COMPUTER SIMULATION TECHNOLOGY

3D Monte Carlo Integration Over a Complicated 3D Surface.

Frank Fayia Kendemah(玛拿西)

ABSTRACT

Monte Carlo Simulation is one of the earliest and most used statistical techniques for drawing conclusion from a small sample. Monte Carlo Simulation uses randomly generated variables or numbers to solve problems. It is applicable in many different fields across science. The most difficult job in computer vision today is 3D object occlusion estimate, which has implications for several areas in the technology sector such as virtual reality, autonomous driving car, robotics, and pedestrian spotting. In this paper, the study described tests with five groups of randomly chosen parameters for three-dimensional objects in Euclidean Space and calculated the size of the occluded region and volume of the intersection part created using the Monte Carlo approach (integration) between 3D objects.

Keywords: *Monte Carlo Simulation, Computer Vision, Euclidean Space, 3D occluded objects.*

1. INTRODUCTION

As compared to three-dimensional objects (3D), it is comparatively simple to calculate the volume of these shapes, length, and area. The collectively perpendicular X and Y axes of the two-dimensional objects (2D) (circle, square, rectangle, and polygons) are on a flat plane which makes the calculation of their volume easier as compared to the three-dimensional objects (3D) (Mustapha Musa et al, 2020). The solid shapes we interact with in the physical world are 3D objects, which have width, height, and depth expressed on axes (Jana, 2010). A Mathematical definition of object occlusion is a discontinuity in a smoothed surface's function. Finding out whether there is a major discontinuity (within the discrete domain) in nearby regions is the aim of occlusion detection. In engineering simulation, robotics, and computer vision, occlude detection has a wide range of uses, including real-time rendering, virtual reality, medical tissue modelling, and 3D printing (Zelek, 2004). There are several methods for finding an occlusion in three-dimensional objects (3D) in Euclidean space that have been proposed. It is still difficult to recognize actively obstructed objects. Occluded net was first introduced by (Reddy et al, 2019) and uses MaskRCNN, which has been only trained on visible key point annotations, to forecast the 2D and 3D positions of occluded object key points. A graph decoder network then corrects the first detector's occluded key point position after a graph encoder network explicitly identifies unseen edges. (J. Xu, 2019) uses the Backbone Network to create a 3D object detection framework on the point cloud that fuses low-level and high-level data. The experiments with the automobile, bike, and pedestrian detection systems have shown effective, averagely accurate results. Planar and solid occludes are combined utilizing a unified selection strategy for dynamic environments pre-calculated visibility data, according to a method given by (G. Papaioannou, 2006). The methodology used to commercially deploy virtual reality equipment and the data from genuine virtual reality program. A method for identifying and locating obstructed apples using the k-means clustering algorithm and convex hull was also published by (D. Wang, 2015). An effective approach of identifying and locating occluded apples was concluded based on the experiment results, which demonstrate that

the method could achieve a considerably better location rate than the Hugh transformation method and contour and curvature method.

This study uses the Monte Carlo Approach (MCA) to estimate the occlusion of 3D objects. Numerous quantitative scientific problems are solved using this effective method (D. P. Kroese, 2014).

2. PROBLEM DEFINITION

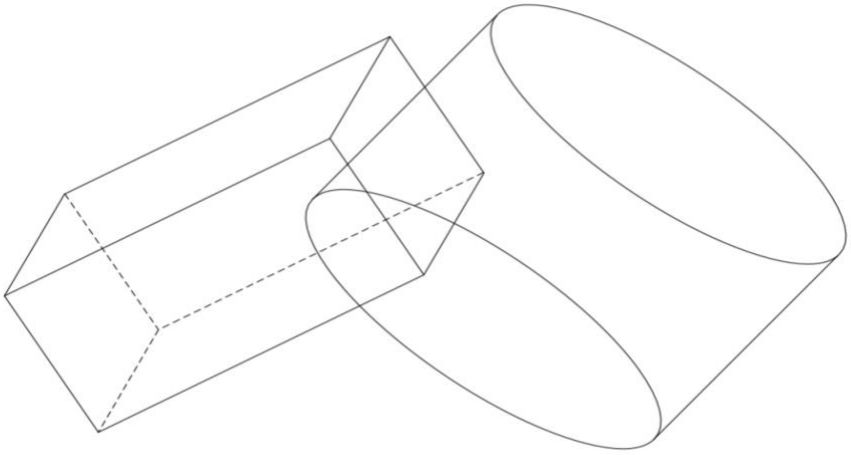

Figure 1: A rectangular box and a cylinder in 3D space.

