Seamless bare-hand interaction in Mixed Reality

Caterina Battisti, Stefano Messelodi, Fabio Poiesi
Technologies of Vision
Fondazione Bruno Kessler
Trento, Italy
{*cbattisti, messelod, poiesi*}*@fbk.eu*

Figure 1: Bare-hand interaction with the augmentation of a hat and a bunny using our proposed approach. When the depth of the hand is processed and associated to the pixels labelled as hand, object rendering can account for occlusions thus enabling seamless MR experiences.

***Abstract*—Unrealistic Mixed Reality (MR) experiences can be caused by unprocessed occlusions between real and augmented objects during interactions. Depth knowledge is indeed key to achieve a seamless visualisation when a bare hand interacts with an augmented object. This can be addressed by blending real-time 3D finger tracking information with the visualisation of the hand in MR. We propose an approach that automatically localises the hand in RGB images and associates the respective depth, estimated with an auxiliary infrared stereo camera used for hand tracking, to each RGB hand pixel. Because misalignments between the outline of the hand and its depth may occur due to tracking errors, we use the distance transform algorithm to densely associate depth values to hand pixels. In this way hand and augmented object depths can be compared, and the object can be rendered accordingly. We evaluate our approach by using an MR setup composed of a smartphone and a Leap Motion mounted on it. We analyse several hand configurations and measure the erroneously classified pixels when occlusions occur.**

***Keywords*-Human Computer Interaction, Mixed Reality, Hand Interaction, Leap Motion.**

I. Introduction

Mixed Reality (MR) has shifted from being used only by experts to being accessible by anyone. MR content can be visualised and interacted with via different devices such as smart glasses, HoloLens and smartphones, and despite its broad use, R&D is still carried out to further improve user experience. Typically, objects are manipulated through the smartphone touchscreen, or by using a joypad [1] or via hand gestures (e.g. pinch) [2], [3]. Immersive and intuitive interactions with virtual objects using only bare hands is still a challenge, as it requires a users hands and fingers to be accurately localised and segmented in 3D space to account for occlusions during scene rendering. These visualisation challenges are less impacting in Virtual Reality as both the environment and the hands, if they are tracked, are jointly rendered [4]. Fig. 1 shows an example of what can be achieved when depth is properly assigned to the hand in an MR experience using our proposed approach.

To enable bare-hand interactions with augmented objects, fingertips and the convexity defect points between fingers can be detected and tracked after the hand is segmented using skin detection [5]. The position and orientation of the hand can be estimated relatively to the camera position to augment the object on the palm. Occlusions are implicitly addressed because the hand is located behind the augmented object with respect to the camera position. RGB-D sensors like Microsoft Kinect are an alternative to achieve bare-hand interactions and have been employed to detect finger touch regions [6], to estimate the hand pose [7] and to render augmented objects based on the occlusions generated by the hand [8]. However, Kinect is designed to function with objects of a size greater than that of a hand and located at a minimum distance of 50cm from the sensor. Moreover, Kinect cannot estimate fingers' position with millimetre accuracy. Alternatively, one can compute the depth using infrared stereo cameras [9]. Infrared images can be processed more reliably as illumination variations are less likely to affect the tracking of the hands. Machine learning methodologies can be used to infer the metric depth of the hand through infrared images [10]. However, an additional camera is needed to acquire the visible spectrum and to enable the MR visualisation. Lastly, stereo cameras can also be used along with stereo matching algorithms to compute

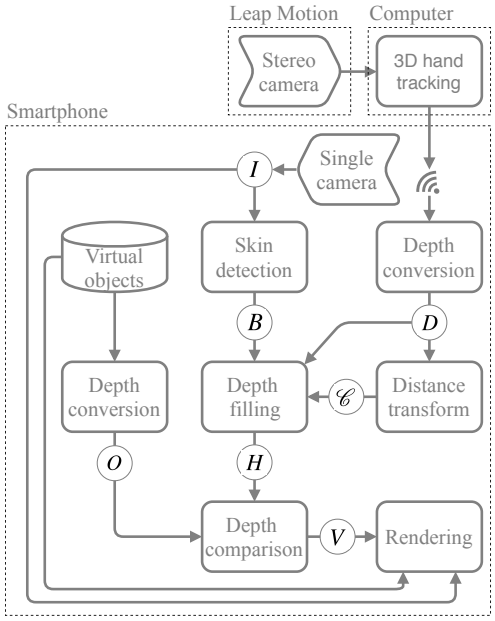

Figure 2: Block diagram. Depth is obtained from tracking data and augmented objects in the scene. Hand image is obtained from the smartphone camera feed. Depth and skin are elaborated to output an occluded skin overlay to be displayed in the augmented environment.

