: prattichizzo@ing.unisi.it

IEEE World Haptics Conference 2011
21-24 June, Istanbul, Turkey
978-1-4577-0298-3/11/$26.00 ©2011 IEEE

data) and to control the motor of Kinect.

The most relevant data to this study are taken from Kinect's depth sensor. These are 11 bit values and represent the raw value d_{raw} of the depth of a point p of the 3D scene. According to the calibration procedure developed in [12, 13], one gets

$$d = K\text{tang}(Hd_{\text{raw}} + L) - O$$

where $H = 3.5\,10^{-4}$rad, $K = 12.36$cm, $O = 3.7$cm and $L = 1.18$rad and d is the Kinect camera depth of p expressed in cm. This tangential approximation has a sum of squared difference of 0.33cm^2 for the calibration data. Once the depth has been obtained using the calibration function above, we can recover the complete coordinate vector for point p in the Kinect camera frame. Let (i, j) be the coordinates (pixels) of the projection of point p onto the Kinect camera frame (Fig. 1). Let (x, y, z) be the coordinates of the 3D point p in the camera frame expressed in cm. In [14] the authors proposed the following equations to compute vector (x, y, z) from projection (i, j) and depth d for point p

$$\begin{aligned} x &= (i - c_x) f_x d \\ y &= (j - c_y) f_y d \\ z &= d \end{aligned}$$

where $f_x = 0.5942143$, $f_y = 0.5910405$, $c_x = 339.3078$ and $c_y = 242.739$. Finally, let us report some consideration on accuracy. In [14] the authors found that the accuracy for point p reconstruction from the Kinect depth camera is lower than 1cm. For more details on the accuracy of measurements that one can get with this sensor we refer the reader to the *Kinect node* of the ROS project at MIT [15].

3 The Hand Tracking Algorithm

The proposed hand tracking algorithm consists of the following steps performed at each time sample:

1. read and process the depth image,
2. compute the bounding box of the hand,
3. extract the main hand points, like fingertip positions,
4. filter trajectories.

The tracking algorithm is implemented with the OpenCV library [16]. OpenCV is a stable and advanced library for 2D and 3D images with a set of useful functions for object recognition and geometric interpretation. The main OpenCV functions necessary for the algorithm implementation are presented.

Process the depth image

Read the Kinect depth map and build a grayscale image of the 3D scene. For each point with image projection (i, j) the 11bits depth variable read from the Kinect sensor is converted in an 8bits depth variable. In Fig. 2 an example of the depth image is reported.

Compute the hand bounding box

From the depth image, find the bounding box including the hand which is supposed to be the closest object to the Kinect sensor. In haptic interactions with virtual environments, this assumption is not a limitation because the user typically interacts with virtual objects and consequently no object is interposed between Kinect and the human hand to be tracked. To compute the bounding box of the hand a threshold approach is used. With lower and upper thresholds, depths of points too far or too close to the Kinect camera are set to zero. In our experiments the optimal value for the hand distance resulted to be 70cm. It is a appropriate value since not too much details is lost regarding the hand's points with the Kinect sensor.

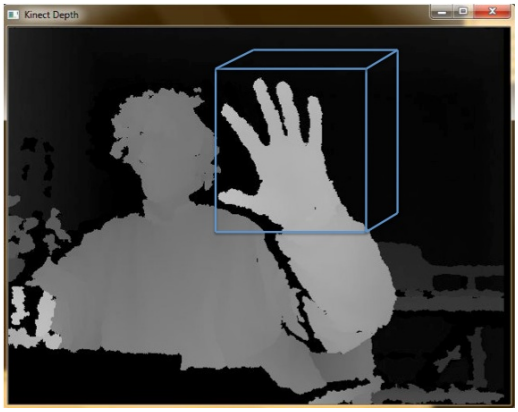

Figure 2: Depth image obtained from the Kinect's sensor with the hand's bounding box.

An example of the bounding box of the hand is reported in Fig. 2. Note that a 3D bounding box is considered here. In the next step we will refer to the contour of the 2D image.

From 2D hand contour to fingertip positions

The contour of the hand in the image cropped by the bounding box is evaluated. The function used to find the contour for objects in OpenCV [16] is `cvFindContours()`: it takes a binary image and returns the number of retrieved contours. The binary image is computed from the cropped image. Once the contour is obtained, it is possible to compute its convex hull using the OpenCV function `cvConvexHull()`. The points of the hull represent the external contour of the hand, from the wrist to the fingers if the hand is open as in Fig. 3.

