ACTA UNIVERSITATIS AGRICULTURAE ET SILVICULTURAE MENDELIANAE BRUNENSIS

Volume LX 23 Number 2, 2012

USAGE OF MICROSOFT KINECT FOR AUGMENTED PROTOTYPING SPEED-UP

J. Landa, D. Procházka

Received: November 30, 2011

Abstract

LANDA, J., PROCHÁZKA, D.: *Usage of Microsoft Kinect for augmented prototyping speed-up.* Acta univ. agric. et silvic. Mendel. Brun., 2012, LX, No. 2, pp. 175–180

Physical model is a common tool for testing of the product features during the design process. This model is usually made of clay or plastic because of the modifiability of these materials. Therefore, the designer could easily adjust the model shape to enhance the look or ergonomics of the product. Nowadays, some companies use augmented reality to enhance their design process. This concept is called augmented prototyping. Common approach uses artificial markers to augment the product prototype by digital 3D models. These 3D models that are shown on the markers positions can represent e.g. car spare parts such as different lights, wheels, spoiler etc. This allows the designer interactively change the look of the physical model.
Further, it is also necessary to transfer physical adjustments made on the model surface back to the computer digital model. Well-known tool for this purpose a professional 3D scanner. Nevertheless, the cost of such scanner is substantial. Therefore, we focused on different solution – a motion capture device Microsoft Kinect that is used for computer games. This article outlines a new augmented prototyping approach that directly updates the digital model during the design process using Kinect depth camera. This solution is a cost effective alternative to the professional 3D scanners. Our article describes especially how depth data can be obtained by the Kinect and also provides an evaluation of depth measurement precision.

augmented reality, augmented prototyping, Kinect, image-based rendering, prototype

General principle of augmented reality (AR) usage is to merge the real and virtual world. It allows user to interact with computer in a natural way. There is a number of examples: controlling the computer using natural gestures (e.g. game consoles), projecting status and navigation information on the car windshield, etc. In our research, we focused on problematic of augmented prototyping (AR usage for product design). We proposed and tested several methods that allow to augment a car prototype with virtual objects such as new lights, spoiler, etc. (outlined in Šťastný *et al.* (2011) and Procházka *et al.* (2011)).

This feature substantially simplifies the overall design process. It is not necessary to create every new component on CNC device and place it on the car model. The designer is able to make a preview of this adjustment using the AR, and just in case of satisfaction, a CNC machine could make a solid model. These adjustments are done by placing square markers on the car surface. Each marker contains unique binary code that allows to pair the marker with a virtual spare part model (wheel, spoiler). In camera stream, an appropriate virtual object is then placed on a position given by the marker (see Fig. 1).

However, there is other important challenge. Product (e.g. car) model shape is during the design process adjusted. It would be useful to adjust automatically also the digital computer model. Obviously, this reconstruction can be done using a specialized 3D scanner, however, the price of such technology is substantial, therefore, it cannot be used in many cases. Therefore we looked for an alternative solution. Such solution could potentially be the Microsoft Kinect device.

Microsoft Kinect is originally an advanced game controller designed for the Xbox game consoles. It

1: *Camera stream with a virtual object placed on a car surface*

is equipped by both common and infrared camera. Kinect also contains microphone array and motor. Motor can be used to tilt Kinect up and down. There are many applications for Kinect. The applications can range from systems for sign language translation (Li *et al.*, 2011) to usage the Kinect as a touch sensor (Wilson, 2011). In many articles, we can find a description of Microsoft Kinect potential to reconstruct the shape of surrounding objects (Zollhöfer *et al.*, 2011), Cui and Stricke (2011), Izadi *et al.* (2011)). Similar approach is used in professional optical tracking solutions for CAVEs. Therefore, we focused on this device and measured and evaluated in detail its precision, hence, its suitability for usage for shape reconstruction.

Following section briefly outlines toolkits available for Kinect. Further, measurements of the precision are published and summarized. The last section discusses the suitability for professional solutions.

