

Robust Silhouette Extraction from Kinect Data

Michele Pirovano^{1,2}, Carl Yuheng Ren³, Iuri Frosio¹, Pier Luca Lanzi²,
Victor Prisacariu³, David W. Murray³, and N. Alberto Borghese¹

¹ Department of Computer Science, University of Milano, Milano, Italy
{pirovano, frosio, borghese}@di.unimi.it

² Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Italy
pierluca.lanzi@polimi.it

³ Department of Engineering Science, University of Oxford, Oxford, England
{carl, victor, david}@robots.ox.ac.uk

Abstract. Natural User Interfaces allow users to interact with virtual environments with little intermediation. Immersion becomes a vital need for such interfaces to be successful and it is achieved by making the interface invisible to the user. For cognitive rehabilitation, a *mirror view* is a good interface to the virtual world, but obtaining immersion is not straightforward. An accurate player profile, or silhouette, accurately extracted from the real-world background, increases both the visual quality and the immersion of the player in the virtual environment. The Kinect SDK provides raw data that can be used to extract a simple player profile. In this paper, we present our method for obtaining a smooth player profile extraction from the Kinect image streams.

1 Introduction

Natural user interfaces based on devices such as the Microsoft Kinect have revolutionized video-gaming: the player is represented by an avatar inside the game and interacts with virtual objects, resulting in a more realistic and intuitive control experience. Furthermore, Microsoft has released a SDK [1] that allows easily integrating Kinect as an input device that provides the motion of the player, defined as a set of segments connected by joints. The pose is described by a set of quaternions, one for each joint, along with the translation of the root joint located at the hip level.

A player tracking modality that is much less explored, is that of tracking the player's *profile*. This has been shown to be particularly effective in cognitive rehabilitation games as it allows the patient to see himself mirrored inside the virtual environment while performing exercises. It has been hypothesized that such modality activates *mirror neurons* [2, 3] that might drive a cortical reorganization to regain the lost function. Moreover, such a *mirror view* largely increases the sense of immersion inside the game. The DuckNeglect platform [4] has been taking advantage of this to design exploratory mini-games that, from preliminary results, have been shown to be

effective in treating neglect patients¹. DuckNeglect forces the patient to explore his neglected hemi-field requiring him to reach targets inside a virtual scenario and to avoid distractors. The patient receives feed-back on his movement in real-time through his profile moving inside the scenario. The limitation of DuckNeglect is the use of a 2D camera that requires uniform lighting and background to achieve a robust and accurate extraction of the patient profile: the gaming space has to be specially set-up and this prevents the patient from exercising at home where he could exercise more intensively and feel more comfortable.



Fig. 1. - The input data for the sitting player. (a) The color image. (b) The depth image. Magenta points in the depth map have an unknown value. (c) our extracted player profile.

The Kinect has the potential to make silhouette extraction more robust through the integration of depth and color data. However, its SDK does not produce a good enough profile. This is due to two main reasons: the depth-to-color calibration parameters provided internally in the SDK are not sufficiently accurate [6], so that the alignment of the two images is inaccurate (Fig. 3b). Moreover, once the two images are aligned, a sharp identification of the profile is not produced due both to the low resolution of the depth map and to noise. The result may be sufficient for entertainment games such as *Get Fit With Mel B*², but it can be annoying and distracting to the rehabilitating patient who generally moves slowly and therefore is aware of all the little inaccuracies on the screen, resulting in a decreased sense of presence.

We show here how we can improve the quality of the player profile through different steps. We first achieve a better alignment of the color and depth images through an innovative on-the-field calibration of the Kinect sensor that only requires moving a ball in front of the device. With the computed parameters, we extract in real time a smooth and robust player silhouette by filtering the player color image identified through the depth map.

The paper is structured as follows. In section 2, we present the materials and the proposed methods for extracting the silhouette. Section 3 presents the application of our methods to a serious game. We draw conclusions in Section 4.

¹ Neglect is the inability of a person to process and perceive stimuli on one side of the body or environment. Such inability is not due to a lack of sensation [5].

