Detection of Holes in Point Clouds Using Statistical Technique

Zain ul Abideen[1], Hamza Ali[1], Muhammad Sajjad[1], Muhammad Abeer Irfan[1], Atif Jan[2], Yasir Saleem[1]

[1]Departement of Computer Systems Engineering University of Engineering and Technology, Peshawar, Pakistan

[2]Department of Electrical Engineering University of Engineering and Technology, Peshawar, Pakistan

*Correspondence:zainikhan3434@gmail.com, hamzaali.dcse@gmail.com, muhammadsajad2710@gmail.com, abeer.irfan@uetpeshawar.edu.com, atifjan@uetpeshawar.edu.pk, yasirsaleem@uetpeshawar.edu.pk ,

Citation | Abideen. Z. U, Ali. H, Sajjad. M, Irfan. M. A, Jan. A, Saleem. Y, "Detection of Holes in Point Clouds Using Statistical Technique", IJIST, Special Issue pp 1-8, May 2024

Received | May 04, 2024, **Revised |** May 09, 2024, **Accepted |** May 15, 2024, **Published |** May 20, 2024.

A point cloud is a dynamic, three-dimensional geometric representation of data that has different qualities for every point, including geometry, normal vectors, and color. However, holes that often occur during the 3D point cloud collection process provide an immense obstruction to the analysis and reconstruction of point clouds. Thus, detecting these holes is a crucial initial step toward obtaining precise and comprehensive representations of the real surfaces. Although there are several methods available for hole detection and filling, the problem is exacerbated by their shortcomings, which include high computation complexity or limited effectiveness. Our method is based on a sequence of basic but efficient statistical techniques. Our method is based on a sequence of basic but efficient statistical techniques. First, we find the mean distances between each point using the K Nearest Neighbors (KNN) technique. Next, we can categorize normal points and points that belong to holes and borders by using this mean as a threshold. Our method's simplicity and low computational resource needs offer significant advantages over other approaches

Keywords: Point clouds; Hole detection; Surface reconstruction; Statistical analysis.

Introduction:

The evolution of laser scanning technology has democratized the process of digitizing objects, enabling precise and efficient data acquisition at scale. This advancement has catalyzed the emergence of reverse engineering; a process wherein digital models are generated from point cloud data as a valuable tool in various industries. A point cloud is a collection of points of data plotted in 3D space, using a 3D laser scanner. Such as, when scanning a building, every virtual point would correspond to a real spot on the metalwork, wall, window, or any other surface where the laser beam comes into contact. A point cloud can be defined as "A Point Cloud is an unordered set of 3-dimensional points in a frame of reference (Cartesian coordinate system) on the surface of objects." The point cloud is a set of discrete points obtained by scanning the object's surface with a three-dimensional (3D) scanning device. The point cloud can directly and stereoscopically represent the geometric features of the object's surface. Various additional details can be used along with 3D coordinates to represent them. A few are listed below:

- RGB color information associated with each point.
- Normal to surface at each point.
- Information about meshes (vertices & edges).

3D laser scanner technology is the foundation of 3D point cloud data. These electronic devices use visible light and lidar-based technology to gather detailed information about a specific area. The performance of the scanning device in use has a major impact on the final point cloud's clarity and accuracy. Nowadays, there are various sensors available for 3D scanning which are based on various principles some of which are mentioned below:

- Structured Light Cameras
- Time-of-flight sensors
- Stereo Vision
- Microsoft Kinect Cameras

Point cloud is of paramount importance in 3D object reconstruction, and object recognition with its high precision and fast processing speed [1][2]. It is also useful in Simultaneous Localization and Mapping (SLAM) and Preservation & restoration. It is possible to scan the complete surface of an object using high-fidelity scanners, but holes may appear in the final integrated model due to occlusions and maneuverability of the scanner while scanning. The holes in 3D objects as shown in Figure 1 may also occur due to factors such as occlusion, low surface reflectance coefficients, high grazing angles, missing parts in the original object, limited number of range scans from various viewing directions, laser range scanner functionality, etc.