This set of points is necessary for the function used to identify the fingers: `cvConvexityDefects()`. The routine takes the contour and the convex hull and it computes the defects of convexity returning for each defect a structure with the start point, corresponding to the tip of a finger, the depth point and the end point, corresponding to the tip of the adjacent finger. The end point is the point where the convexity defect ends. The maximum distance of the convexity from the hull is also returned. In Fig. 3 an example of defects computation is reported.

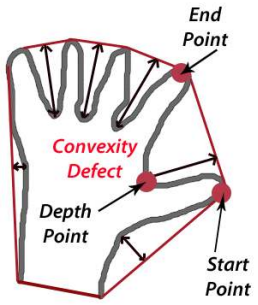

Figure 3: An example of defects computation. The algorithm detects the fingertips, in particular of the index and the thumb, which correspond to the start and end points of a convexity defect.

Convexity defects provide relevant information for localizing the

hand in the 3D space. In particular the defects, which are closer to the convex hull, determine the fingertip's position. Other important points are the center of mass of the hand and the two wrist points at the vertices of the convex hull. These three points allow for the localization of the palm and its reference frame in the virtual reality. Fig. 4 reports a more detailed image of the main elements of the proposed hand tracking algorithm.

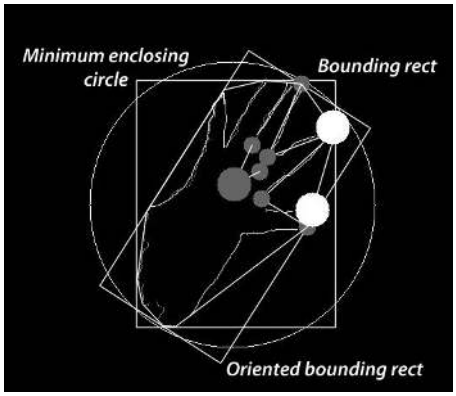

Figure 4: Tracking window: the algorithm computes the hand's bounding box, minimal enclosing circle, and the oriented bounding box. The tips of the thumb and index finger and the hand's center, are also reported.

Filtering

All the relevant values obtained from the previous steps are filtered and smoothed by a Kalman filter. The main OpenCV function in this case is `cvCreateKalman()`. For a generic relevant point $p = (x, y, z)$, the filter estimates the state vector s consisting of the point position p and its velocity v. The dynamic and the measurement (m) equations are given by

$$\begin{cases} s(k+1) &= As(k) + w(k) \\ m(k) &= Cs(k) + v(k) \end{cases}$$

where

$$A = \begin{bmatrix} I & \Delta t \\ 0 & I \end{bmatrix} \quad \text{and} \quad C = \begin{bmatrix} I & 0 \end{bmatrix}$$

being Δt the time interval between two iterations (it is not lower than $1/30$, being 30fps the frame rate of the Kinect device) and $w(k)$ and $v(k)$ represent the dynamics and measurement noise. They are assumed to be independent from each other, white, and with normal probability distributions with covariance Q and R, respectively. Details on the choice of covariance matrices and filter initialization are reported in [17].

4 Limitations of the proposed algorithm

In this section the main limitations of the proposed algorithm are discussed. The main error affecting the proposed tracking algorithm regards the second step. The hand tracker is based on heuristics. In order for the algorithm to work properly, it is important that the largest part of the hand is visible. Several heuristics are used for instance to define the finger's identity and these may fail under occlusions of the whole or part of one or more fingers. The worst condition generating occlusions is when the fingers are perpendicular to the Kinect camera plane xy. In this case no depth information can be retrieved for large part of the hand. The optimal distance between the hand to be tracked and the Kinect sensor is 70cm. Farther away the hand starts to become too small, and some depth information gets lost or becomes very poor in terms of accuracy.

Figure 5: Two Omega devices were used in [18] for 3D real–time haptics interaction.

5 Hand animation and haptic feedback

The proposed hand tracking algorithm allows for the animation of the hand avatar as discussed in [18] where two grounded haptic devices are used to track the motion of the index and of the thumb as in Fig. 5. Here we substitute the measurement of the fingertip positions with data provided by the Kinect hand tracking algorithm: the index and thumb tip measurements are provided by Kinect as in Fig. 3. Of course in this case we have more measurements to better track the human hand than those provided by the haptic interfaces only. The kinematic model of the human hand avatar is the one proposed in [18]. The biomechanical model of the hand avatar has 26 joints. The hand avatar is animated by the data provided by the tracker. The center of mass of the hand and the measurement of the wrist position allow for the position and rotation of the wrist in the virtual environment.

In Fig. 6 the experimental setup is reported. The Kinect sensor is on the desk pointed towards the hand. The user wore two wearable devices and the main application's windows are shown on the screen of the computer: in the right window the hand tracking is reported and in the left window the rendering of the hand avatar.