the depth from stereo images [11]. In particular infrared stereo cameras are more robust and can be used to locate and track the position of the hand joints with a higher accuracy than the sensors mentioned before. Leap Motion is an example of infrared stereo camera that can enable effective bare-hand interactions with augmented objects [12].

In this paper we propose a methodology to render augmented objects based on the position of the hands in MR. The depth of the hand is estimated using an off-the-shelf (low-cost) hand tracking device (i.e. Leap Motion [12]) and combined with the image of the hand that is captured from an RGB camera. Because finger poses may be inaccurately predicted during tracking, the estimated depth and the image of the hand may be misaligned. We address this problem by using the distance transform algorithm in the depth map space to associate depth values to hand pixels detected in each RGB image. We use the Unity 3D game engine as development environment and qualitatively evaluate the proposed method by showing the rendering results while hands manipulate an augmented cube. Then we quantify the accuracy by counting the number of incorrectly rendered pixels when occlusions occur in the case of several interaction configurations.

II. Problem formulation

Given the hand tracking data processed by a stereo infrared camera and the view of the hand captured from a single RGB camera, we densely associate each RGB pixel, that is automatically labelled as hand through skin detection, to a corresponding measured depth value. Let $I : \mathbb{Z}^2 \mapsto \mathbb{Z}^3$ be an image, where $x \mapsto I(x) = [i_r(x), i_g(x), i_b(x)]$ is the RGB value of the pixel in position x. Let $D : \mathbb{Z}^2 \mapsto \mathbb{R} \cup \{\infty\}$ be a depth map. $D_1 = \{x : d(x) \in [0, \infty)\}$, where $D(x) = d(x)$ is the measured depth value of the hand and $D(x) = \infty$ is an unmeasured depth value at pixel x. Let B be the set of RGB pixels labelled as hand. I and D are acquired from two different coordinate systems whose relative transformation is unknown. The objective is to associate a pixel $x \in B$ to an appropriate depth value $d(x)$ in order to estimate the objects, or portions of them, in MR that are occluded by the hand. Therefore, let $H : B \mapsto \mathbb{R}$ be the depth that is associated to the RGB pixels of the hand, where $H(x)$ is a measured depth value of pixel $x \in B$. Let $O : \mathbb{Z}^2 \mapsto \mathbb{R} \cup \{\infty\}$ be the depth map of the augmented objects defined as $O(x) = o(x) \in [0, \infty)$, where $o(x)$ is a measured depth value in x and $O(x) = \infty$ is a pixel x where the depth is not measured. H and O are defined on the same coordinate system. By comparing the depth of the associated hand pixels with the depth of the augmented objects, we create a binary visibility mask $V : \mathbb{Z}^2 \mapsto \{0, 1\}$ that is defined as

$$V(x) = \begin{cases} 1 & \text{if } x \in B \wedge H(x) < O(x) \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

$V(x) = 1$ corresponds to a visible pixel of the hand (not occluded by the object) that will be rendered. Fig. 2 shows the block diagram of the proposed approach.

III. Proposed approach

We associate the two physical cameras on different coordinate systems to two virtual cameras in our rendering space in order to separate the hand model data from the augmented object data, thus producing a depth map for each virtual camera [13].

A. Hand depth retrieval

Hand tracking data processed by the stereo camera have a vectorial representation, that is the direction in a three-dimensional space of the joints defined with a skeletal tracking model (distal, intermediate, proximal and metacarpal bones) [14]. In order to transform this vectorial representation into a depth map, we apply the skeletal joints onto a 3D hand model. The 3D hand model can deform its shape according to the position of the tracked hand joints.

The depth values of D can be computed with Unity 3D through the virtual camera associated to the stereo camera [15]. In practice, each $d(x)$ is encoded with values that specify the distance between a point in the space and the origin of the camera. The distance is defined for a fixed range, and lies within the near and a far clipping plane having depth values encoded with 8 bits, thus producing a quantised depth map represented with 256 levels.