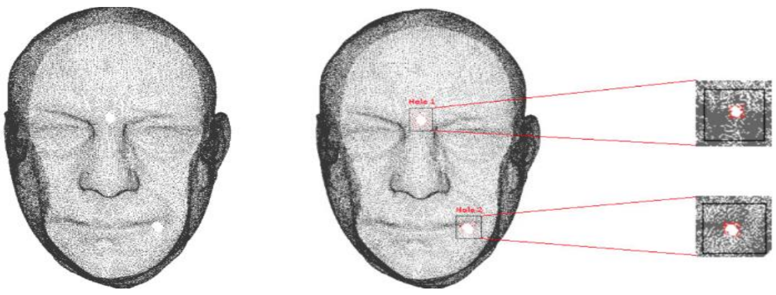


Figure 1: Holes in point clouds [3].

For example, some object portions may be absent because of accessibility or visibility issues, or because of unique physical characteristics of the scanned surface (transparency, reflectivity, etc.). This produces holes in the dataset, which do not correspond to any holes in the object. If the point cloud's boundary and hole points are not properly processed, it will influence subsequent research, such as point set simplification [4][5], point cloud registration [6], and hole repair [7]. It is worth mentioning that not all holes appearing in the estimated model are virtual, i.e. due to missing reconstructions of featureless surfaces [8]. A physical hole that is a part of the structure of the object being reconstructed may also be classified as a hole. Unfortunately, just looking at the point cloud, it is almost impossible to differentiate between these two types of holes (real and virtual). These real holes should be left as they are during the filling process. These days, a high-quality reconstruction of objects is an essential demand by many applications. Therefore, surface completion or hole-filling has grown in importance in the process of reconstructing 3D images. However, no hole filling can be applied without the detection of holes in point clouds. So, hole detection in the point cloud is an important component. Usually, 3D image reconstruction methods produce unstructured point clouds, in other words, the object's surface is not explicitly encoded with any connectivity information which makes the problem of the detection of holes on the surface an ill-defined problem [9]. In this paper, we introduce a statistical approach intended to identify holes within 3D point cloud data by using geometrical features in three-dimensional space. Our method relies on the principle of K-nearest neighbors (KNN) to identify neighboring points, followed by the computation of their average Euclidean distance. This average distance serves as a threshold, enabling the detection of irregularities in the point cloud. By comparing the distances between points, we can effectively pinpoint regions where data is missing, indicating the presence of holes.

Literature Review:

In this section, we study some existing methods for the detection of the hole boundary in the 3D point cloud. As we know the detection of point cloud boundaries and hole points is a vital task. In the state of the art of boundary extracting methods, different methods have been studied and developed to extract the boundary. These methods fall into two categories: (1) The methods executed on the grid; and (2) The methods executed directly on the data points of the point cloud. The former methods triangulate the point cloud and look for the adjacent triangular meshes. If there is no adjacent triangle, the associated triangle is considered as the boundary of the hole [4]. Authors in [10] subdivided the methods on meshes into volume-based methods and surface-based methods. In an early study [8], a seed boundary is selected on the point cloud grid to search for the next boundary according to the half-edge data structure. When the closed loop is reached to complete the boundary detection of holes, the search is finished. Many studies have recently focused on the methods of direct hole detection, which do not need to triangulate the point cloud in advance and can save a lot of time. In [9] the authors computed boundary probability for each point and classified the points in the point set into boundary or interior points through the application of the angle criterion. The coherence of the points is leveraged to extract a boundary loop that represents a simple hole boundary. As the point cloud is usually unstructured, therefore, to enhance the efficiency of neighborhood searches kD-tree, octree, or BSP-tree can be utilized. Along with the angle criterion authors in [9] also presented some other criteria to determine whether a point is a boundary point or not i.e. half disc criterion, shape criterion, and various enhancements based on these criteria. The author in [11], finds the neighborhood of each point in the point cloud and approximates the direction of the normal for detection of hole boundary in the point cloud. After finding the neighborhood of points, the author detected the candidate boundary points and created boundary polygons from boundary points. Following the detection of boundary points of the hole, an algebraic surface patch is fitted to the neighborhood and sample auxiliary points. Surfaces like spheres, cylinders, and planes are all recreated by this process. The author in [10] extracted the polygonal hole boundary with the Mean Shift approach to find out the vertices that have similar geometry properties with the hole region in the neighborhood of the boundary.

