2021

A SOFTWARE TOOL FOR USING AN AUGMENTED REALITY SANDBOX TO CALCULATE VOLUME CHANGE

Dante Paglia
Michigan Technological University, dfpaglia@mtu.edu

A SOFTWARE TOOL FOR USING AN AUGMENTED REALITY SANDBOX TO CALCULATE VOLUME CHANGE

By

Dante F. Paglia

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2021

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Computer Science

Report Co-Advisor:	*Dr. James Walker*
Report Co-Advisor:	*Dr. Paul van Susante*
Committee Member:	*Dr. Keith Vertanen*
Department Chair:	*Dr. Linda Ott*

Table of Contents

List of Figures iv

List of Tables v

Acknowledgements vi

Abstract vii

1 Project Introduction and Goals 1

2 Preliminary Work 3
- 2.1 Building the AR Sandbox and Selecting Hardware 3
- 2.2 Calibration of the AR Sandbox 4
- 2.3 Running the AR Sandbox 5

3 Methods 8
- 3.1 Development of the Volume Tool 8
- 3.2 Depth Image Collection 9
- 3.3 Volume Calculation Design 10

4 Initial Results 12

5 Error in Volume Calculation 14
- 5.1 Reducing Inconsistencies with Multiple Collections 14
- 5.2 Error Propagation 14
- 5.3 Error in the x and y Dimensions Based on z 16
  - 5.3.1 Mitigation of Error in the x and y Dimensions 17
  - 5.3.2 Adjusted Volume Calculations 19

6 Conclusion 21

7 Future Work 22

8 References 23

List of Figures

Figure 1. AR sandbox with water simulation running 1
Figure 2. Projector and Kinect mounted above the sandbox 4
Figure 3. Calibration disk aligned with the projected cross for tie point collection 5
Figure 4. Initial output of the SARndbox application 6
Figure 5. Screenshot of running AR sandbox application 7
Figure 6. Projection of the AR sandbox application onto the sandbox surface 7
Figure 7. Raw Kinect Viewer application 9
Figure 8. Flow diagram of depth image collection 10
Figure 9. Diagram of approximated rectangular prism calculated by the volume tool 11
Figure 10. Diagram of x and y dimension inconsistencies 17
Figure 11. Depth over percentage of error in the x dimension 18
Figure 12. Depth over percentage of error in the y dimension 18

List of Tables

Table 1. Average Volume Calculations of Different Objects Over 10 Calculations.....12
Table 2. Percent Error of Volume Calculations of Different Objects.....13
Table 3. Average Difference Between Before and After Depth Images 14
Table 4. Error Propagation Calculated for Each Volume Calculation.....16
Table 5. Average x and y Dimensions at a Given z.....17
Table 6. Average Volume Calculation over 10 Calculations 19
Table 7. Percentage of Error of Volume Calculations.....20

Acknowledgements

I would like to thank my advisors Dr. Walker of the Department of Computer Science and Dr. van Susante of the Department of Mechanical Engineering-Engineering Mechanics. Their support and expertise were invaluable resources throughout my research and assisted me in working through any challenge I faced.

In addition, I would like to thank all the members of the Planetary Surface Technology Development Laboratory (PSTDL). Specifically, Ben Wiegand who assisted by drawing the CAD model and building of the projector/Kinect mount as well as the Mining Innovation Enterprise (MINE) Senior Capstone Trencher team for designing and building the sandbox. Everyone in the lab was extremely helpful and always willing to spend some of their precious time lending a hand when needed.

Finally, I would like to thank all my family and friends for their support over the years. None of what I have accomplished would have been possible without you.

Abstract

The Augmented Reality (AR) Sandbox application and software frameworks provide an interactive tool for freshwater and watershed education. This is done by projecting the real-time topography of the sandbox surface and simulating waterflow. This tool has the potential to assist in analyzing excavation tools that will be used on lunar or Martian surfaces. This project extended the software to include a volume tool capable of calculating the change in volume of the sandbox after manipulation.

This report details the setup and calibration of the sandbox as well as the development of the volume calculation tool. It outlines the methods used to calculate the volume changed based on two depth image collections. This is followed by exploring the results of multiple volume calculations against their expected values and investigating the amount of error that occurs.

1 Project Introduction and Goals

The Augmented Reality (AR) Sandbox is part of an NSF-funded project for the education of freshwater and watershed science. It is developed by the UC Davis' W.M. Keck Center for Active Visualization in the Earth Sciences (KeckCAVES), together with the UC Davis Tahoe Environmental Research Center, Lawrence Hall of Science, and ECHO Lake Aquarium and Science Center. The AR Sandbox software consists of the Vrui Virtual Reality (VR) development toolkit, Kinect 3D video processing framework, and the SARndbox software package, all of which were developed at UC Davis. The three software packages are available for use under the GNU General Public License. The AR Sandbox provides a real-time topographical model of a sand surface using a Microsoft Kinect sensor and a projector. The projection shows the topography of the sandbox surface with contour lines and colors representing change in elevation as well as simulated water to depict how it flows over the surface, seen in Figure 1.

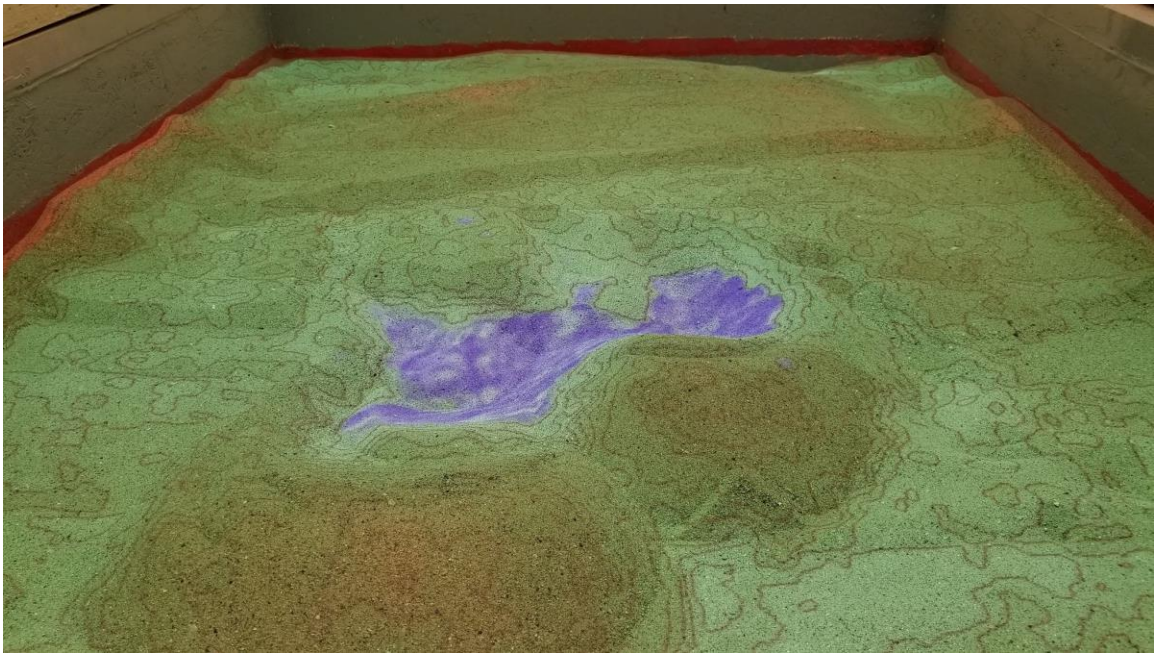

Figure 1. AR sandbox with water simulation running

The purpose of the AR Sandbox is to provide a hands-on experience for learning about freshwater and watershed environments, but it has the potential to provide many other scientific applications. It can also assist in analyzing the performance of various excavation tools on a small-scale level. This aligns with the goals of the Planetary Surface Technology Development Laboratory (PSTDL) at Michigan Technological University, to build and test technology for lunar and Mars missions. The features of the AR sandbox can help to analyze excavation tools that could be used on the lunar or Martian surface. The real-time visualizations of the AR sandbox give the user a general idea of how the surface of the sandbox has changed after an excavation test. However, actual metrics need to be collected to gain a better understanding of the excavation's

performance. The focus of this project is to use the frameworks provided with the AR sandbox application to calculate the change in volume of the sandbox surface.