There exist a rectangular box and a cylinder in a 3D space. Their sizes, locations and orientations are arbitrary. They may or may not intersect with each other. If they do intersect with each other, theoretically, the volume of their intersection part can be expressed as a 3D integration over a complicated 3D region. If they are intersected, the volume of their intersection is zero. The rectangular box is described by its eight vertices $\boldsymbol{P_1, P_2, \ldots, P_8}$. The cylinder is described by the two center points $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ of its top and bottom faces and the circle radius $\boldsymbol{R}$ of its bottom face.

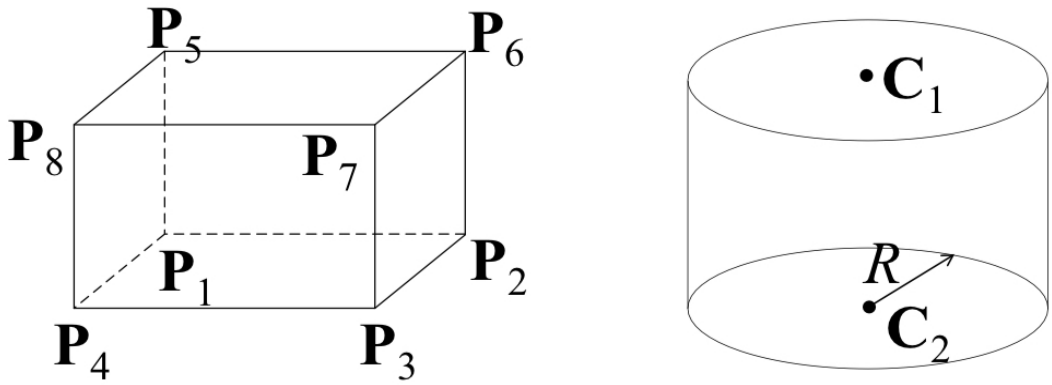

Figure 2: The 3D shapes with their vertices definitions.

2.2. MONTE CARLO SAMPLING STRATEGIES.

I. **Cuboid** – The six faces of a cuboid, which can be any quadrilateral, form a convex polyhedron in three dimensions. A cuboid is made up of six rectangles arranged at right angles. Its vertices are $\boldsymbol{P_1, P_2, \ldots, P_8}$, and it has twelve edges, six sides, and eight angles. $\boldsymbol{h}$ stands for height, $\boldsymbol{l}$ for length, and $\boldsymbol{w}$ for width. One must compute the distance between two locations using the vertical points in the Euclidean distance equation in order to obtain the cuboid's width, length, and height.

$$d(P_i, P_j) = \sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2 + (Z_j - Z_i)^2}$$

$$d = \sqrt{X^2 + Y^2 + Z^2}$$

$$h = d(P_5, P_1)$$

$$l = d(P_4, P_1)$$

$$w = d(P_2, P_1)$$

$$P(X, Y, Z) = \begin{pmatrix} X_1 & \times & l \\ X_2 & \times & h \\ X_3 & \times & w \end{pmatrix}$$

$$P = [X + Y + Z] + P_1$$

$X_1, X_2, X_3 \in [0, 1]$ are random points. The volume of the cuboid is: $v = l \times w \times h$.

II. **Cylinder** – A three-dimensional object with circles at its two ends is known as a right circle cylinder. When determining the volume of an oblique cylinder, the height (h) or altitude is the perpendicular distance between the bases. The radius (r) of a cylinder is its base radius. The definition of cylindrical coordinates is as shown below:

$$P(X, Y, Z) = \begin{pmatrix} \sqrt{X} & \times & r\cos(\emptyset) \\ \sqrt{X} & \times & r\sin(\emptyset) \\ \sqrt{X} & \times & h \end{pmatrix}$$

$$r \geq 0, \emptyset \in [0, 2\pi Y], h \in R.$$

$X, Y, Z \in [0, 1]$ are random points.

