Multi-sensor data fusion for hand tracking using Kinect and Leap Motion

Conference Paper · April 2014

DOI: 10.1145/2617841.2620710

2 authors:

Benoît Penelle
Université Libre de Bruxelles
7 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Olivier D Debeir
Université Libre de Bruxelles
93 PUBLICATIONS **2,088** CITATIONS
SEE PROFILE

Multi-Sensor Data Fusion for Hand Tracking using Kinect and Leap Motion

Benoît Penelle
Université Libre de Bruxelles (ULB)
50 avenue F-D Roosevelt
1050 Brussels, Belgium
bpenelle@ulb.ac.be

Olivier Debeir
Université Libre de Bruxelles (ULB)
50 avenue F-D Roosevelt
1050 Brussels, Belgium
odebeir@ulb.ac.be

ABSTRACT

Often presented as competing products on the market of low cost 3D sensors, the Kinect™ and the Leap Motion™ (LM) can actually be complementary in some scenario. We promote, in this paper, the fusion of data acquired by both LM and Kinect sensors to improve hand tracking performances. The sensor fusion is applied to an existing augmented reality system targeting the treatment of phantom limb pain (PLP) in upper limb amputees. With the Kinect we acquire 3D images of the patient in real-time. These images are post-processed to apply a mirror effect along the sagittal plane of the body, before being displayed back to the patient in 3D, giving him the illusion that he has two arms. The patient uses the virtual reconstructed arm to perform given tasks involving interactions with virtual objects. Thanks to the plasticity of the brain, the restored visual feedback of the missing arm allows, in some cases, to reduce the pain intensity. The Leap Motion brings to the system the ability to perform accurate motion tracking of the hand, including the fingers. By registering the position and orientation of the LM in the frame of reference of the Kinect, we make our system able to accurately detect interactions of the hand and the fingers with virtual objects, which will greatly improve the user experience. We also show that the sensor fusion nicely extends the tracking domain by supplying finger positions even when the Kinect sensor fails to acquire the depth values for the hand.

Categories and Subject Descriptors

I.3.6 [**Computer Graphics**]: Methodology and Techniques—*Interaction techniques*

General Terms

Algorithms, Performance

VRIC'14, April 9-11 2014 Laval, France
Copyright is held by the owner/author(s).
Publication rights licensed to ACM.
ACM 978-1-4503-2626-114/04$15.00
http://dx.doi.org/10.1145/2617841.2620710.

Keywords

Kinect, Leap Motion, hand tracking, augmented reality, PLP

1. INTRODUCTION

Amputees frequently develop pain that they locate in the amputated limb [4]. This type of pain, called phantom limb pain (PLP), is typically a neuropathic pain. PLP greatly impacts the quality of life of the patient and is difficult to treat. Most PLP patients are under heavy medication.

Recent studies [6, 8] show the crucial role of the central nervous system in these pathologies and suggest a link to the plasticity of the latter. Mirror visual feedback (MVF) is often used in case of amputation to restore a normal cortical organization and to lower the pain intensity. We have conceived an augmented reality system that applies the principle of MVF without requiring the use of a physical mirror. The system strengthens the patient's immersion and concentration by showing him photo realistic 3D images that are acquired, processed and displayed back in 3D and in real time using the Kinect sensor (http://www.microsoft.com/en-us/kinectforwindows). This system has been described in detail in [1], including the results obtained after the first clinical study which are encouraging.

One feature of the system allows the patient to interact with virtual objects. This has been initially implemented with a rudimentary algorithm for detecting collisions between the patient's hand and the virtual targets. Basically, it consists in counting, for each frame, the number of vertices from the mesh representing the patient that are located inside the bounding box of the target. If this number exceeds a certain threshold the target is considered as being touched. The main drawback of this approach lies in the fact that we do not segment the patient's hand in the 3D point cloud from the Kinect: all the vertices are taken into account regardless of the body part to which they belong. This means that the patient can touch the target with any part of his body, and that the system is not able to restrict the collision detection to the hand or finger level.