Fig. 3 shows the comparison between the depth computed through 3D model fitting (c) and block matching (d) applied to the stereo images [16]. We can observe that we have a noise-free estimated depth in the former as opposed to that of the latter. The 3D model is fit onto the hand through a tracking process. The interaction points associated to the fingers within the augmented space depend on the tracking output. However, tracking may introduce errors due to prior constraints on the states (i.e. pose and joints) and on the motion of the hand. Despite these likely errors, we have deemed more appropriate to retrieve the depth measured through the 3D model as it is related of the position of the joints during the interaction with augmented objects.

B. Skin detection

In order to segment the region of the RGB image that includes the hand, we perform skin detection in the HSV colour space using thresholding [17]. The HSV format separates the light intensity from the colour information, improving the robustness of the detection in different lighting conditions. For simplicity we omit the dependency on x in this section.

Given the RGB camera image I, the conversion to HSV is calculated as

$$i_v = max(i_r, i_g, i_b) \tag{2}$$

$$i_s = \begin{cases} \dfrac{i_v - min(i_r, i_g, i_b)}{i_v} & \text{if } i_v \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$i_h = \begin{cases} \dfrac{60(i_g - i_b)}{i_v - min(i_r, i_g, i_b)} & \text{if } i_v = i_r \\ \dfrac{120 + 60(i_b - i_r)}{i_v - min(i_r, i_g, i_b)} & \text{if } i_v = i_g \\ \dfrac{240 + 60(i_r - i_g)}{i_v - min(i_r, i_g, i_b)} & \text{if } i_v = i_b, \end{cases} \tag{4}$$

where i_h, i_s, i_v represent the HSV colour components, respectively.

The hand skin can be detected by identifying the pixels that lie within an appropriate range of HSV values [18]. Therefore, the set of pixels labelled as hand is defined as

$$B = \{x : (0 < i_h < 40) \wedge (40 < i_s < 150) \wedge (60 < i_v < 255)\}. \tag{5}$$

C. Hand-depth pixel association

We associate each pixel in B to a measured depth value of the hand in D. This association is computationally intensive as it is to be performed for each pixel in B (Sec. III-B). Therefore, we use the distance transform to perform such a "dense association" efficiently [19]. The distance transform assigns to each pixel x the distance from the set of pixels different from zero, and typically it is applied to binary images to compute the minimum distance between each pixel equal to zero and its nearest non-zero pixel. In our case we use the distance transform to find the closest measured depth value of a pixel that has unmeasured depth. Therefore all pixels labelled as hand will have an associated measured depth value.

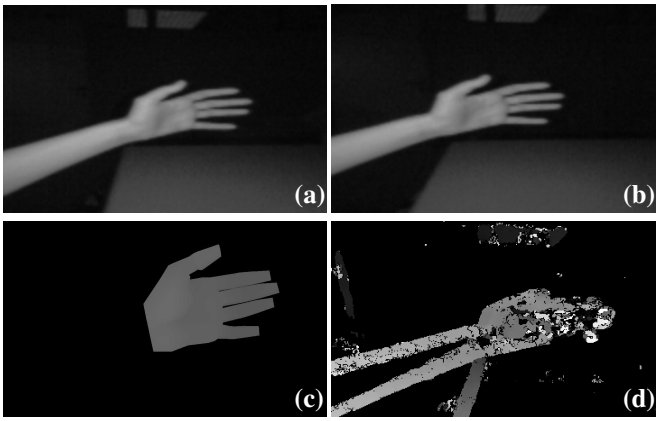

Figure 3: Stereo images ((a) left and (b) right) acquired with the Leap Motion sensor [12]. Comparison between depth computed through (c) 3D model tracking and (d) stereo block matching.

The algorithmic definition of distance transform of a sampled function applied to our case is

$$\mathcal{D}(x) = \min_{x' \in D_1} (\delta(x, x')), \tag{6}$$

where $\mathcal{D}(x) \in \mathbb{R}$ and $\delta(x, x')$ is an appropriate distance, e.g. L_1 or L_2, between pixels located at x and x'. Because we deal with a bi-dimensional data structure (i.e. image), this minimisation can be solved in $O(N)$ time, where N is the number of pixels of the image [19].