The volume of the cylinder is: $\boldsymbol{v = \pi r^2 h}$, where $\boldsymbol{h = v/\pi r^2 \equiv d(C_1, C_2)}$.
The cylinder height is the distance between two bases of the cylinder.

2.2. ALGORITHM DESCRIPTION.

For this, one must select a randomly generated point from the cuboid and determine if it falls on the cylinder's volume. If it does, then choose the point; if not, then toss it. The total number of points approved divided by the total volume of the cuboid results in the volume of the occluded area. A distribution can be sampled using the Accept-Reject method, a traditional sampling technique, even though an inverse transformation would make this distribution difficult or impossible to reproduce (Casella, 2013).

- ***Determine if the cuboid point intersected with the cylinder.***

As to determine if a cuboid point falls within the cylinder volume. The cylinder's bases are $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$, its radius $\boldsymbol{R}$ and its test point as $\boldsymbol{P}$.

The number one step is to determine if the points fall within the height of the cylinder.

$$\left(\vec{P} - \overrightarrow{C_1}\right).\left(\overrightarrow{C_2} - \overrightarrow{C_1}\right) \geq 0 \;\&\; \left(\vec{P} - \overrightarrow{C_2}\right).\left(\overrightarrow{C_2} - \overrightarrow{C_1}\right) \leq 0$$

The equation will check if $\boldsymbol{P}$ is between the planes of the cylinder's two circular faces.

$$\frac{\left|\left(\vec{P} - \overrightarrow{C_1}\right) \times \left(\overrightarrow{C_2} - \overrightarrow{C_1}\right)\right|}{\left(\overrightarrow{C_2} - \overrightarrow{C_1}\right)} \leq R$$

The formula above finalizes if $\boldsymbol{q}$ falls within the arc surface of the cylinder.

Algorithm to Count the Intersection

1. f**or all** points *i* on the cuboid do:
2. **loop** cylinder
3. **if** intersect () **then**:
4. **count** intersect + *i*
5. **end**
6. **end**
7. result = (count / N) * cuboid_volume.

The source code for this study work can be obtain from GitHub on the link below:
Monte-Carlo-3D-Occlusion/3D_Occluded_Objects.py at main · Manasseh19k/Monte-Carlo-3D-Occlusion (github.com)

