World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:14, No:12, 2020

Method for Auto-Calibrate Projector and Color-Depth Systems for Spatial Augmented Reality Applications

R. Estrada, A. Henriquez, R. Becerra, C. Laguna

***Abstract*—**Spatial Augmented Reality is a variation of Augmented Reality where the Head-Mounted Display is not required. This variation of Augmented Reality is useful in cases where the need for a Head-Mounted Display itself is a limitation. To achieve this, Spatial Augmented Reality techniques substitute the technological elements of Augmented Reality; the virtual world is projected onto a physical surface. To create an interactive spatial augmented experience, the application must be aware of the spatial relations that exist between its core elements. In this case, the core elements are referred to as a projection system and an input system, and the process to achieve this spatial awareness is called system calibration. The Spatial Augmented Reality system is considered calibrated if the projected virtual world scale is similar to the real-world scale, meaning that a virtual object will maintain its perceived dimensions when projected to the real world. Also, the input system is calibrated if the application knows the relative position of a point in the projection plane and the RGB-depth sensor origin point. Any kind of projection technology can be used, light-based projectors, close-range projectors, and screens, as long as it complies with the defined constraints; the method was tested on different configurations. The proposed procedure does not rely on a physical marker, minimizing the human intervention on the process. The tests are made using a Kinect V2 as an input sensor and several projection devices. In order to test the method, the constraints defined were applied to a variety of physical configurations; once the method was executed, some variables were obtained to measure the method performance. It was demonstrated that the method obtained can solve different arrangements, giving the user a wide range of setup possibilities.

***Keywords*—**Color depth sensor, human computer interface, interactive surface, spatial augmented reality.

I. Introduction

VIRTUAL, augmented and mixed reality technologies (VR, AR, MR respectively) are revolutionizing the way humans interact with software. Companies, like Samsung, Microsoft, Facebook, Google, Unity among others, are investing in the development of these technologies, in both software and hardware. Currently, the are many applications for VR, AR and MR technologies, in fields like medicine, education, research, architecture, and entertainment [1]-[3]. Every system has its limitations, ranging from wired headsets, the need for external devices for motion tracking, low battery life, low process power, the requirement of complementary devices for interaction with the system itself among others. The need off using a head mounted display (HMD) can be a limitation for applying an AR solution, these prerequisite limits the access to the system and adds an extra step for its use.

Roberto Giordano Estrada Leyva is with the Sidia Instituto de Ciência e Tecnologia, Brazil (e-mail: roberto.leyva@sidia.com).

Spatial augmented reality (SAR) uses other kinds of technologies to provide a different type of AR experience. SAR implementation is convenient for applying AR into solutions where the need of an HDM is a drawback [2]. To achieve this, SAR must substitute all the technological elements of AR, namely a projection device and an input method. For the system to work properly, the input and projection elements must be calibrated, which is the focus of this research. For the input is used an RGB-D capture device, namely, a Kinect v2 [4]. Depth sensors, such as the Kinect, provide depth images that can be converted into a point cloud. In this case, the point cloud is a representation in 3D points of the scene that the sensor is capturing in real-world measures. Using the color to depth correspondence calibration of the sensor, a colored mesh of the environment can be created, this capability is fundamental for the approach. The accuracy of the point cloud depends on the sensor itself, its resolution, depth map capture technology, and calibration.

The solution was implemented using Unity3d as the development environment. Unity is one of the most popular game engines available for both professional and hobbyist game developers [5]. It has multiplatform compiling capabilities and great integration for creating VR, AR and MR applications. For the color image processing, also Open Source Computer Vision (OpenCV) was used [6]. OpenCV is a popular library that implements computer vision algorithms and it has wrappers to several programming languages and platforms, in this case, the OpenCV for Unity plugin from Unity Assets Store was used. This plugin implements functionalities for generating and detecting ArUco markers [7], [8] that are fundamental in the approach.

Similar studies have been made [9], [10], this approach differs in the absence of a physical marker. This allows a solution that is easier to implement, minimizing effort, resources and human intervention in the process.