METHODS AND RESOURCES

Libraries for Kinect

There are many APIs designed for Kinect. Majority of them is open-source, nevertheless, some are available just for non-commercial use. The common feature of all of them is the ability to access raw depth data. However, they differ in approaches used for accessing these data. This part describes few selected libraries: OpenNI, KinectSDK, Libfreenect, KinectRGB Demo and PCL (Point Cloud Library). The first three use their own drivers to access raw data. RGBDemo and PCL use OpenNI to access raw depth data. Summary of the APIs functions can be found in Tab I.

OpenNI

OpenNI[1] is an open-source framework that provides an API for writing applications using

I: *Overview of Kinect APIs*

	OpenNI	KinectSDK	Libfreenect	KinectRGB Demo	PCL
Kinect Driver	PrimeSense	Microsoft	Libfreenect	PrimeSense	PrimeSense
Access to depth data	OpenNI	KinectSDK	Libfreenect	OpenNI	OpenNI
Point Cloud creation	X	X	YES	YES	YES
Skeleton Tracking	YES	YES	YES	YES	X
Calibration	X	X	X	YES	X
Reconstruction	X	X	YES	YES	X
Multiple Kinect Support	X	YES	YES	YES	X
Operating System	Linux, Windows	Windows 7	Linux/Os X/ Windows	Linux/Os X/ Windows	Linux/Os X/ Windows
System type	32b/64b	32b/64b	32b	32b	32b/64b
Number of dependencies (for Windows)	1	0	3	5	8
Current version	1.3.3.6	1.0.0.45	0.1.1	0.6.1	1.3

1 More information on: http://www.openni.org

natural interaction[2] (NI). OpenNI was created in 2010 shortly after Kinect was released. OpenNI is easy to install, however, to use OpenNI with Kinect, the proper drivers (SensorKinect) must be installed. The drivers were created by PrimeSense, one of the companies that developed Kinect. OpenNI provides access to raw depth data, full body tracking, hand tracking, gesture recognition and more.

There are many advantages in using OpenNI. The biggest advantage is the licence which includes commercial use. This allows companies to use OpenNI in their applications. OpenNI is also a cross-platform and supports Microsoft Windows, Mac OS X and GNU/Linux, both 32b and 64b systems. The other advantages are multiple sensors support and more.

OpenNI has two disadvantages: does not provide access to microphone array and does not allow access to motor, hence users cannot tilt Kinect up and down.

KinectSDK

KinectSDK[3] is an API for Kinect released by Microsoft. The features include access to raw depth data, skeleton tracking and advance audio capabilities. KinectSDK is not an open source application and the licence is limited to non-commercial use (it is used mostly for academic purposes).

This API has several advantages and disadvantages. The advantages are support for audio, the ability to control motor/tilt and support of multiple Kinect devices. The biggest advantage for novice users is an easy installation with almost no dependencies. On the other hand, it also has many disadvantages. Among many are licence limited to non-commercial use only, does not support hand tracking and gesture recognition, does not work with SensorKinect drivers and requires Windows 7 to run.

Libfreenect

Libfreenect[4] is software created by OpenKinect community. OpenKinect is a community of people who created libfreenect in order to use Kinect on PC and other devices. Libfreenect is, same as OpenNI cross-platform (Microsoft Windows, Mac OS X. GNU/Linux) but uses different hardware drivers to access Kinect depth data. Among features of libfreenect are hand and skeleton tracking, point cloud generation and 3D reconstruction.

Libfreenect is easy to install. All the dependencies can be downloaded from OpenKinect website. It also has a large community behind it, which makes it very promising for the future.

The disadvantage of libfreenect is in hardware drivers. Libfreenect cannot be used with PrimeSense drivers which, in practice means manually changing hardware drivers.

PCL

Point Cloud Library (PCL)[2] is an open source library that allows capturing point clouds directly from a Kinect depth data and subsequently processing them. PCL uses OpenNI to access depth data. The library contains methods for operations on point clouds. Among many are filtering, segmentation, surface reconstruction, visualization, features and keypoints detection. PCL is also a cross-platform library and supports 32b and 64 systems. It is not as easy to install as the others because it has a lot of dependencies.