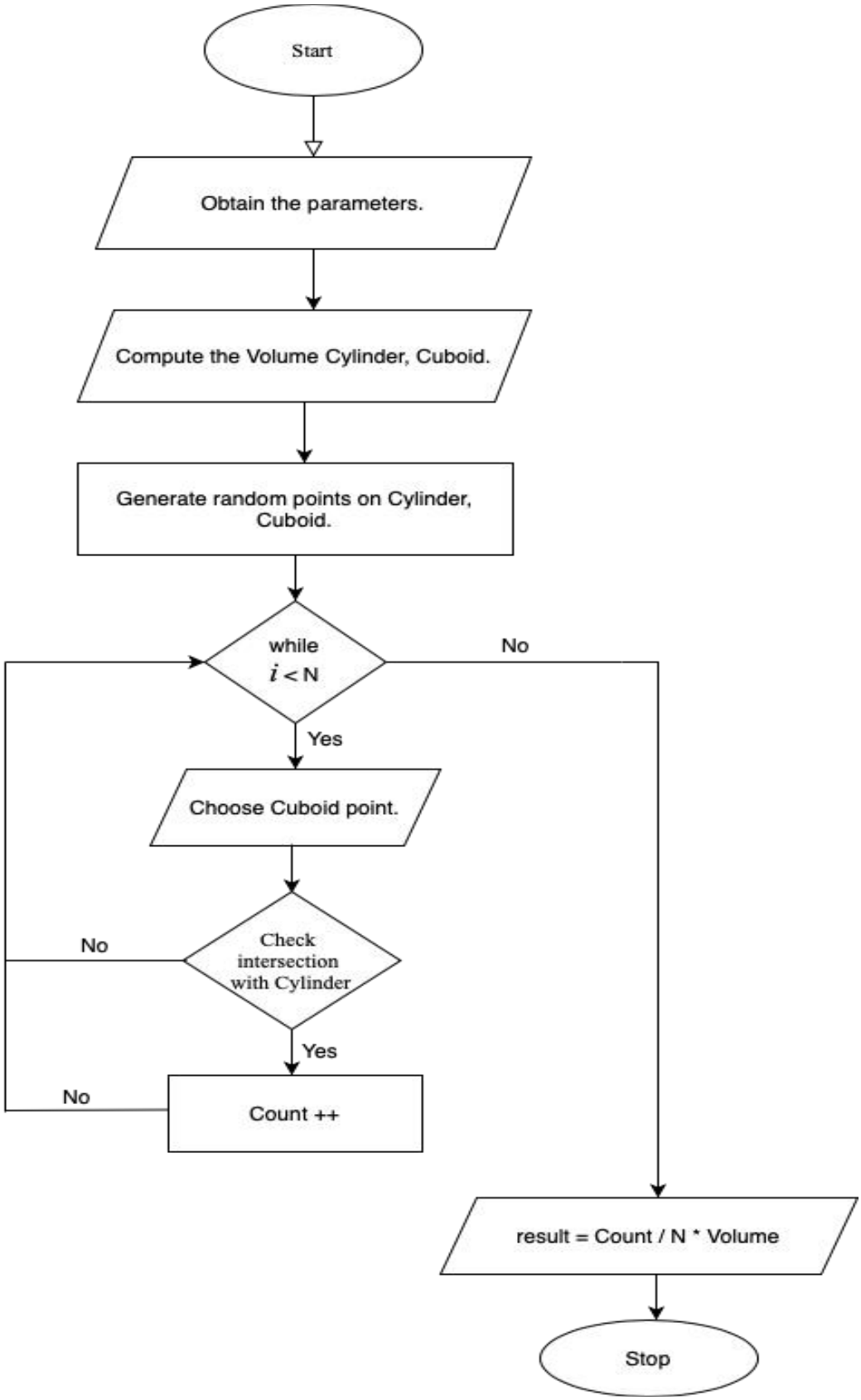

Figure 3: Flowchart of the problem.

Table 1: Objects, Vertices, and Parameters.

Experimental Parameters: The vertices parameters of the 3D objects, Cylinder, and Cuboid.							
Objects	Vertices	Group 1			Group 2		
		x-axis	y-axis	z-axis	x-axis	y-axis	z-axis
Cuboid	P_1	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
	P_2	0.000e+00	1.000e+00	0.000e+00	3.305e-01	3.389e-01	8.808e-01
	P_3	1.000e+00	1.000e+00	0.000e+00	1.165e+00	6.694e-01	4.404e-01
	P_4	1.000e+00	0.000e+00	0.000e+00	8.347e-01	3.305e-01	-4.404e-01
	P_5	0.000e+00	0.000e+00	1.000e+00	4.404e-01	-8.808e-01	1.736e-01
	P_6	0.000e+00	1.000e+00	1.000e+00	7.709e-01	-5.419e-01	1.05e+00
	P_7	1.000e+00	1.000e+00	1.000e+00	1.605e+00	-2.113e-01	6.140e-01
	P_8	1.000e+00	0.000e+00	1.000e+00	1.275e+00	-5.502e-01	-2.668e-01
Cylinder	C_1	5.000e-01	5.000e-01	1.000e+00	1.023e+00	-5.461e-01	3.938e-01
	C_2	5.000e-01	5.000e-01	0.000e+00	5.826e-01	3.347e-01	2.202e-01
	R	5.000000e-01			5.000000e-01		

Table 2: Objects, Vertices, and Parameters.

Objects	Vertices	Group 3			Group 4		
		x-axis	y-axis	z-axis	x-axis	y-axis	z-axis
Cuboid	P_1	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00	0.000e+00
	P_2	1.326e-02	8.674e-01	4.975e-01	0.000e+00	1.000e+00	0.000e+00
	P_3	1.012e+00	8.806e-01	4.478e-01	1.000e+00	1.000e+00	0.000e+00
	P_4	4.478e-01	1.326e-02	-4.975e-02	1.000e+00	0.000e+00	0.000e+00
	P_5	4.975e-02	-4.975e-01	8.660e-01	0.000e+00	0.000e+00	1.000e+00
	P_6	6.302e-02	3.698e-01	1.364e+00	0.000e+00	1.000e+00	1.000e+00
	P_7	1.062e+00	3.831e-01	1.314e+00	1.000e+00	1.000e+00	1.000e+00
	P_8	1.048e+00	-4.843e-01	8.163e-01	1.000e+00	0.000e+00	1.000e+00
Cylinder	C_1	5.557e-01	-5.721e-02	1.089e+00	5.698e-01	3.159e-01	1.037e+00
	C_2	5.059e-01	4.403e-01	2.238e-01	4.212e-01	5.624e-01	7.937e-02
	R	1.000000e+00			5.000000e-01		

Table 3: Objects, Vertices, and Parameters.