² uk.gamespot.com/get-fit-with-mel-b

2 Materials and Methods

The Kinect sensor integrates a color and a depth camera. The color camera outputs an RGB image I_c at a standard resolution of 640x480 pixels at 30 frames per second (fps) (Fig. 1a). The depth camera consists of an infrared sensor paired with an infrared emitter that projects onto the scene an irregular circular pattern (Fig. 2a). The depth map image I_d (Fig. 1b) is recovered through triangulation and speckle pattern decorrelation [7]; it has a maximum resolution of 640x480 pixels, which is actually up-sampled from a resolution of 320x240 pixels at 30 fps. The depth map returned by the Kinect sensor is bound to be noisy, thus a bilateral filter is internally applied to it. The result is a smoother depth map that preserves sharp edges but is still noisy and holds many unknown values. In addition, even with a static scene, the depth reading for each pixel is often not consistent between two consecutive frames, resulting in a flickering visualization of the depth stream. The SDK isolates and identifies up to 6 different players in I_d assigning a different index to each player. This can be used to create an effective silhouette extraction procedure (cf. [4]): an *alpha bitmask* image I_b is created independently for each player where each pixel is set to 1 (visible) if the player index for the pixel corresponds to the player and to 0 (invisible) otherwise. An RGBA player image I_p is created by combining the RGB color of each pixel in I_c with the corresponding alpha value in I_b .

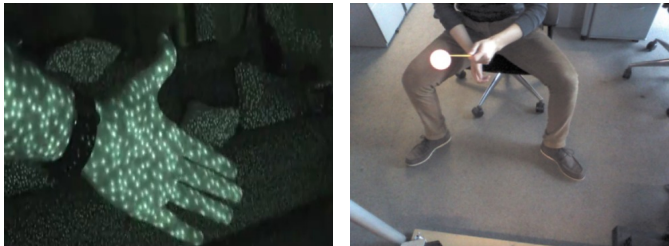


Fig. 2. On the left the typical pattern projected by the infrared emitter³. On the right, calibrating Kinect with a LED ball.

However, the direct overlap of I_b with I_c produces an incorrectly aligned I_p as seen in Fig. 3a. This happens because the Kinect's color and depth sensors are not aligned, either in time (the acquisitions from the two sensors, even at the same frequency, are up to 7ms apart), or in space, as the images are acquired from different viewpoints. If time difference can be dealt directly with the function *AllFramesReady* of the SDK that guarantees a maximum of 3ms delay between the frames of different cameras⁴, the spatial alignment obtained through the SDK can instead be improved (Fig. 3b-c).

³ from www.youtube.com/watch?v=nvvQJxgykcU

⁴ from <http://social.msdn.microsoft.com/Forums/s1-SI/kinectsdknuapi/thread/2e172449-3d18-4914-9370-fbc1cfa31aa6>

2.1 Depth to Color Camera Calibration

Methods for a complete photogrammetric calibration of the Kinect camera have been proposed (e.g. [7-8]), but they require grids and structured markers and are not suitable for the home environment. We propose here an efficient on-the-field calibration method for any Kinect device that can accurately recover the geometrical parameters of the stereo-system constituted of the color and depth camera by tracking a moving ball. It is based on an adapted version of bundle adjustment [14].

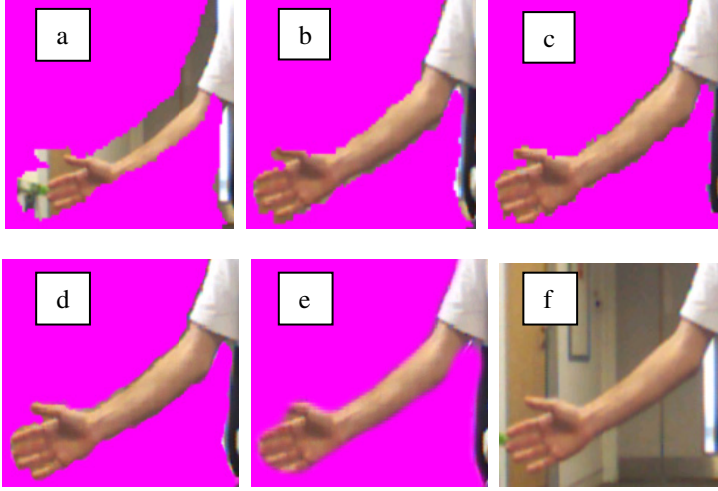


Fig. 3. In panel (a) a close-up of the silhouette produced by directly overlapping depth and color data: the offset is evident. A residual offset is still present when using SDK to extract silhouette (b): the background is visible below the arm and on the left of the chair, whereas the upper part of the forearm has been partially eroded. A better result has been obtained with the method proposed here (c). Notice that in all cases a jagged border is obtained. The silhouette is then smoothed with the moving average method (d) or with the frequency filtering method (e). Panel (f) shows the original color image.

