Automatic and Adaptable Registration of Live RGBD Video Streams

Afsaneh Rafighi* Sahand Seifi† Oscar Meruvia-Pastor‡

Department of Computer Science
Memorial University of Newfoundland

Abstract

We introduce DeReEs-4V, an algorithm that receives two separate RGBD video streams and automatically produces a unified scene through RGBD registration in a few seconds. The motivation behind the solution presented here is to allow game players to place the depth-sensing cameras at arbitrary locations to capture any scene where there is some partial overlap between the parts of the scene captured by the sensors. A typical way to combine partially overlapping views from multiple cameras is through visual calibration using external markers within the field of view of both cameras. Calibration can be time consuming and may require fine tuning, interrupting gameplay. If the cameras are even slightly moved or bumped into, the calibration process typically needs to be repeated from scratch. In this article we demonstrate how RGBD registration can be used to automatically find a 3D viewing transformation to match the view of one camera with respect to the other without calibration while the system is running. To validate this approach, a comparison of our method against standard checkerboard target calibration is provided, with a thorough examination of the system performance under different scenarios. The system presented supports any application that might benefit from a wider operational field-of-view video capture. Our results show that the system is robust to camera movements while simultaneously capturing and registering live point clouds from two depth-sensing cameras.

CR Categories: I.2.10 [ARTIFICIAL INTELLIGENCE]: Vision and Scene Understanding—3D/stereo scene analysis I.4.1 [IMAGE PROCESSING AND COMPUTER VISION]: Digitization and Image Capture—Camera Calibration I.4.3 [IMAGE PROCESSING AND COMPUTER VISION]: Enhancement—Registration;

Keywords: wide-screen 3D video, registration of multiple RGBD cameras, field of view extension

1 Introduction

With the introduction of the Kinect, RGBD sensing cameras became widely acknowledged as input devices for gesture-controlled and immersive virtual reality games, offering affordable hardware to obtain hand and body gesture recognition. Since then, RGBD sensing cameras have been adopted for a wide range of applications in arts, science, and the health sector among others [Lun and Zhao 2015]. One limitation of these cameras is in their field of view. In a typical playing scenario two adult users standing side by side and playing against each other in an action game might occasionally hit each other during gameplay. A solution to address this problem is having a system where each user has his or her own separated playing area, sometime over a network. However, this might not be ideal for family or group play. In this work, a solution is proposed to widen the operational field-of-view of Kinect-type sensors by using two sensors placed next to each other to capture elements that would otherwise fall outside of the scope of a single camera. The cameras can be placed in arbitrary locations in front of the captured scene; they do not need to be at the same height or otherwise aligned. The main requirement for this system to work is that the cameras share part of their corresponding fields of view to allow for automatic 3D video registration to take place. Using this setup, users can move the sensors and combine the views from two cameras to involve more users in the game, to provide more personal space for players to move and act more freely, or to change the focus of attention to a different part of the room.

Apart from its use in games, depth-sensing cameras are used extensively in other research and commercial applications. A recent survey on the applications of Kinect sensors lists 9 different areas of applications and over 80 different documented cases [Lun and Zhao 2015]. In Virtual Reality games, RGBD sensors are also used to provide an immersive experience [Laskowski ,Raghuraman et al. 2012, NVIDIA]. In the health-care sector, active and immersive games are being used together with RGBD sensors to help rehabilitate patients with motor disabilities and for other treatments as well [Koenig et al. 2014, Webster and Celik 2014, Altamimi and Skinner 2012]. To increase the sense of immersion in video conferences, [Zhang et al. 2013b] proposed one of the first Kinect-based 3D video conferencing systems. While most of these applications are based on the use of one camera per person, the simultaneous use of multiple RGBD cameras has also great potential. For example, as part of a medical exam, multiple calibrated cameras are being used to record the gait patterns of patients walking within large rooms [Czarnuch and Ploughman 2014, Cippitelli et al. 2015].

Applying multiple depth sensing cameras not only helps capture a larger portion of the environment: With the help of another camera, parts of the scene which are occluded by one camera can be captured by the other. For example, the left corner of Figure 1(a) is not captured due to the shadow of the user in front of camera one. However, the second camera has captured that space (Figure 1(b)). As a result, the registered point cloud provides a more comprehensive representation of the scene, as shown in Figure 1(c).

Having a multi-camera system typically requires careful calibration of the cameras as an initial setup [Zhang 2000]. After finding a coarse transformation of point clouds through calibration, registration methods such as ICP [Besl and McKay 1992] and its variations are used to perform a fine alignment on them. The calibration process can be time consuming and has to be repeated whenever the cameras are moved. It is a cumbersome task for users, whether they are in a video conferencing or playing a game; even for regular researchers, this step can be frustrating and time-consuming. In all of these cases, users could benefit from a system that is both automatic and robust, particularly if the cameras are moved while the system is working, as can easily occur in gaming situations that involve multiple parties and physical activity.