PCL has very large community behind it, and the project is still growing which is proved by the release of a new version (1.3) in November 2011. PCL is very well documented with a lot of samples and with a large forum. More about PCL can be found in Rusu and Cousins (2011).

Kinect RGBDemo

Kinect RGBDemo[6] is an open source project created by Nicolas Burns. The idea behind RGBDemo is to provide a simple toolkit to start playing with Kinect. RGBDemo has many functionalities including access to depth data, skeleton tracking, reconstruction, object tracking and many more. RGBDemo is cross-platform and supports both OpenNI and libfreenect backends. RGBDemo is easy to install although it requires 5 dependencies. RGBDemo is the best way for a novice user to get started with Kinect.

RESULTS

To test the Kinect measurement precision we used the OpenNI library. OpenNI library allows us to access raw depth data. Depth data are stored in a one-dimensional matrix where individual image pixels are indexed as follows:

Resulting information is a point distance (in millimetres) from the plane representing the position of a Kinect sensor. This means, we have X, Y and Z coordinate where X and Y are in pixels, and Z is in millimetres. X and Y are called Projective coordinates and do not represent physical distance. To calculate object size, the X and Y coordinates must be transferred into millimetres thus converting them from Projective coordinates to Real World coordinates. This can be done by calling OpenNI method *ConvertProjectiveToRealWorld*. Once

2 More information on: http://www.primesense.com/en/openni
3 More information on: http://kinectforwindows.org
4 More information on: http://openkinect.org
5 More information on: http://pointclouds.org
6 More information on: http://nicolas.burrus.name/

II: *Precision measurements*

Real distance	Average measured distance	Standard deviation in distance	Average measured object size	Standard deviation in object size
Object size: 96,5 mm				
500	513.8	3.2	93.8	1.7
750	764.2	6.4	95.1	2.2
1 000	1 018.9	6.3	95.9	2.1
1 500	1 521.1	9.7	95.3	3.7
2 000	2 041.8	10.4	98.9	5.1
3 000	3 080.8	21.2	97.9	9.8
4 000	4 257.7	76.8	118.2	32.9
Object size: 252 mm				
500	516.2	3.4	248.9	2.2
750	769.1	5.0	253.1	3.6
1 000	1 015.6	5.2	252.0	5.0
1 500	1 519.0	6.6	252.9	4.3
2 000	2 033.9	11.3	254.5	9.2
3 000	3 082.0	15.8	249.7	11.6
4 000	4 160.0	54.2	253.0	10.5
Object Size: 476 mm				
500	519.6	5.0	473.9	3.3
750	774.1	10.5	476.5	2.5
1 000	1 026.0	4.8	476.1	3.9
1 500	1 532.1	6.3	484.5	3.4
2 000	2 050.3	22.3	483.5	6.6
3 000	3 098.5	29.7	481.3	6.3
4 000	4 185.3	65.4	498.2	18.0

the point coordinates are transferred into real world coordinates, we can calculate the Euclidian distance between two points in 3D space.

We measured several different objects, three of had sizes: 96,5 mm, 252 mm and 476 mm. The objects were measured in 7 different distances from Kinect. The distances were 500, 750, 1 000, 1 500, 2 000, 3 000 and 4000 mm. The distance did not exceed 4 meters because 4.5 meters is a depth limit given by Microsoft. Measurements results are in Tab II. The real world distance was measured from the edge of Kinect Sensor to the edge of the measured object. All measurements are in millimetres.

The measurements clearly show that further the object is from Kinect, the larger is the difference between the real world distance and distance measured by the Kinect. This is very apparent with the smallest object (96.5mm) where the difference is almost 26 centimetres. It correlates with the standard deviation which is 76.8 mm.