The feedback from the physiotherapists that experimented the system on their patients, is that it would benefit from a more precise way of interacting with the virtual objects. Two features are requested to allow the patient to touch and displace virtual objects: with the whole hand and with the fingers. This requires whole hand tracking and fingertips detection and tracking. In this paper we show how the system has been improved by incorporating a collision detection algorithm based on the use of the Leap Motion (www.leapmotion.com), a low cost accurate 3D sensor ded-

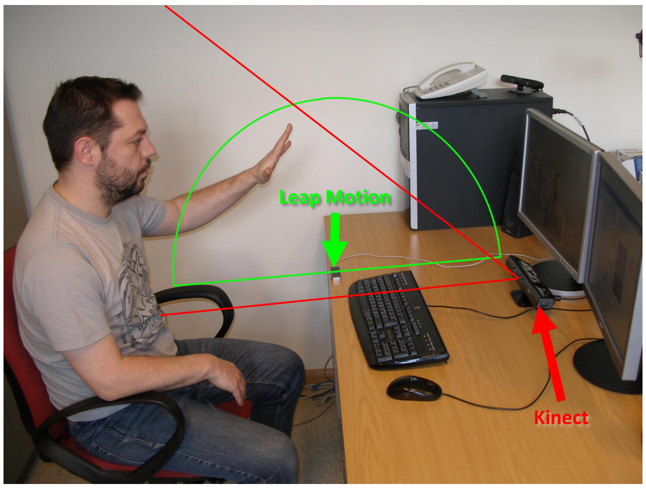


Figure 1: Setup of the two sensors in front of the subject.

icated to hand motion tracking.

2. MATERIAL AND METHODS

From the depth and color images acquired by the Kinect, we generate a 3D image of the subject in the form of a textured mesh. At the same time, the Leap Motion directly provides information on the location of the subject's hand and fingers. Coordinates returned by the two sensors are expressed in two different frames of reference. To exploit them jointly, we have developed a calibration method that calculates the rigid transformation required to align the two frames of reference. Once the calibration is done, data from the Kinect and from the LM may be fused together and used respectively to display the 3D image of the subject and to detect interactions between his hand and virtual objects. This fusion improves the system in two ways: (1) it provides a more accurate tracking of the hand and the fingers, and (2) it extends its operating range by providing fingertips locations even when the hand is not visible to the Kinect (i.e. oriented parallel to the line of sight).

2.1 Description of the setup

The patient is seated in front of the computer screen. On the table before him, about a meter away, the Kinect is aligned with the center of his body, with a tilt angle set to view the upper body as a whole. The LM is laid flat on the edge of the table, in alignment with the shoulder on the side where the subject will move his arm. In order to touch the virtual targets that appear before him, the subject moves him hand forward ; the area where the targets appear is set to be in the operating range of the LM, that is to say, a $50\,cm$ radius hemisphere above the sensor, as illustrated in figure 1.

2.2 3D Images

The 3D images of the subject are acquired with the Kinect. It produces depth and color images that are synchronized and calibrated, with a resolution of 640×480 pixels at a frame rate of $30\,Hz$. The depth image is transformed into a point cloud by using the calibration parameters of the sensor [5]. The vertices of the point cloud are meshed together

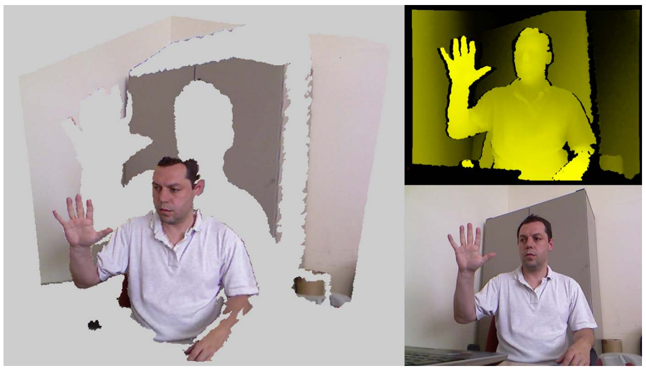


Figure 2: Example of a 3D image (left) constructed from a depth image (top right) and a color image (bottom right) generated by the Kinect.

based on the relative positions of their pixels in the depth image. Finally, the color image is used to texture the resulting mesh (see figure 2). A more detailed description of this process may be found in [1].

