Software and Hardware Limitations in Model-Surface Registration

DAVID RIX, UNIVERSITY OF CAPE TOWN

We introduce model-surface registration and iterative closest point (ICP), a popular solution to this problem, with reference to surveys by Rusinkiewicz and Levoy [2001] and Salvi et al. [2007]. Implementations of ICP using the Kinect for Xbox 360 are reviewed. Hardware characteristics for the Kinect and other commercial sensors (Leap Motion, Kinect for Xbox One) are summarised. Finally, limitations and requirements for academic reference material are discussed.

• **Computer vision ~ Range sensors** • **Computer vision ~ Model-surface registration**

Additional Key Words and Phrases: iterative closest point (ICP), RGBD, Kinect for Xbox 360 (K4X360), Leap Motion, Kinect for Xbox One (K4X1)

ACM Reference Format:

David Rix. 2015. *Software and hardware limitations in model-surface registration.* Honours report, University of Cape Town. 12 pages. April 2015. http://people.cs.uct.ac.za/~previz2015/registration-review/

1. INTRODUCTION

Registration of objects using range images

Computer vision is the study of techniques for processing image-based data. Data sources may include traditional images (monocular RGB bitmaps), but a wide range of sensors are available that produce richer or alternative datasets, also suitable for vision processing. A subset of these are *range images*, where the pixel value does not represent an RGB colour. Instead, it represents the distance of that pixel from the camera. This is conceptually equivalent to a heightmap, where the pixel value represents the height of the pixel from a nominal surface. However, range images do not have to represent a top-down view (see Figure 1).

One interpretation of a range image is as a 3D surface: each depth pixel is equivalent to the <x,y,z> coordinate of a point, and together these points define a surface. This surface is a partial 3D view of the real-world scene (sometimes called a 2.5D view). The collection of points from the sensor is known as the source cloud.

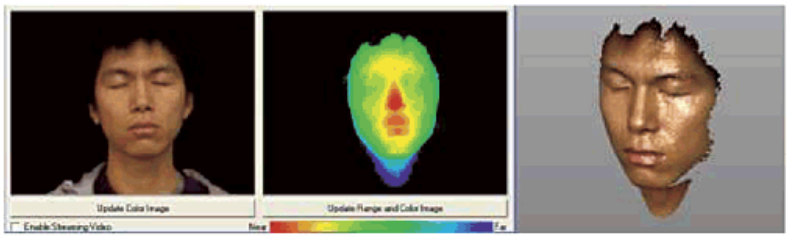

Fig. 1. Example depth scan of a human face, taken from Lu et al. [2006]. (a) Traditional RGB bitmap. (b) Range image (false colour indicates depth). (c) Reconstruction based on range data. Note that the model is limited to the surface defined by the range image.

We might also have an independent point cloud defining a full 3D object, for example drawn in a 3D modelling tool. This collection of known points is called the reference cloud. Given these two datasets, we can consider locating the known model in the unknown surface. One way to do this is to align the two 3D datasets (so that partially visible surface of the object in the scene matches the corresponding surface of the full model). This alignment task is known as *registration* [Besl and McKay 1992; Chen and Medioni 1991]. Model-surface registration is only one application of this technique. Another common application is generating a large surface by stitching smaller overlapping surfaces together, for example when performing large-scale 3D scans.

Salvi et al. [2007] distinguish between coarse and fine registration tasks. Coarse registration searches for an initial estimate aligning the two point clouds. Local solutions, false positives and false negatives should be avoided, but the accuracy of fit is not important. Fine registration relies on a good initial estimate, and refines this estimate such that the distance between corresponding cloud points is minimised, and the best fit is found.

2. ITERATIVE CLOSEST POINT

Iterative closest point (ICP) is a matching algorithm commonly used for fine model-surface registration. The source and reference clouds are compared, with the closest reference point found for each source point. The distance between corresponding points is measured, and the mean squared error (MSE) of difference calculated. The source cloud is moved and rotated slightly to reduce the mean squared error. This process is iterated until the MSE is reduced to an acceptable level, or an iteration limit is reached. ICP has been shown to be geometry-preserving [Besl and McKay

1992], able to find at local solutions [Besl and McKay 1992], and have average case complexity of O(n log n) [Sharp et al. 2001].