The Kinect sensor used to determine the position and depth of the sandbox surface can also be used to calculate the change in volume. Other researchers have developed software to detect the volume in 3D space, but they rely on having an object in the 3D space and calculating its volume, while it is present, from various angles. For example, taking depth images from 4 different angles of household objects with a Kinect, creating a point cloud and utilizing that to compute volume (Dellen, Rojas, & Andres, 2013). Another technique outlined in the paper *VOLUMNECT – Measuring Volumes with Kinect* is using the point cloud from a depth image to create planes in which the x, y, and z dimensions are determined for the volume calculation (Ferreira, Grine, Gameiro, Costeira, & Santos, 2014). However, these techniques differ from the goals of this project.

The main goal of this project is to develop a software tool within the AR sandbox frameworks that can calculate the volume change that takes place within the sandbox. This will provide a valuable tool to be used with current and future projects of the PSTDL. Other project teams can utilize the sandbox and volume tool to collect data on the performance of excavation tools or other hardware.

Another goal of the project is to allow the user to select the area of the sandbox over which the change in volume is calculated. If the volume is always calculated over the entire sandbox and no sand is removed, the overall change in volume would be zero. Having the capability to select an area of the sandbox allows the user to select a section based on where their excavation will take place and where the removed sand will end up. Additionally, the volume reported should outline the volume added, removed and overall net volume.

In order to accomplish the above goals, a sandbox needed to be built and the AR sandbox application needed to be set up and calibrated with in the PSTDL.

2 Preliminary Work

Before the volume tool could be developed and tested, the AR sandbox was built and calibrated. Detailed instructions for setting up an AR Sandbox, installing the software packages and calibrating the Kinect with the projector can be found on the UC Davis team's website (Build your own AR Sandbox, 2016).

2.1 Building the AR Sandbox and Selecting Hardware

Through cooperation with the Senior Capstone Trencher team, the size of the sandbox was decided to be 60 inches by 45 inches. The AR sandbox software requires it to be in a 4:3 aspect ratio to match the intrinsic aspect ratio of the Kinect depth image. The final size was determined to ensure it was large and deep enough to benefit the Trencher team's project. The sandbox was then filled to 12 inches deep with play sand.

Due to the graphically intensive real-time visualization of the AR sandbox application, a new computer was purchased with the recommended NVIDIA GeForce graphics card. Additionally, the AR sandbox frameworks were built and tested using Linux Mint 19 which was installed on the new computer to ensure a smooth setup.

A new projector was also purchased with a 4:3 aspect ratio to match the surface size of the sandbox and Kinect depth image. The projector also required a short enough throw distance, or the distance between the projector lens and screen it is projecting on, to fit within the designated area in the lab. Both the Kinect-for-Xbox-360 (first generation Kinect camera) and the Kinect V2 for Xbox One are compatible with the AR sandbox and were available from members of PSTDL. The Kinect V2 was originally chosen because it was the newer hardware option. Next, a mount for the projector and Kinect needed to be designed and built. Per the recommendations in the UC Davis instructions, the Kinect needed to be mounted in the center of the sandbox so that the distance between the Kinect and the sand surface was approximately equal to the width of the sandbox. Figure 2 depicts the built sandbox and assembled projector/camera mount before the Kinect was attached.

Figure 2. Projector and Kinect mounted above the sandbox

2.2 Calibration of the AR Sandbox

After the projector and Kinect were mounted, the calibration between the projector display and the camera was performed. This allows for the AR Sandbox application (SARndbox) to scale the Kinect's depth image to fill the projection on the sandbox surface and accurately display the topography. Per the UC Davis instructions, a calibration disk was made by cutting out and attaching a sheet of paper to a compact disc (CD) then drawing a cross that intersects at the center of the CD. The calibration disk was then attached to the end of an allen wrench to avoid interference by holding it be hand.

Calibration was performed by running the command "./bin/CalibrateProjector -s 1024 728" where 1024 and 728 represent the respective width and height of the projector's image in pixels. As Figure 3 shows, this projects a white cross onto the sandbox surface in which the cross on the calibration disk is aligned and a tie point is captured. A tie point is a point that the calibration file uses as a point of reference to align the projection to the depth image of the Kinect. Starting with a flat sandbox surface, tie points are collected at different heights. After approximately half a dozen captures, a hole was dug to the bottom of the sandbox to capture more tie points at a lower level. After about a dozen captures are taken, the projector displays a red cross which tracks the calibration disk while it is

moved around the sandbox. This is used to verify the calibration was performed correctly. If the red cross does not accurately track the calibration disk's movement around the sandbox, more tie points at varying heights can be collected. The collected calibration data was stored into a projector transformation file to be used by the AR sandbox application.

Figure 3. Calibration disk aligned with the projected cross for tie point collection

2.3 Running the AR Sandbox

Upon successful calibration, the sandbox framework can be run in the terminal with the following command: "./bin/SARndbox -uhm -fpv". The -uhm flag enables the elevation color mapping and loads the elevation color map. The -fpv enables the use of the projector transformation file which was created during calibration so that the Kinect camera and projector are aligned. However, the application displayed a mostly black screen with a few colored pixels (Figure 4), instead of the expected colored topographical map of the sand surface. To test if this resulted from an error in the calibration, the command was run without either of the flags, which allows the display image to be manipulated within the application window. Rotating the display image revealed that the Kinect was successfully calibrated. Next the -uhm flag was reenabled, verifying that the depth was also interpreted correctly. However, the image appeared to be inverted along the horizontal axis.