Given I_d , we define each depth image pixel x_d at depth images coordinates (u_d, v_d) having depth d . The corresponding color camera pixel x_c on I_c has coordinates (u_c, v_c) . The homogeneous 3D coordinates of x_d and x_c are respectively

$$x_d = [u_d d \quad v_d d \quad d]^T \quad (1)$$

$$x_c = [u_c \quad v_c \quad 1]^T \quad (2)$$

With the world coordinate system centered on the depth camera, we represent a 3D point in the world coordinate as

$$X = [x \quad y \quad z]^T. \quad (3)$$

Both cameras are modeled using the pinhole model and distortions are not considered. The relationship between the pixels and the 3D points is thus

$$x_d = A_d X \quad (4)$$

$$sx_c = A_c(RX + T) \quad (5)$$

where R is the 3x3 relative orientation matrix between the two cameras, T the relative position and s is an arbitrary scale factor. A_d and A_c are the intrinsic matrices of the two cameras in the form:

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

Given the depth pixel x_d as input, we can reproject it on the camera image as

$$sx_c = A_c(RX + T) = A_c(RA_d^{-1}x_d + T) = A_cRA_d^{-1}x_d + A_cT = Hx_d + M, \quad (7)$$

where H is a 3x3 matrix and M is a 3x1 translation vector. Note that using this reprojection model we do not need to recover the actual depth or color camera intrinsic parameters explicitly, which gives great computational efficiency. However, given known color camera intrinsic parameters, A_c , both the depth camera intrinsic parameters A_d and the relative pose R, T can be uniquely recovered. Given n pairs of corresponding points on the depth and color images, we find H and M through a least squares estimate

$$[M, H] = \arg_{H, M} \min \left(\sum_{i=1}^n \|x_{c_i} - x_c(M, H, x_{d_i})\|^2 \right). \quad (8)$$

The n pairs of corresponding points are obtained by tracking a known object both in the depth and color camera. Our object of choice is a LED ball, which proves useful due to its bright color and simple shape. We use color detection in the HSV space on I_c to find the 2D center of the LED ball x_c (Fig. 2b). We use the 3D level-set based tracker described in [9] to track the position of the LED ball in I_d and we find its 3D center x_d . Each tracking frame provides a corresponding ball-center pair (x_c, x_d) . Eq. (7) can then be used to find in real-time for each depth pixel x_d the corresponding pixel x_c .

2.2 Silhouette Extraction and Smoothing

The silhouette of the player is jagged (Fig. 3b-c) due to the low resolution of the depth stream and to the irregular pattern used to estimate depth (Fig. 2a). The flickering depth stream, in turn, produces a flickering silhouette as well.

We have tackled this problem through adequate filtering of the player segmentation data. We have developed two alternative solutions: one in the space and one in the frequency domain. Both methods take I_b as input. The first method is inspired by [10] and has two steps. In the first step, we apply a moving average low-pass filter to I_b with a sliding window of 5x5 pixels. To obtain a graded transition, the alpha values are converted from a binary value to a byte value (0 to 255), obtaining I'_b . In the second step, we perform a weighted average over the pixel values in the last N frames $I'_b(t - i)$ with $i = 0 \dots N$, assigning a weight $N - i$ to each frame to avoid adding a

significant delay. As a result, we achieve a smoother player profile I_p , reducing the flickering effect of the silhouette with a smooth transition between different frames (Fig. 3d). The method produces a smooth silhouette, but its drawbacks are the local smoothing, that doesn't allow an accurate identification of the silhouette, and the trailing effect that can be noticed especially during fast motion. This can be acceptable and desirable for some implementations but it can be annoying for rehabilitation. For this reason, we have developed a second method that is based on frequency filtering. We first compute the Discrete Cosine Transform (DCT) of the input bitmask $I_b(u, v)$ where $u \in [0, U - 1]$ and $v \in [0, V - 1]$. We then eliminate the higher frequency components, thus realizing a box low-pass filter. Finally, we perform an inverted DCT (IDCT) to obtain a smoother player profile I_p (Fig. 3e). The DCT of I_b , $C(k_u, k_v)$, is a sum of cosine functions computed as

$$C(k_u, k_v) = \alpha_u(k_u) \alpha_v(k_v) \sum_{u=0}^U \sum_{v=0}^V \beta_u(k_u, u) \beta_v(k_v, v) I_b(u, v) \quad (9)$$

with $k_u \in [0, K_U]$ and $k_v \in [0, K_V]$, K_U and K_V being the number of harmonics on the u and v coordinate that we want to keep in the DCT. Usually, K_U and K_V are set to $U - 1$ and $V - 1$ respectively and cannot be larger.