Although mathematically reliable [Besl and McKay 1992], the following limitations were identified in the ICP approach:

(1) Over-reliance on a good initial rotation [Besl and McKay 1992; Sharp et al. 2001]. Without a suitable intial rotation ICP may converge on an incorrect local solution. This dependency limits its application when full automation is required, but initial rotation is not known (for example, locating an object in an arbitrary scene) [Sharp et al. 2001].
(2) Inability to account for outliers in the datasets [Besl and McKay 1992]. This means it is not robust against noise generated by real-world surfaces and/or the sensor [Berger et al. 2013].
(3) Surfaces that differ in only fine detail may be subject to false registration. For example, two flat surfaces with light incisions, as might be found when performing a 3D scan of ancient inscriptions [Rusinkiewicz and Levoy 2001].

Chen et al. [1991] is generally considered a variant of ICP (although it is classified by Salvi et al. [2007] as a distinct solution). Instead of measuring distance from point to corresponding point, distance is measured from point to the tangent plane of the corresponding point. This approach has been found to be more robust to local solutions, resulting in more accurate fits [Rusinkiewicz and Levoy 2001]. Historically, Chen's variant relied on calculating normal values in order to obtain the tangent planes, which reduced its speed, but since c2006 depth cameras have typically included normal data in their output [Salvi et al. 2007]. Rusinkiewicz and Levoy [2001] use Chen's variant and increased sampling in regions where the normals were similar, which improved the registration of inscribed surfaces, amongst others (see Figure 2).

Rusinkiewicz and Levoy [2001] also identified six stages of the generalised ICP algorithm:

(1) Selection. Selecting some or all points from each point cloud.
(2) Matching. Finding corresponding points in the two sets.
(3) Weighting. Applying a weighting measure to point pairs.
(4) Rejecting. Applying a validity constraint to point pairs.
(5) Measuring error. Calculating a measure of error based on the final set of pairs.
(6) Minimizing error. Performing a geometric transformation that reduces the error.

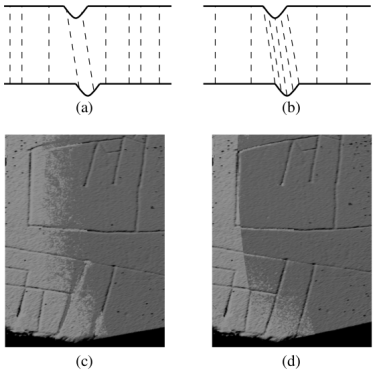

Fig. 2. Increased sampling improves registration of light incisions. Contrast even sampling and resulting registration (a, c) with increased sampling in incised areas (b) and the more accurate registration (d) – demonstrated by crisper image and no duplicate lines [Rusinkiewicz and Levoy 2001]

By combining independently published improvements made to each stage of the algorithm they were able to optimize the speed of the implementation enough to suggest the feasibility of real-time registration from video input.

Despite the popularity of the ICP algorithm, only two approaches (in the period 1992-2003) meet the robustness criteria set out by Salvi et al. [2007]. This is only because they are reviewing ICP in contexts beyond surface-model registration, including motion detection and non-overlapping point clouds. For the surface-model problem, they consider ICP a suitable algorithm. Robustness is still a widely recognised issue, though, and several improvements have been suggested. Broadly, these approaches can be categorized [Salvi et al. 2007] as those introducing statistical techniques (reducing the number of points sampled and/or the number of iterations based on statistical inferences and/or noise reduction) and those introducing coarse registration measures into the ICP distance calculation.

Of note is Sharp et al. [2001] who introduce invariant geometric features as a factor in the point-to-point distance calculation. They note that, for example, the curvature at a point is independent of its position in space. Thus if the curvature of corresponding points do not match, we know that these points do not in fact correspond. Invariant features significantly improved the convergence rate of ICP.

