# Toward synchronized, interference-free, distributed 3D multi-sensor real-time image capture

Chengsi Zhang
*Electrical and Computer Engineering*
*Faculty of Engineering and Applied Science*
*Memorial University*
*cz2075@mun.ca*

Zizui Chen
*Electrical and Computer Engineering*
*Faculty of Engineering and Applied Science*
*Memorial University*
*zizui.chen@mun.ca*

Stephen Czarnuch
*Electrical and Computer Engineering*
*Faculty of Engineering and Applied Science*
*Memorial University*
*sczarnuch@mun.ca*

***Abstract*—Multiple three dimensional (3D) computer vision devices capturing from different perspectives provides better 3D scene coverage. This multi-sensor approach has been widely used in 3D model scanning, scene reconstruction, and motion streaming or recording applications. However, collecting data from multiple 3D cameras is challenging due to limitations such as bandwidth, synchronization, or interference. Common basic 3D sensors such as Microsoft Kinect v2 and Asus Xtion Pro have limited resolution and frame rate, so the number of 3D sensors that one computer can support is restricted. More advanced 3D sensors like Intel RealSense D4xx series and Asus Xtion 2 provide wider variety of resolution and frame rate, but multiple-sensor capture is still limited by camera interference and synchronization. Therefore, we have developed a new multi-sensor grabber based on the new Microsoft Azure Kinect in order to address the bandwidth, synchronization and interference challenges. The grabber contains: *configurable synchronization component* which handles synchronization and avoids interference between sensors; *capture component* which receives RGB and depth outputs from each sensor and stores them into the host computer; and *process component* which converts stored raw data to the dedicated 3D point cloud data. Our results show that this grabber can stably capture and produce synchronized 3D data from multiple sensors, and it can support up to six sensors when using multiple host computers. We evaluated the captured results on our perspective independent 3D point cloud registration algorithm, and provided robust registration results.**

***Index Terms*—3D data capture, Homography, homography decomposition, point cloud segmentation, 3D data analysis.**

## I. Introduction

Compared to two dimensional (2D) information, three dimensional (3D) data are more useful at providing additional spatial geometry information and representing more complex structures[9], which allows intuitive visualization, geometric measurement, and space analysis. Therefore, the scenes and objects that are represented by 3D information have been widely used in many industries, such as training and simulation[10][11], augmented reality(AR) / virtual reality(VR)[12][13], gaming[14] and movies[15].

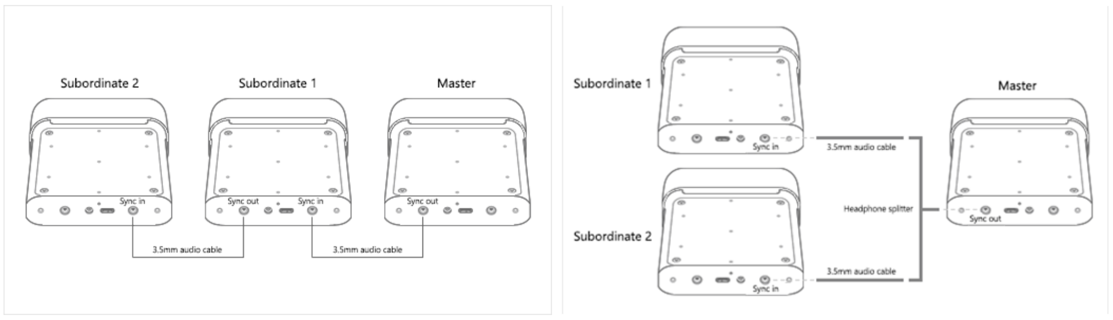
However, due to the limited field of view, capture perspective and accuracy requirements, 3D sensors still cannot capture all the information in a large scene or demonstrate 360 degree full coverage of an object within a single frame. In order to provide more useful information, a 3D digital copy of a scene or an object is usually produced by combining multiple frames from one 3D sensor[16] (requiring temporal inference) or registering inputs from multiple 3D sensors[1] (introducing complexity).