As we are not directly interested in the minimum value of the distance but in the location of the depth value closest to $x \in B$, we compute the pixel position as

$$\mathcal{C}(x) = \arg\min_{x' \in D_1} (\delta(x, x')), \tag{7}$$

where $\mathcal{C}(x) \in \mathbb{Z}^2$. We use the L_2 distance for $\delta(x, x')$ in order to promote associations of spatially close depth values.

The function H (Sec. II) associates a depth value $D(x)$ located at $\mathcal{C}(x)$ to a pixel $x \in B$. H is defined as

$$H(x) = D(\mathcal{C}(x)) \; \forall \; x \in B. \tag{8}$$

Fig. 4 shows an example of skin-depth association. After extracting the hand depth (a), we can overlay it to the original image extracted from the camera and notice their offset (b). Because we do not perform camera calibration, this offset increases as the position of the hand moves closer to the image border. After we associate skin-depth using the distance transform algorithm, we can observe how the depth gets aligned to the hand region on the image (d).

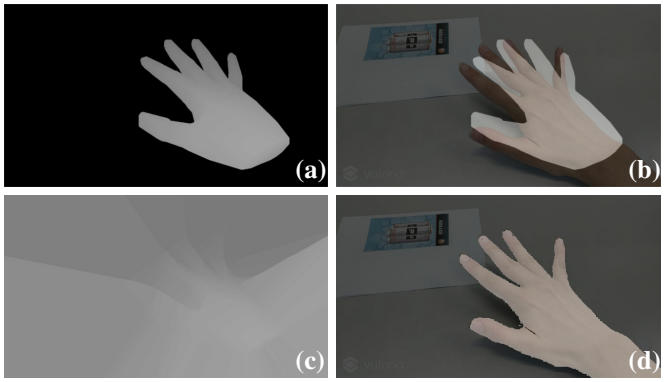

Figure 4: Alignment of the depth of the hand retrieved from a 3D model onto the hand detected in an RGB image by using the distance transform algorithm. (a) Depth retrieved from a 3D hand model generated from a stereo acquisition device (black pixels represent unmeasured depth). (b) Overlay of the depth onto the hand captured through an RGB image. (c) Depth values associated to pixels with unmeasured depth. (d) Depth aligned onto the hand after the association performed with the distance transform algorithm.

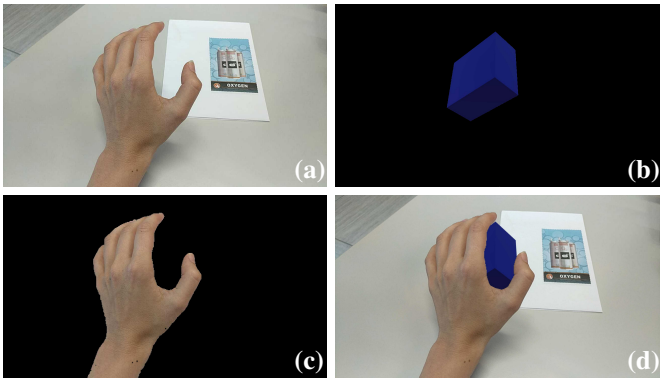

Figure 5: Queue-based rendering assigns a priority to each image that is rendered with a specific order. (a) Original RGB image captured from the smartphone camera. (b) Object to augment in the scene. (c) Image containing the pixels that are labelled as hand. Black pixels in (b) and (c) represent 100% transparency. (d) Final rendering where occlusions are accounted for.

D. Hand rendering

We compare the depth values of H with the depth values of O to build the visibility mask V (Sec. II). In our experiments we experienced unwanted artifacts due to the quantisation of the values of D (Sec. III-A), especially in those regions where the augmented objects and the hand intersect. We addressed this problem by spatially smoothing V with a Gaussian kernel $G(0, \sigma)$. Then we add an alpha layer of the same size as the image I that defines the transparency of each pixel according to V. Each pixel $x \in B$ with $V(x) = 1$ is rendered with its original RGB value, whereas each x with $V(x) = 0$ is set to transparent.

Next we render the original RGB image, the image of the augmented objects and the image of the visible pixels of the hand with a specific order to handle the relative occlusions. We employ a queue-based rendering approach that assigns a priority queue to each layer to render. Objects with higher priority are rendered before objects with lower priority. With this mechanism it is possible to display the hand overlaid on top of the augmented objects.