*e-mail:afsaneh.rafighi@mun.ca
†e-mail:sahands@mun.ca
‡e-mail:oscar@mun.ca

MIG '15, November 16 – 18, 2015, Paris, France.
© 2015 ACM. ISBN 978-1-4503-3991-9/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2822013.2822027

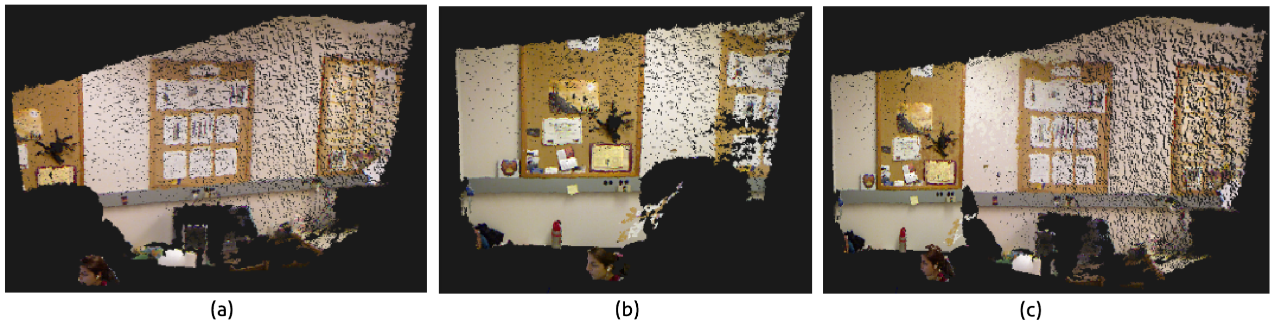

Figure 1: *Effect of having a multi-depth sensor system. Areas occluded by each of the cameras,(a) and (b), are complete in the registered point cloud(c).*

Motivation and Contributions The aim of this research is to have a robust and flexible automatic registration system when using multiple depth-sensing cameras for capturing video streams. The registration of cameras sharing partial views of the scene provides more room for players to act more freely or for more people to take part in the game. The system must be user-friendly, so that no calibration should be required for operation. In addition, the system must be robust to camera displacements as long as they share part of the viewed scene after displacement. To do this, we modified DeReEs [Seifi et al. 2014], a registration method that has been proven to work for registering individually captured RGBD images, but that we have adapted for video processing. DeReEs works for cameras that share small overlapping regions within their field of view. In contrast, other algorithms such as ICP require point clouds to be fairly similar and roughly aligned with each other prior to attempting the registration. The key contribution of this work is the introduction of an adaptable registration system that captures wider scenes by combining RGBD point cloud video streams from two Kinects that can be moved during interaction. Further, we provide a thorough demonstration of its robustness in maintaining point clouds' alignment under various conditions. Additional contributions, described in Section 3, include extending the Octree point cloud compression method for two depth sensing cameras, applying ICP in the last step of the algorithm, detecting camera movement and a mechanism to enhance the quality of the registration over multiple iterations during the first seconds of use.

Since no calibration is necessary, the system is fast and removes the hassle of the initial calibration of the cameras. In this work, we show that users can easily move cameras around to change the portion of the scenario that is captured. This feature is also useful in situations where users enter in contact with the cameras and move them inadvertently or in situations where cameras cannot be set at fixed locations permanently.

The rest of the article is structured as follows: In Section 2, a review of the most related systems along with an overview on camera calibration and point cloud registration is provided. In Section 3 our system is presented in detail. In Section 4 we present an evaluation of the proposed system, and Section 5 presents our conclusions and suggestions for extending this work.

2 Related work

A large body of work exists regarding regular camera calibration methods, which predates depth-sensing cameras. As discussed in [Zhang 2000], calibration methods are classified in two general ways as **photogrammetric calibration** and **self-calibration**. In photogrammetric calibration, an object for which its 3D geometry in the real world is known is used as reference. The latter does not require an object. By moving the camera in a scene, internal and external parameters are recovered by finding the correspondence between only three images. A common approach is to use Zhang's method [Zhang 2000], which is implemented both as a toolbox in Matlab [Bouguet 2004] and in the OpenCV library [Bradski 2000]. In this method, a planar pattern, usually a checkerboard, is observed from at least two different orientations. Then, the intrinsic and extrinsic camera parameters are obtained. The use of RGBD sensors also requires calibration methods, since the depth sensor and the RGB camera are separated by a small baseline. After taking several snapshots of the object, the 3D coordinates of the feature points are extracted from the RGB camera. Depth values are obtained from the depth sensor's coordinate system, and a feature matching between the RGB and depth image results in features getting their true depth value based on the RGBs coordinate system.

RGBD sensors have also been used for 3D scanning through registration with a single sensor, such as in [Newcombe et al. 2011] or [Henry et al. 2012], where a camera is moved around a scene or object to acquire a 3D model of it, or for indoor 3D mapping and localization, to provide 3D maps in real-time (e.g. [Zou et al. 2012]).

Since the use of multiple cameras provides an effective way to more comprehensively capture a scene and provide a sense of immersion, videoconferencing systems have long made use of multiple cameras. [Zhang et al. 2013a] proposed an immersive teleconferencing system where users are rendered in a virtual room as if they are seated around a table. Each site is equipped with 3 IR cameras, 3 color cameras and two IR laser projectors, carefully calibrated. The assumption in their setup is that there is only one user at each end during the conference and eye-tracking methods are used to render the virtual room according to each user's gaze. In a related approach, [Maimone and Fuchs 2011a] proposed a telepresence system that works by merging overlapping views from multiple Kinect cameras. There, a fully dynamic 3D scene capture system that also preserves gaze is presented. This work was improved later in [Maimone and Fuchs 2011b] by using bigger and higher resolution 3D displays to convey the feeling of a window through another room. In [Beck et al. 2013], an immersive telepresence system is presented that allows meeting of a group of people in a shared virtual 3D world. In their setup, two projection-based multi-user display systems, two arrays of Kinects, and a distributed virtual reality framework named AVANGO is used. It is important to note that in all multi-camera scenarios above, the cameras must be carefully calibrated and must be fixed and placed in a certain position during the whole session, whereas the system we propose is designed to allow users to freely move the cameras around, as long as the partial overlap requirement is satisfied.