Objects	Vertices	Group 5		
		x-axis	y-axis	z-axis
Cuboid	P_1	0.000000e+00	0.000000e+00	0.000000e+00
	P_2	2.923852e-01	-1.844820e-01	9.383375e-01
	P_3	-6.604095e-01	-1.567369e-01	1.240682e+00
	P_4	-9.527947e-01	2.774510e-02	3.023449e-01
	P_5	8.181147e-02	9.824442e-01	1.676613e-01
	P_6	3.741967e-01	7.979621e-01	1.105999e+00
	P_7	-5.785981e-01	8.257072e-01	1.408344e+00
	P_8	-8.709833e-01	1.010189e+00	4.700061e-01
Cylinder	C_1	5.697592e-01	3.159923e-01	1.037074e+00
	C_2	4.211637e-01	5.624242e-01	7.937364e-02
	R	5.000000e-01		

2.3. SIMULATION OUTCOMES.

In this study, two simulation tests were conducted utilizing 400, 000, and 500, 000 uniform random points using the python library NumPy and its ***random.rand()*** function and filled the volume of the 3D objects depicted in Figure 4 with the use of the 3D point vertices parameters supplied in Table 1, 2, and 3 above.

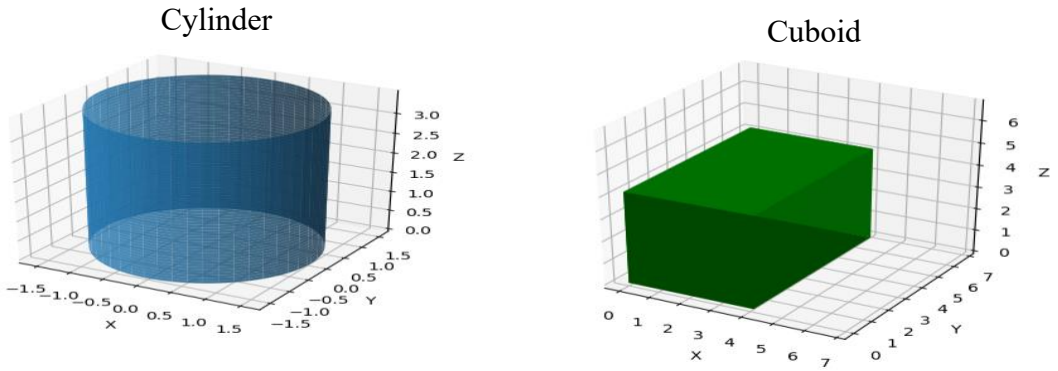

Figure 4: The objects in 3D with filled random points.

To conduct this work or experiment, the following tools were used: MacBook Air (Retina) with a Processor of 1.1 GHz Dual-Core Intel Core m3 and a Memory 8 GB 1867 MHz. Python3 programming language was used along with its libraries NumPy and matplotlib.