Registering inputs from multiple 3D sensors more reliably captures high-quality 3D data if some challenges can be resolve. For example, infrared camera reconstruction suffers from relatively low accuracy [30], which further leads to precision loss in generated point clouds. Registering multiple point clouds requires significant efforts toward coordination and synchronization of the sensors, as well as addressing bandwidth and memory limitations. We propose a novel, distributed architecture image grabber that supports sensor synchronization and capture from up to six commercial depth sensors.

## II. Related work

Existing sensors capable of capturing 3D data including infrared-based systems (e.g. Microsoft Kinect [20], Intel RealSense [21] and Leap Motion [22]), LiDAR-based systems [23], Stereo vision-based systems [24] and more. Among these systems, inferred-based sensors feature as low-cost, widely available and with acceptable accuracy in short-range sensing [25], [26]. For example, *Microsoft Kinect* is widely used in research, largely because of the release of the open-source SDK [27]. With the SDK, it is possible to capture a 320x240 16-bit depth images at 30 frames per second. The more recent models in Kinect series further improve the resolution and performance [28], [29] to 512x512 16-bit depth images at 30 FPS.

To register 3D images, multiple sensors need to be synchronised in order to minimise the errors that arise from



(a) Daisy-chain configuration (b) Star configuration

Fig. 1: Azure Kinect DK synchronization configurations

object movements in scene. Most existing sensors, such as Microsoft Kinect 2, do not have cross-camera or cross-machine synchronisation capabilities.

A computer, without special hardware, usually can support up to two depth sensors through the USB 3.0 interface, or four depth sensors using the USB 2.0 interface. Extension cards can be installed to overcome the USB bandwidth limitation to a certain point, but interference still restricts the number of sensors that can generally be synchronised. Depth cameras emit invisible infrared light and measure the time taken to receive the reflected infrared light. When multiple sensors are emitting at the same time, the measurements interfere with each other, reducing the overall accuracy of each sensor. This is because, in general, a sensor cannot tell the difference between the infrared light emitted by itself and the infrared light emitted by other sensors. Thus, the accuracy of the depth image is significantly reduced, and increases with the introduction of each new sensor. The most recent Microsoft depth sensor, the Azure, addresses this issue through a software-based time sharing approach which quickly turns on and off the infrared source of each camera. We build on this software-based solution, which can theoretically support up to eight *subordinate* sensors along with a ninth *master* node, to develop our distributed registered image grabber.

## III. Methodology

Our grabber is built based on the Azure Kinect camera which consists of a RGB camera, an infrared camera and an inertial measurement unit (IMU). Compared to other existing 3D sensors, the Azure Kinect camera offers a large variety of image resolutions. There are six different resolutions for the RGB sensor with four configurable frame rates, while the depth sensor has four different resolution with four configurable frame rates based on the user-set depth sensor field of view (FOV) mode[17]. Unlike the Kinect V2, where the capture frame rate only depends on the bandwidth and ambient lighting conditions, all the configurations of the Azure Kinect are customizable. Accordingly, our grabber can be flexibly configured based on the research or data requirements. In addition, another essential feature of the Azure Kinect is that it natively supports multiple sensor synchronization. Each Azure Kinect device has one "Sync In" and one "Sync Out" portal. By connecting two Azure Kinect devices with a 3.5mm audio cable, both devices can be synchronized on the sensor level [18].

Our grabber consists of a configurable synchronization component, capture component, and post-process component. The configurable synchronization component handles the capture mode of each sensor as well as synchronization among all the connected Azure Kinect devices; the capture component is responsible for receiving the raw RGB and depth data from all the sensors; and the post-processing component converts the raw data capture by each Microsoft Azure sensor into an overall registered 3D point cloud image.