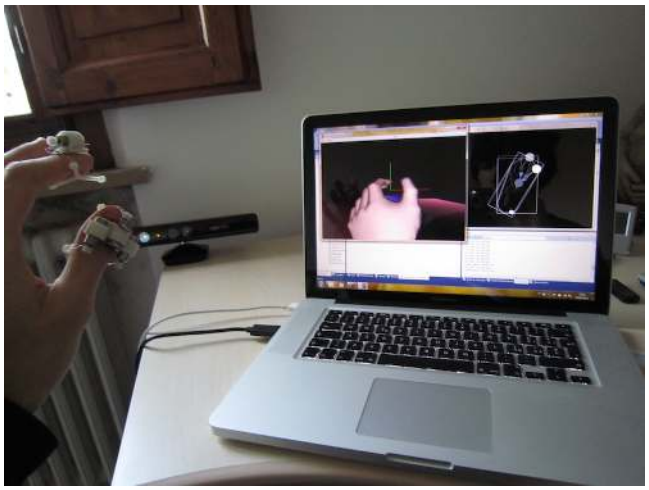

Figure 6: The Kinect–based wearable haptics: the Kinect sensor is in front of the hand. On the screen the rendered scene with the hand avatar (left) and the main points of the tracking algorithm (right).

As far as haptic rendering is concerned, differently from [18]

where the force feedback at the index and thumb was provided by two grounded haptic devices, in this paper we use wearable devices. These devices were developed at the University of Siena and are similar to those developed by K. Minamizawa in [6]. The prototypes of the new devices are those reported in Fig. 6. The force feedback is applied at the contact point on the finger pad and is controlled by three motors. Three independent contact force direction at the contact can be controlled in our haptic devices which as the ones developed in [6], provide cutaneous feedback only and lack of kinesthetic feedback [8]. The haptic devices have no sensing.

As discussed in the introduction we believe that the Kinect technology will help spreading the use of wearable haptics in consumer electronics. The complete analysis of haptic devices used is beyond the scope of this paper. Description and characterization of the new devices in Fig. 6 can be found in [19]. The hand tracking performance in Section 3 is not affected by the presence of the haptic devices worn by the user as in Fig. 6. These devices are small and can be considered minimally invasive for the Kinect sensor when compared to the whole hand. To improve the hand tracking with the worn device, additional image pre-processing phases were required in order to remove parts in the image that did not belong to the hand. This was done partly with color filtering techniques and with erosion and smoothing techniques of images as implemented by the `cvErode()` in OpenCV.

In the following we present a preliminary experiment showing the viability of using the Kinect sensor in haptics and in particular with wearable devices with or without poor position sensing. The experiment consists of grasping and moving a cube as in Fig. 7, with the thumb and the index. The cube whose edge length is $l_b =$ 10cm is assumed to have uniformly distributed mass $m_b = 0.1$Kg. The Open Dynamics Engine (ODE) library [20] has been used to simulate dynamics. The two contact points have been simulated as point contacts with friction. A proxy algorithm [21] is used for force rendering. The stiffness is the same along the three axes of the reference frame at the two contacts and in particular for each direction $K = 60$N/m. The friction coefficient for both contacts is set to $\mu = 0.5$. The contact points on the finger are assumed to be at the tips of the thumb and index fingers as measured by the tracking algorithm.

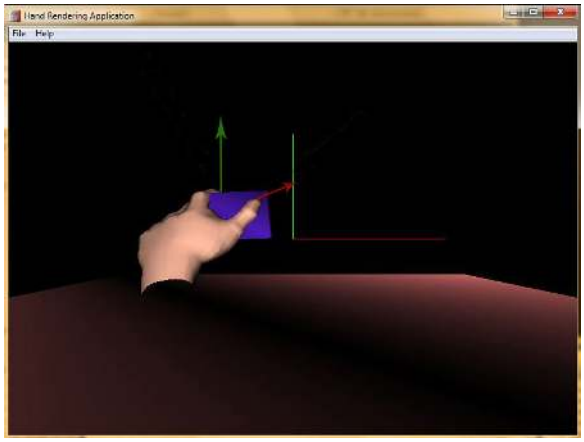

Figure 7: Example of object grasping simulation: the colored lines at the contact points are the force vectors proportional to the force intensity.

A grasping task is described here with reference to the Kinect camera frame. Once the user has reached the virtual object, she/he grasps the object with the thumb and the index. Then the cubic object is raised up for about 12cm from its original position. It is moved to the right along the x-axis for about 20cm. Finally it is moved down to the floor. The object's trajectory is reported in Fig. 8. During the experiment, the force rendering algorithm computed the forces to be displayed by means of the wearable devices [19]. The intensity of the contact forces are reported in Fig. 9 for the contact at the thumb and in Fig. 10 for the index finger.