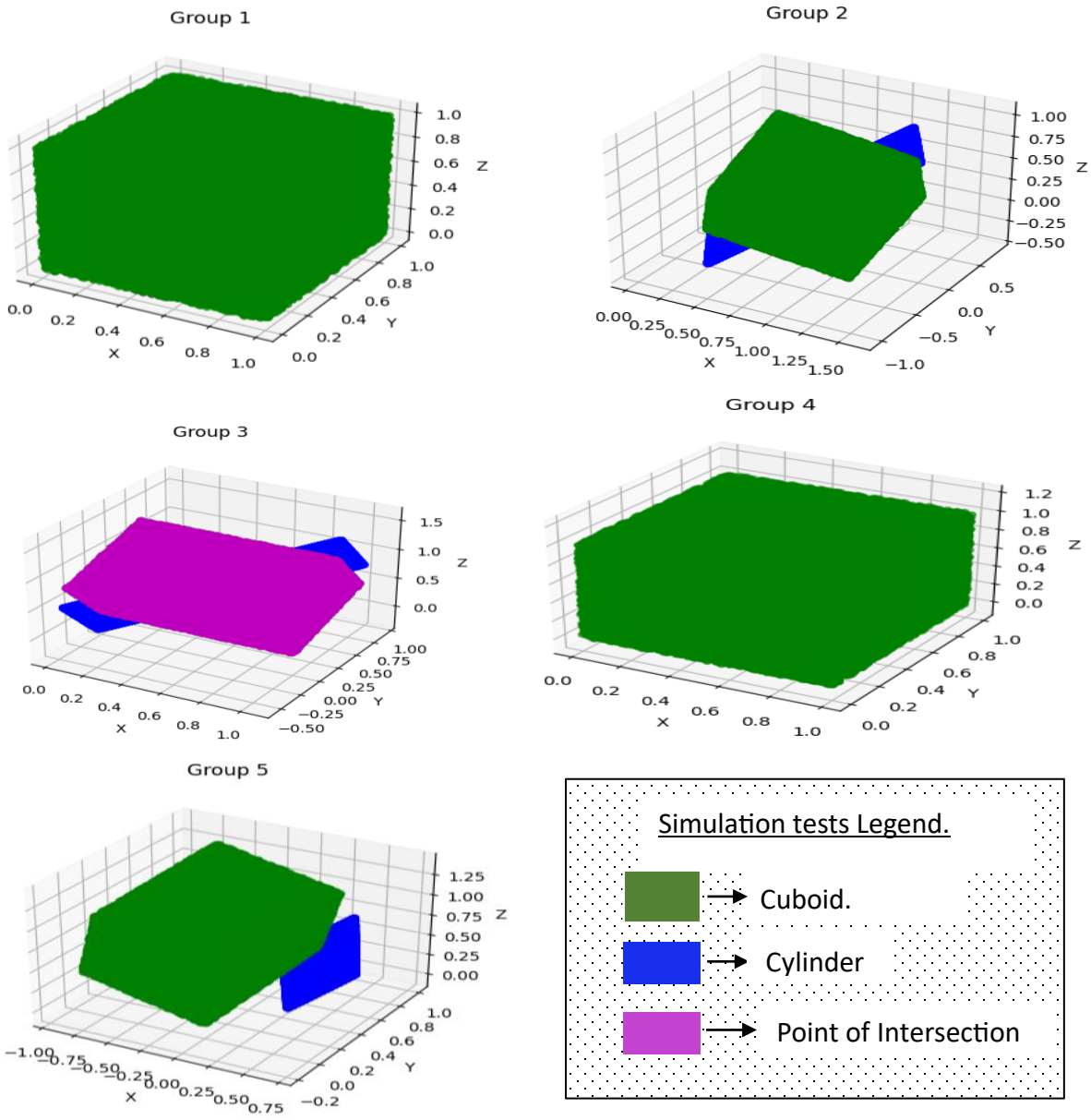

Figure 5: The simulation results of the occlusion, the magenta region is the point of occlusion between the objects in 3D.

Table 4: The Simulation Tests Outcomes.

		VOLUMES (COMPUTED)		INTERSECTION.	
Group.	**N (Sample).**	**Cylinder.**	**Cuboid.**	**Points.**	**Volume.**
1	400, 000	0.7854	1.0	314, 339	0.78585
	500, 000	0.7854	1.0	392, 582	0.785164
2	400, 000	0.7854	0.9999 (1.0)	314, 299	0.78747
	500, 000	0.7854	0.9999 (1.0)	392, 118	0.78424
3	400, 000	3.141593	0.9999 (1.0)	400, 000	0.9999 (1.0)
	500, 000	3.141593	0.9999 (1.0)	500, 000	0.9999 (1.0)
4	400, 000	0.7854	1.0	281, 317	0.7033
	500, 000	0.7854	1.0	351, 610	0.7032
5	400, 000	0.7854	0.9999 (1.0)	450, 26	0.11256
	500, 000	0.7854	0.9999 (1.0)	559, 55	0.1119