II. Calibration Method Description

The projector is calibrated if the dimensions of a virtual object match its real-world projected dimensions. In this case, a depth sensor is calibrated when the spatial relation between the sensor point cloud coordinates and a point in the projection plane is known.

A. Method Constrains

The first step is defining a setup for the projector and input sensor. There are some constraints that are basic for designing the configuration, those are:

1. The projection surface must be flat.
2. The depth sensor color-depth calibration must be correct.
3. A part of the projection area must be visible by the input system depth and color streams.
4. The color stream of the input system must be able to detect an ArUco marker in the projection area.
5. The positions of the projection and the input sensor must be static.

The first requirement is established because the reference system used for calibrating the depth sensor and projector relies on a plane. The plane is computed selecting points in the scene point cloud and assessing how well they fit a plane. The points are selected searching an ArUco marker in the color stream of the RGB-D sensor. We also recommend minimizing the distortion in the projection for better calibration.

In the case of the second requirement, the user must check if the calibration of the color stream and the depth stream is correct meaning that the correspondence of a depth map pixel and a color map pixel can be obtained in an accurate manner.

The third requirement refers to the color-depth cameras being able to capture a part, or all of the projection area. If the projection viewed area by the color-depth sensor is not sufficient the calibration algorithm will not converge. Also, in some cases the depth stream is not able to capture some areas of the scene, leaving holes in the depth map (see Fig. 3). In any case, those limitations are out of the scope of this research.

The fourth constraint is related to the successful detection of an ArUco marker in the RGB-D sensor color stream. As part of the algorithm, an ArUco marker is displayed by the projector into a flat surface, and captured by the color stream of the input sensor. The ArUco detection algorithm can fail if the captured image is blurry, the marker area in the image is too small, or the ArUco shape is not well defined, which can be the case if the projector is too bright. To solve these issues, it is recommended to minimize the distance between the sensor and the projection plane and adjust the brightness and focus of the projections system.

The fifth constraint is a basic one, if there is a displacement on the input sensor or the projector sensor the whole system will be un-calibrated.

If the first and second requirements are not met the calibration may not be correct. The third and fourth requirements are related, if they are not met, the calibration algorithm will not converge. It is also suggested to minimize the occlusion in the field of view of the input sensor removing objects that are between the depth sensor and the projection plane.

B. Environment Setup

The following physical configuration is recommended to increase the calibration quality. The depth sensor is placed looking down at the projection plane at an inclination of approximately 30 degrees. It is not necessary to have the input and projector perfectly aligned, as stated earlier, the calibration algorithms take care of finding the relative positions of the input system and the projection plane. The center of the Kinect is approximately 250 cm of the projection plane, the distance between the depth sensor and the wall depends on the capabilities of the depth sensor and the needs of the application. Any distance is valid as long as the requirement 3 and requirement 4 are fulfilled.

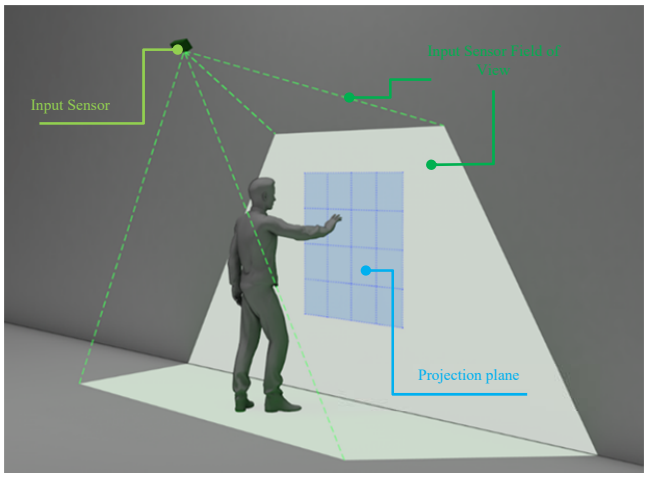
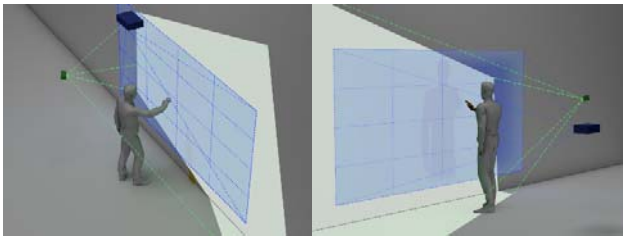