In general, a coarse registration technique must be selected before invoking ICP. In non-automated systems, this could include user input [Besl and McKay 1992; Rusinkiewicz and Levoy 2001]. Surveys from c2000 [Rusinkiewicz and Levoy 2001; Salvi et al. 2007] consistently reference the potential of colour as a coarse registration

technique. As with invariant geometry, points of similar or dissimilar colours suggest correspondence or lack thereof.[1] Thus, points under consideration are assigned 3D coordinates from the range image, and an RGB value from a regular image. The source data is then known as RGBD (RGB-plus-depth).

One potential application of RGBD registration is facial recognition. Lu et al. [2006] use registration to construct a 3D facial model of the subject from 2.5D RGBD scans, storing this as the reference for the recognition task. They then retrieve a new 2.5D image of the subject, and use a hybrid ICP algorithm[2] to register the new source image against the reference model. Finally, RGB data is used to complete the facial recognition task. They note, however, that application is limited due to the cost of 3D data acquisition. The release of commodity RGBD sensors in 2010 led to an surge in computer vision publications that has not yet abated [Berger et al. 2013].

3. HARDWARE CONSIDERATIONS

3.1 Kinect for Xbox 360

The Kinect for Xbox 360 was a gaming accessory released in 2010. A side-effect of its availability in the consumer market was that it lowered the barrier to entry for computer vision research [Berger et al. 2013]. The Kinect is a structured-light depth scanner [Smisek et al. 2011]. It projects a known infrared (IR) pattern into the real-world scene, which is recorded by an IR camera. The geometry of the scene is derived from the deformation of the known pattern, and output as a range image [Berger et al. 2013; Smisek et al. 2011]. An RGB image is supplied from a separate camera, so that the final dataset is RGBD. The RGBD video stream is processed to locate human figures in the space in front of the sensor, and a decision-forest is used to identify these figures and the gestures they perform [Berger et al. 2013]. A high-level API was provided so that developers could make games using whole-body gestures as input, with sufficient abstraction from the underlying computer vision task.

However, the Kinect sensor (and other RGBD cameras in the consumer market) has become a popular tool for low-level computer vision tasks using just the

1 Colour cannot be considered truly invariant, though, as it changes with lighting conditions [Salvi et al. 2006]. But under similar lighting conditions, for example from one video frame to the next, it can provide a guide [Henry et al. 2012; Hu et al. 2012].

2 They use Besl and McKay [1992] to estimate fine registration, and Chen and Medioni [1991] to refine the estimation.

RGBD datastream. This raw data has been used to register natural and manufactured objects, both static and in motion, with appropriate compensations for environmental factors and image noise [Berger et al. 2013]. These implementations tend to use RGB data for coarse registration and ICP for fine registration, and rely on RGB registration as a fallback when ICP registration fails [Henry et al. 2012; Hu et al. 2012]. The KinectFusion system achieved real-time 3D reconstruction with a parallelised, GPU-based implementation of ICP [Izadia et al. 2011].

The range imaging technique (structured-light in the IR spectrum) puts certain limitations on what scenes can be examined, and therefore under what conditions vision tasks can be performed [Berger et al. 2013]. In some cases compensations can be made, in others new techniques must be found, and in still others the sensor is simply unusable:

(1) The depth sensor cannot perform in ranges closer than 1.8m [Amon and Fuhrmann 2014; Smisek et al. 2011].
(2) There is mild distortion towards the edge of the image, and the distortion pattern for the depth and RGB images differs [Smisek et al. 2011]. (See Figure 3.)
(3) Bright light, such as direct sunlight, interferes with the IR camera, limiting the sensor to evenly-lit indoor settings [Butkiewicz 2014].
(4) The IR pattern only appears on matte objects. Transparent objects (such as a glass of water) and reflective objects (such as a mirror) do not reflect the light in the normal way. Compensation techniques have been found to account for transparent objects in some cases [Berger 2013].
(5) A viewing shadow is created by offset between the IR emitter and IR camera, and the offset between RGB camera and IR camera [Smisek et al. 2011]. The practical outcome is that depth is uncertain at the boundaries of objects [Berger 2013].