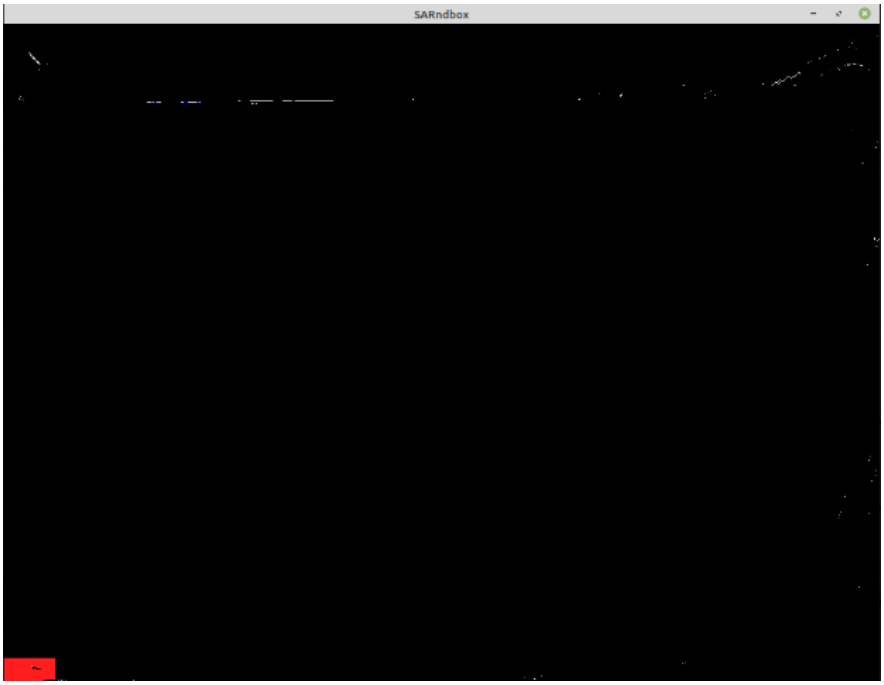

Figure 4. Initial output of the SARndbox application

One of the calibration files, BoxLayout.txt, stores the recorded base plane of the sandbox, which is the depth of the flat sandbox surface, as well as the x, y and z positions of the sandbox corners. It is noted within the UC Davis instructions that the second-generation Kinect may report the depth plane with inverted values for an unknown reason. This was taken into account during the calibration by initially flipping the received values, but in an attempt to fix the inverted depth image, these values were changed back to the camera's default values within BoxLayout.txt. This flipped the depth image shown in the AR sandbox application but caused it to be mirrored. After further discussions on the AR Sandbox forums run by the UC Davis team, it was determined to be an issue solely with the Kinect V2. Since a Kinect-for-Xbox-360 was already available to the lab, a new adapter was purchased, and the cameras were swapped out. After repeating the calibration with the first-generation Kinect, the AR sandbox application was run successfully. Figure 5 shows a screenshot of the application running correctly and Figure 6 shows the working AR sandbox with the topographical lines and coloring being correctly projected onto the surface of the sandbox.

> **Note:** As of March 9th, 2021, the original developer of the AR sandbox software packages updated the software packages to fix the issue experienced above. The Kinect V2 should function properly now, if used with the updated packages, and is available as an option for the PSTDL in the future.

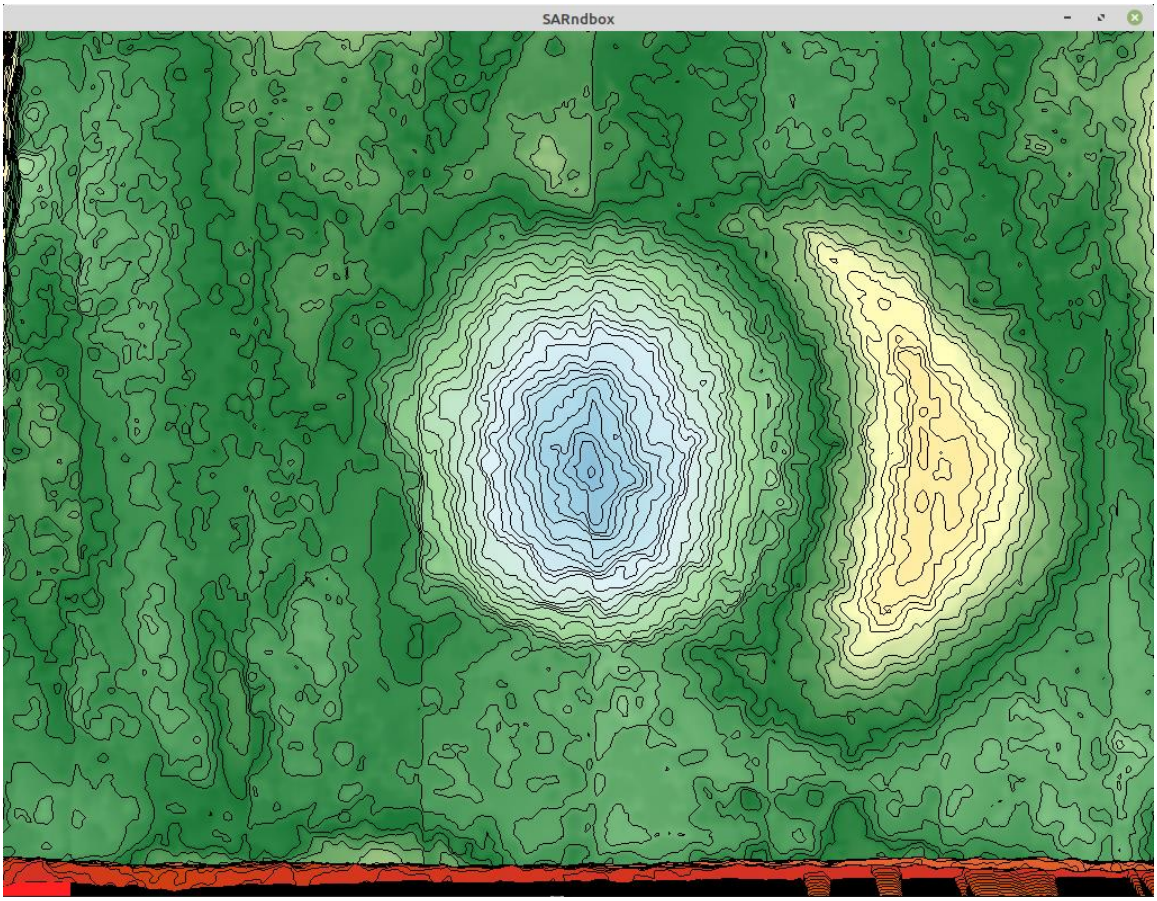

Figure 5. Screenshot of running AR sandbox application

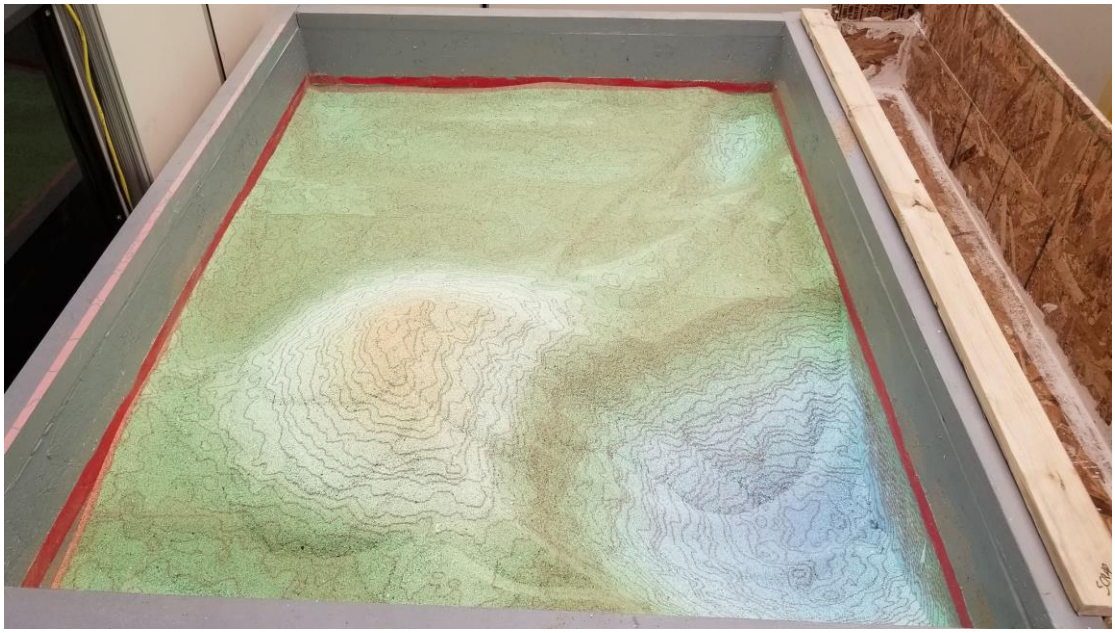
Figure 6. Projection of the AR sandbox application onto the sandbox surface