Due the nature of randomness, the Monte Carlo simulation does not always provide the same result, and the outcome may somewhat differ as stated in Table 4. Although uncertain conditions exist, it is still the most dependable and useful approach of evaluating measurements in practice. Only when the number of trail samples rises do the results begin to asymptotically assemble (Mustapha Musa et al, 2020). In a Monte Carlo calculation, the error is proportional to $\boldsymbol{\sigma}/\sqrt{\boldsymbol{n}}$. The number of values we utilize for the average determine its accuracy (Feigin, 2012). By using more samples, we can decrease the error. But keep in mind that we need to add four times as many samples to cut the error in half. Minimizing variation is another technique to increase accuracy. The variance described by σ^2 is a potential indicator of the error $\boldsymbol{\sigma^2 = \langle f^2 \rangle - \langle f \rangle^2}$. Where $\boldsymbol{\langle f^2 \rangle = \frac{1}{N}\sum_{i=1}^{N} f(x_i)}$ and $\boldsymbol{\langle f \rangle^2 = \frac{1}{N}\sum_{i=1}^{N} f(x_i)^2}$. The sigma quantity won't change, but we should anticipate that the error will decrease with $\boldsymbol{N}$ points. As the result, this approach is unsuitable for estimating mistakes. Imagine that we are carrying out several integrated measurements, each of which yields $\boldsymbol{I_n}$. These values were generated using $\boldsymbol{N}$ random integers from various sequences. The central limit theorem states that these values typically have a mean value of $\langle \boldsymbol{I} \rangle$. Let's say we have a collection of these measurements, $\boldsymbol{I_n}$. The standard deviation of the means $\boldsymbol{\sigma M}$ is an appropriate metric for determining how these measurements differ from one another $\boldsymbol{\sigma_M^2 = \langle I^2 \rangle - \langle I \rangle^2}$, where $\boldsymbol{\langle I \rangle = \frac{1}{N}\sum_{n=1}^{N} I_n}$ and $\boldsymbol{\langle I \rangle^2 = \frac{1}{M}\sum_{n=1}^{N} I_n^2}$, this can be proven that $\boldsymbol{\sigma M \approx \sigma/\sqrt{N}}$.

At the upper and lower boundaries of a sizeable number of metrics, this relationship is accurate. It is crucial to remember that this statement shows that the error reduces with the square root of the trails, meaning that if we want to lower the error by a factor of 10, we will require, on average, 100 times more points.

3. CONCLUSION

For the given randomly generated 3D objects, this study approximated the occluded region in Euclidean space. To accomplish this, random points were produced and distributed to fill the volumes of all the 3D objects, then evaluation was done on each generated point to see if it fell within the volume of all provided items. Using two different random points sample categories, the occlusion estimation efficiency was assessed. Due to the randomness of the numbers, the simulation's outcomes vary from 0.06% to 0.068%.

4. ACKOWLEDGEMENT

Thanks, and appreciation to Prof. Chen Chunyi for his patience and offering of knowledge throughout the semester, which has been a great contributing factor for the completion of this study.

5. REFERENCES

Casella, C. R. (2013). *Monte Carlo statistical methods.* Springer Science & Business Media.

D. P. Kroese, T. B. (2014). Why the Monte Carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 386-392.

D. Wang, H. S. (2015). *Recognition and localization of occluded apples using K-means clustering algorithm and convex hull theory: a comparison.* Multimedia Tools and Applications.

Feigin, A. E. (2012). *"Phys 5870: Modern Computational Methods in Solids,".* Northeastern University.

G. Papaioannou, A. G. (2006). *Efficient occlusion culling using solid occluders.*

J. Xu, Y. M. (2019). *3D-GIoU: 3D Generalized Intersection over Union for Object Detection in Point Cloud.* Sensors (Basel).

Jana, D. P. (2010). *Computer Graphics: Algorithms and Implementations.* PHI Learning.

Mustapha Musa et al. (2020). 3D OBJECTS OCCLUSION ESTIMATION USING MONTE CARLO SIMULATION. *International Journal of Computer Science and Mobile Computing*, 55-64.

Reddy et al. (2019). Occlusion-Net: 2D/3D Occluded Keypoint Localization Using Graph Networks. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7318-7327.

Zelek, D. J. (2004). *Real-time tracking for visual interface applications in cluttered and occluding situations .* Image and Vision Computing.