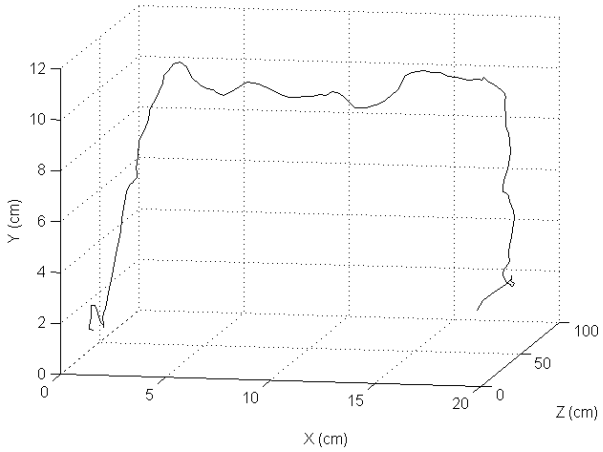

Figure 8: The trajectory of the object during the grasping simulation.

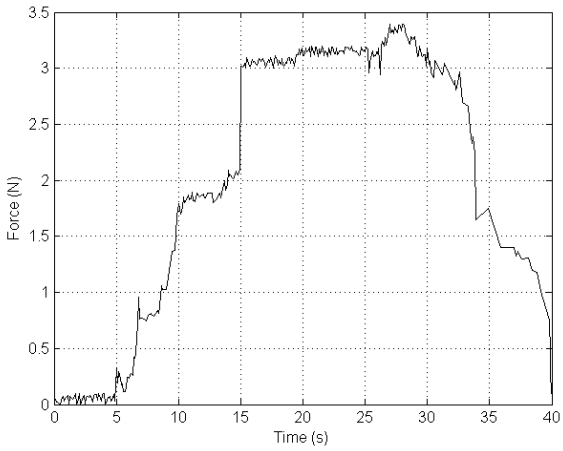

Figure 9: Rendered contact force at the thumb. The simulated contact stiffness is 60 N/m.

The entire process: data acquisition from Kinect, hand tracking, collision detection, force rendering, and 3D scene dynamics run in real time at the frame rate of Kinect. The preliminary experiment was performed with a Core 2 Duo CPU at 3.06Ghz, under Windows 7. Note that there was a relevant computational margin since the CPU's use rarely exceeded 50% percent of its capabilities. In Fig. 11, the execution time is reported for the first 70 iterations. Note that the initial value of the execution time is higher because of the initialization procedures of the software environment. After the first couple of iterations, the range of execution times reaches an average of 0.03 seconds.

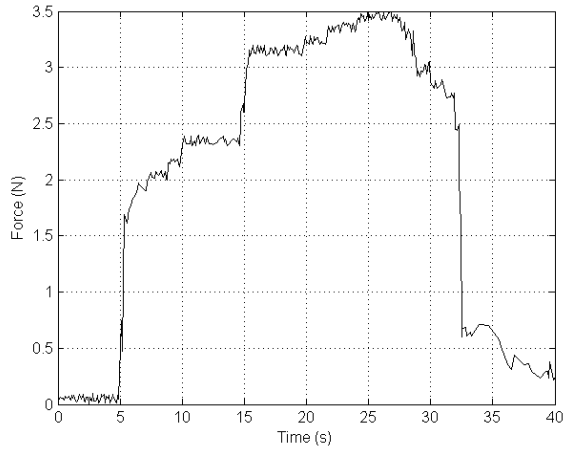

Figure 10: Rendered contact force at the index finger. The simulated contact stiffness is 60 N/m.

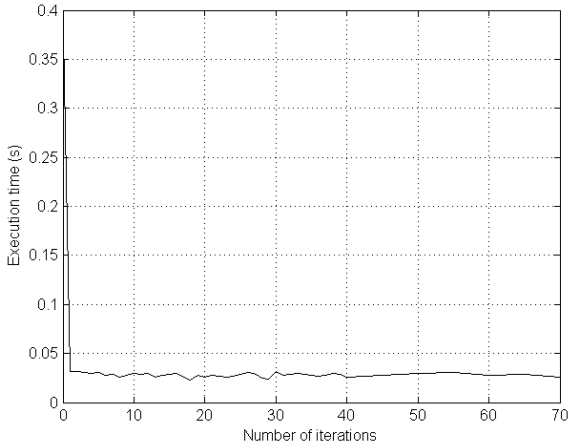

Figure 11: Execution time per iteration for the first 70 time samples. On the y axis the execution time of each single iteration in seconds is reported.