The work of [Miller et al. 2013] proposing camera registration without human supervision or fully textured environments is also related. Their goal is to calibrate two cameras placed in a cross-eyed view setup (where the cameras do not share a background), based on the analysis of the motion of a user or an object moving within the field of view of both cameras. Instead of using feature matching, they interpret the optical flow across multiple frames and find their correspondence to obtain the extrinsic calibration parameters. Foreground objects are used to obtain an initial transformation using RANSAC [Fischler and Bolles 1981]. This step gives a rough transformation which is then refined by an occlusion-aware energy minimization step. While their approach involves multiple cameras and dynamic scenes, it has an average execution time of about 10 minutes, which makes the solution unsuitable for live operation or on-line adjustments of camera arrangements.

In this article we present a rigorous evaluation of our systems performance and its robustness in registering RGBD video streams. We also assessed our systems capability to successfully register users found in either the non-overlapping, the partially overlapping, or at the boundaries of the overlapping regions of each cameras views. We present a multiple depth-sensing camera system that allows for interactive positioning of the cameras and does not require a calibration operation, even after the cameras are displaced.

2.1 Point Cloud Registration

Point cloud registration refers to aligning point clouds captured either using one depth-sensing camera (that is capturing a scene from different viewpoints) or using multiple depth-sensing cameras. Each of these point clouds have their own coordinate system. In order to bring these point clouds to a single framework, so that overlapping parts of point clouds intersect correctly, registration methods are used. As shown in [Seifi et al. 2014], registration techniques with fast execution times, such as ICP, are mostly successful when they are used for fine alignment but are not suitable to align point clouds sharing small overlapping sections and having significant differences in their geometry.

To achieve video registration of the two RGBD streams, we implemented a modified version of DeReEs, called DeReEs-4V (DeReEs for Video). We have introduced modifications to provide registration stability and adaptability over time to enable applications that go beyond single image pair registration. We have also implemented such as Octree point cloud compression for data transmission.

3 Implementation

Two Kinect cameras, K_l and K_r, are used at the sender side and they are placed arbitrarily in an indoor environment in a way that an overlap of at least 23% exists between two viewpoints. Streams of two point clouds and RGB images, $RGBD_l$ and $RGBD_r$, are constantly being captured.

3.1 Methodology

The first stage of the registration is to find corresponding features in both point clouds. In our system data is streaming constantly. RGBD data of each camera is being buffered in separate threads at the frame rate of $30fps$. The input of the registration algorithm, which is being executed on a separate thread, is a pair of RGBD images. RGB images of two cameras are obtained from each camera's RGB buffer and are constantly being uploaded to the GPU to apply SURF feature matching on them. In the second step of DeReEs, after finding a coarse transformation, the transformation is applied on feature pairs and then, features larger than a distance threshold are removed from the set. To increase the stability of the fine transformation estimation, we use ICP on a small subset of the original point cloud which contains only the 3D feature pairs that are left after false-pair rejection, the last step of the algorithm. We observed that by performing ICP on the point clouds made from this small set of remaining 3D feature pairs we obtain a registration transformation that gives us a correct alignment and is more stable over several iterations of DeReEs-4V. This modification does not introduce much delay, as ICP is performed only on a small set of points, executing in less than $1ms$ on average. Once the transformation registration is found, it is applied on one of the point clouds. As a result, point clouds are registered and updated in $65ms$, or 15 fps on average. Point clouds can be registered continuously from scratch at this rate, and the quality of the registration produces matches that are successful in producing well matched backgrounds. However, the registration of elements in the foreground may still need some fine tuning after the initial transformation has been found, so we use the registration information gathered over several iterations of DeReEs-4V to improve the quality of the registration results when the cameras are not moving for a few seconds. After a registered point cloud is produced, it is compressed using the Octree compression method implemented in Point Cloud Library(PCL) [Aldoma et al. 2012]. Then, it is sent over the network to the client's side to decompress the registered point cloud and view it, being updated continuously. The algorithm is outlined in Algorithm 1.

```
Data: Two live video streams $RGBD_l$,$RGBD_r$
Result: Registered point clouds
Cam1IsMoved
← $DetectCameraMovement(RGBD_l$prev,$RGBD_l$curr)
Cam2IsMoved
← $DetectCameraMovement(RGBD_r$prev,$RGBD_r$curr)
T ← $DeReEs$-$4V(RGBD_l$, $RGBD_r)$
if Cam1IsMoved or Cam2IsMoved then
    Erase memory
    score ← $countfeatures$
    memory ← $T$
else
    if countfeatures > score then
        memory ← $T$
        update score
    else
        T ← $memory$
    end
end
```

Algorithm 1: Algorithm 1: An iteration of the registration system using DeReEs-4V.