2.3 Hand Tracking

The Leap Motion is a small sensor, connected via USB, which allows to accurately track the movements of the hand and the fingers. Laid flat on a horizontal support, the detection field is approximately a $50\,cm$ radius hemisphere above the sensor. It is composed of three infrared LEDs and two infrared CCD cameras. The data acquisition frequency is about $115\,Hz$, but can vary according to the selected parameters and the lighting conditions. The technology behind the LM is currently waiting to be patented. For this reason, the manufacturer has not released any information describing the operating principle nor the implemented tracking algorithms.

The accuracy announced by the manufacturer for the position detection of the fingertips is approximately 0.01 mm. In [9], the authors have conducted a study where they analyze the accuracy and the robustness of the LM. They have developed an experimental setup making use of an industrial robot for measuring the position of a reference pen. From this, they have measured a deviation below 0.22 mm for static setups and of 1.2 mm for dynamic setups. In this study, the precision was evaluated in the case of the motion tracking of a pen (a *trackable* in the jargon of the LM), for which the location of the end point can be determined unambiguously. In the case of the tip of a finger, the determination of a reference point is less obvious, and since no information has been published on how the LM determines this reference, no validation of the accuracy in that case could be performed at this stage.

After each frame acquisition, the LM software development kit (SDK) publishes a detailed set of location data for the detected hand(s). Among these, we use the estimated center of the palm and the fingertips locations. They are expressed in a Cartesian coordinate system centered on the device. Although the LM allows for simultaneously tracking of multiple hands, our context of use only requires the tracking of one hand at a time.

2.4 Calibration

In order to compute the rigid transformation (translation and rotation) that best aligns the frames of reference of the two sensors we use the Corresponding Point Set Registration (CPSR) algorithm described in [2]. It requires to identify landmarks on the target object (here the hand of the subject) that will be tracked by both sensors, each sensor returning the locations of the landmarks in his own coordinate system. In our case, the choice of the landmarks has been dictated by the fact that the LM does not return dense 3D information for the hand, but only a limited set of locations for specific points such as the center of the palm and the fingertips. We have decided to work with the fingertips. The acquisition of the landmarks coordinates with the LM is straightforward as they are directly returned by the SDK. On the other hand, the acquisition of their coordinates by the Kinect requires some processing that is detailed below.

The segmentation of the hand in the Kinect's depth image in a robust way is a complex problem, especially because in a completely general case, we can not make any a priori assumption on the pose of the subject and his hand, nor on the presence or absence of occlusions [7]. Hopefully, by making some restrictive assumptions on the pose, it is possible to simplify the problem: the subject must be seated facing the Kinect, hand open and positioned vertically, fingers outstretched, palm facing the camera and making sure that his hand is in the forefront of any object with respect to the camera, including his trunk and head (see figure 4). It is important to note that these assumptions are only necessary during the calibration phase. When these assumptions are met, the segmentation of the hand and finger tips is considerably simplified: first the pixels corresponding to the hand are segmented in the depth image, then the algorithm checks if the hand is correctly opened, and finally the fingertips are localized. These three main steps of the algorithm are detailed below.

2.4.1 *Hand Segmentation*

Let P be the set of pixel coordinates for the depth image Z of size $H \times W$.

$$P = \{p = (r, c) \mid r \in \mathbb{N},\, c \in \mathbb{N},\, 0 \leq r < H,\, 0 \leq c < W\}$$

The pixel coordinates $(0, 0)$ correspond to the upper left corner of the image. Using the calibration parameters of the Kinect it is possible to compute the x and y coordinates for each pixel of the depth image, which gives X and Y images. We delimit an horizontal exclusion band I_h of height h_i at the bottom of the image (see figure 3).

$$I_h = \{p = (r, c) \mid p \in P,\, r < W - h_i\}$$

The useful area of the depth image is designated by the pixels that belong to $U = P \setminus I_h$. In order to segment the hand, we compute z^{min} as the smallest depth value within U, then we compute the set A of all the pixels having a depth value greater than z^{min} of at least Δz (see figure 4). Since, by hypothesis, the hand is positioned vertically, A will contain all the pixels of the hand, but it will also contain pixels from the wrist and potentially pixels of the arm.

$$\begin{aligned} z^{min} &= \min_{p \in U} Z(p) \\ A &= \left\{p \mid p \in U,\, Z(p) - z^{min} \leq \Delta z\right\} \end{aligned}$$