3 Methods

Once the Kinect was mounted above the sandbox, the development of the volume calculation tool was started. This required an understanding of how the Vrui VR development toolkit, Kinect 3D video processing framework, and the SARndbox software package worked together to create the existing tools and corresponding applications. The Vrui VR development toolkit provides the base visualization functionalities as well as all the tools used in each application the package provides. The Kinect 3D video processing framework provides access to the Kinect depth image and camera feed and functions for utilizing them. The SARndbox software package is where the AR sandbox application is built using the other two packages.

3.1 Development of the Volume Tool

Initially, the volume tool was created as a tool within the SARndbox application. All tools within the packages are built using the Vrui VR development toolkit. The design was modeled off the plane tool (PlaneTool.h and PlaneTool.cpp), which is built into the RawKinectViewer application that is used for calibration. The plane tool maps to two buttons (or keys on a keyboard). By pressing and holding the first key, the user can drag their mouse and draw a box, releasing when the area they wish to select is encompassed. The second key can then be pressed to run the base plane calibration step over that area. This is the basis for the desired functionality of the volume tool since the goal is to select an area of the sandbox and then choose when to calculate volume change over that area after the sandbox surface is manipulated.

Therefore, the volume tool maps to two buttons. The first allows the user to drag and draw a box around the area of the sandbox over which they wish to calculate volume change. The sandbox surface can then be manipulated by an excavation tool. Once complete, the user presses the second button that was mapped, and the volume change is calculated over the same area that was originally selected.

The user-selected box is measured by collecting the initial (x, y) point that the box is drawn from and the final (x, y) point at the corner where it is released. The dimensions of the box are then calculated by subtracting the floored values of the two points' respective x and y coordinates and taking their absolute value. The floor function is used get the sizes as integers and the absolute value ensures the values are positive since they are used later to create an array of that size.

After building and compiling the volume tool within the SARndbox application, it was discovered that functions needed within the RawKinectViewer application were not easily available outside of it. Therefore, in order to gather the depth image data from the camera over the selected area and convert them to x, y, and z values in centimeters, the tool was moved to run within the RawKinectViewer application (Figure 7) instead of the SARndbox application.

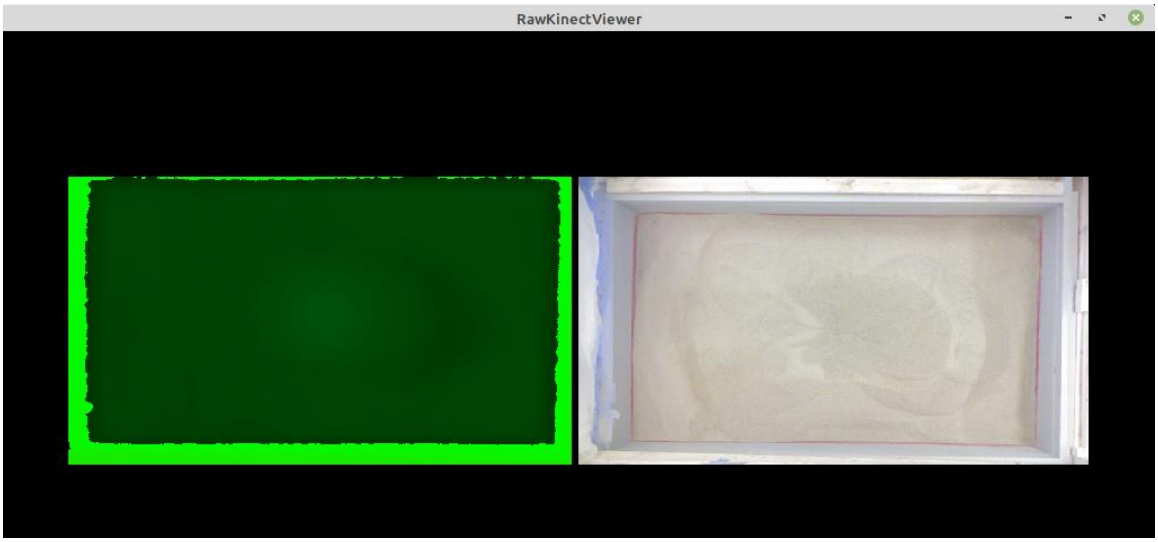

Figure 7. Raw Kinect Viewer application

3.2 Depth Image Collection

Once the size of the user selected box is calculated, a 3D array is initialized to hold the x, y, and z values at each point within the box. Each corresponding x and y position of the 3D array is set to its respective x, y and z values in centimeter form. This is done by collecting the point and feeding it into a function to convert the point to its equivalent (x, y) depth image pixel and z depth image value. The depth image point is then put through another transform function to convert it to its position in the real world with respect to the camera. These final values are reported in centimeters with the center of the sandbox being (x, y) = (0, 0) and the depth values as negative distance away from the lens. Both of the conversion functions already existed within the Kinect 3D video processing framework. The depth conversion was verified by ensuring the depth value provided for the surface of the flat sandbox matched its actual distance of ~165 centimeters.

When the user clicks the second button to indicate they are ready for volume to be calculated over the previously selected area, the same process is performed. A new array is created with the updated depth image values based on how the surface of the sandbox changed. From there, these two depth value arrays are used to calculate the change in volume. Figure 8 provides a flow diagram of the depth image collection.