To accomplish this without introducing much delay, the different tasks are executed by different threads. One thread is responsible for capturing point clouds and RGB images and putting them in their buffers, while the other thread executes the registration by reading from the input buffers. Once a registration is found, another thread performs the visualization step using Point Cloud Library visualizer.

Since the system is designed to work with live video streams, several enhancements were made to account for the two scenarios described below.

3.1.1 Scenario one: Cameras are stable

One important observation is that during live capture and registration, each registration transformation obtained will be slightly dif-

Table 1: *The speed of the different stages of DeReEs-4V measured on a case where the cameras are located apart from each other with* 30% *scene overlap (Figure* 1*).*

Step	Camera Registration (ms)	Camera Movement Detection (ms)
Feature Matching	24.15	21.33
Coarse Transformation Estimation	35.59	40.32
Bad Features Removal	0.19	0.95
ICP	0.86	1.02
Total DeReEs-4V(ms)	60.79	63.62

Table 2: *The comparison between our algorithm and Miller et al. [Miller et al. 2013]. Results of Sequence* 1 *to* 3 *are retrieved from their article.*

Our algorithm	Error$_t(cm)$	Error$_r(^\circ)$	Miller et al.	Error$_t(cm)$	Error$_r(^\circ)$
Figure 1(30% overlap)	0.62	0.44	Sequence 1	3.43	0.70
Figure 1(50% overlap)	0.59	0.61	Sequence 2	2.75	0.49
Figure 1(75% overlap)	0.70	0.98	Sequence 3	4.67	1.23

ferent than the previous one. This results in the point clouds updating themselves with different values in the registration matrix. This becomes noticeable by the viewer as a form of ”flicker” effect or vibration stemming from the continuous update of the registration information. Our solution to this problem is to detect if either camera is moved, updating the registration when any of the cameras are moved. We noticed that over a period of time of 15 seconds or more, a registration could be chosen as the best overall, as long as cameras are stable and no considerable changes in the user's movements are observed. To take advantage of this fact, we implemented a memory mechanism to keep the best possible registration found so far, as long as cameras do not move. In order to do this, we stored the score of the metric DeReEs-4V used to find the best registration. DeReEs-4V randomly selects three feature points for a number of iterations and finds transformations between the features in one camera and their corresponding pairs in the other camera. After the transformation matrix is found, it is applied to all features. The number of features which have distance less than a threshold ($5cm$) is counted. After that, the transformation with the highest number of successful feature pairs is chosen as the best transformation. In our case, after the score is found and the transformation is used, this score and its transformation are saved in memory. Each time a new registration at a different time stamp is obtained, the score is compared to the previous one in the memory. If the score is higher, the best transformation over time is then changed and new score and its transformation are saved in memory.

Table 3: *Execution times of camera motion detection and registration per iteration and across multiple iterations to achieve seamless registration, measured on different video stream capture scenarios:. Case 1-One camera is located higher than the other (Figure 5). Case 2-Cameras are located far from each other with* 30% *overlap (Figure* 1*). Case 3-Two users with one being captured by only one of the cameras (Figure 3).*

Step	Case 1	Case 2	Case 3
Camera Movement Detection (ms)	68.55	63.42	72.75
DeReEs-4V Registration (ms)	77.81	66.40	57.73
Seamless Registration (seconds)	8.2	15.30	12.25

3.1.2 Scenario two: Cameras are moved

When the cameras are moved, the memory has to be refreshed, since the best transformation found so far is not correct anymore for the new coming point clouds. To detect camera movement, we take advantage of DeReEs-4V itself. A separate thread is responsible for applying DeReEs-4V on each camera's buffer. Assuming we have two RGBDs from the same camera (left one, for instance) taken at time t_{prev} and t_{curr}. Each new RGBD obtained from each camera is given as input to DeReEs-4V. The result is a registration matrix that shows the transformation and orientation of the two frames. This matrix is saved and the current RGBD is saved as the previous frame while the new frame is obtained. DeReEs-4V is then applied on these two RGBDs. If the cameras have not been moved, the difference between the new transformation matrix and the previous one would be minimal. If this differential matrix is higher than a certain threshold (0.05), the main thread receives the notification that cameras have moved, causing the internal memory to be erased and the registration data to be reset.

4 Evaluation

In our experiments, two Microsoft^{TM}XboX 360 Kinect cameras are used at each end to capture a portion of an indoor office space. The Kinects are connected to a PC equipped with an Intel(R) Core(TM) i7-960 CPU with a NVIDIA GeForce GTX 570 GPU, 1280MB memory, 480 CUDA cores, running Ubuntu 14.04 64-bit.

The GPU implementation of SURF feature matching in OpenCV is used. The Point Cloud Library is used for point cloud stream capturing from both Kinects simultaneously to visualize the aligned point clouds, and to use Octree point cloud compression implementation for streaming the point clouds over the network.

Our experiments are based upon different video capturing scenarios where one, two or three users are captured under varying conditions as shown in Figures 2 to 5. The ground truth to which we are comparing our results is obtained by calibrating cameras using the RGBDemo software [Burrus 2012], designed for Kinect visualization and calibration.