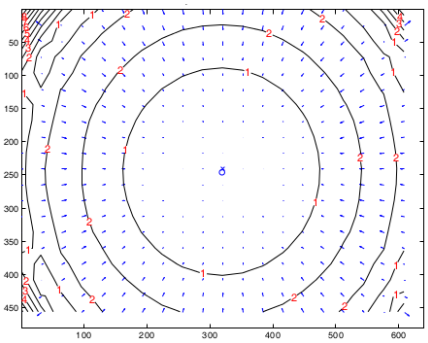

Fig. 3. Radial component of distortion for the Kinect for Xbox 360 IR (depth) camera [Smisek et al. 2011]

(6) The patterns from multiple Kinect sensors used at once interfere with one another's depth perception. Shutter mechanisms can compensate for this [Berger 2013].

Thus, despite its popularity, the peculiarities of the hardware device must be taken into consideration and in some cases are discovered only by empirical observation [Smisek et al. 2011]. The Kinect was itself part of a wave of gesture-based gaming accessories that started with the Wiimote. Since 2010, other vision-based consumer devices have been released, including the Leap Motion and Kinect for Xbox One. As one would expect, these have different camera characteristics, such as resolution, FOV and range depth. However, they also use different range imaging hardware, which has implications for their potential application.

3.2 Leap Motion

The Leap Motion uses a close-range stereoscopic infrared camera [Adhikarla et al. 2015; Guna et al. 2014], and provides a high-level API for articulated hand movements. For low-level processing, the device offers only a stereoscopic IR image and it was not designed as a generic range sensor [Leap Motion 2015c].

As with the Kinect, hardware characteristics determine the suitability of the Leap device:

(1) Effective depth detection is limited to 30cm. [Adhikarla 2015]
(2) Because it is a close-range device, the Leap suffers from image distortion that is extreme compared to the Kinect [Leap Motion 2015b].
(3) There is poorer performance under bright and strong IR conditions. (This is somewhat accounted for with a "Robust Mode" for API-supported hand tracking, but even this has limitations [Leap Motion 2015d].)

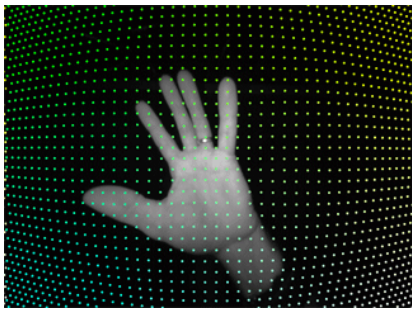

Fig. 3. Visualization of the distortion present in a raw image from the Leap Motion Controller [Leap Motion 2015b]

These limitations, the relatively recent release of the low-level API [He 2015], and the different underlying data set, probably account for the lack of publications applying computer vision techniques to Leap data. Recent publications tend to focus on applying the high-level API to hand-oriented tasks such as sign language [Potter et al. 2013], or augmenting Kinect data with finger position detail [Marin et al. 2014]. It is possible to generate mesh data from the stereoscopic images [Lahoz 2014], but this technique has not seen formal publication. Given the relative mobility of the Leap device, and it's imminent compatibility with mobile operating systems [Buckwald 2015], investigation into it's application for geometric data may bear fruit for other close-range, high-fidelity and/or mobile applications. On the other hand, another device explicitly designed as a close-range generic depth sensor may be more suitable.