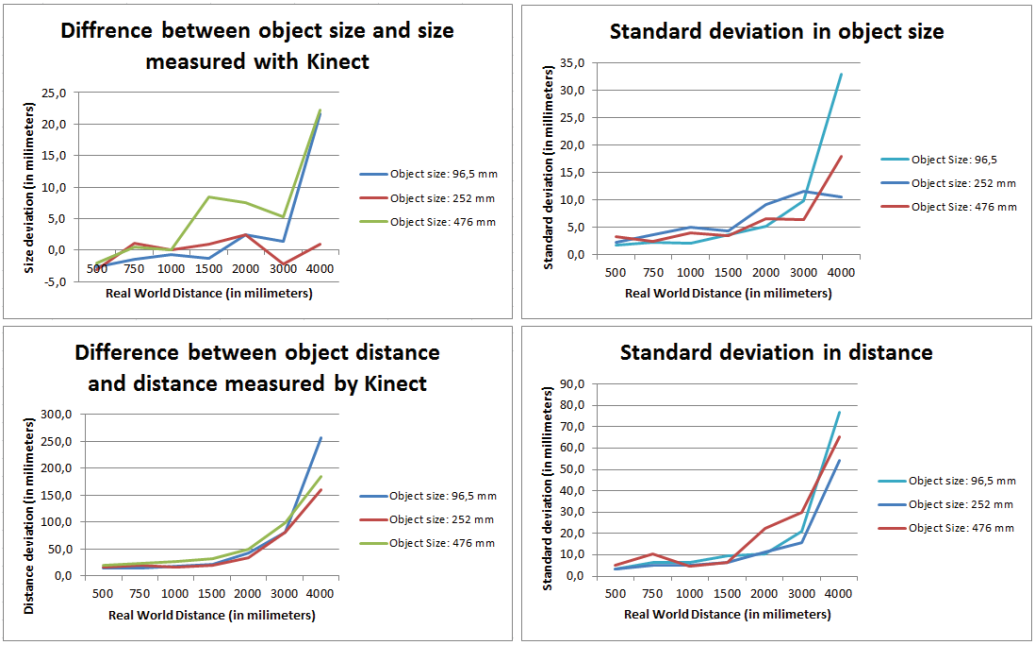
The most accurate object size was measured with all 3 measured objects at 1 meter from Kinect. The best results were with the middle object where the average size for all measurements is 252.1 mm. The worst over all measured object size is with the largest object.

Standard deviation in the distance and object size grows with the distance from Kinect. The largest standard deviation is with smallest object. The standard deviation jumped from 9.8mm (for 3 meters) to 32.9 mm (for 4 meters). This clearly shows that Kinect is not well suited for measuring small object at large distances.

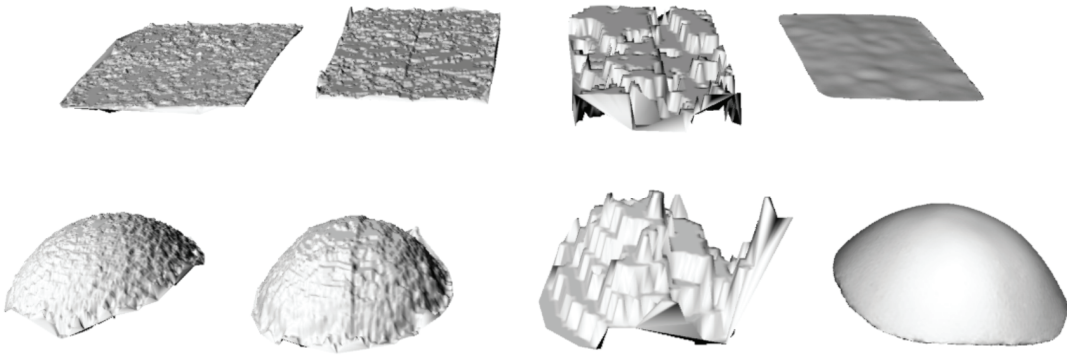
DISCUSSION

As is obvious from the measurements, the measurement precision of distance between point and Kinect is less precise than the measurement of line between two points within the same distance from the Kinect. These results were anticipated. Generally, line between two points could be rather easily estimated using the camera projection matrix, however, to measure the distance from the camera must be used more complex methods (in this case infrared reflection).