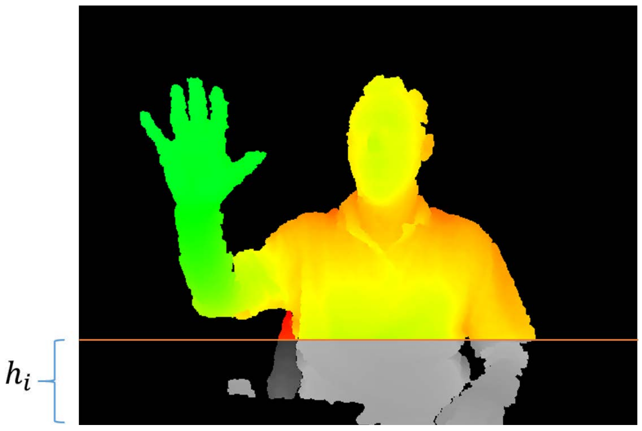


Figure 3: The exclusion band (in gray color) defined in the depth image for the hand segmentation.

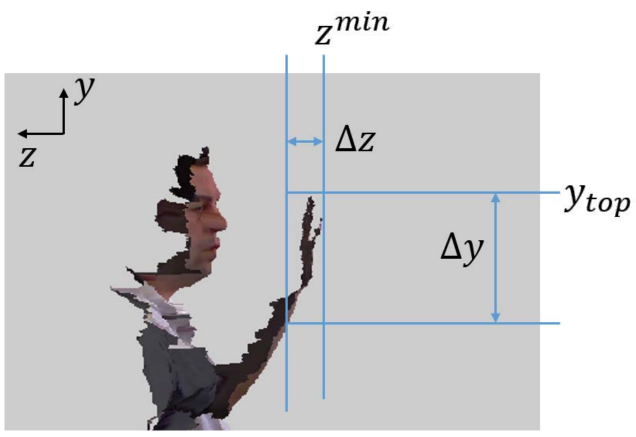


Figure 4: Limits and parameters used to segment the hand in the U zone.

The set H of the pixels corresponding to the hand segmentation is obtained by restriction of A by keeping only pixels whose height (y coordinate) is lower than the maximum height y^{top} of at least Δy (see figure 4).

$$\begin{aligned} y^{top} &= \max_{p \in A} Y(p) \\ H &= \{p \mid p \in A,\, Y(p) - y^{top} \leq \Delta y\} \end{aligned}$$

2.4.2 *Hand State Verification*

We consider two possible states for the hand: open or closed. It is considered open if the five fingers are in partial or complete extension and if they can be distinguished from each other in the depth image. Here we explain how we proceed to check if the hand is in the open state based on the X, Y and Z images. We start by computing a binary mask H_b of the depth image corresponding the hand segmentation H.

$$H_b(p) = \begin{cases} 1 & \text{if } p \in H \\ 0 & \text{otherwise} \end{cases}$$

On this mask, we apply the *findContours*() function of the OpenCV library [3] in order to retrieve N^C external contours c_i for all the blobs (set of 8-connected pixels) detected in H_b. Each contour c_i is a polygon defined by a

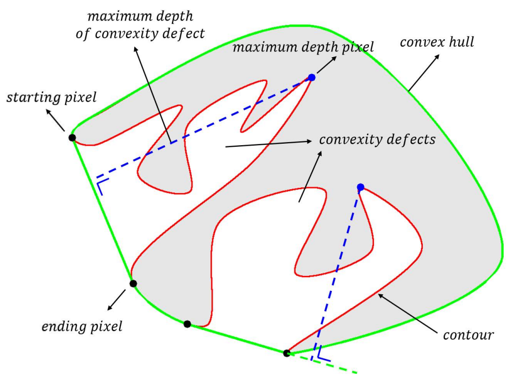


Figure 5: Illustration of the convex hull and the convexity defects for a closed curve.

sequence of N^{c_i} pixels.

$$\begin{aligned} C &= findContours(H_b) \\ &= \{c_i = \{p_{ij}\}\} \quad i \in \left\{0, \dots, N^C\right\}, j \in \{0, \dots, N^{c_i}\} \end{aligned}$$

Since, by hypothesis, the hand is the main object in U, it should correspond to contour c^{max} delimiting the largest area value, the area being computed using OpenCV $contourArea()$ function. The $convexHull()$ function allows us to retrieve the convex hull c^{hull} of c^{max}, which is then used as argument for $convexityDefects()$ to retrieve the N^D convexity defects d_i of the contour c^{max} with regards to c^{hull}. Convexity defects are defined as the concave sections of the contour relative to its convex hull, as illustrated in figure 5 for a general case.