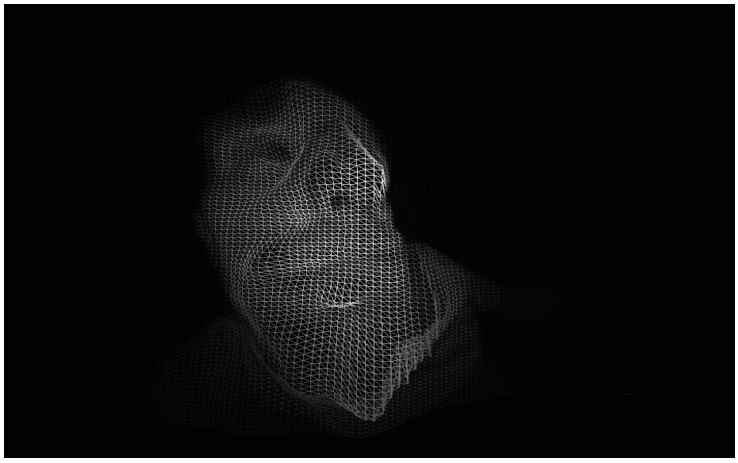

Fig. 4. Human face reconstructed from raw Leap Motion data [Lahoz 2015].
Mesh based on stereo IR data. Note perspective distortion.

3.3 Kinect for Xbox One

The Kinect for Xbox One uses a time-of-flight range sensing [Amon and Fuhrmann 2014; Sell and O'Connor 2014], which is recognised as a reliable, high-fidelity range imaging technique [Berger 2013]. It measures the time taken to receive the reflection of an IR light pulse. The high-level API expands on the features offered by the previous version, including detection of more participants and novel features such as heartbeat detection. As with the other sensors reviewed, the Kinect for Xbox One has a distinct set of distortion maps [Butkiewicz 2014].

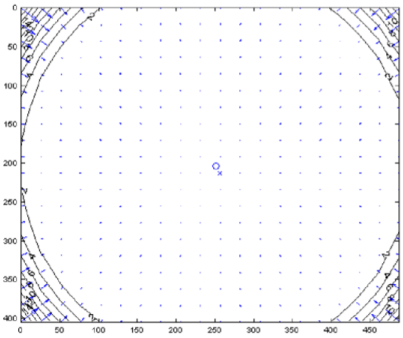

Fig. 5. Radial component of distortion for the Kinect for Xbox One IR (depth) camera [Butkiewicz 2014]

Significant hardware characteristics of the Kinect for Xbox One include:

(1) High-fidelity geometry in similar ranges to the Kinect for Xbox 360 [Amon and Fuhrmann 2014; Sell and O'Connor 2014]. This suggests that invariant geometry may be a viable coarse registration measure, and that registration of fine detail will be possible.
(2) A viewing shadow between RGB and range images [Butkiewicz 2014], but no viewing shadow in range images (due to the time-of-flight approach).
(3) Limited but improved performance in an outdoor setting [Butkiewicz 2014].
(4) Different noise model between time-of-flight and structured light [Berger 2013]. This means changes from research using the Kinect for Xbox 360 will need to be accounted for, but pre-Kinect research using time-of-flight noise models will again be directly applicable.

The Kinect for Xbox One was also released relatively recently, and there was some confusion around how convenient it might be to use in the non-gaming context [Orland 2014; Machkovech 2014]. This accounts for the relatively low number of publications currently available that directly investigate its characteristics.[3] It is reasonable to assume that as the 360 product line comes to an end, hardware begins to fail, and free software library support inevitably improves [Blake et al. 2015], at least some researchers investigating the same physical range will migrate to the new model. Thus it is worth pursuing comparative and device-specific investigation [Butkiewicz 2014], or diving into applications in the same range [Marin et al. 2014].

3 Some of the referenced work is based on the discontinued but near-identical Kinect for Windows v2 sensor [Machkovech 2014].