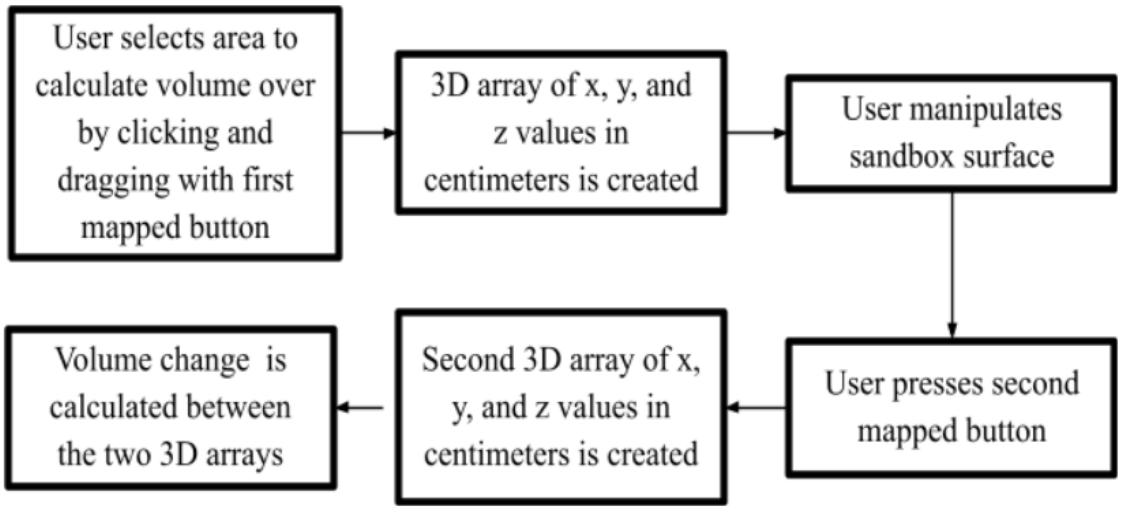

Figure 8. Flow diagram of depth image collection

3.3 Volume Calculation Design

The change depth can be observed by checking if the z dimension of a given point has a different value, either higher or lower, between the two depth arrays. In order to determine the volume change from this, the change in z must be observed over a change in the x and y dimension as well since volume at its most basic form is x times y times z.

To do this with the two depth arrays, four initial/before points from the first array and their respective final/after points from the second array are observed together as a prism. The volume of the prism is calculated for every set of 4 points in the arrays. The volume is summed together and the amount of volume gained, volume remove, and net volume are reported. However, due to intrinsic error and inconsistencies in the Kinect sensor, discussed further in Section 5, and the potential for a very uneven surface in the sandbox, the 8 points form an uneven quadrilateral prism (Figure 9-A).

To account for this, the volume of an approximated rectangular prism is calculated instead. The x and y dimensions are calculated by taking the average of the two before x/y dimensions and the two after x/y dimensions (Equations 1 and 2). The before values represent the x and y values from the first 3D array and the after values represent them in the second 3D array that is collected after the sandbox is changed.

$$xDim = \begin{pmatrix} |x1Before - x2Before| + |x3Before - x4Before| \\ +|x1After - x2After| + |x3After - x4After| \end{pmatrix} /4 \quad (1)$$

$$yDim = \begin{pmatrix} |y3Before - y1Before| + |y4Before - y2Before| \\ +|y3After - y1After| + |y4After - y2After| \end{pmatrix} /4 \quad (2)$$

The z dimension is calculated by subtracting the maximum depth from the minimum depth. If these dimensions are used, it would produce an overestimate of the volume, depicted by the red dotted lines in Figure 9-B. To reduce this overestimate to a more

accurate volume calculation, the z dimension is reduced. The depth between the maximum and minimum z values is calculated for the before and after sets of points (Equation 3 and 4). After adding these two values and dividing the result by two, this can be subtracted from the z dimension to reduce the rectangular prism from the red dotted box to the green in Figure 9-C. Therefore, the volume calculated for the irregular prism is the green rectangular prism (Figure 9-D) that cuts off part of the original prism but covers a portion outside of it, resulting in an estimated volume calculation. Equation 5 represents the final volume calculation computed for each set of points.

$$\Delta ZBefore = maxZbefore - minZBefore \quad (3)$$

$$\Delta ZAfter = maxZAfter - minZAfter \quad (4)$$

$$V = xDim * yDim * \left((maxZ - minZ) - \left(\frac{1}{2}\right) * (\Delta ZBefore + \Delta ZAfter)\right) \quad (5)$$

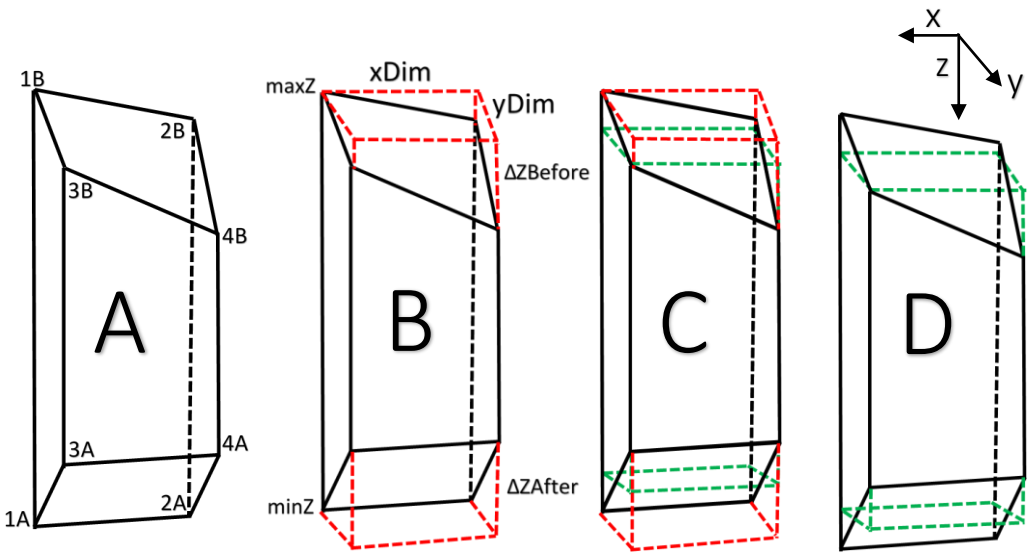

Figure 9. A: The irregular prism formed by the four points of the before array (1B-4B) and the four points of the after array (1A-4A). B: The x and y dimensions of the prism with the z values of interest and the overestimated rectangular prism formed by the dotted red lines. C: The overestimated rectangular prism compared to the one used for volume calculation formed by the green dotted lines. D: The green dotted line rectangular prism used to calculate the volume of the irregular prism.

4 Initial Results

Once the volume tool was completed, it was tested with multiple shapes whose expected volumes were calculated by hand. The four shapes tested were two different sized cardboard boxes or rectangular prisms, a triangular prism made of cardboard and a glass bowl to represent a hemisphere. Each objects expected volume was calculated by hand. Each object was placed within the sandbox and the selection box was drawn around it. The object was then removed and the volume tool was triggered to calculate the change in volume. This was performed five times for each object. Five more tests were completed where the selection was made and the object was placed into the sandbox. By doing this, both the volume added and volume removed could be tested, with the expectation that the absolute value of the result should be the same. In addition to the four objects, volume was collected over an unchanged sandbox surface and one that had been manipulated by randomly digging by hand. Both of these measurements have an expected net volume change of zero since no sand was removed or added to the sandbox.

Due to the inconsistencies of the Kinect sensor which will be elaborated on in the next section, the volumes for all the tests were collected in three ways, a single collection as well as the mean and median of five collections of the before and after arrays. All of the volume calculations are reported in centimeters cubed. Table 1 displays expected value along with the average of the ten collections for each object and collection technique. The results from the calculation on the unchanged sandbox and the random digging produced the closest to the expected results of a net zero volume change. However, all of the other objects reported large overestimates of the expected volume.