Each convexity defect is characterized by the index in c^{max} of its starting pixel (s_i) and its ending pixel (e_i), and by the maximum depth of the defect expressed in pixels (l_i).

$$\begin{aligned} c^{max} &= \arg\max_{c \in C} contourArea(c) \\ c^{hull} &= convexHull(c^{max}) \\ D &= convexityDefects(c^{max}, c^{hull}) \\ &= \{d_i = (s_i, e_i, l_i)\} \quad i = \left\{0, \dots, N^D\right\} \end{aligned}$$

Let D_j be the sets of convexity defects whose maximum depth is respectively greater than L_1, L_2 and L_3 (parameters of the system, see table 1), with $L_1 > L_2 > L_3$, and let N_j be their cardinality.

$$N_j = |D_j = \{d_i \mid d_i \in D, \, l_i > L_j\}| \quad j \in \{1, 2, 3\}$$

The hand will be considered open if there is at least one convexity defect having a maximum depth greater than L_1 or at least two convexity defects having a maximum depth greater than L_2, and in any case, only if the number of convexity defects having a maximum depth greater than L_3 is equal to six. This condition may be written as:

$$(N_1 \geq 1 \vee N_2 \geq 2) \wedge (N_3 = 6)$$

2.4.3 *Localization of the finger tips*

We define o_c as column index of the centroid H, and o_r as the row index of the lowest convexity defect from D_3 in

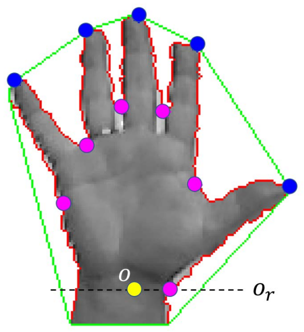


Figure 6: Illustration of the various contours and features used to identify the fingertips. In red we have the contour c^{max}, in green the convex hull c^{hull}, in magenta the points of maximum depth for the convexity defects d_i, in yellow the origin o, and in blue the fingertips $\hat{p}_i$. The mask H_b used to find the contour is computed from the depth image, but for illustration purpose we have filled the interior of the contour with the pixels from the color image converted to gray scale.

the image. The origin pixel is defined as $o = (o_r, o_c)$.

$$\begin{aligned} o_c &= \frac{1}{|H|} \sum_{(p_r, p_c) \in H} p_c \\ o_r &= \max_{(d_r, d_c) \in D_3} d_r \end{aligned}$$

The convexity defects of D_3 delimit contour sections c_i^f corresponding to the different fingers and the wrist. For each c_i^f, the pixel $\hat{p}_i$ is defined as the pixel form c_i^f which is the furthest from o. The $\hat{p}_i$ correspond to the finger tips, except for one of them that lies on the contour section containing the wrist. This section may easily be identified and discarded because it is the only one that contains pixels which are lower in the image than o_r. Figure 6 illustrates these landmarks.

$$\hat{p}_i = \arg\max_{p \in c_i^f} \|p - o\|_2$$

The finger tip pixels $\hat{p}_i$ are sorted in increasing (decreasing) order of pixel column index if the hand is located in the right (left) half of the image. The 3D coordinates of the finger tips are given by $[X(\hat{p}_i) \; Y(\hat{p}_i) \; Z(\hat{p}_i)]^T$.

2.4.4 *Registration*

To increase the robustness of the registration and make it less sensitive to noise and variations in the position and pose of the hand, we collect fingertips location for a sequence of successive frames, during which the subject is asked to move his hand to cover as much as possible the operating range of the Kinect.