Fig. 1 Screen projection and input sensor system configuration example

The environment setup is represented in Fig. 1. One of the advantages of this proposal is the flexibility in the physical configuration of it, meaning that the depth sensor and projector can be placed in a wide variety of arrangements. Fig. 1 shows the target setup, but other configurations were tested during the investigation progress. This configuration is preferred because it reduces the occlusion and maximizes the projection area captured by the Kinect sensor which will increase the interaction area of the application and is less space invasive when assembled. Figs. 2 (a) and (b) show other examples of useful setup, using close-range and a light base projector respectively.



(a) (b)

Fig. 2 (a) Configuration using near range projection and unaligned input sensor, (b) Configuration using a light base projector and unaligned input sensor

C. Method Description

The method consists of six steps and nine input parameters. These parameters can be adjusted to get a good result depending on the user configuration. The input parameters are:

1. ArUco Marker Id
2. ArUco Marker dictionary
3. Maximum ArUco screen size

4. Minimum ArUco screen size
5. ArUco screen size iteration step decrease
6. Minimum allowed ArUco real size
7. Plane capture iteration count
8. Plane subdivision count
9. Plane fit maximum allowed error

The marker and the dictionary used are given by the user; by default it sets the ArUco id one with the dictionary 4X4_50 from OpenCV. Any other dictionary or ArUco id can be used; a different selection can modify the time taken to detect the marker. Once these are defined OpenCV can generate a texture with the ArUco marker. The marker texture is applied to a quad mesh in Unity. The parameters Maximum ArUco screen size, Minimum ArUco screen size, and ArUco screen size iteration step decrease are used by the algorithm to iterate the markers quad size through the algorithms lifetime. The marker size parameter is referred to as the reason between a side of the marker square and the minimum side of the projection rectangle. Meaning, the ArUco size is a normalized measure, going from zero to one. When the ArUco size is one, the ArUco is displayed as the maximum physical size possible in the projection. The marker size iteration is done from maximum to minimum using the step decrease as the reduction value. This will increase the chance of a better calibration. Since the projection plane dimensions are unknown, the minimum allowed ArUco real size is settled to be a threshold of the permissible size of the ArUco marker in real-world dimensions. The plane capture iteration count defines the number of depth frames used to obtain the candidate marker plane; this parameter is used in the third step of the algorithm. The plane subdivision count defines the number of points taken from the point cloud to generate the plane of the marker. If this parameter is zero, the four corners of the marker quad will be used for computing the plane. Once the input points are obtained we fit a plane to them; this will be discussed later. The point cloud and the fitted plane are compared to determine the difference between them. The plane fit maximum allowed error parameter quantifies the maximum difference allowed between the plane and the point cloud selected, this parameter is measured in meters, for real-world scale.

The process steps are described as follows:

1. Display the ArUco marker at a given position and size in the projection plane
2. Generate an area patch from the ArUco marker corners
3. Obtain a surface point cloud from the area patch
4. Plane fit to surface point cloud
5. Compute virtual camera position and scale
6. Compute input sensor transformation matrix relative to ArUco marker origin

Step one displays the ArUco marker using the projection system in the projection plane, at a given position and random size. This step is the beginning of the search iteration. In every iteration, the marker is positioned in a grid point. The grid is generated using the projection plane dimensions and the current marker size. The position's grid describes different points where the marker is displayed completely in an ordained fashion. The algorithm will require to repeat this step if the calibration cannot be achieved. On each call, this step will return a different grid point, if all the points on the grid are used, this step will decrease the marker size and generate a new grid. The algorithm will fail if all points for al marker sizes are tested and not converged.

On step two, the color stream of the depth sensor is used to record the projected image and search for the ArUco marker on it. If the ArUco marker is not found, the iteration breaks and starts again in step one. Once the ArUco marker is found OpenCV provides the pixel coordinates of the marker corners. The marker corner is used to generate a Coons patch [11]. The patch depends on the plane subdivision count, the amount of points in the patch equals to the plane subdivision count plus two squared. At this point, the patch generates a uniform 2D point array inside the area of the marker, includes the marker corners detected by OpenCV.