Table 1. Average Volume Calculations of Different Objects Over 10 Calculations

	Unchanged sandbox (cm^3)	**Random Dig (cm^3)**	**Large Rectangular Prism (cm^3)**	**Small Rectangular Prism (cm^3)**	**Triangular Prism (cm^3)**	**Hemisphere (cm^3)**
Expected Value	0.0	0.0	30223.3	12105.7	11830.0	2999.0
Single Collection	0.8	121.0	36450.3	13186.5	14012.6	3229.9
Mean of 5 Collections	0.5	49.2	35961.6	13159.5	13812.2	3252.7
Median of 5 Collections	0.3	38.0	35960.8	13099.9	13913.6	3276.6

The percentage of error for each of the measured objects was calculated and is listed in Table 2. Taking the mean and median over five collections appears to reduce error slightly in almost every category but does not make a dramatic impact overall. Furthermore, it can be observed that the taller the object, with the large rectangular prism and the triangular prisms having the largest change in the z dimension, the more error there is in the calculation. This will also be explored further in the next section. Overall, the initial results contained much more error than anticipated.

Table 2. Percent Error of Volume Calculations of Different Objects

	Large Rectangular Prism (%)	**Small Rectangular Prism (%)**	**Triangular Prism (%)**	**Hemisphere (%)**	**Average (%)**
Single Collection	17.1	8.2	15.6	7.1	12.0
Mean of 5 Collections	16.0	8.0	14.4	7.8	11.5
Median of 5 Collections	16.0	7.6	15.0	8.5	11.7

5 Error in Volume Calculation

The Kinect v1 sensor suffers from noise in the depth data and inconsistencies at different distances. Each dimension of the volume calculation contains error not only from the approximation formula but from the data produced by the sensor. To better understand the results produced by the volume tool, error propagation was calculated and the sources of error were investigated in an attempt to find future mitigation strategies.

5.1 Reducing Inconsistencies with Multiple Collections

In an attempt to mitigate some of the inconsistencies in the depth image data, two primitive filtering techniques were used. The mean and median were done over five collections for each the before and after depth image arrays. While the results show this does not provide any significant improvements in the accuracy of the volume calculations, it provided a small amount of smoothing for flat surfaces. This provided consistent z values for points that were known to be of the same height as opposed to having a variation of a few millimeters in either direction.

5.2 Error Propagation

Each depth image taken by the Kinect varies slightly due to inherent noise in the system. This produces a small range that each x, y and z value can take each time they are collected. In order to quantify some of the error in the volume calculation, that range of error in each dimension of the Kinect was determined. This was performed by collecting the average difference between points in each dimension over an unchanged sandbox surface. If the sensor was consistent, the collections would produce the same x, y and z values for each point since the area is unchanged. Unfortunately, this is not the case and Table 3 shows the potential error for in each dimension for a given collection.

Table 3. Average Difference Between Before and After Depth Images Over an Unchanged Sandbox Surface

	X	**Y**	**Z**
Average over entire sandbox (~140,000 points)	0.12 cm	0.10 cm	0.90 cm

The base equations for calculating error propagation differ for addition/subtraction and multiplication/division. Given Equation 6 which consists of addition and subtraction, Equation 7 represents the error propagation formula which is the square root of the sum of squares, where δ represents the error for a given measurement (Glen, 2016).

$$Q = a + b + \cdots + c - (x + y + \cdots + z) \tag{6}$$

$$\delta Q = \sqrt{(\delta a)^2 + (\delta b)^2 + \cdots + (\delta c)^2 + (\delta x)^2 + (\delta y)^2 + \cdots + (\delta z)^2} \tag{7}$$

Similarly, Equations 8 and 9 represent the error propagation formula for multiplication and division (Glen, 2016).

$$Q = \frac{a,b,\ldots,c}{x,y,\ldots,z} \quad (8)$$

$$\delta Q = |\mathrm{Q}| * \sqrt{\left(\frac{\delta a}{a}\right)^2 + \left(\frac{\delta b}{b}\right)^2 + \cdots + \left(\frac{\delta c}{c}\right)^2 + \left(\frac{\delta x}{x}\right)^2 + \left(\frac{\delta y}{y}\right)^2 + \cdots + \left(\frac{\delta z}{z}\right)^2} \quad (9)$$

Due to the complexity of the volume formula (Equation 5), the error propagation was performed in stages. The values in Table 3 were used as δx, δy, and δz respectively. Each error propagation calculation was performed alongside its corresponding volume calculation. Equations 10 and 11 were used to calculate the error in the x and y dimensions.

$$\delta xDim = \left(\sqrt{(\delta x)^2 * 8}\right) * \left(\frac{1}{4}\right) \quad (10)$$

$$\delta yDim = \left(\sqrt{(\delta y)^2 * 8}\right) * \left(\frac{1}{4}\right) \quad (11)$$

The z dimension error is then propagated through the multiple instances where z values are used for volume calculation (Equations 12 and 13).

$$\delta(\Delta ZBefore + \Delta ZAfter) = \sqrt{(\delta z)^2 * 4} \quad (12)$$

$$\delta(maxZ - minZ) = \sqrt{(\delta z)^2 * 2} \quad (13)$$

The volume formula is then broken down to separate the error propagation of addition and subtraction. Equations 14 through 19 outline the remaining error propagation formulas used to determine the overall error δV that is propagated through the volume calculation.

$$A = \left(\frac{1}{2}\right) * (\Delta ZBefore + \Delta ZAfter) \quad (14)$$

$$\delta A = \left(\frac{1}{2}\right) * \delta(\Delta ZBefore + \Delta ZAfter) \quad (15)$$

$$B = \left((maxZ - minZ) - A\right) \quad (16)$$

$$\delta B = \sqrt{\left(\delta(maxZ - minZ)\right)^2 + (\delta A)^2} \quad (17)$$

$V = xDim * yDim * B$ Eq. (18)

$$\delta V = |V| * \sqrt{\left(\frac{\delta xDim}{xDim}\right)^2 + \left(\frac{\delta yDim}{yDim}\right)^2 + \left(\frac{\delta B}{B}\right)^2} \quad (19)$$

Since the error propagation is done over each volume calculation which are then summed together to produce the total volume, the error at each step must also be summed together (Equations 20 and 21).

$$\delta VTotal = \delta VTotal + (\delta V)^2 \quad (20)$$

$$\delta VTotal = \sqrt{\delta VTotal} \quad (21)$$

The average error propagation calculated alongside ten volume calculations is reported in Table 4. The amount of error that the error propagation reports is significantly less than the error the volume measurement produces. Aligning each result with its corresponding volume from Table 1, the error calculated does not account for much of the error experienced. For example, the large rectangular prism had over 5000 cm^3 of error and reports about $\pm$ 100 cm^3. Since the expected value of the large rectangular prism is 30223.3 cm^3, the error propagation should bring the expected value into range. However, there are still a few thousand cubic centimeters of error. This means that there is a lot of error coming from other sources and not just sensor inconsistency over unchanged depth images.