The hand's avatar and the entire scene has been developed using the Microsoft DirectX libraries. Further details on hand animation are given in [17].

6 Conclusions

In this paper we propose a solution based on the new Kinect technology by Microsoft to compensate for the lack of position sensing in wearable haptics. The qualitative results of the preliminary experiments showed the viability of combining wearable haptics and the Kinect technology using a hand tracking algorithm based on some heuristics. We are convinced that the preliminary results of this paper will open new horizons to the use of wearable haptics in consumer electronics.

Acknowledgments

The authors wish to thank Francesco Chinello and Monica Malvezzi for his help in designing the wearable haptic devices and Adrian Ramos for his help in the software implementation.

This work has been partially supported by the European Commission with the Collaborative Project no. FP7-ICT-2009-6 270460, "ACTIVE: Active Constraints Technologies for Ill-defined or Volatile Environments" and with the Collaborative Project no. 248587, "THE Hand Embodied", within the FP7-ICT-2009-4-2-1 program "Cognitive Systems and Robotics."

References

[1] Wikipedia. Kinect — Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Kinect, 2011. [Online; accessed 13-January-2011].

[2] J. Giles. Inside the race to hack the Kinect. *The New Scientist*, 208(2789):22–23, 2010.

[3] Prime Sense. Prime sense kinect's hardware. http://www.primesense.com, 2010. [Online; accessed 8-January-2011].

[4] How kinect depth sensor works. stereo triangulation? http://mirror2image.wordpress.com/2010/11/30/how-kinect-works-stereo-triangulation, Nov 2010.

[5] University of Washington. Microsoft's Craig Mundie demos Haptics-Kinect work from Biorobotics Lab. http://www.ee.washington.edu/news/2011/CraigMundieHaptic.html, January 2011. [Online; accessed 7-January-2011].

[6] K. Minamizawa S. Fukamachi H. Kajimoto N. Kawakami and S. Tachi. Gravity grabber: wearable haptic display to present virtual mass sensation. *ACM SIGGRAPH*, 2007.

[7] A. Frisoli, F. Rocchi, S. Marcheschi, A. Dettori, F. Salsedo, and M. Bergamasco. A new force-feedback arm exoskeleton for haptic interaction in virtual environments. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint*, pages 195–201. IEEE, 2005.

[8] K. Minamizawa, D. Prattichizzo, and S. Tachi. Simplified design of haptic display by extending one-point kinesthetic feedback to multipoint tactile feedback. In *IEEE Haptic Symposium*, Waltham, Massachusetts, USA, 2010.

[9] F. Barbagli, D. Prattichizzo, and J.K. Salisbury. *Multi-Point Physical Interaction with Real and Virtual Objects*, volume 18 of *STAR, Springer Tracks in Advanced Robotics*. Springer Verlag, Berlin Heidelberg, 2005.

[10] Code Laboratories. Clnui sdk and drivers. http://www.codelaboratories.com, 2011. [Online; accessed 14-January-2011].

[11] Prime Sense. Openni platform. http://www.openni.org, 2011. [Online; accessed 13-January-2011].

[12] Open kinect imaging information. http://openkinect.org/wiki/Imaging_Information, 2010.

[13] ROS (MIT). Ros kinect calibration, 2011. [Online; accessed 10-January-2011].

[14] Matthew Fisher. Kinect study. http://graphics.stanford.edu/~mdfisher/Kinect.html, 2010.

[15] ROS (MIT). Ros implementation of hand tracking. http://www.ros.org/wiki/kinect_tools, 2011. [Online; accessed 13-January-2011].

[16] Opencv computer vision library. http://opencv.willowgarage.com/wiki/, 2011.

[17] V. Frati and D. Prattichizzo. Using kinect technology to track hands in haptic applications. Internal report of the University of Siena, January 2011.

[18] M. Malvezzi S. Mulatto, A. Formaglio and D. Prattichizzo. Animating a deformable hand avatar with postural synergies for haptic grasping. *Computer Science*, 6191(2789), 2010.

[19] F. Chinello, M. Malvezzi, and D. Prattichizzo. Design and control of a new wearable haptic device for fingerpad contact simulation. Internal report of the University of Siena, December 2010.

[20] Open Dynamics Engine. Ode library. http://www.ode.org, 2010. [Online; accessed 13-January-2011].

[21] CB Zilles and JK Salisbury. A constraint-based god-object method for haptic display. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pages 146–151. IEEE, 2002.