Each one of the N valid acquisitions from this sequence (i.e. frames for which the fingertips have been successfully detected by both sensors) provides Cartesian coordinates for five fingertips in both frames of reference. At the end of the calibration sequence, we have two matrices F_{Kinect} and F_{LM} of dimensions $5N \times 3$ that will be used as input for the CPSR algorithm. The output is a 4×4 transformation matrix T corresponding to the rigid transformation that best aligns

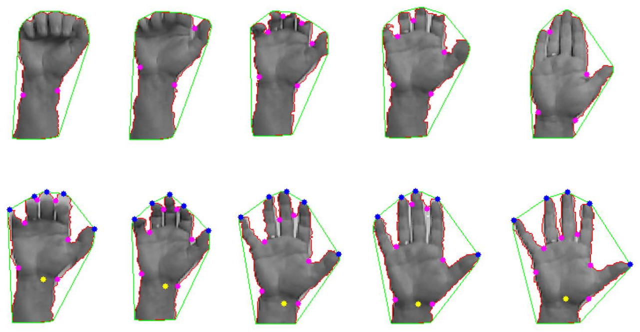


Figure 7: The top row of images illustrates cases where the fingertips were not segmented because the convexity defects did not verify the required geometric conditions (i.e. there are too few defects or some defects are too shallow). The bottom row gives examples of hand images where the fingertips were successfully segmented.

(in the least square sense) the two points clouds represented by F_{Kinect} and F_{LM}.

2.4.5 Typical values for the parameters

Table 1 gives the values that were used for the parameters of the calibration algorithm for our experiments.

Table 1: Typical values for the parameters used by the calibration algorithm.

Parameter	Description	Value
h_i	Height of the horizontal ignored band at the bottom of the image	150 pix
Δz	Thickness of the hand (incl. wrist)	15 cm
Δy	Height of the hand (incl. wrist)	20 cm
L_1	Minimum defect depth for first type of convexity defects	30 pix
L_2	Minimum defect depth for second type of convexity defects	20 pix
L_3	Minimum defect depth for third type of convexity defects	5 pix

2.5 Tracking

Once the calibration phase is completed, T is used in all subsequent acquisitions to express the coordinates of the hand and fingertips returned by the LM in the frame of reference of the Kinect. This transformation remains valid as long as both sensors stay static. If one of them is moved, it is necessary to redo the calibration phase. The advantage of the calibration method presented here is that it may easily and rapidly be executed by the subject himself because it does not rely on any marker system or external calibration tool.

3. RESULTS AND DISCUSSION

Figure 8 shows the different steps of the calibration procedure. The images show the 3D reconstructed mesh of the subject's body with the LM estimated hand positions in blue (palm center and fingertips). The first snapshot has been taken before the execution of the calibration: it clearly shows the offset between the frames of reference of the sensors. The second snapshot has been taken after a few frames of the calibration phase: it shows the segmented hand (in red), the estimated fingertips locations on the Kinect image (in green) and the corresponding estimated positions coming from the LM (in blue). We see here that the frames of reference have been aligned. The third snapshot illustrates that after the calibration phase we have a good spatial concordance between the Kinect-based image and the LM-tracked locations for the fingertips. The fourth snapshot illustrates another advantage of working with these two sensors: even when the Kinect is "blind" (not able to measure depth for certain angles) as in the case illustrated here where the hand is in the XZ plane, the LM continues to track the fingertips.

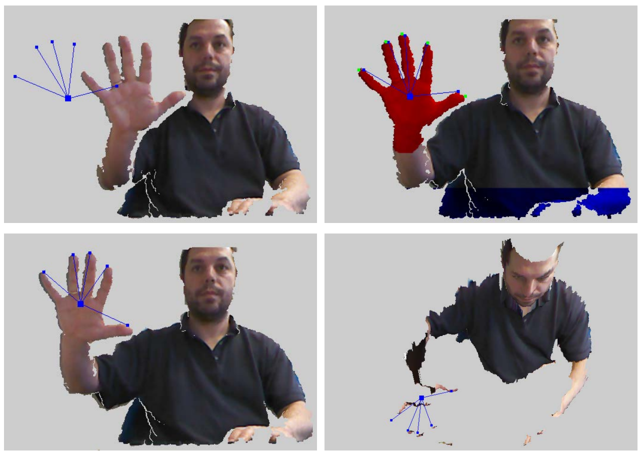


Figure 8: The different steps of the calibration procedure (in reading order): before calibration, during calibration, after calibration and a case where the hand is not visible to the Kinect.