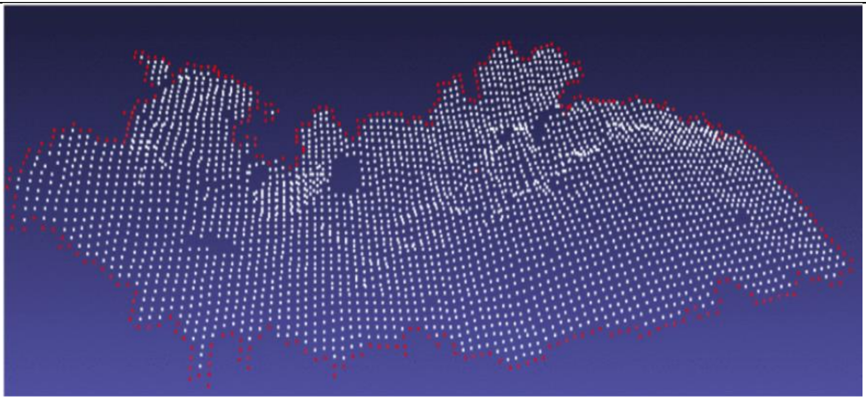


Figure 2: Detection of the surficial boundary of the point cloud described in [12].

Authors in [13] use the Distance-Centroid technique, which employs maximum squared distance and K-Nearest neighbors to detect the boundary points. The points that do not belong in the hole boundary are filtered out using Statistical Outlier Removal (SoR). To cluster out the different holes, the author used region-grow segmentation. The author in [14] proposed a method based on triangular mesh models which aims to find solid holes in 3D models. The author grouped the interconnected coplanar triangles and extracted the contour of the model using the boundaries of the nearby planes. Then, the author uses the extracted contour to form several disjoint clusters of model vertices and detect holes by analyzing the relationship between the clusters and plane. The author in [15] also uses the triangular mesh for the detection of holes in a point cloud. The author uses boundary edge tracking to automatically identify holes, an edge is a boundary edge if it is part of only one triangle; otherwise, it is an interconnected edge that is part of multiple triangles. Authors in [12] extracted exterior boundary points of the surface S and further classified them into exterior boundary points and inter points. Then looping is done for all points of S, if each of them is not an exterior boundary point, and one of its neighbors (in 4-connectivity) is empty, it is determined as a boundary point of a hole. The boundaries detected by the author are shown in Figure 2. The methods for boundary extraction based on the computation of convex hull have also been widely applied. The author in [16] proposed a method that involves modifying the convex hull to detect the boundary of the point cloud. For every point on the boundary p_b, the computation is repeated to calculate the smallest angle between p_b and its neighboring points for finding the next boundary points, but this method is time-consuming. The author in [17] computed the convex hull of a group of points, starting from an initial core point and its neighbors, and referred to it as a cluster. Then, by repeating the procedure and using the convex hulls of the core points as new core points, this cluster is extended repeatedly until each convex hull of a core point is joined with the cluster's main body. When all exterior points have been identified and the final boundary has been reached, clusters will not be combined or expanded further. The author in [18] proposed a method to detect the boundary of a surface of 3D points set. The author triangulated the convex hull C of the surface first and C is then optimized by computing and removing some outward triangles to obtain the exterior boundary of the surface. The author in [19] introduces a new method for identifying and fixing holes in point clouds. The author first calculated the density and 2D projection points of the point cloud. Then, the author used Euclidean distance to find the boundary points of the holes. It was followed by a preprocessing segmentation algorithm that helped to identify the boundary points more accurately. However, the author failed to detect the boundary of the whole point cloud simultaneously, so the method failed to detect the holes at boundary points.

Methodology:

Our proposed methodology consists of two steps first we find K-Nearest neighbors and then we calculate Euclidean distances. These two steps are employed to compare each point within the point cloud. A threshold is settled that facilitates the identification of boundary and hole points. If the distance between points exceeds the predefined threshold, then the point is a potential boundary point, and if it is not then it is a normal point. The threshold has been set as the mean of measured distances between points in the point cloud.

Dataset:

We have used the Kaggle digit dataset the "3D MNIST Dataset". The dataset includes 3D point clouds, which are sets of (x, y, z) coordinates produced from about 5,000 images in the original 2D MNIST dataset [20]. The point clouds' maximum dimension range is one, and their mean is zero.