### A. Synchronization component

There are two type of configurations when using a 3.5 mm audio cable to connect multiple Azure Kinect sensors. With normal 3.5 mm male-to-male audio cables, multiple Azure Kinect devices can be connected in a daisy-chain configuration by connecting the "Sync out" port of each sensor to the "Sync in" port of the next sensor sequentially as shown in Figure 1 (a). Within this connected sensor group, the first sensor does not have anything connected to it's synchronization input, and is therefore the *master* camera. Alternatively, multiple cameras can be configured in a star configuration (Figure 1 (b)), where the *master* camera is central to the devices, and all other *subordinate* devices are directly connected to the *master* in parallel.

Unlike the Kinect sensor, When using most other existing 3D sensors, the capture action periodically happens only based on the frame rate of each individual sensor. In this way, the grabber is only able to roughly synchronize the frames based on the timestamp, and as a result frames will be discarded or interpolated when the cameras' frame rates are different. When using the Azure Kinect, one the other hand, since the capture action of all the subordinate devices are triggered by the master camera, each frame provided by one sensor will always align with data from all other sensors perfectly. Therefore, the synchronization of multiple Azure Kinect sensors is achieved on the sensor level, and one instance of our grabber only controls and reads data from one

Azure Kinect device at a time, reducing the USB bandwidth requirements.

In addition, the configurable synchronization component also provides an input parameter to define the capture delay between the master sensor and a subordinate sensor. Within an Azure Kinect device group, only one sensor's IR is turned on while all other sensors are in an idle state. Hence, the interference between each sensor is entirely eliminated. The minimum capture delay between two sensors is $160\mu s$ [18], so the largest delay will be $160\mu s\ /sensor \times 8\ sensors = 1280\mu s$, which is smaller the minimum idle time $1450\mu s$ of a Azure Kinect sensor, between the master camera and last subordinate camera, assuming all eight subordinate cameras are connected with the master camera.

## B. Capture component

The capture component allow users to specify the RGB image resolution, the depth camera capture mode (depth camera FOV and resolution), and frame rate based on the data requirement. Since we are using the captured results for 3D data registration, we keep all the configuration parameters the same for all Azure Kinect sensors within the post-process component. In order to ensure the generated 3D point cloud has the same frame rate as the raw RGB/Depth data, the capture component obtains the raw RGB and depth data frames from a sensor and puts them into the memory First in First out (FIFO) queue with only the current timestamp and the sensor serial number. At the same time, since we want to ensure the queue cannot be quickly filled up by the raw data frames, we dedicate another thread to pull out frames in the FIFO queue and save them to the local disk as binary files with the combination of their captured timestamp and camera serial number as the unique file name. Finally, we save the device calibration parameters and capture configurations to disk as the Azure Kinect capture configuration file to support future post-processing of the data. Our sensor level synchronization ensures that the capture component of our grabber simply reads in data from each sensor and saves it to the local disk without any processing which dramatically improves the capture efficiency of our grabber.

## C. Post-process component

The post-process component is an isolated component, because converting RGB and depth 2D images into 3D point cloud frames is computational expensive. Furthermore, this component needs to be easily changeable based on specific applications outside our intended multi-sensor registration. After the capture process is completed and all the RGB and depth images are saved to the local disks as binary files, the post-process component can be triggered by the user to process the data of each sensor by following the timestamp sequence. In order to generate the correct 3D point cloud data from the raw RGB and depth image data, the user has to use the same RGB image resolution, the depth camera capture mode, and frame rate capture configurations from the Azure Kinect capture configuration file as when these image data were captured. Unlike the previous Kinect sensors, Azure Kinect device captures the depth image by using a fish eye sensor. The raw RGB and depth image is first transformed into a normal depth image by assigning the pixels in the fish eye depth image to the pixels in the undistorted color and depth images based on the transform mapping table, which is generated by the Microsoft Azure internal API function *k4a_transformation_t_get_context*. Since this API function requires the extrinsic and intrinsic parameters of the Azure Kinect device that captured these data, we again read the sensor calibration parameter data from the configuration file. Finally, we generate the 3D point cloud frames by combining the undistorted RGB and depth images and storing on the local disk as *ply* format files.