Step three uses the RGB to depth correspondence transformation from the input sensor to know the depth values of the 2D points obtained in step two. In some cases, the depth sensor will not be able to compute the depth of a region of the image, see Fig. 3. If the depth of any patch point cannot be found the algorithms return to step one.

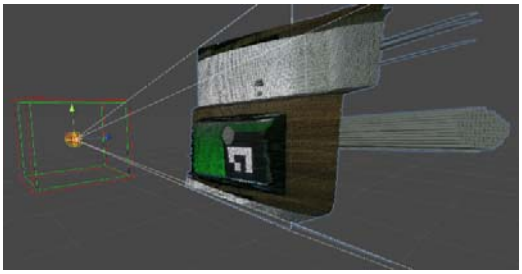


Fig. 3 Kinect scene reconstruction with a hole in the projection plane

Since the Kinect and other depth sensors have an error margin in capturing the depth map, we determinate the final depth of a point as the average of several depths captures for that point. As said before, the plane capture iteration count input parameter defines the amount of frames depth frames used in this operation. The Kinect sensor can make around 30 depth captures by second.

Once the depth values of the patch points are known, their relative position to the sensor origin in real-world measures can be reconstructed. Via least square error method [12], a plane is fitted to the resulting point cloud. In this case, the z-normal of the solution is constrained to one since it is known that the points are arranged in a plane in front of the sensor. The distance between each point of the patch point cloud and the closest point in the fitted plane must be less than the plane fit maximum allowed error input parameter. If that condition is not met, the algorithm breaks and jumps to step one.

The first requirement stated is the projection surface to be flat, from that assumption the algorithm searches for a plane. But depending on the input sensor capabilities the captured plane could not be a plane at all, in the case of the Kinect V2, the depth stream is affected by reflectance and light-emissive

surface, like a led based display. Figs. 4 (a) and (b) illustrate examples of this case. It is recommended displaying the scene mesh to ensure that the captured projection plane is, in fact, a plane surface or at least some area of it is plane. The algorithm will search for this plane area and used it as a calibration reference for the next steps.

(a) (b)

Fig. 4 (a) and (b) Kinect V2 point cloud reconstruction errors of flat led screen

The plane fit maximum allowed error parameter is used as a real-world measure of how non-flat the captured points are allowed it to be. Also we notice that it is more likely to find a wrong solution if the marker area is small. Fixing the z normal coordinate of the plane fit method works as a configuration regulator since it is most likely to obtain better results if the input system is perpendicular to the projection plane.

In step five, the virtual camera scale is calibrated; this means that a virtual object will maintain its dimensions in the real-world through the projection plane. On finishing this step, a virtual sphere of 50 cm radius will be displayed on that same screen size, independent of the projection plane dimensions. For this research is used a virtual orthogonal camera; this kind of projection does not distort the perceived size of an object upon distance. In unity, orthogonal cameras use an orthographic size to regulate the frustum. By changing the orthographic size of the camera the perceived size of an object in the projection is changed.

From the patch plane points, the dimensions of the marker in the real world can be computed as the average of the patch side distance. A scene scale ratio (SSR) is computed using the relation between the marker real size and the marker virtual size. An orthogonal camera was used in the previous steps to display the marker prefab in the projection plane. The target display camera orthographic size (TOS) equals to the marker display camera orthographic size (MOS) multiplied by SSR, see (1). The target display camera position is constrained to the (0, 0, -1) position and oriented to the virtual world origin of coordinates with the up vector of (0, 1, 0).

$$TOS = MOS * SSR. \quad (1)$$

At the final step, the input sensor and the projection plane relative positions and rotations are solved. A reference Cartesian origin is created from the fitted plane, as shown in Fig. 5. The origin of the reference point is the points cloud center. The depth edge of the reference origin is the normal of the fitted plane. The vertical orientation vector is computed from the marker corners of vertical vectors.