K-Nearest Neighbors (KNN):

K-nearest neighbors refer to k () points close to the query point in the spatial distance. K-nearest neighbors search can be described as; given a point set S containing n points, for query point p_i ($p_i \in S$) there is a subset set C containing k points (not including point p_i), $C \in S$, $k < n$ and for any $p_1 \in C$, $p_2 \in S - C$, the following equation 1 can be met [21].

$$\text{dist}\,(p_i, p_1) \leq \text{dist}\,(p_i, p_2) \qquad (1)$$

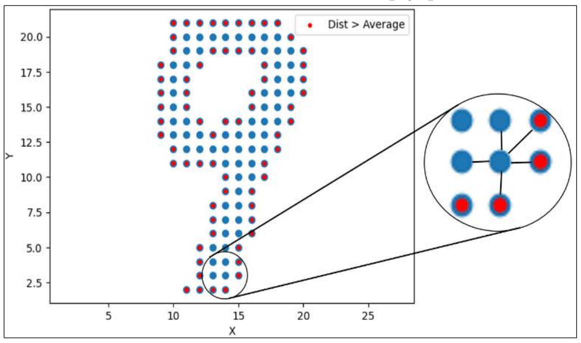


Figure 3: k-nearest neighbor in two-dimension

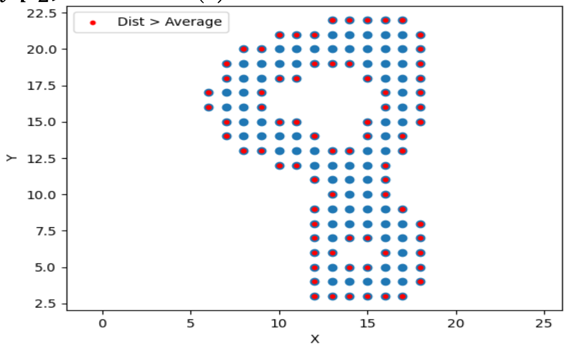


Figure 4: Boundary detection in the Kaggle digit

The k-nearest neighbors in two dimensions are shown in Figure 3, where the points are distributed in the two-dimensional space dispersedly, p is a query point and five points in the circle are the k (k=5) nearest neighbors. When the number of points is small, k-nearest neighbors can be obtained by calculating the query point to all other points' Euclidean distance directly, sorting the distance values, and then taking the first k values as the k-nearest neighbors [21].

Euclidean Distance:

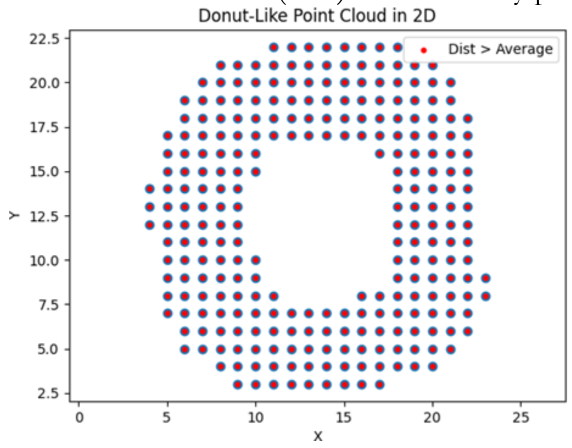
In our approach, we used KNN (K-Nearest Neighbors) for selecting the number of neighbors based on their Euclidean distances in 3D space. Next, we compute the distance between each point of the point cloud, identifying points with distances greater than this threshold as potential hole boundary points or point cloud boundaries. This process not only allows for effective hole detection but also provides insights into the overall structure and boundary of the point cloud. The Euclidean distances for 3-dimensional space are calculated in Equation 2.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2} \quad (2)$$