| Session | # of participants | frames |
|---|---|---|
| 1 | 1 | 176 |
| 2 | 3 | 328 |
| 3 | 2 | 193 |
| 4 | 3 | 464 |
| 5 | 2 | 730 |
| 6 | 2 | 717 |
| 7 | 1 | 711 |
| 8 | 2 | 813 |
| 9 | 2 | 695 |
| 10 | 2 | 745 |
| 11 | 1 | 717 |
| 12 | 1 | 774 |
| 13 | 2 | 789 |

TABLE I: Caption

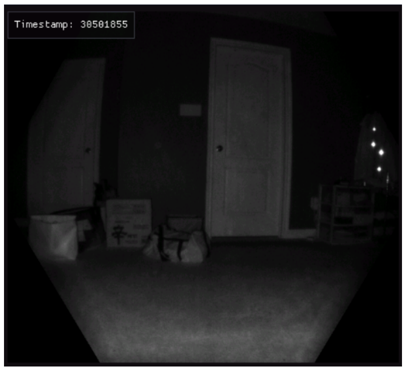


Fig. 2: Azure Kinect fish eye depth image

## IV. Data

By using this grabber, we have captured 13 data trials with at least one of four human participants walking in the scene (see Table I) for our previous perspective independent ground plane estimation research [19] and multiple sensor registration work. The recorded raw data are organised in pairs consisting of one frame of RGB image encoded in 3-channel 8-bit binary format, with a size of approximate 8100 kilobytes, and one frame with the corresponding depth image encoded in single-channel 16 bit binary format, with a size of around 720 kilobytes. The generated 3D point cloud frame encoded in *ply* format has a size of around 7700 kilobytes.

## V. Discussion and Conclusions

We developed a new grabber that can capture RGB and depth images from multiple sensors, based on the Microsoft Azure Kinect device and Microsoft Azure API. Our grabber is efficient for multiple sensor capture, resolving common issues that other 3D sensor grabbers have encountered, such as limitations associated with transfer bandwidth, data synchronization, and sensor interference. In addition, our grabber is highly configurable and user friendly so that the user can capture the data based on their customized requirements. Due to the high accuracy when compared with our attempts to use other 3D sensors, we have used this grabber to capture the data trials that were required for our multiple sensor registration work as well as additional datasets for our previous ground plane estimation research [19]. In the future, we will verify the performance of this grabber while using up to eight Azure Kinect devices and even further improve the efficiency and accuracy of this grabber.

## References

[1] Jia, Zhen, et al. ”Method and system for multiple 3D sensor calibration.” U.S. Patent No. 10,371,512. 6 Aug. 2019.

[2] Sultani, Zainab Namh, and Rana Fareed Ghani. ”Kinect 3D point cloud live video streaming.” Procedia Computer Science 65 (2015): 125-132.

[3] Schröder, Yannic, et al. “Multiple kinect studies.” Computer Graphics 2.4 (2011): 6.

[4] Bhandari, Ayush, et al. “Resolving multipath interference in Kinect: An inverse problem approach.” SENSORS, 2014 IEEE. IEEE, 2014. In SENSORS, 2014 IEEE (pp. 614-617). IEEE.

[5] Lee, Ju-Hwan, Eung-Su Kim, and Soon-Yong Park. “Synchronization error compensation of multi-view RGB-D 3D modeling system.” Asian Conference on Computer Vision. Springer, Cham, 2016.

[6] Lachat, Elise, et al. ”Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling.” Remote Sensing 7.10 (2015): 13070-13097.

[7] Grunnet-Jepsen, Anders, et al. ”Using the RealSense D4xx depth sensors in multi-camera configurations.” Santa Monica, CA, USA (2018).