3.4 Summary of hardware characteristics

The hardware characteristics of the three sensors under discussion are summarised below:

Table I. Summary of sensor hardware

Component	K. Xbox 360	Leap Motion	K. Xbox One
Technique	Structured-light	Stereoscopic infrared	Time-of-flight
Depth sensor	1.8 to 3.5m	3 to 30cm [a]	1.3 to 3.5m
IR depth image	320 x 240	-	512 x 424
Colour image	640 x 480	-	1920 x 1080
IR image	-	640 x 240 (stereo) [b]	512 x 424
Audio	4 kHz, 16-bit	-	48 kHz, 16-bit
FOV horizontal	57 deg.	150 deg. [c]	70 deg.
FOV vertical	43 deg.	-	60 deg.
Minimum latency	102 ms	Unknown [d]	20-60 ms

Source: Summary of Kinect data from Amon and Fuhrmann [2014].

[a] Adhikarla [2015].
[b] As reported in [Derivative 2014] but unconfirmed.
[c] As stated by manufacturer in [Leap Motion 2015a]. Adhikararla [2015] summarises the range volume as an inverted pyramid extending 20cm behind the device, and 20cm in either direction laterally. Guna [2014] notes a decrease in accuracy as the point of interest moves away from the sensor.
[d] Guna [2014] notes that the rate of recognition is limited.

4. CONCLUSIONS

Even with a well-defined task (real-time model-surface registration), a well-known algorithm (ICP) and readily-available hardware, consideration of the research context is neccessary so that we can account for the characteristics of choices made at both the algorithmic and hardware levels.

Hardware characteristics in particular have a significant influence on potential application of a computer vision system. They may even influence the computational tasks required (for example, requiring a deformation step before

registration can occur), which in turn influence the processing speed, which further influences potential application of the system.

Having popular sensors available in the consumer market means that hardware-level characterstics can be measured and shared, even with otherwise proprietary devices. For long-standing popular devices, this is of significant benefit to peers. It is preferable that new product lines are evaluated before attempting application, a need which is heightened when old product lines are discontinued.

REFERENCES

Vamsi Kiran Adhikarla, Jaka Sodnik, Peter Szolgay, and Grega Jakus. 2015. Exploring direct 3D interaction for full horizontal parallax light field displays using Leap Motion controller. *Sensors* 15, 4 (2015), 8642-8663.

Clemens Amon and Ferdinand Fuhrmann. 2014. Evaluation of the spatial resolution accuracy of the face tracking system for Kinect for Windows v1 and v2. In *Proceedings of the 6th Congress of the Alps Adria Acoustics Association* (16-17 October 2014).

Kai Berger, Stephan Meister, Rahul Nair, and Daniel Kondermann. 2013. A state of the art report on Kinect sensor setups in computer vision. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications* (2013), Springer Berlin Heidelberg, 257-272.

Paul J. Besl and Neil D. McKay. 1992. A Method for Registration of 3-D Shapes. *IEEE Trans. Pattern Analysis Mach. Intell*. 14, 2 (February 1992), 239-256.

Joshua Blake, Florian Echtler, and Christian Kerl. 2015. libfreenect2/README.md (April 2015). Retrieved April 24, 2015 from https://github.com/OpenKinect/libfreenect2/blob/master/README.md

Michael Buckwald. 2015. Leap Motion + VR 2015: A Look at the Year Ahead. *Leap Motion Blog* (January 8, 2015) Retrieved April 24, 2015 from http://blog.leapmotion.com/leap-motion-vr-2015-look-year-ahead/

Thomas Butkiewicz. 2014. Low-cost coastal mapping using Kinect v2 time-of-flight cameras. *Oceans – St. John's* (14-19 September 2014), 1-9.

Yang Chen and Gérard Medioni. 1991. Object modeling by registration of multiple range images. In *Proceedings of the IEEE International Conference on Robotics and Automation* 3, 9-11 (April 1991), 2724-2729.

Derivative. 2014. Image Changes Resolution. *Leap Motion Community* (September 17, 2014) Retrieved April 24, 2015 from https://community.leapmotion.com/t/image-changes-resolution/1765

Jimmy He. 2015. Image API Now Available for v2 Tracking Beta. *Leap Motion Blog* (August 15, 2014) Retrieved April 24, 2015 from http://blog.leapmotion.com/image-api-now-available-v2-tracking-beta/

Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. 2014. An analysis of the precision and reliability of the Leap Motion sensor and its suitability for static and dynamic tracking. *Sensors* 14, 2 (2014), 3702-3720.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. 2012. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Rob. Res.* 31, 5 (April 2012), 647-663.