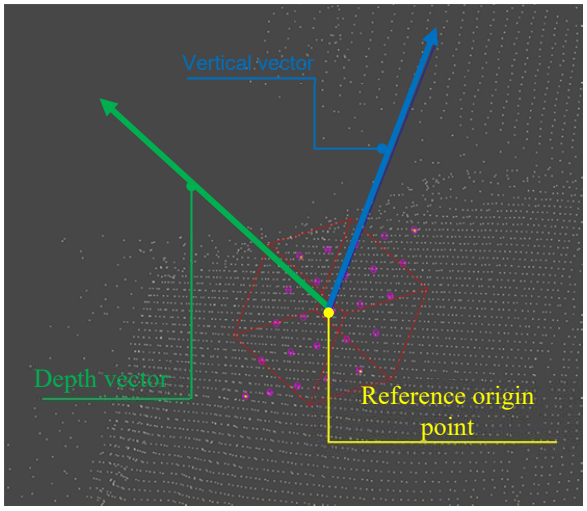


Fig. 5 Reference origin point, depth vector in green, blue vertical vector

A hierarchy is created to facilitate the position and rotation extraction. This hierarchy is composed of two elements: A parent coordinate system (P) will represent the marker position and orientation in the virtual scene and a child coordinate system (C) will represent the sensor position and orientation in the virtual scene. The P coordinate system is placed at the Cartesian reference origin position. Then, P is transformed to gaze the depth vector of the Cartesian edge and the vertical orientation vector is used as the up world vector of the method. Next, C is placed in the zero vector position and zero quaternion rotation in global coordinates. At this point, the transformations are as displayed in Fig. 6.

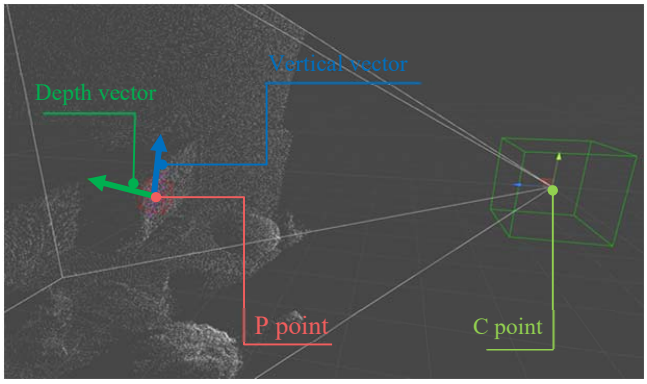


Fig. 6 Marker point (P), input sensor point (C) and scene point cloud representation. Marker to sensor relative position and relative orientation solved

The red cube represents the position and orientation of P, the marker in the point cloud. Each point cloud position is relative to the input sensor, in this case, the sensor is located in the virtual world origin, C position and rotation is represented by the green box. Later, the parent is placed and rotated as the relative position of the marker in the display camera. C will maintain its relative position and rotation to the P after the last one is transformed. As the display camera was fixed in (0, 0, -

1) and its orientation to look at (0, 0, 0) with an up vector of (0, 1, 0) the marker relative position to this camera, marker scaled position (MSP) is computed in the following way:

$$MSP = MDP * SSC. \quad (2)$$

where MDP is the marker relative position to the render marker camera and SSC is the scene scale ratio computed in the previous step.

III. Algorithm Test Description and Analysis

Three tests were made to test the method. Two of them use a near-range laser projector and the other uses a 50'' OLED display. All captures were made using the Kinect sensor version 2. The first two tests use a configuration similar to the one described in Fig. 1. On each test iteration, the distance of the sensor to the display was increased by approximately 500 cm. For the first test, a near-range laser projector was used, obtaining samples from 1.5 m to 4 m.

The second test was similar to the first, changing the display to a 50 inches OLED display. In this case, the algorithm could converge only in distances between one meter and 2.5 meters. The third test was made using the near-range laser projector in a random Kinect position-rotation configuration. The calibration configuration parameters for all tests were as described in the Table I.