Table 4. Error Propagation Calculated for Each Volume Calculation Averaged over 10 Calculations

	Unchanged sandbox (cm^3)	**Random Dig (cm^3)**	**Large Rectangular Prism (cm^3)**	**Small Rectangular Prism (cm^3)**	**Triangular Prism (cm^3)**	**Hemisphere (cm^3)**
Single Collection	$\pm$ 0.46	$\pm$ 10.1	$\pm$ 101.1	$\pm$ 40.0	$\pm$ 50.3	$\pm$ 20.2
Mean of 5 Collections	$\pm$ 0.59	$\pm$ 3.9	$\pm$ 102.1	$\pm$ 42.8	$\pm$ 50.5	$\pm$ 20.0
Median of 5 Collections	$\pm$ 0.56	$\pm$ 9.5	$\pm$ 99.2	$\pm$ 42.5	$\pm$ 49.8	$\pm$ 20.0

5.3 Error in the x and y Dimensions Based on z

As outlined before, the x and y points used to calculate the x and y dimensions for volume calculation were (x, y) pairs of depth image pixels being converted to their corresponding centimeter values with respect to the center of the depth image. However, the current depth of those pixels directly relates to the value they are converted to. The closer to the camera a pixel is, the closer together the pixels are considered. Consequently, pixels that are more distant from the camera are interpreted as more spread out. This is depicted in Figure 10 where the x and y values match the expected real world values at the green depth but are smaller at the yellow and larger at the red. This causes inconsistencies in the x and y dimensions calculated when using the before points and the after points.

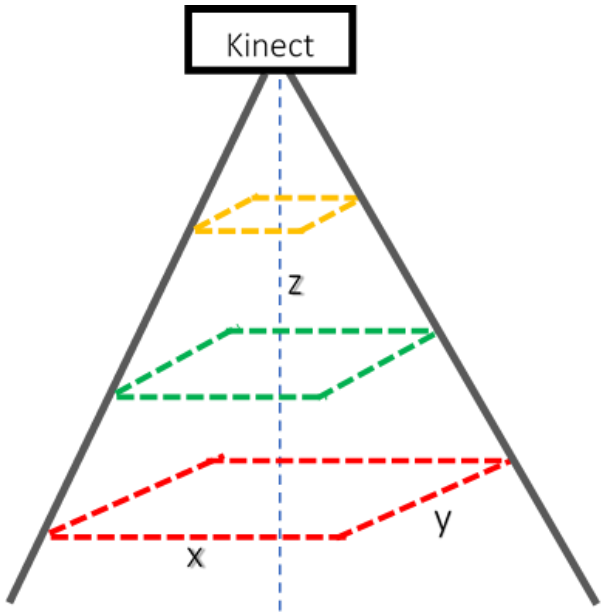

Figure 10. Diagram of x and y dimension inconsistencies

Table 5 shows the average x and y dimension calculated at five different distances away from the camera in centimeters. To illustrate this, consider the scenario where the before point reports x = 0.23 cm and the after point report x = 0.29cm. After taking the average, the resulting x dimension that is used for the volume calculation is 0.26cm. While this difference from the actual value is on the scale of less than a millimeter, which the Kinect cannot even detect, it affects the end result of each volume calculation by a few millimeters. This difference quickly adds up over thousands of calculations being summed together, leading to error in the volume calculation outlined in Tables 1 and 2.

Table 5. Average x and y Dimensions at a Given z

	z (cm)	**x (cm)**	**y (cm)**
Depth 1	167.4	0.29	0.29
Depth 2	158.7	0.27	0.26
Depth 3	143.7	0.24	0.23
Depth 4	134.7	0.23	0.23
Depth 5	125.4	0.22	0.22

5.3.1 Mitigation of Error in the x and y Dimensions

In order to mitigate the error in the x and y dimensions, the total x and y lengths of the small rectangular prism were collected at 5 different depths. Each value was then compared to its expected value and percent error was calculated. By plotting the percent error over the depth (z dimension) for both the x and y dimensions, a linear line could be fitted to the points. Figures 11 and 12 show the graphs and trendlines for the x and y dimensions, respectively.

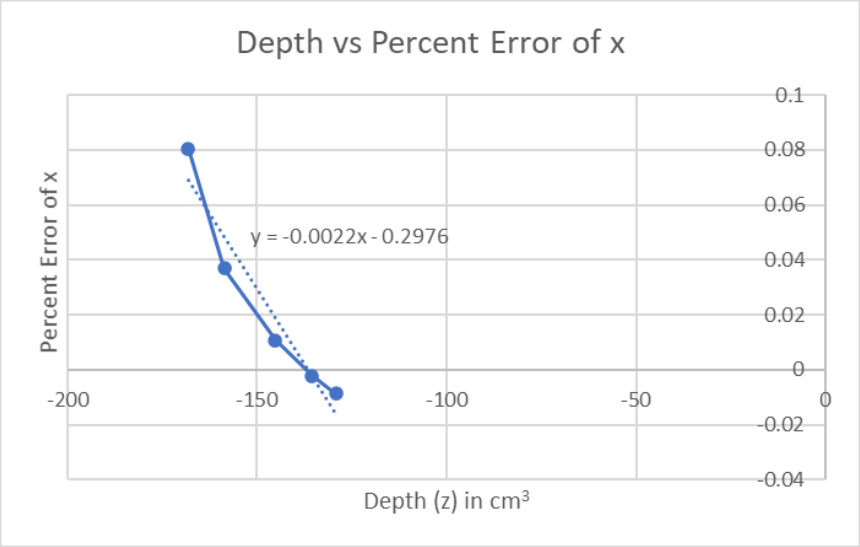

Figure 11. Depth over percentage of error in the x dimension

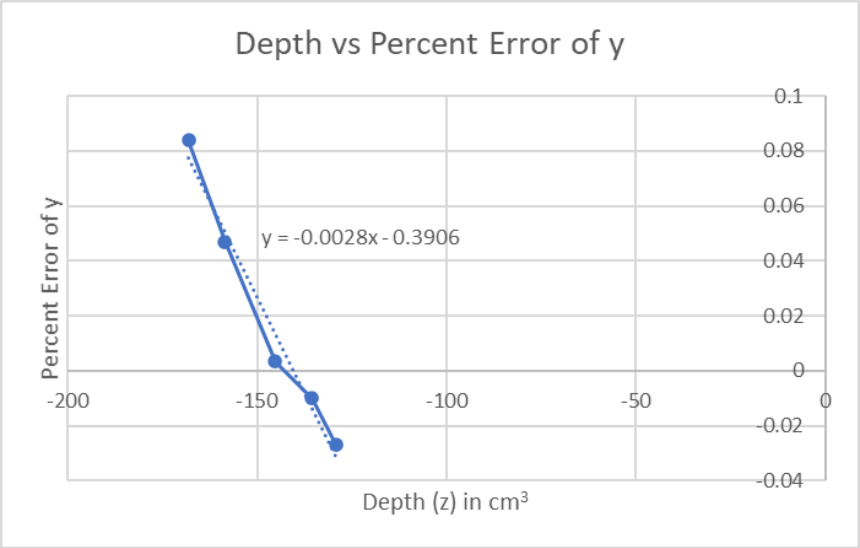

Figure 12. Depth over percentage of error in the y dimension

These trendlines help to depict the expected error of the x and y dimensions based on their distance away from the camera. To account for this error in the volume formula, the expected error is accounted for using the formula of the trendline. The x and y calculations were changed by subtracting the expected error based on their z values (Equations 22 and 23).

$$\Delta X1 = |x1Before - x2Before| - |x1Before - x2Before| * \left(-0.0022 * \left(\frac{z1Before + z2Before}{2}\right) - 0.2976\right) \quad (22)$$

$$\Delta Y1 = |y3Before - y1Before| - |y3Before - y1Before| * \left(-0.0028 * \left(\frac{z3Before+z1Before}{2}\right) - 0.3906\right) \quad (23)$$