As mentioned in Section 2, the work of [Miller et al. 2013] is the closest to what we achieved here in terms of alignment accuracy. A direct comparison to their results is not possible since we did not have access to their configurations and their video sequences; however, we use the same methodology to obtain the ground truth. The results and sequence pictures mentioned in Table 2 of Miller et al.'s method are obtained from their paper [Miller et al. 2013].

We compared our results to the classic calibration method. Miller et

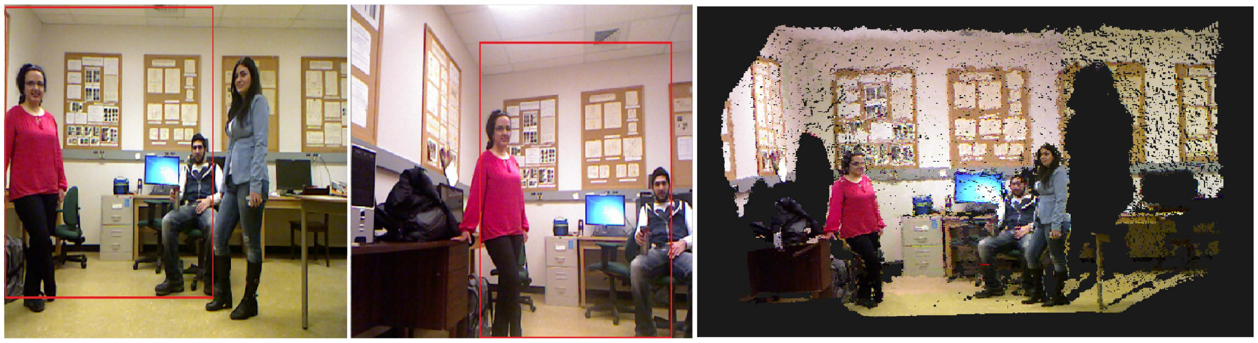

Figure 2: *An example of successful registration of a multi user scene. Two users are captured by two Kinects, while the other one is captured by only one camera. Images on the left shows the portions of the scene each Kinect cameras are capturing.*

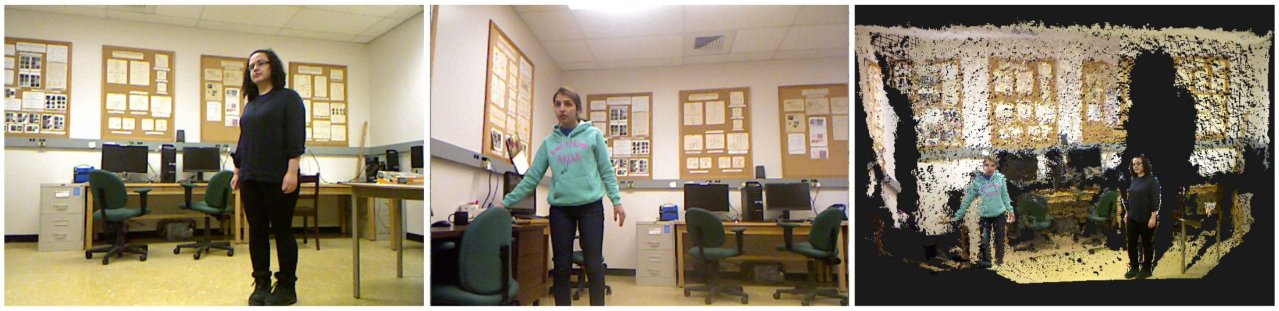

Figure 3: *An example of successful registration of a multi-user scene where each user is captured by only one Kinect.*

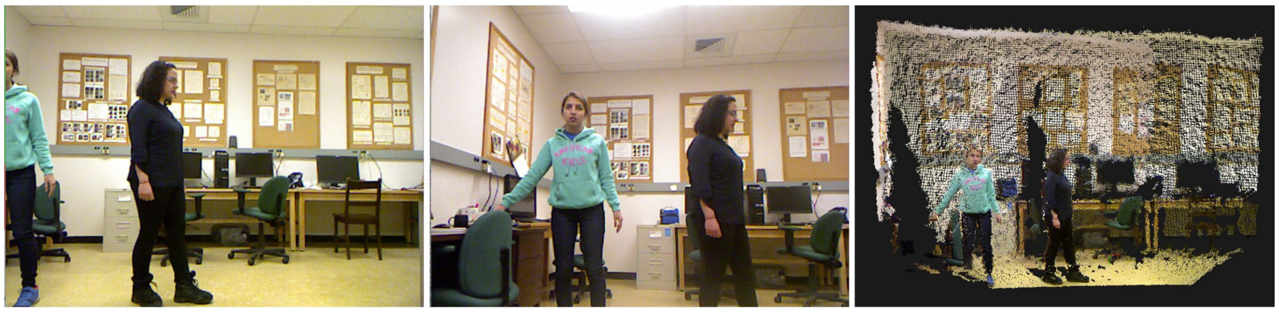

Figure 4: *Example result of two users where half of one user is captured by one camera and the full body is captured by the other.*