Result variables captured on each test iteration are:

- Sensor distance to display Surface
- Plane fit determinant
- Plane fit error average
- Converge marker size
- Sensor Euler angles

Plane fit determinant and plane fit error is the value of this parameter at the end of the execution. Converge marker size is the size in real-world scale of the marker side when the calibration converged. The Sensor Euler angles contain the global rotation of the sensor in Euler angles, by axis, after the calibration is made. Furthermore, the projections plane is calibrated and the re-projection of the user can be seen matching the real world, meaning that the calibration was correctly done.

A. Results Discussion

In Fig. 7, there is an example of a calibration made on the experiment. The position of the Kinect sensor and the projection plane in the real-world and the virtual-world solved result can be seen.

Table II shows the results of the first test, using the near-range laser projection and increasing the distance between the sensor and the projection surface as described before. Notice that there is a minimum change on the Euler angles parameters as expected for this experiment.

It is important to analyze the correlation between the result variables. As stated before, the plane fit determinant and plane fit error average are indicators of the quality of the calibration. The ground truth is unknown, so it is not possible to compare the results obtained by the calibration.

TABLE I
Calibration Parameters of Tests

Parameters	Value
ArUco Marker Id	5
ArUco Marker dictionary	DICT_4X4_50
Maximum ArUco screen size	0.6
Minimum ArUco screen size	0.1
ArUco screen size iteration step decrease	0.05
Minimum allowed ArUco real size	0.02
Plane capture iteration count	10
Plane subdivision count	3
Plane fit maximum allowed error	0.005

The greater linear correlation found in this test was between the plane fit determinant variable and the converge marker size with a value of 0.928 [12]. The other correlation worth mentioning is between the sensor distance and the error average, indicating that the error will increase as the distance of the sensor to the display plane increases. This behavior matches the Kinect sensor specification, meaning that the error on the depth map increases with the distance to the surface. The correlation between the plane fit determinant and the sensor distance was also high with a value of 0.62 supporting the distance importance to the calibration process.

There was a low correlation between the plane fit error and the plane fit determinant and also between the plane fit error and the converge marker size indicating that for this test the error did no change much, in general the calibrations were good. Fig. 7 shows the result parameters of the first test, every parameter in this chart is normalized so the correlation between them can be seen more easily.

The results for the second experiment are displayed on Table III. In this case, the Kinect Depth Map is affected by the change on the display in a similar fashion as the artifacts shown in Figs. 3 and 4. Also the display area is smaller. These conditions affected the results of tests; the system was able to converge on a maximum of 2.5 m. Fig. 8 shows the normalized variation of the results on the second test. In this case there is also a strong positive correlation between the sensor distance result and the plane fit determinant. We see an increase on the positive correlation between the plane fit error average variable and the plane fit determinant variable due to the Kinect depth map error being more affected by the elements' positions. Also the error average increased with the marker size. Likewise, it is interesting that there is a strong negative correlation between the variable sensor distance and the variables plane fit determinant, plane fit average error and converge marker size with values of -0.86, -0.76 and -0.88, respectably. Oddly in this test, the distance improved the error and fit determinant but the real marker size decreased on each iteration. The deformation displayed on Fig. 4 gives an indicator of this problem, planes can be found in smaller part of the display surface, the minimum allowed ArUco real size configuration parameter is designed to mitigate this situation.

TABLE II
RESULTS OF THE FIRST CONFIGURATION TEST

Iteration	Sensor Distance	Plane fit determinant	Plane fit error average	Converge marker size	Euler Angles X	Euler Angles Y	Euler Angles Z
1	1040	0.10437	0.00075	0.32795	16.51034	179.89840	356.44040
2	1496	0.97690	0.00092	0.57759	17.73703	181.68660	357.46080
3	2008	1.02351	0.00073	0.58424	17.62832	181.98830	356.89700
4	2516	1.70056	0.00117	0.66339	17.24300	187.28320	357.30370
5	3002	2.78366	0.00061	0.74705	14.59960	170.54790	347.13880
6	3505	3.02110	0.00185	0.75156	0.810952	170.46880	353.73780
7	4046	1.05997	0.00173	0.57284	0.927804	183.93000	354.11470