The coefficients $\alpha_n(k_n)$ assume the form

$$\alpha_n(k_n) = \begin{cases} \frac{1}{\sqrt{k_n}} & \text{for } k_n = 0 \\ \sqrt{\frac{2}{k_n}} & \text{otherwise} \end{cases} \quad (10)$$

The functions $\beta_n(k_n, n)$ represent the cosine basis functions on the u axis:

$$\beta_n(k_n, n) = \cos\left(\frac{\pi k_n (2n-1)}{2N}\right). \quad (11)$$

Similarly, $\alpha_v(k_v)$ and $\beta_v(k_v, v)$ can be derived for the v axis. One important property of the DCT is its separability, due to which it can be rewritten as

$$\begin{aligned} C(k_u, k_v) &= \alpha_v(k_v) \alpha_u(k_u) \sum_{v=0}^V \beta_v(k_v, v) \sum_{u=0}^U \beta_u(k_u, u) I_b(u, v) \\ &= \alpha_v(k_v) \alpha_u(k_u) \sum_{v=0}^V \beta_v(k_v, v) D_u(k_u, v) \end{aligned} \quad (12)$$

We can thus separate the computation in a first matrix multiplication that computes $D_u(k_u, v)$ and a second matrix multiplication that computes $C(k_u, k_v)$. The procedure is illustrated in Fig. 4a. A similar separation can be performed for the IDCT (Fig. 4b), that can be written as

$$\begin{aligned} I_p(u, v) &= \sum_{k_v=0}^{K_v} \alpha_v(k_v) \beta_v(k_v, v) \sum_{k_u=0}^{K_u} \alpha_u(k_u) \beta_u(k_u, u) C(k_u, k_v) \\ &= \sum_{k_v=0}^{K_v} \alpha_v(k_v) \beta_v(k_v, v) D_v(u, k_v) \end{aligned} \quad (13)$$

In our implementation, we observe that we do not need to compute the full DCT of the input bitmask, since we are going to filter out most of the higher frequency data. As such, we compute the DCT only for the first M harmonics, from 20 to 30 in the present case, that we wish to keep in the final image. As a result, we can work with

smaller DCT images and thus decrease both computation and memory load. We then apply a circular window in the frequency domain setting to zero the coefficients of the DCT components for which $\sqrt{k_u^2 + k_v^2} > M$. To increase the speed, we also pre-compute the required coefficients and samples of the cosine functions: $\alpha_n(k_n)$ and $\beta_n(k_n, n)$.

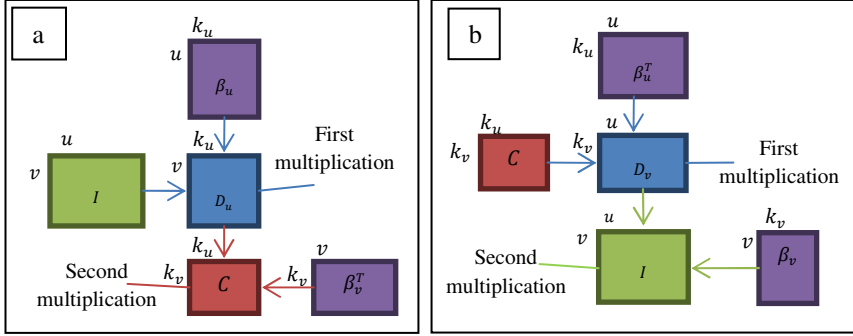


Fig. 4. Visual representation of DCT (a) and IDCT (b) computation using two matrix multiplications

We implemented the algorithm in CUDA for an additional speed-up using a parallel method to compute the DCT and its inverse [11]. We first load the input bitmask I_p into device global memory. We then perform two subsequent matrix multiplications using parallel CUDA Kernels to compute C , using one parallel Kernel thread for each matrix coefficient, thanks to the separability of the DCT. After the DCT is computed, we apply a step filter with a circular shape to remove the high frequencies, then we compute the IDCT with the same parallel multiplication procedure, obtaining I_p .

However, the sharp transition of the step filter introduces oscillations (Gibb's artifacts) on the final image, especially close to the border of the patient profile where I_p goes from one to zero in a few pixels. We thus apply a sigmoid function to each pixel value of I_p to map the resulting pixel values in a 0-255 range and drastically reduce the oscillations. The filter is heavily parameterized in order to easily tune the shape of the outline. A flowchart of the different steps is reported in Fig. 5.