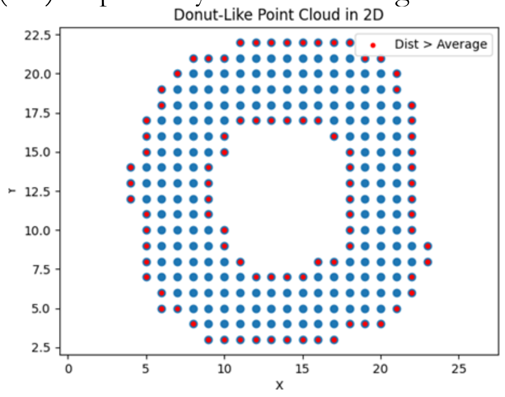
Euclidean distance is applied to express the distance between two points on account of its convenience and simplicity of use. Moreover, it offers the advantage of requiring less computational power compared to more complex learning algorithms. We calculated the mean of the point cloud points while considering the mean of the K Nearest Neighbors (KNN). This involves computing distances between the points using KNN and then aggregating the mean distances of the neighbors. Subsequently, the average of these mean distances serves as the threshold, effectively separating points as hole or boundary points and normal points.

Results and Discussion:

As previously discussed, various approaches have been employed for hole detection, each with its own set of advantages and limitations. However, our approach, as detailed in this paper, takes a unique path toward hole detection, primarily relying on statistical techniques such as Euclidean distances. The Euclidian distances between neighboring points are given in Table 1. The distances between 5 points concerning each other are calculated using equation 2 and are recorded in the table. A small value of Euclidean distance between the points means the points are near to each other. For example, the distance between N1 and N5 is 0.11 and the distance between N1 and N3 is 0.22 meaning that N5 is nearest to N1 as compared to N3. The results obtained from this technique have proven to be highly satisfactory, as illustrated in Figure 5. Through this simple approach, we have achieved effective hole detection and boundary delineation within the point cloud. In Figure 4 our proposed method is applied to the Kaggle digit dataset and the points in the point cloud have been classified. The blue points show normal points while the red points are considered boundary points. In Figure 5a donut-like point cloud is shown. Before, all the points in the donut are red after looping through our method the points are classified as normal (blue) and boundary points (red) respectively as shown in Figure 5b.



(a) Input Point cloud



(b) Point cloud with detected boundaries

Figure 5: Hole boundary detection in point cloud

Table 1: Euclidean distances between neighboring points (k = 5)

5- Nearest Neighbors	N1	N2	N3	N4	N5
N1	0	0.12	0.22	0.18	0.11
N2	0.12	0	0.13	0.17	0.14
N3	0.22	0.13	0	0.15	0.16
N4	0.18	0.17	0.15	0	0.15
N5	0.11	0.14	0.11	0.15	0

Conclusion:

Hole detection in point clouds is a crucial initial step toward obtaining precise and comprehensive representations of the real surfaces. Our technique for the detection of holes is simple but effective and it takes a unique path towards hole detection, primarily relying on statistical techniques such as Euclidean distances and K Nearest Neighbors (KNN). For now, our method is limited to the detection of hole boundaries and point cloud boundaries in surficial point clouds, but in future work, we go on to modify this statistical technique and make it robust for the accurate detection of holes in complex point clouds.

References:

[1] Y. Guo, F. Sohel, M. Bennamoun, J. Wan, and M. Lu, "A novel local surface feature for 3D object recognition under clutter and occlusion," Inf. Sci. (Ny)., vol. 293, pp. 196–213, Feb. 2015, doi: 10.1016/J.INS.2014.09.015.

[2] G. Sansoni, M. Trebeschi, and F. Docchio, "State-of-The-Art and Applications of 3D Imaging Sensors in Industry, Cultural Heritage, Medicine, and Criminal Investigation," Sensors 2009, Vol. 9, Pages 568-601, vol. 9, no. 1, pp. 568–601, Jan. 2009, doi: 10.3390/S90100568.

[3] R. A. Tabib et al., "Learning-Based Hole Detection in 3D Point Cloud Towards Hole Filling," Procedia Comput. Sci., vol. 171, pp. 475–482, Jan. 2020, doi: 10.1016/J.PROCS.2020.04.050.

[4] H. Song and H. Y. Feng, "A progressive point cloud simplification algorithm with preserved sharp edge data," Int. J. Adv. Manuf. Technol., vol. 45, no. 5–6, pp. 583–592, Nov. 2009, doi: 10.1007/S00170-009-1980-4/METRICS.