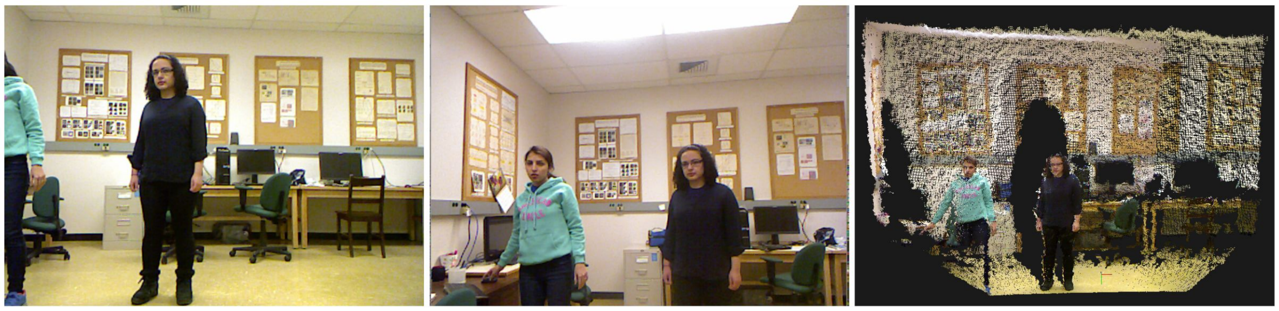

Figure 5: *An example of successful registration of a multi user scene where one camera is located higher than the other one.*

al. also evaluated their results based on the same method. In order to compare our method to the ground truth, for each scene captured we first calibrated the Kinects and then compared the calibration parameters to our results being applied on the same camera pose. As shown in Table 2, the results are very close to those obtained from the calibration method with an average of $0.63cm$ translation and $0.67\ ^\circ$ rotation errors. The results in Table 2 suggest that our algorithm outperforms the work of Miller et al., in terms of translation and rotation error, using the same method to obtain the ground truth. However, it is important to note that the video sequences and configuration parameters are not the same and thus the differences could be explained by other factors outside of our control.

As Table 1 shows, we measured the execution times of different stages of the algorithm in a scenario of point clouds sharing 30% overlap and the execution times for camera movement detection, both included 100 RANSAC iterations to obtain a coarse transformation estimation. Since camera movement detection is implemented in a separate thread, its execution time does not influence the execution time of the dual camera registration. Table 3 shows the speed of the algorithm in different cases, both for camera motion detection and dual camera registration, and the execution time across several iterations of DeReEs-4V until it is not possible to notice any misalignment artefacts, when a seamless registration of the point clouds is achieved.

We also evaluated the alignment improvement over time under varying overlapping conditions of 10%, 25%, 50% and 70% for a scene where one user is standing in front of the camera (Figure 1). The number of features in Figure 6 shows the metric described in Section 3.1.1, which is the number of feature pairs that update the alignment to a better one over time. Each time a higher number is found, the transformation is updated. According to the results, camera arrangements where the overlap is small take longer time to find a good transformation. For instance, for 10% of overlap, it took $120.23ms$ to find a good transformation; However, it only took $60.52ms$ for the cameras sharing 50% overlap to find a good one.

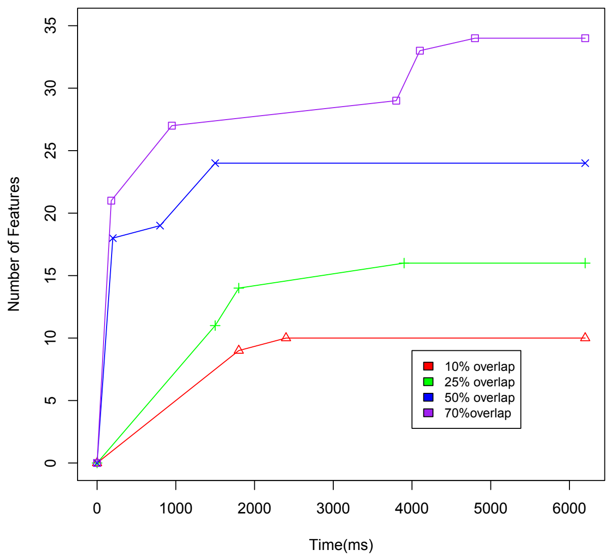

Figure 6: *Performance of the algorithm over time under varying overlapping conditions.*

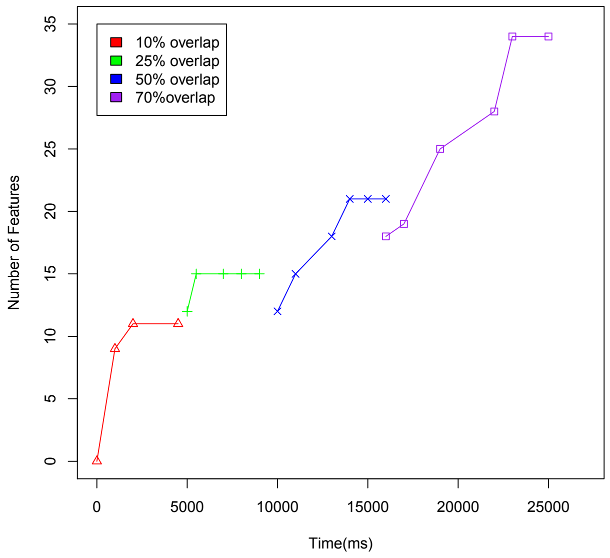

Figure 7: *Performance of the algorithm over time under camera movements.*

Note that plots do not reach to the same point on the number of features. This is because each scene has a different number of matching features to start with. For example, scenes with high overlap have high number of matching pairs, as well as scenes which many distinctive features.