In the second image of figure 8, we may observe that the fingertips locations from the Kinect (green) and those from the LM (blue) are not perfectly aligned. As explained above, this comes from the fact that there is no available information on how the LM algorithm estimates the location of the fingertips, nor on what geometric or anatomic reference points are taken into account. Is it, as we do for the Kinect image, the furthest point of the finger along its axis, or is it the central points on the surface of the distal phalanx, or some other point? Currently we do not have the answer to this question.

We compute the registration error by applying the rigid transformation T returned by the CPSR algorithm to all the fingertips locations measured by the LM and by computing the distance between the transformed coordinates and the fingertips locations estimated using the Kinect images.

Figure 9 shows the boxplots of the distribution of this error for 10 sequences of 500 frames, all taken for the same subject, when the hand is hold static during each sequence (left) and when the hand is moving (right). 500 frames is enough to have a good idea of the error distribution since, from our observations, both the Kinect and the LM internal tracking algorithms do not undergo any drift (i.e. frames are processed independently). Figure 10 shows the boxplots of the error distribution for 5 different subjects (5 sequences of 500 frames per subject), when the hand is hold static during each sequence (left) and when the hand is moving (right). We see from these results that the median error stays between 5 and 10 mm for the static and the dynamic setups. We see also that the average value for the maximum error (upper whisker) for all subjects stays lower than 20 mm.

A video illustrating the calibration procedure and the tracking capabilities of our system is available at: http://youtu.be/ofQjTA2W8PE.

4. CONCLUSIONS

In this paper we have shown that Kinect and Leap Motion data me be fused together in a complementary way. The Kinect data are used to generate 3D images of the subject while LM data allow for accurate tracking of his hand and fingers. We have proposed a fast and easy to use calibration method.

The presented combination of these two sensors will greatly improve the ability of the subject to interact with virtual objects with his whole hand or more accurately with his fingertips. It will allow for instance to simulate accurate prehension of small virtual objects between the finger and the index.

Future work will integrate the presented multi-sensor tracking into the neuropathic pain treatment application. New interaction scenarii will be implemented in order to evaluate the impact on both efficacy and user experience.

5. REFERENCES

[1] B. Penelle, D. Mouraux, E. Brassinne, T. Tuna, A. Nonclercq, and N. Warzée. 3D augmented reality applied to the treatment of neuropathic pain. In *Proc. 9th Intl Conf. on Disability, Virtual Reality and Assoc. Technologies*, pages 61–68, Laval, France, September 2012. P M Sharkey, E Klinger (Eds).

[2] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, February 1992.

[3] G. R. Bradski and A. Kaehler. *Learning OpenCV, 1st Edition*. O'Reilly Media, Inc., first edition, 2008.

[4] H. Flor. Phantom-limb pain: characteristics, causes, and treatment. *The Lancet Neurology*, 1(3):182 – 189, 2002.

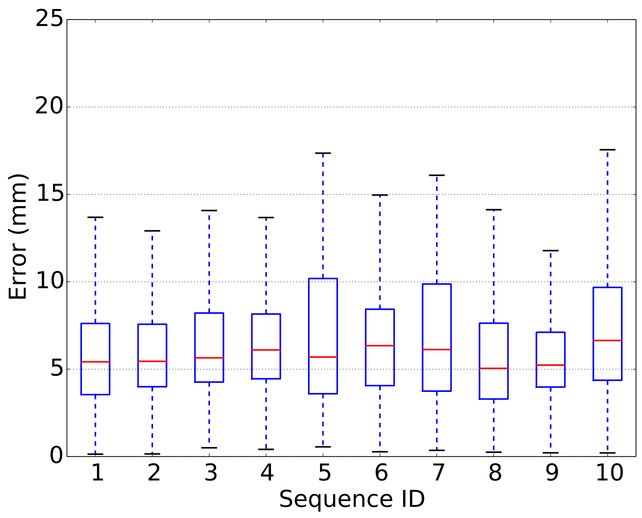
[5] K. Khoshelham and S. O. Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(12):1437–1454, February 2012.

[6] G. L. Moseley. Using visual illusion to reduce at-level neuropathic pain in paraplegia. *PAIN*, 130(3):294 – 298, 2007.