[8] Martín, Roberto Martín, Malte Lorbach, and Oliver Brock. ”Deterioration of depth measurements due to interference of multiple RGB-D sensors.” 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014.

[9] Meyer-Marcotty, Philipp, et al. ”Impact of facial asymmetry in visual perception: a 3-dimensional data analysis.” American Journal of Orthodontics and Dentofacial Orthopedics 137.2 (2010): 168-e1.

[10] Waran, Vicknes, et al. ”Injecting realism in surgical training—initial simulation experience with custom 3D models.” Journal of surgical education 71.2 (2014): 193-197.

[11] Bruzzone, Agostino G., and Francesco Longo. ”3D simulation as training tool in container terminals: The TRAINPORTS simulator.” Journal of Manufacturing Systems 32.1 (2013): 85-98.

[12] Garon, Mathieu, et al. ”Real-time high resolution 3D data on the HoloLens.” 2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct). IEEE, 2016.

[13] Cox, Donna J., Robert M. Patterson Jr, and Marcus L. Thiebaux. ”Virtual reality 3D interface system for data creation, viewing and editing.” U.S. Patent No. 6,154,723. 28 Nov. 2000.

[14] Adams, Khaled. ”3D enhancements to gaming components in gaming systems with real-world physics.” U.S. Patent No. 10,115,261. 30 Oct. 2018.

[15] Emoto, Michiko, Shinya Miyamoto, and Kazuyoshi Yamamoto. ”Navigation apparatuses, methods, and programs for generation of a 3D movie.” U.S. Patent No. 7,974,781. 5 Jul. 2011.

[16] Whelan, Thomas, et al. ”Real-time large-scale dense RGB-D SLAM with volumetric fusion.” The International Journal of Robotics Research 34.4-5 (2015): 598-626.

[17] https://docs.microsoft.com/en-us/azure/kinect-dk/hardware-specification, ”Azure Kinect DK hardware specifications”, Microsoft Azure.

[18] https://docs.microsoft.com/en-us/azure/kinect-dk/multi-camera-sync, ”Synchronize multiple Azure Kinect DK devices”, Microsoft Azure.

[19] Zhang, Chengsi, and Stephen Czarnuch. ”Perspective Independent Ground Plane Estimation by 2D and 3D Data Analysis.” IEEE Access 8 (2020): 82024-82034.

[20] Microsoft Corp ”Kinect for Windows” https://developer.microsoft.com/en-us/windows/kinect/ Kinect - Windows app development

[21] Intel ”Stereo Depth - Intel® RealSense™ Depth and Tracking Cameras” https://www.intelrealsense.com/stereo-depth/

[22] UltraLeap ”Digital worlds that feel human — Ultraleap” https://www.ultraleap.com/

[23] Taylor, Travis S ”Introduction to Laser Science and Engineering” 2019 CRC Press

[24] Ayache, Nicholas ”Artificial vision for mobile robots: stereo vision and multisensory perception” 1991 Mit Press

[25] Smisek, Jan and Jancosek, Michal and Pajdla, Tomas ”3D with Kinect” Consumer depth cameras for computer vision pg.3–25 2013 Springer

[26] Zhang, Zhengyou Microsoft kinect sensor and its effect IEEE multimedia vol.19 no.2 pg.4–10 2012 IEEE

[27] El-laithy, Riyad A and Huang, Jidong and Yeh, Michael ”Study on the use of Microsoft Kinect for robotics applications” Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium pg.280–1288 2012 IEEE

[28] Microsoft Corp ”Kinect for Windows SDK” https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10)

[29] Microsoft Corp ”About Azure Kinect DK” https://docs.microsoft.com/en-us/azure/kinect-dk/about-azure-kinect-dk

[30] Khoshelham, Kourosh ”Accuracy analysis of kinect depth data” ISPRS workshop laser scanning vol.38 no.1 2011