Fig. 5 shows an example where the RGB image acquired from the camera has the lowest queue priority, the hand overlay has the highest queue priority and the augmented object has medium queue priority. The black pixels in (b) and (c) are transparent. The final rendering of these images based on their priority leads to the seamless visualisation shown in (d).

IV. Results

A. Experimental setup

We mounted the Leap Motion on the back of a smartphone (Google Pixel XL), which in turn was mounted on a smartphone stand (Fig. 6). Like in [20], we connected the Leap Motion to a desktop computer via USB. The computer processes the video stream and generates the hand tracking data that is streamed to the smartphone via WiFi. Tracking data is transmitted to the smartphone through a JSON message over a web socket with a rate that can reach 200 frames per second. This frame rate is high for our application, therefore we designed a custom web socket application running on the computer that intercepts, modifies and forwards the tracking frames at a slower rate, to a custom port on the smartphone. Each JSON message received by the smartphone is stored as a string in a buffer that can then be polled by the smartphone application when the system is ready to update the scene. The messages retrieved from the buffer are converted from *string* format to a Frame object. The object is thus ready to be used to render the virtual hand.

In all the experiments we set the near clipping plane and the far clipping plane at distances of $0.01m$ and $0.5m$, respectively. Images are processed at a resolution of 640×360. We use a Gaussian kernel of size 13×13 pixels with $\sigma = 2.3$. We initialise the system by coarsely superimposing the 3D model of the hand onto the hand captured with the RGB camera.

B. Evaluation

We evaluated the proposed approach through hand interactions with a 3D cube that was augmented on an image target (Fig. 6). The experiments were carried out indoor with typical office illumination conditions. We quantified the performance by measuring the number of pixels that had been correctly rendered during the manipulation of the cube. For example, if a finger was located in front of the cube and some of the pixels were behind the cube, we counted them and deemed them as incorrectly rendered. In order to

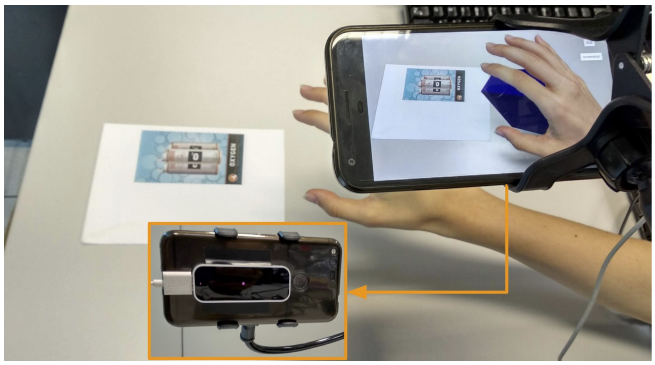

Figure 6: The Leap Motion sensor is mounted on the back of the smartphone in our experiments. The origin of the smartphone camera's coordinate system and of the Leap Motion are different.

make the results comparable across different experiments, the number of incorrectly rendered pixels were normalised by the number of pixels belonging to the intersection region between the cube and the hand. We define the rendering accuracy as $A = 1 - \frac{|x_{err}|}{|x_{hand} \cap x_{obj}|}$, where x_{err} are incorrectly rendered pixels, x_{hand} are the pixels labelled as hand and x_{obj} are pixels of the augmented object.

In order to count the number of incorrectly rendered pixels we augmented the cube by aligning one of its faces with the border of a sheet of paper. We drew transversal lines on the hand to align it with the same border. In this way we can accurately know where the face of the augmented cube should pass through.

C. Analysis

We defined four experimental configurations that involve different hand poses. Configuration 1 (C1) evaluates the accuracy when the hand when is positioned in front the augmented object. Configurations 2 (C2) and 3 (C3) evaluate the accuracy when the hand passes through the augmented object along the vertical and horizontal axes of the augmented space, respectively. Configuration 4 (C4) evaluates the accuracy when the thumb is positioned in front of the fingers simulating an object grab. For each configuration we captured three screenshots: the augmented object position, real hand position and augmented scene. The count of the incorrectly rendered pixels was manually performed.