[7] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings of the British Machine Vision Conference*, pages 101.1–101.11. BMVA Press, 2011.

[8] V. S. Ramachandran and E. L. Altschuler. The use of visual feedback, in particular mirror visual feedback, in restoring brain function. *Brain*, 132(7):1693–1710, 2009.

[9] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the accuracy and robustness of the Leap Motion controller. *Sensors*, 13(5):6380–6393, May 2013.



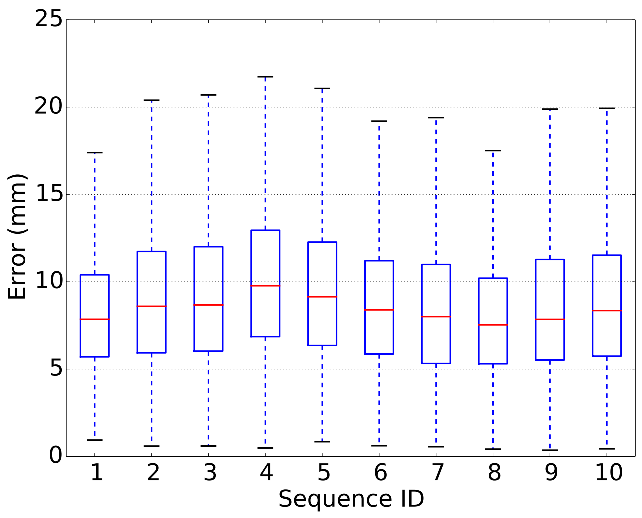
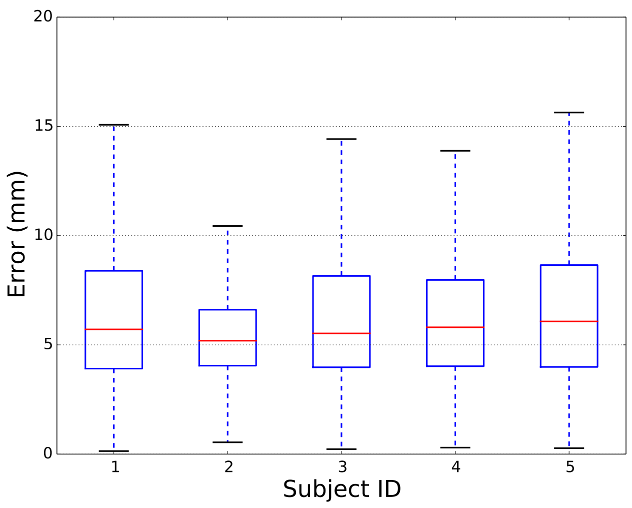


Figure 9: Error distribution for 10 sequences of 500 frames, all taken for the same subject, when the hand is hold static during each sequence (left) and when the hand is moving (right). In these boxplots, the whiskers extend to 1.5 times the inter-quartile range.



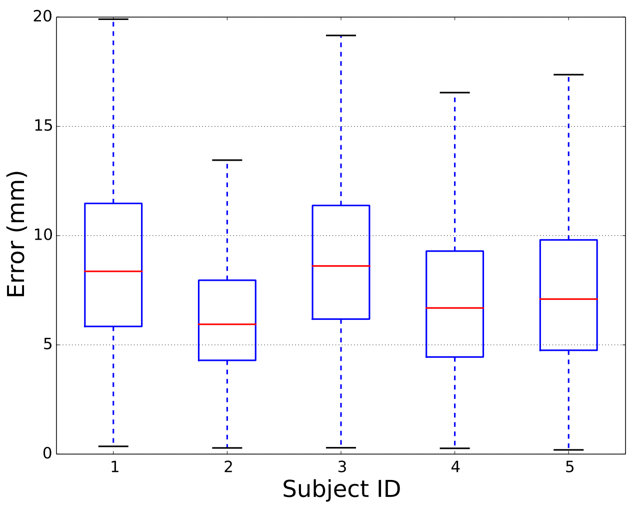


Figure 10: Error distribution for 5 different subjects (5 sequences of 500 frames per subject), when the hand is hold static during each sequence (left) and when the hand is moving (right). In these boxplots, the whiskers extend to 1.5 times the inter-quartile range.