TABLE III
CORRELATION OF NEAR-RANGE LASER PROJECTION TEST RESULTS

Linear Correlations	Plane fit determinant	Plane fit error average	Converge marker size	Sensor Distance	Euler Angles X	Euler Angles Y
Plane fit determinant	-	-	-	-	-	-
Plane fit error average	0.58536	-	-	-	-	-
Converge marker size	0.91082	0.59918	-	-	-	-
Sensor Distance	0.40760	0.66044	0.42089	-	-	-
Euler Angles X	0.04303	-0.04852	-0.00490	-0.26472	-	-
Euler Angles Y	-0.36043	-0.00995	-0.40920	0.20916	-0.15598	-
Euler Angles Z	0.40760	-0.03113	0.29801	-0.16697	-0.27894	-0.11541

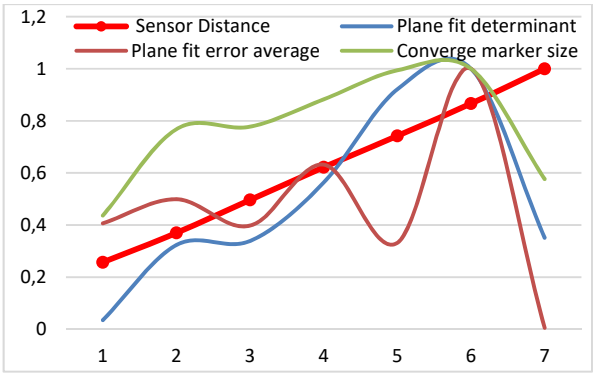


Fig. 7 Normalized results variables of the first test

From the results of the second test, it can be concluded that the method can be applied to OLED displays, but the deformations occasioned in the depth map need to be taken into count. Also the configuration parameters need to be more demanding for obtaining better results.

From the observation made on the first and second experiment, a third test was designed. On this case the near-range laser projector was used, since it does not affect the Kinect sensor input. The physical configuration of this test is random, exploring different distances and orientations of the sensor. The distance range was from 700 cm to 3700 cm, and 18 samples were taken.

Fig. 8 shows the behavior of the variables on the third test, the values are normalized so it is easier to see the correlations. Fig. 8 also includes the Euler angles behavior.

For a more accurate computation of the correlation, the results of the first test and second test area merged into Table III. Once again a strong positive correlation between plane fit determinant and the converged market size is displayed. These results confirm the direct relation existing among the sensor distance and the plane fit determinant, plane fit error average and converge marker size variables. The trade-off on the distance is that the user will lose interaction area if the sensor is closer to the display surface, but will lose accuracy on the capture and calibration the further the sensor is from the display plane.

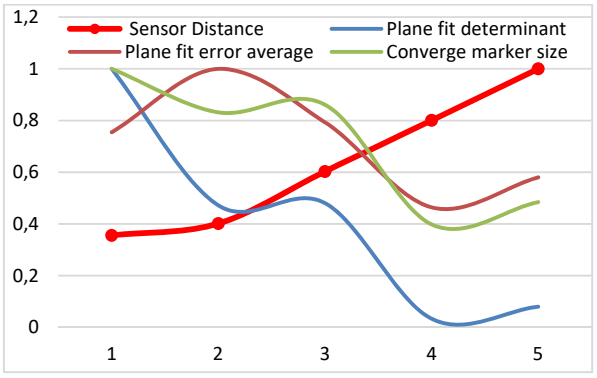


Fig. 8 Normalized results variables of second test

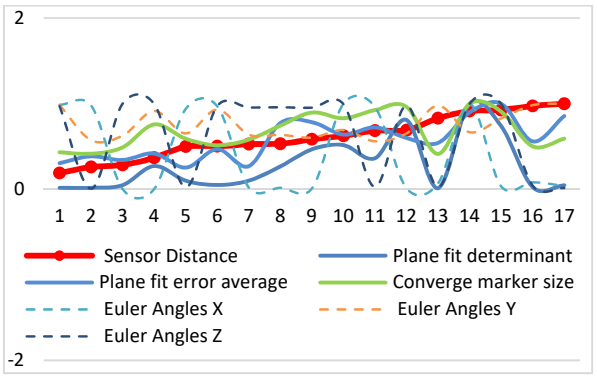
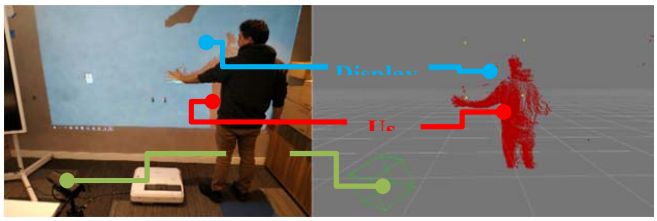