Figure 7 shows the performance of the algorithm under camera movements. The captured scene is shown in Figure 1 in which one user is standing in front of the cameras. Video capturing starts from 10% overlap. As the transformation stabilizes in each step, one of the cameras are moved. The algorithm detects camera movement and the memory is erased; therefore, a drop in the number of features detected is observed. Once the transformation is found again, point clouds are aligned again and continue to improve over time.

We also evaluated the algorithm by comparing the translation and rotation error over time. Errors are obtained by comparing our method to the standard calibration method. For this purpose, three different overlaps of 30%, 50% and 75% are used in our measurements. Figure 8 shows translation error in cm over a period of time. Figure 9 shows the rotation error for the same overlapping conditions over time. Results show that rotation and translation error are kept to very low levels of less than $1cm$ in translation and 1° in rotation error.

A video accessible at `http://www.cs.mun.ca/~omeruvia/research/research.html` shows several usage scenarios. Since this system deals with video streams, the video helps readers observe how the algorithm performs under varying conditions. Four different scenarios are depicted in the video. In the first one, cameras are stable and registration is performed. A user walks in the scene, and registration is updated as the person walks in. The second scenario shows two users being captured by both Kinects and successfully registered. In scenario 3, one person is only visible in one camera, while the other user is being captured by both. The first user walks in from the first camera's viewpoint towards the second camera, and the alignment remains successful. Finally, in the fourth scenario, one of the

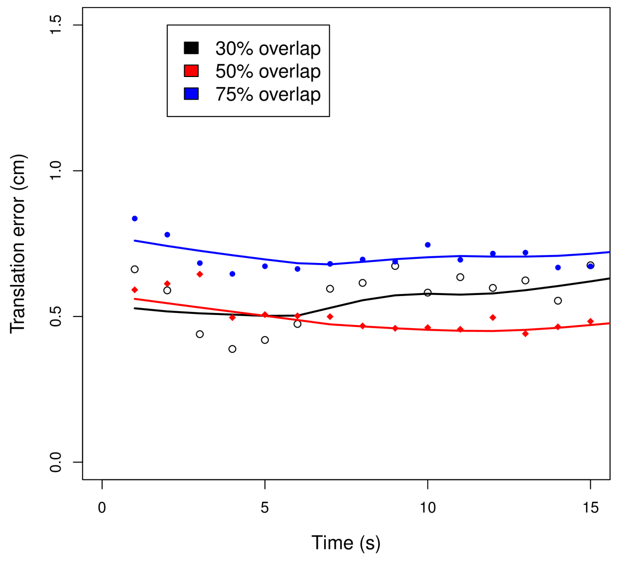

Figure 8: *Translation error over time for three different overlaps.*

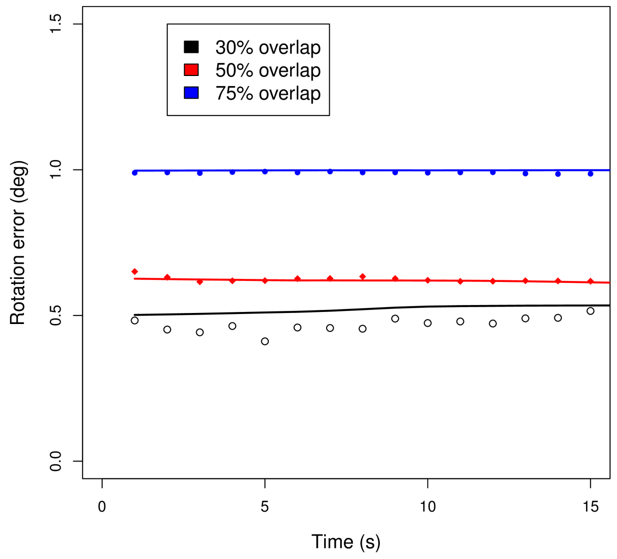

Figure 9: *Rotation error over time for three different overlaps.*

cameras is moved and the result shows a successful registration after the camera is moved.

5 Conclusions And Future Work

We have presented an automatic and adaptable registration system using two Kinect cameras to capture and combine two RGBD video streams providing a wider field of view. Our multi-camera system allows for arbitrary camera placements and does not require camera calibration. Our system successfully aligns point clouds on the fly and is robust to camera movements while capturing point clouds and visualizing them, automatically finding a good approximate registration transformation in little less than 100 milliseconds and a high-quality registration transformation in a few seconds. Moreover, the cameras need not be placed at the same height, they can be placed at different heights and perform registration successfully.

Many potential extensions to this work and applications can be envisioned, for example, we are exploring active people tracking over the extended field of view of both sensors and other algorithms that can make use of the registered point cloud as a whole. To improve the aspect of the rendering quality, hole- filling and point-rendering algorithms can be used to present a smoother display of the environment. Kinect cameras are suitable for capturing objects further than $50cm$, but the system proposed here could be used to register video from depth sensing cameras with smaller depth-sensing range, such as Intels Creative or it could be used to facilitate the combination of different types of cameras. This work could also be extended by using more than two Kinect cameras at each end. In terms of applications, more point cloud processing algorithms could be used on the resulting point cloud produced after registration, and the system could be used to support more users fitting inside the wider field of view, or can provide a wider range of motion for a single user, which could be of benefit for motion capture and gaming, robot and computer vision, simulation, training, and health-related applications.