3 Results

These algorithms have been integrated inside the Intelligent Game Engine for Rehabilitation (IGER), developed for the Rewire project [12], to guide and monitor patients during rehabilitation at home and have been extensively tested. In the experiment presented here, the player sits at about 1.5m from the Kinect device that is installed above a large TV screen (50 inches) connected to the PC on which the Game Engine is run. The calibration procedure required a maximum of 15s to acquire the data and calibrate: 300 frames were typically sufficient to guarantee a good calibration with residuals lower than 2 pixels.

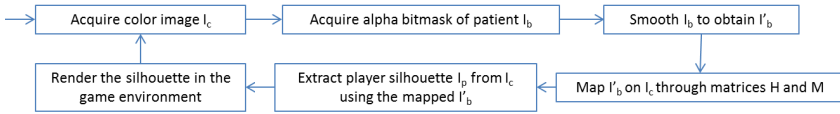


Fig. 5. The block diagram associated to processing each Kinect frame

The player profile is used during a cognitive rehabilitation game called *Balloon Popper*, where the player has to reach and touch a set of balloons that appear in the virtual scene, avoiding the bombs. We played the game using several settings for the silhouette extraction and obtained different visual results, as shown in Fig. 6 and in the related series of videos⁵. The background was cluttered with office objects such as chairs, desks, computers and their displays. Fig. 6a shows the raw misaligned silhouette. The player's profile is jagged and noisy and a large part of the background can be seen. It is also hard to understand when the player is actually touching a game object. In Fig. 6b, we applied our calibrated depth-to-color mapping to the silhouette. The background is correctly subtracted and the player sees only himself inside the profile area. In Fig. 6c, we apply moving average smoothing. The jagged silhouette disappears and there is a less clear separation between the real player and the virtual scene, as if the two were merging together. However, the smoothing is only local and thus the resulting silhouette has a "wavy" look and there is a noticeable trailing. In Fig. 6d, we apply DCT smoothing instead. We obtain a smooth contour around the player, and the player's silhouette is merging into the virtual background as if the player was actually part of the virtual scene, increasing immersion.

4 Discussion

Rehabilitation exercises can be dull and boring due to their repetitive and slow movements. Accordingly, consciousness has raised in the last years towards the need for more compelling experiences, under the form of virtual rehabilitation games, to hold the focus of patients for longer periods and thus address compliance. Such rehabilitation games are often set in a virtual fictitious environment, such as the Rewire farm, in order to create a compelling fantasy setting.

Moreover, a mirrored view of oneself allows a greater immersion inside the virtual world. This increased sense of presence can be a useful way to promote neural reorganization [4]. However, to achieve this, the patient must not be reminded of the difference between his own realistic image, reflected in the mirrored view, and the virtual environment. For this reason, the player's mirrored image must be of high quality and it must be segmented with accuracy and in real-time, in order to let it blend in the virtual environment. Otherwise, the result can easily fall inside the uncanny valley [13] (Fig. 6b) and the display will alienate the patient from exercising.

Using as a base Kinect's official SDK, we obtained a robust and smooth extraction of the player's silhouette in real-time. We improved the depth-to-color mapping

⁵ homes.di.unimi.it/~pirovano/rewire/iciap2013

through an efficient and robust calibration. The accuracy of the algorithm can be further improved using circle fitting insensitive to occlusions [6]. We explicitly remark that the projection of the sphere on the image plane is, in general, an ellipse whose center is not the projection of the sphere center. However, under very general assumptions, such displacement can be safely neglected [15]. The resulting procedure is particularly suitable for the home environment and it allows to easily recalibrate the device, whenever required (e.g. if the device falls to the ground).

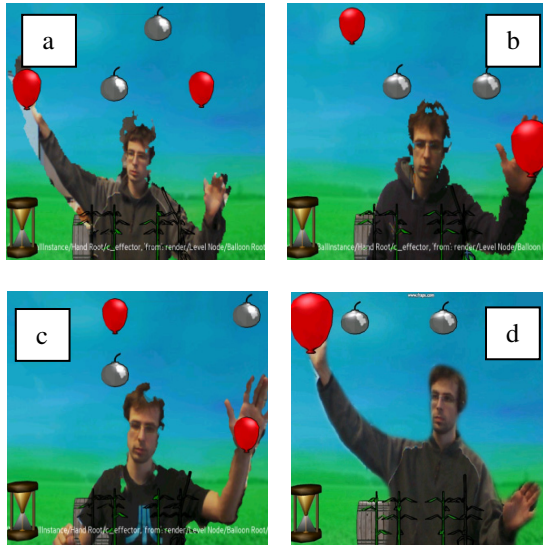


Fig. 6. Playing the Balloon Popper game using different silhouette settings. (a) No mapping nor smoothing. (b) Calibrated mapping, with no smoothing. (c) Calibrated mapping with moving average smoothing. (d) Calibrated mapping with DCT smoothing