Fig. 9 Obtained variables of third test



(a) (b)

Fig. 10 SAR system calibrated, example from third test

IV. Conclusions

A solution was to design an easy setup and integrate a SAR experience. The implementation was made using Unity 3D, one of the most popular virtual reality and augmented reality implementation tools and OpenCV, one of the most popular computer vision libraries. The solution does not require any marker and it is oriented so the implementer has as little intervention on the process as possible. The method proposed is flexible to changes in physical configurations. The implementer can modify the input parameters of the process to obtain better results or to adapt to its physical limitations. Tests were made using different display technologies identifying artifacts that may occur during the implementation. An ideal physical configuration was proposed; this configuration is the result of the correlation analysis of the output results of the process. Figs. 7-9 show the correlation between the obtained variables on the third experiment. As said before, this experiment was designed with random physical configurations. In all variants the algorithm converged.

Finally, Fig. 10 is an example of one of the configurations in the third experiment. The real-world elements are shown in Fig. 10 (a) and their respective solved 3d-virtual world positions and rotations on Fig. 10 (b). The real-world part of this image has the user point cloud projected, showing the correct calibration of all the elements of the system.

Acknowledgment

The author thanks the people at SIDIA for allowing the time and resources.

References

[1] M. K. Bekele, R. Pierdicca, E. Frontoni, E. S. Malinverni, y J. Gain, «A Survey of Augmented, Virtual, and Mixed Reality for Cultural Heritage», J Comput Cult Herit, vol. 11, n.o 2, p. 7:1–7:36, mar. 2018, doi: 10.1145/3145534.

[2] O. Bimber y R. Raskar, Spatial augmented reality: merging real and virtual worlds. AK Peters/CRC Press, 2005.

[3] P. Chen, X. Liu, W. Cheng, y R. Huang, «A review of using Augmented Reality in Education from 2011 to 2016», en Innovations in smart learning, Springer, 2017, pp. 13–18.

[4] H. Gonzalez-Jorge et al., «Metrological comparison between Kinect I and Kinect II sensors», Measurement, vol. 70, pp. 21–26, 2015.

[5] T. Shareef, «5 best cross-platform game engines for rising game developers».

[6] G. B. García, O. D. Suarez, J. L. E. Aranda, J. S. Tercero, I. S. Gracia, y N. V. Enano, Learning image processing with opencv. Packt Publishing Ltd, 2015.

[7] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, y R. Medina-Carnicer, «Generation of fiducial marker dictionaries using mixed integer linear programming», Pattern Recognit., vol. 51, pp. 481–491, 2016.

[8] F. J. Romero-Ramirez, R. Muñoz-Salinas, y R. Medina-Carnicer, «Speeded up detection of squared fiducial markers», Image Vis. Comput., vol. 76, pp. 38–47, 2018.

[9] S. Voida, M. Podlaseck, R. Kjeldsen, y C. Pinhanez, «A study on the manipulation of 2D objects in a projector/camera-based augmented reality environment», en Proceedings of the SIGCHI conference on Human factors in computing systems, 2005, pp. 611–620.

[10] T. Motta, M. Loaiza, L. Soares, y A. Raposo, «Projection Mapping for a Kinect-Projector System», en 2014 XVI Symposium on Virtual and Augmented Reality, 2014, pp. 200–209.

[11] S. A. Coons, «Surface patches and B-spline curves», en Computer Aided Geometric Design, Elsevier, 1974, pp. 1–16.

[12] «Best Fitting Plane given a Set of Points». https://math.stackexchange.com/questions/99299/best-fitting-plane-given-a-set-of-points.