Acknowledgements

The authors of this work wish to acknowledge the assistance of Dr. Lourdes Peña-Castillo for her support in interpreting and analyzing the data. This work has been funded by a grant from the Office of the Dean of the Faculty of Science, Memorial University of Newfoundland, Canada.

References

Aldoma, A., Marton, Z.-C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R. B., Gedikli, S., and Vincze, M. 2012. Point cloud library. *IEEE Robotics & Automation Magazine 1070*, 9932/12.

Altamimi, R., and Skinner, G. 2012. A survey of active video game literature. *Journal of Computer and Information Technology 1*, 1, 20–35.

Beck, S., Kunert, A., Kulik, A., and Froehlich, B. 2013. Immersive group-to-group telepresence. *Visualization and Computer Graphics, IEEE Transactions on 19*, 4, 616–625.

Besl, P. J., and McKay, N. D. 1992. Method for registration of 3-d shapes. In *Robotics-DL tentative*, International Society for Optics and Photonics, 586–606.

Bouguet, J.-Y. 2004. Camera calibration toolbox for matlab.

Bradski, G. 2000. *Dr. Dobb's Journal of Software Tools*.

BURRUS, N., 2012. Rgbdemo: Demo software to visualize, calibrate and process kinect cameras output. `https://github.com/rgbdemo`. Accessed: February 2015.

CIPPITELLI, E., GASPARRINI, S., SPINSANTE, S., AND GAMBI, E. 2015. Kinect as a tool for gait analysis: Validation of a real-time joint extraction algorithm working in side view. *Sensors 15*, 1, 1417–1434.

CZARNUCH, S., AND PLOUGHMAN, M. 2014. Automated gait analysis in people with multiple sclerosis using two unreferenced depth imaging sensors: Preliminary steps.

FISCHLER, M. A., AND BOLLES, R. C. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6, 381–395.

HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. 2012. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research 31*, 5, 647–663.

KOENIG, S., ARDANZA, A., CORTES, C., DE MAURO, A., AND LANGE, B. 2014. Introduction to low-cost motion-tracking for virtual rehabilitation. In *Emerging Therapies in Neurorehabilitation*. Springer, 287–303.

LASKOWSKI, J. Ultra immersive game using kinect, vuzix glasses and udk. `https://youtube.com/watch?v=ch3TD6u461I`. Accessed: May 2015.

LUN, R., AND ZHAO, W. 2015. A survey of applications and human motion recognition with microsoft kinect. *International Journal of Pattern Recognition and Artificial Intelligence*.

MAIMONE, A., AND FUCHS, H. 2011. Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, IEEE, 137–146.

MAIMONE, A., AND FUCHS, H. 2011. A first look at a telepresence system with room-sized real-time 3d capture and life-sized tracked display wall. *Proceedings of ICAT 2011, to appear*, 4–9.

MILLER, S., TEICHMAN, A., AND THRUN, S. 2013. Unsupervised extrinsic calibration of depth sensors in dynamic scenes. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, 2695–2702.

NEWCOMBE, R. A., DAVISON, A. J., IZADI, S., KOHLI, P., HILLIGES, O., SHOTTON, J., MOLYNEAUX, D., HODGES, S., KIM, D., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, IEEE, 127–136.

NVIDIA. Gaming's future plays out at nvidia's computex demo hall. `http://blogs.nvidia.com/blog/2015/06/02/gaming-future-computex`. Accessed: June 2015.

RAGHURAMAN, S., VENKATRAMAN, K., WANG, Z., WU, J., CLEMENTS, J., LOTFIAN, R., PRABHAKARAN, B., GUO, X., JAFARI, R., AND NAHRSTEDT, K. 2012. Immersive multi-player tennis with microsoft kinect and body sensor networks. In *Proceedings of the 20th ACM international conference on Multimedia*, ACM, 1481–1484.

SEIFI, S., RAFIGHI, A., AND MERUVIA-PASTOR, O. 2014. Derees: Real-time registration of RGBD images using image-based feature detection and robust 3d correspondence estimation and refinement. In *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand, IVCNZ 2014, Hamilton, New Zealand, November 19-21, 2014*, 136.

WEBSTER, D., AND CELIK, O. 2014. Systematic review of kinect applications in elderly care and stroke rehabilitation. *J. Neuroeng. Rehabil 11*, 1, 108.

ZHANG, C., CAI, Q., CHOU, P. A., ZHANG, Z., AND MARTIN-BRUALLA, R. 2013. Viewport: A distributed, immersive teleconferencing system with infrared dot pattern. *MultiMedia, IEEE 20*, 1, 17–27.

ZHANG, D., YAO, Y., LIU, D., CHEN, Y., AND ZANG, D. 2013. Kinect-based 3d video conference system. In *Global High Tech Congress on Electronics (GHTCE), 2013 IEEE*, IEEE, 165–169.

ZHANG, Z. 2000. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 11, 1330–1334.

ZOU, Y., CHEN, W., WU, X., AND LIU, Z. 2012. Indoor localization and 3d scene reconstruction for mobile robots using the microsoft kinect sensor. In *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, IEEE, 1182–1187.