Thanks to CUDA and the specific DCT implementation, the silhouette is extracted and smoothed efficiently, freeing the CPU for game engine processing. The resulting silhouette lets the player immerse into the game without being distracted or misled by inaccuracies or the bad quality of the silhouette. Thanks to the depth camera of Kinect, this result can be achieved even in bad light conditions and with a really noisy background, making this method preferable in respect to color-based methods. As future work, the authors aim to assess quantitatively the quality of our results.

The method depends on the number of DCT components, M , that are retained. The actual choice is a trade-off between the amount of details retained and noise suppression, due to the frequency of the noise being similar to that of the silhouette detail (i.e. the fingers). A similar trade-off on the border of the silhouette between showing the background beneath and the eroding part of the player profile is achieved by tuning the steepness of the sigmoidal function. Still, improvements are required to stabilize the obtained silhouette: unknown depth data and hair, that do not produce a clear reflection of the infrared pattern, can contribute to a variation over time of the shape of

the input bitmask. To allow a different cut-off frequency in the different regions of the body, one possibility is to substitute DCT with wavelet decomposition and retain a different number of components in different areas. This approach can be combined with a local color image processing with the aim of an identification of the hair region that is more stable over time.

Acknowledgements. This work was partially supported by the FP7 Project REWIRE, grant 287713 of the European Union.

References

1. Kinect for Windows SDK,
<http://www.microsoft.com/en-us/kinectforwindows/>
2. Rizzolatti, G., Craighero, L.: The mirror neuron system. *Annual Review of Neuroscience* 27, 169–192 (2004)
3. Cameirão, M.S., Badia, S.B.I., Oller, E.D., Verschure, P.F.M.J.: Neurorehabilitation using the virtual reality based Rehabilitation Gaming System: methodology, design, psychometrics, usability and validation. *Journal of Neuroengineering and Rehabilitation* 7, 48 (2010)
4. Mainetti, R., Sedda, A., Ronchetti, M., Bottini, G., Borghese, N.A.: Duckneglect: video-games based neglect rehabilitation. *Technology and Health care* (in press, 2013)
5. Unsworth, C.A.: Cognitive and Perceptual Dysfunction. In: Schmitz, T.J., O’Sullivan, S.B. (eds.) *Physical Rehabilitation*, pp. 1149–1185. Davis Company, Philadelphia (2007)
6. Frosio, I., Borghese, N.A.: Real-time accurate circle fitting with occlusion. *Pattern Recognition* 41(3), 1041–1055 (2008)
7. Menna, F., Remondino, F., Battisti, R., Nocerino, E.: Geometric investigation of a gaming active device. In: *Proc. of Videometrics, Range Imaging and Applications XI, SPIE Optical Metrology* (2011)
8. Khoshelham, K., Elberink, S.O.: Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. In: *Sensors 2012*, vol. 12, pp. 1437–1454 (2012)
9. Yuheng Ren, C., Reid, I.: A unified energy minimization framework for model fitting in depth. In: *2nd Workshop on Consumer Depth Cameras for Computer Vision, CDC4CV* (2012)
10. Sanford, K.: Smoothing Kinect Depth Frames in Real-Time (2012)
11. Obukhov, A., Kharlamov, A.: 8x8 Discrete Cosine Transform with CUDA, Whitepaper, <http://docs.nvidia.com/cuda/cuda-samples/index.html>
12. Pirovano, M., Mainetti, R., Baud-Bovy, G., Lanzi, P.L., Borghese, N.A.: Self-Adaptive Games for Rehabilitation at Home. In: *Proc. IEEE Conference on Computational Intelligence and Games*, pp. 179–186 (2012)
13. Mori, M.: The Uncanny Valley. *Energy* 7(4), 33–35 (1970)
14. Hartley, R., Zisserman, A.: *Multiple View Geometry*, Cambridge University Press (2000)
15. Frosio, I., Borghese, N.A.: 6dof tracking, with axial markers. *Medical & Biological Engineering & Computing* (in press, 2013), doi:10.1007/s11517-013-1052-7