In C1 the 98.2% of pixels have been correctly rendered. The proposed approach can reliably handle hand poses like that in Fig. 6. In C2 and C3 the 93.5% and 85.0% of pixels have been correctly rendered, respectively. The quantisation of the depth map (Sec. III-A) produced blunt discontinuities near the intersection point between the cube and the hand. Fig. 7 shows the effect of this errors. In C4 the 69.0% of pixels have been correctly rendered. Errors in C4 were due to the dense association that failed when the thumb overlapped with the other fingers. Because the 3D model of the hand is not accurately matching the outline of the hand, the depth

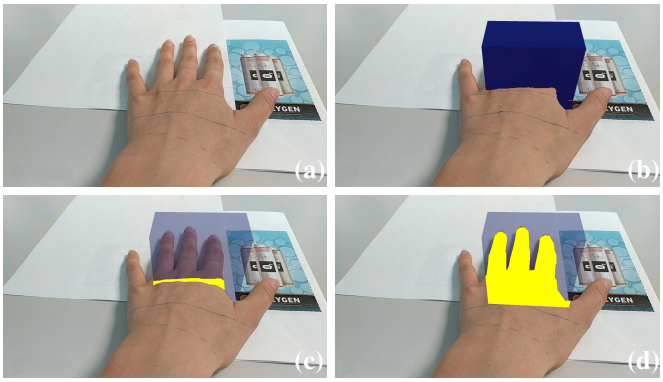

Figure 7: Experimental configuration to measure the number of incorrectly labelled pixels. (a) The lines that are drawn on the hand are aligned to the border of the sheet of paper and the marker to have an accurate reference with respect to the augmented space. (b) Rendering of the augmented object. (c) The augmented object is set to transparent and the incorrectly rendered pixels are highlighted in blue. (d) Highlight of the area where hand and object occlude each other.

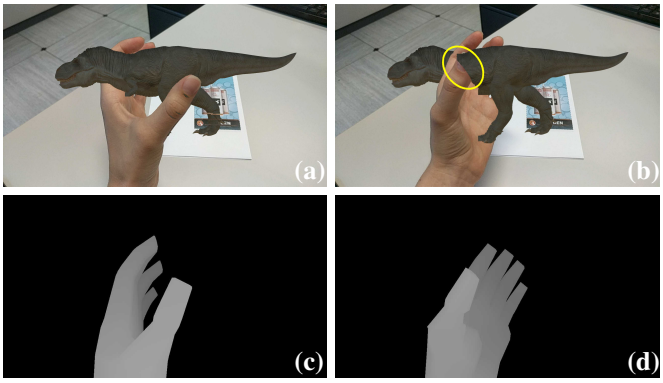

Figure 8: Examples of (a) correctly and (b) incorrectly estimated depth values associated to the thumb. (c) When there is no overlap between fingers the depth can be reliably associated. (d) When there is overlap between fingers the depth values associated to the pixels of the thumb tip are those of the fingers underneath.

values that get associated to the pixels of the thumb tip are those of the fingers underneath. Fig. 8 shows an example of this error.

In our experiments we also observed that similar cases, where the hand goes beyond the boundaries of the augmented object, are due to undefined physical constraints of the object volume. See video in the supplementary material to observe this effect. This challenge can be addressed via haptic feedback [21].

D. Computational time

We have measured the computational time of key blocks of the proposed pipeline and presented it in percentage in

Table I: Computational time of selected blocks in the proposed pipeline. The second column shows the percentage of the average execution time measured over 200 frames.

	time [%]
skin detection	7.2
distance transform	22.8
depth mask smoothing	31.0
transparency adding	10.3
others (e.g. thresholding, resizing, colour conversions)	28.7

order to make it device-independent. The time is calculated as the average over 200 frames.

From Table I we can observe that the Gaussian filtering is the most expensive operation, followed by the computation of the distance transform. Our application runs at about 7 frames per second on the Google Pixel XL. We additionally measured that the distance transform employs 75% of its 22.8% of the time to solve Eq. 7. The remaining 25% is to retrieve the depth values from the image and to assign them to the pixels labelled as hand.

V. Conclusions

We presented an approach to associate depth from hand tracking data to skin detected on RGB images in order to enable hand interactions with augmented objects seamlessly. We performed this association efficiently using the distance transform algorithm. Results show that the depth can be transferred reliably to the hand, but there are still ambiguous situations where the association fails, for example when the fingers overlap.