[5] H. Han, X. Han, F. Sun, and C. Huang, "Point cloud simplification with preserved edge based on normal vector," Opt. - Int. J. Light Electron Opt., vol. 126, no. 19, pp. 2157–2162, Oct. 2015, doi: 10.1016/J.IJLEO.2015.05.092.

[6] J. Yang, Z. Cao, and Q. Zhang, "A fast and robust local descriptor for 3D point cloud registration," Inf. Sci. (Ny)., vol. 346–347, pp. 163–179, Jun. 2016, doi: 10.1016/J.INS.2016.01.095.

[7] Y. Quinsat and C. lartigue, "Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics," Int. J. Adv. Manuf. Technol., vol. 81, no. 1–4, pp. 411–421, Oct. 2015, doi: 10.1007/S00170-015-7185-0/METRICS.

[8] O. H. Nader H. Aldeeb, "Detection and Classification of Holes in Point Clouds", [Online]. Available: https://www.semanticscholar.org/paper/Detection-and-Classification-of-Holes-in-Point-Aldeeb-Hellwich/2feb6ee457c21a3565763fa4b472cd6164a38d1e

[9] "Detecting holes in point set surfaces".

[10] Q. He, S. Zhang, X. Bai, and X. Zhang, "Hole filling based on local surface approximation," ICCASM 2010 - 2010 Int. Conf. Comput. Appl. Syst. Model. Proc., vol. 3, 2010, doi: 10.1109/ICCASM.2010.5620018.

[11] P. Chalmovianský and B. Jüttler, "Filling Holes in Point Clouds," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2768, pp. 196–212, 2003, doi: 10.1007/978-3-540-39422-8_14.

[12] V. S. Nguyen, T. H. Trinh, and M. H. Tran, "Hole Boundary Detection of a Surface of 3D Point Clouds," Proc. - 2015 Int. Conf. Adv. Comput. Appl. ACOMP 2015, pp. 124–129, Feb. 2016, doi: 10.1109/ACOMP.2015.12.

[13] "Automatic Hole Detection for Selective Hole Filling in Point Cloud Data - Results." Accessed: May 05, 2024. [Online]. Available: https://www.researchgate.net/publication/316975536_Automatic_Hole_Detection_for_Selective_Hole_Filling_in_Point_Cloud_Data_-_Results

[14] Y. Wang, R. Liu, F. Li, S. Endo, T. Baba, and Y. Uehara, "AN EFFECTIVE HOLE DETECTION METHOD FOR 3D MODELS".

[15] J. Wang and M. M. Oliveira, "Filling holes on locally smooth surfaces reconstructed from point clouds," Image Vis. Comput., vol. 25, no. 1, pp. 103–113, Jan. 2007, doi: 10.1016/J.IMAVIS.2005.12.006.

[16] A. Sampath and J. Shan, "Building boundary tracing and regularization from airborne lidar point clouds," Photogramm. Eng. Remote Sensing, vol. 73, no. 7, pp. 805–812, 2007, doi: 10.14358/PERS.73.7.805.

[17] S. Suer, S. Kockara, and M. Mete, "An improved border detection in dermoscopy images for density based clustering," BMC Bioinformatics, vol. 12, no. SUPPL. 10, pp. 1–10, Oct. 2011, doi: 10.1186/1471-2105-12-S10-S12/FIGURES/9.

[18] G. J. HUANG Xianfeng, CHENG Xiaoguang, ZHANG Fan, "SIDE RATIO CONSTRAIN BASED PRECISE BOUNDARY TRACING ALGORITHM FOR DISCRETE POINT CLOUDS", [Online]. Available: https://www.isprs.org/proceedings/xxxvii/congress/3b_pdf/69.pdf

[19] C. Zhang, H. Zhou, and J. Duan, "A method for identifying and repairing holes on the surface of unorganized point cloud," Measurement, vol. 210, p. 112575, Mar. 2023, doi:

10.1016/J.MEASUREMENT.2023.112575.

[20] "MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges." Accessed: May 05, 2024. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[21] D. Li and A. Wang, "Improved KNN algorithm for scattered point cloud," Proc. 2017 IEEE 2nd Adv. Inf. Technol. Electron. Autom. Control Conf. IAEAC 2017, pp. 1865–1869, Sep. 2017, doi: 10.1109/IAEAC.2017.8054336.