Gibson Hu, Shoudong Huang, Liang Zhao, Alen Alempijevic, and Gamini Dissanayake. A robust RGB-D SLAM algorithm. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS '12). IEEE, 1714-1719.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (UIST '11). ACM, New York, NY, USA, 559-568.

Benoît Lahoz. 2014. Use of Distortion() to get an OpenCV usable calibration map (post #9). *Leap Motion Community* (August 22, 2014). Retrieved April 24, 2015 from https://community.leapmotion.com/t/use-of-distortion-to-get-an-opencv-usable-calibration-map/1605/9

Leap Motion. 2015a. Leap Motion (product page). Retrieved April 24, 2015 from https://www.leapmotion.com/product

Leap Motion. 2015b. Camera Images. *Leap Motion C# SDK v2.2 documentation* (2015). Retrieved April 24, 2015 from https://developer.leapmotion.com/documentation/csharp/devguide/Leap_Images.html

Leap Motion. 2015c. Is it possible to get raw point cloud data? *Leap Motion Support* (2015). Retrieved April 24, 2015 from https://support.leapmotion.com/entries/40337273-Is-it-possible-to-get-raw-point-cloud-data

Leap Motion. 2015d. Using the Leap Motion Control Panel (robust tracking mode). *Leap Motion C# SDK v2.2 documentation* (2015). Retrieved April 24, 2015 from https://developer.leapmotion.com/documentation/csharp/supplements/Leap_Application.html#robust-tracking-mode

Xiaoguang Lu, Anil K. Jain, and Dirk Colbry. 2006. Matching 2.5D Face Scans to 3D Models. *IEEE Trans. Pattern Analysis Mach. Intell.* 28, 1 (January 2006), 31-43.

Sam Machkovech. 2014. Kinect v2 PC devs receive official SDK, $50 USB 3.0 adapter. *Ars Technica* (October 25 2014). Retrieved April 24, 2015 from http://arstechnica.com/gaming/2014/10/25/kinect-v2-pc-devs-receive-official-sdk-50-usb-3-0-adapter/

Giulio Marin, Fabio Dominio, and Pietro Zanuttigh. 2014. Hand gesture recognition with Leap Motion and Kinect devices. In *IEEE International Conference on Image Processing* (ICIP '14), IEEE, 1565-1569.

Kyle Orland. 2014. Standalone Kinect for Xbox One coming October 7. *Ars Technica* (August 27 2014). Retrieved April 24, 2015 from http://arstechnica.com/gaming/2014/08/27/standalone-kinect-for-xbox-one-coming-october-7/

Leigh Ellen Potter, Jake Araullo, and Lewis Carter. 2013. The Leap Motion controller: a view on sign language. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration* (OzCHI '13), Haifeng Shen, Ross Smith, Jeni Paay, Paul Calder, and Theodor Wyeld (Eds.). ACM, New York, NY, USA, 175-178.

Szymon Rusinkiewicz and Marc Levoy. 2001. Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling* (2001), 145-152.

Joaquim Salvi, Carles Matabosch, David Fofi, and Josep Forest. 2007. A review of recent range image registration methods with accuracy evaluation. *Image Vision Comput.* 25, 5 (May 2007), 578-596.

Gregory C. Sharp, Sang W. Lee, and David K. Wehe. 2002. ICP registration using invariant features. *IEEE Trans. Pattern Analysis Mach. Intell.* 24, 1 (January 2002), 90-102.

Jan Smisek, Michal Jancosek, and Tomas Pajdla. 2011. 3D with Kinect. In *IEEE Workshop on Consumer Depth Cameras for Computer Vision* (2011), 1154-1160.

Received April 2015; revised April 2015; accepted April 2015