We are working towards a fully-automated pipeline for seamless hand-based interactions in Mixed Reality that does not require a manual initialisation, but instead it is based on hand-pose estimation estimated from the frames acquired from the RGB camera [22]. We will improve the algorithm that retrieves the depth from the 3D hand model by accessing to higher-precision depth values (i.e. 32 bit instead of 8 bit). In this way we will be able to bypass the Gaussian smoothing, thus reducing the overall computational time. After these improvements, we will optimise our code aiming to achieve a computational time of at least 25 frame per second.

Acknowledgment

This research has partially received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement number 687757.

References

[1] "Magic Leap," https://www.magicleap.com, accessed: Aug 2018.

[2] S. Lin, H. F. Cheng, W. Li, Z. Huang, P. Hui, and C. Peylo, "Ubii: Physical world interaction through augmented reality," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 872 – 885, May 2016.

[3] "HoloLens gestures," https://docs.microsoft.com/en-us/windows/mixed-reality/gestures, accessed: Aug 2018.

[4] Y. Jang, S. T. Noh, H. J. Chang, T. K. Kim, and W. Woo, "3D Finger CAPE: Clicking Action and Position Estimation under Self-Occlusions in Egocentric Viewpoint," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, pp. 501 – 510, Jan 2015.

[5] Y. Shen, S. K. Ong, and A. Y. C. Nee, "Vision-based hand interaction in augmented reality environment," *International Journal of Human-Computer Interaction*, vol. 27, no. 6, pp. 523–544, May 2011.

[6] D. W. Seo and J. Y. Lee, "Direct hand touchable interactions in augmented reality environments for natural and intuitive user experiences," *Expert Systems With Applications*, vol. 40, pp. 3784–3793, 2013.

[7] F. Hernoux and O. Christmann, "A seamless solution for 3D real-time interaction: design and evaluation," *Virtual Reality*, pp. 1–20, 2014.

[8] A. Leal, L. A. Robles, and J. Gonzales, "Occlusion handling in video-based augmented reality using the kinect sensor for indoor registration," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 18th Iberoamerican Congress, CIARP 2013, Havana, Cuba, November 20-23, 2013, Proceedings, Part II*, Nov 2013.

[9] M. Kanbara, T. Okuma, H. Takemura, and N. Yokoya, "A stereoscopic video see-through augmented reality system based on real-time vision-based registration," in *Proc. of IEEE Virtual Reality*, New Brunswick, US, Mar 2000.

[10] S. Fanello, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, S. Kang, and T. Paek, "Learning to be a depth camera," *ACM Trans. on Graphics*, vol. 33, no. 4, Jul 2014.

[11] "Leap Motion as a Rudimentary Depth Sensor," http://blog.rymnd.com/leap-motion-depth, accessed: Aug 2018.

[12] "Leap Motion," https://www.leapmotion.com, accessed: Aug 2018.

[13] "Unity Camera," https://docs.unity3d.com/Manual/class-Camera.html, accessed: Aug 2018.

[14] "Introducing the Skeletal Tracking Model," https://developer.leapmotion.com/documentation/csharp/devguide/Intro_Skeleton_API.html, accessed: Aug 2018.

[15] "Unity Depth Texture," https://docs.unity3d.com/Manual/SL-DepthTextures.html, accessed: Aug 2018.

[16] M. Brown, D. Burschka, and G. Hager, "Advances in computational stereo," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 993–1008, Aug 2003.

[17] V. Vezhnevets, V. Sazonov, and A. Andreeva, "A survey on pixel-based skin color detection techniques," in *Proc. of GraphiCon*, Moscow, RU, Sep 2003.

[18] P. Kakumanu, S. Makrogiannis, and N. Bourbakis, "A survey of skin-color modeling and detection methods," *Pattern Recognition*, vol. 40, no. 3, pp. 1106–1122, Mar 2007.

[19] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, no. 19, pp. 415–428, Sep 2012.

[20] M. Pani and F. Poiesi, "Distributed data exchange with Leap Motion," in *Proc. of Augmented Reality, Virtual Reality and Computer Graphics*, Otranto, IT, Jun 2018.

[21] C. Bermejo and P. Hui, "A survey on haptic technologies for mobile augmented reality," *arXiv:1709.00698*, Sep 2017.

[22] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proc. of IEEE International Conference on Computer Vision Workshop: HANDS*, Venice, IT, Oct 2017.