For simplicity, only the formulas for the first ΔX and ΔY are provided but adjusted values for all four x and y are calculated. The x and y dimension values are once again the average of those four respective values (Equations 24 and 25).

$$xDim = (\Delta X1 + \Delta X2 + \Delta X3 + \Delta X4)/4 \quad (24)$$
$$yDim = (\Delta Y1 + \Delta Y2 + \Delta Y3 + \Delta Y4)/4 \quad (25)$$

5.3.2 Adjusted Volume Calculations

Ten volume calculations were performed for each object using the adjusted x and y values. The unchanged sandbox and random digging are omitted due to unchanged results. The results are displayed in Table 6. By accounting for the error in the x and y dimensions caused by varying depths and adjusting them accordingly, the volume calculations produce significantly less error. The percentage of error can be seen in Table 7. The rectangular prism still experiences the largest amount of error due to the Kinect having a hard time detecting heights along a 90 degree edge, but it is still a major improvement from the original collections. Overall, 1-8% error is more acceptable than the 8-17% of error that occurred before this adjustment was made. Due to the change in how the x and y dimension are calculated for the volume, the error propagation formulas are no longer valid. They will need to be updated in the future to reflect the new calculations and to better understand where the remaining error is coming from.

Table 6. Average Volume Calculation over 10 Calculations

	Large Rectangular Prism (cm^3)	**Small Rectangular Prism (cm^3)**	**Triangular Prism (cm^3)**	**Hemisphere (cm^3)**
Expected Value	30223.3	12105.7	11830.0	2999.0
Single Collection	31841.4	11496.7	12267.2	3239.6
Mean of 5 Calculations	32487.5	11932	11932.3	3165.9
Median of 5 Calculations	32641.9	11893	12064.6	3175.4

Table 7. Percentage of Error of Volume Calculations

	Large Rectangular Prism (%)	**Small Rectangular Prism (%)**	**Triangular Prism (%)**	**Hemisphere (%)**	**Average (%)**
Single Collection	5.4	5.0	3.7	8.0	5.5
Mean of 5 Collections	7.5	1.4	0.9	5.6	3.8
Median of 5 Collections	8.0	1.8	2.0	5.9	4.4

6 Conclusion

The AR sandbox application provides a real-time topographical map of the sandbox surface which can be used to better visualize how the sand is manipulated. This has the potential to be a beneficial tool for the PSTDL. A 60 inch by 45 inch sandbox was built within the PSTDL and an AR sandbox was setup and calibrated. Within the software frameworks a volume calculation tool was developed to provide a valuable metric to project teams using the sandbox for testing their excavation tools.

The volume tool allows a user to calculate the change in volume over a selected area of the sandbox. Due to error and inconsistencies from the Kinect v1 sensor, the initial volume calculations experienced a large amount of error, ranging from 8% to 17% error based on how the depth image changed in shape. Mean and median filters were applied to the depth image collections but did little to reduce the error. It was then discovered that the x and y dimensions varied at different depths of the sandbox. This had the potential to be a major cause of the experienced error. This was mitigated by adjusting the x and y dimensions to be closer to their expected value, depending on their depth. As a result, the error in the volume calculations reduced down to 1% to 8% error.

These initial results and error mitigation strategies show promise that change in volume can be properly calculated with a depth camera. With more investigation into the causes of error in the calculation as well as exploring other depth camera options, the calculation can be made more accurate.

7 Future Work

There are many improvements and additions that can be made to the volume tool project. As mentioned in the preliminary work section, the Kinect V2 can now be used if the software packages are updated. Nevertheless, there appear to be advantages and disadvantages of using one over the other. In their 2016 paper, *Comparison of Kinect v1 and v2 Depth Images in Terms of Accuracy and Precision*, Wasenmuller and Stricker found that the Kinect v1 has exponential decrease in accuracy as distance increased . However, it was also observed that the Kinect v2 has much lower precision for flat surfaces as well as uneven ones and contains a lot more extreme points than the v1 (Wasenmuller & Stricker, 2016). The reported results are further supported by Zennaro, et al. who found that the Kinect v2 was around two times more accurate at short range and close to ten times more accurate after distances of 6 meters (Zennaro, et al., 2015). These observations could be investigated further, and the volume tool could be tested with both cameras to determine which produces better calculations for this project.

Another improvement that can be made within the volume tool is investigating and mitigating error further. This could be done by using a different filtering technique to achieve more consistent results in the depth image collections. However, the error observed by inconsistent x and y dimensions based on distance away from the sensor would remain unaffected by these filters. The error within the x and y dimension could be improved by taking more collections at various heights and establishing a stronger trendline to be used for the adjustment. Furthermore, the way the error of each dimension is propagated through the volume formulas can be updated to account for the changes made to the formulas, which in turn could bring insight into how much error is expected in each calculation.

One of the original goals of the project that was not achieved was to have the volume tool working within the SARndbox application. This would allow an area of the topographical map to be selected as opposed to using the raw depth image in the RawKinectViewer application. Moving the tool back to the other application should be possible as long as undefined function references are handled. Since many of the functions used to obtain the depth image arrays and convert the values to centimeters are not available in the AR sandbox application, some of the functionality may need to be rewritten. Regardless, at its core, the volume tool will accept two arrays of x, y, and z dimensions and compute the volume difference between them. As long as those arrays can be properly formed, the volume can be calculated.

Furthermore, due to the intrinsic error of the Kinect sensors, other depth cameras could be explored for future advances of this project. Other popular alternatives to using a Kinect sensor for research purposes are Azure Kinect DK camera and the Intel RealSense depth camera series. Similar to the comparison of the Kinect v1 and v2 above, each camera has its own advantages and disadvantages that would need to be explored further before purchasing. Additionally, a different camera would not work within the current software packages since they are built for a Kinect, but a new standalone application could be developed to simply read in the depth images and report the volume change.

8 References

Build your own AR Sandbox. (2016). Retrieved from Augmented Reality Sandbox: https://arsandbox.ucdavis.edu/instructions/

Dellen, B., Rojas, J., & Andres, I. (2013). Volume measurement with a consumer depth camera based on structured infrared light. *Catalan Conference on Artificial Intelligence*, (pp. 1-10). Barcelona.

Ferreira, B., Grine, M., Gameiro, D., Costeira, J., & Santos, B. (2014). VOLUMNECT: measuring volumes with Kinect. *Proceedings of SPIE - The International Socierty for Optical Engineering.*

Glen, S. (2016, August 15). *Error Propagation (Propagation of Uncertainty)*. Retrieved from StatisticsHowTo.com: https://www.statisticshowto.com/error-propogation/

Kreylos, O. (2021, March 13). *Software Instructions*. Retrieved from Augmented Reality Sandbox: https://web.cs.ucdavis.edu/~okreylos/ResDev/SARndbox/

Wasenmuller, O., & Stricker, D. (2016). Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision. In *Lecture Notes in Computer Science.*

Zennaro, S., Munaro, M., Malani, S., Zanuttigh, P., Bernardi, A., Ghidoni, S., & Menegatti, E. (2015). Performance evaluation of the 1st and 2nd generation kinect for multimedia applications. *IEEE Internation Conference on Multimedia and Expo.* Turin, Italy: IEEE.