Measured precision in the z axis (distance from the Kinect) is approximately from 3 to 10 cm within the distance up to two meters. Beyond this point, the error is rising substantially. The precision in the xy plane is usually within 5 cm up to 2 m distance. Therefore, we could say, that this device has not high enough precision for scanning the differences in the product shape. The overall prototype shape could be reconstructed (we could find e.g. statues



2: *Measurement error trends*



3: *Amount of points on a sphere surface from distance 50, 100 and 300 cm. Surface of the last sphere is smoothed by averaging the position of mesh vertices is all 3 axis.*

reconstructed by the Kinect), however, small adjustments could not be precisely evaluated.

Nevertheless, for many applications, this solution is suitable – in case it is not necessary to measure a complex or even non-convex hull. Moreover, overall shape could be significantly improved by an appropriate approximation method in common professional software (e.g. splines approx.). Deviations will be in this case suppressed; therefore, the result will be a smooth hull (Fig. 3). This approximated hull could be then again converted into a triangular network. This smoothing could however significantly reduce the amount of reconstructed details. It is necessary find a compromise between the amount of smoothing and the degree of precision.

This solution is able to make a 3D reconstruction for a fraction of cost than the profession 3D scanner. It is not a substitute of a 3D scanner, but it presents an option for projects where 3D scanner cannot be used from the financial reasons.

SUMMARY

This article is focused on the possibility of usage Microsoft Kinect device as a low-cost alternative to a profession 3D scanner. A comparison of libraries that can be used to obtain depth data from Kinect device is presented. We selected one particular library and evaluated the precision of Kinect measurements in all axes. As has been anticipated, the precision along the z-axis (distance from

the device) is substantially lesser than the precision along x and y-axis. It is possible to reach a sub 10 cm precision within 2 m distance from the Kinect device. These measurements could be further improved by suitable approximation method. However, such approximation could destroy fine details of the surface. Generally, this solution could be used in applications where is necessary to make a 3D reconstruction of larger objects without many sub 10 cm details or non-convex shapes. It does not present a competition to the professional 3D scanners; however, it is a viable low-cost alternative.

Acknowledgements

This paper is written as a part of a solution of the project IGA FBE MENDELU 31/2011 and FBE MENDELU research plan MSM 6215648904.

REFERENCES

RUSU, R. D., COUSINS, S., 2011: 3D is here: Point Cloud Library (PCL). 2011 *IEEE International Conference on Robotics and Automation*, pp. 1–4.

ŠŤASTNÝ, J., PROCHÁZKA, D., KOUBEK, T., LANDA, J., 2011: Augmented reality usage for prototyping speed up. *Acta universitatis agriculturae Mendelianae Brunensis*, 59, 4: 353–360. ISSN 1211-8516.

PROCHÁZKA, D., ŠTENCL, M., POPELKA, O., ŠŤASTNÝ, J., 2011: Mobile Augmented Reality Applications. *Mendel 2011: 17th International Conference on Soft Computing*. p. 469–476. ISBN 978-80-214-4302-0.

ZOLLHÖFER, M. et al., 2011: Automatic reconstruction of personalized avatars from 3D face scans. *Computer Animation and Virtual Worlds*. p. 195–202. ISSN 15464261.

CUI, Y., STRICKE, D., 2011: 3D Shape Scanning with a Kinect. *SIGGRAPH 2011 Posters*. ISBN 978-1-4503-0971-4.

IZADI, S. et al., 2011: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*. p. 559–568. ISBN 978-1-4503-0716-1.

LI, K. F., LOTHROP, K., GILL, E., LAU, S., 2011: A Web-Based Sign Language Translator Using 3D Video Processing. *NBIS '11 Proceedings of the 2011 14th International Conference on Network-Based Information Systems*, p. 356–361, ISBN 978-0-7695-4458-8

WILSON, A. D., 2010: Using a depth camera as a touch sensor. *ITS '10 ACM International Conference on Interactive Tabletops and Surfaces*, p. 69–72, ISBN 978-1-4503-0399-6.

Address

Ing. Jaromír Landa, Ing. David Procházka, Ph.D., Ústav informatiky, Mendelova univerzita v Brně, 613 00 Brno, Česká republika, e-mail: jaromir.landa@mendelu.cz, david.prochazka@mendelu.cz