IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 40, No. 4, August, 2023, pp. 1377-1384
Journal homepage: http://iieta.org/journals/ts

Real-Time Hole-Filling in Mobile Augmented Reality Gaming: A Novel Algorithm to Overcome Depth Sensor Limitations

Do Heon Choi[1], Seok-Kyoo Kim[1], SeongKi Kim[2*]

[1] Department of Game Design and Development, Sangmyung University, 20, Hongjimun 2-gil, Jongno-gu, Seoul 03016, Korea
[2] National Center of Excellence in Software, Sangmyung University, 20, Hongjmun 2-gil, Jongno-gu, Seoul 03016, Korea

Corresponding Author Email: skkim9226@smu.ac.kr

https://doi.org/10.18280/ts.400407

Received: 22 February 2023
Revised: 16 July 2023
Accepted: 28 July 2023
Available online: 31 August 2023

Keywords:
AR, Hole-Filling algorithm, mobile gaming, real-time processing, AR game development

ABSTRACT

In the realm of Augmented Reality (AR) within mobile gaming, the planes recognized by depth sensors delineate the space for content implementation, thereby constraining the scope of representation. Errors in these recognized planes may inhibit the progression of mobile AR. A method is explored that utilizes 'Meshing' facilitated by Unity, generating meshes corresponding to physical space and enabling expansion of content space beyond mere planes. This approach, although promising, is contingent on the depth sensor, leading to the creation of holes beyond the sensor's reach. These holes present a critical issue, allowing game objects to escape. To address this challenge, an algorithm is proposed that consists of two main components: 'Hole-Finding' and 'Hole-Filling'. In 'Hole-Finding', real holes are identified by the calculation of the direction of each loop. Subsequently, 'Hole-Filling' computes the centroid-vertex of each hole and employs it for the hole-filling process. A real-time hole-filling performance with only a 7 μsec degradation was observed, heralding a significant step towards mitigating this problem within AR content. This investigation contributes a novel solution to a crucial technical obstacle, thereby enhancing the functionality and potential of AR in mobile gaming.

1. INTRODUCTION

With the advancement of AR, a significant enhancement has been observed in various domains, notably including the mobile gaming sector. Initially, early mobile AR games were developed to render 3D objects on designated markers, serving as reference coordinates [1, 2]. Subsequent endeavors shifted towards the utilization of a Simultaneous Localization and Mapping (SLAM)-based markerless approach [3, 4]. A notable landmark was achieved with the release of 'Pokemon Go', a markerless game grounded in location-based information, bringing AR games to public attention [3, 5]. Subsequently, ARKit and ARCore, frameworks for AR applications, were unveiled by Apple and Google respectively [6, 7], and applied to 'Pokemon Go', manifesting more realistic AR experiences. Despite this progress, no mobile AR game has eclipsed 'Pokemon Go', and the repetitive pattern exhibited by most mobile AR games is identified as a likely factor for market stagnation [8].

Currently, the deployment of mobile AR games is constrained to planes recognized by the sensor, further hampered by the performance limitations of mobile devices. Inaccuracies in object placement within real space owing to these constraints yield unnatural experiences, undermining immersion [8, 9]. Efforts to address these limitations have been undertaken [10, 11]; improvements in plane recognition have been achieved in accordance with the user's posture [10], while play spaces limited to planes have been extended to rectangular parallelepipeds [11]. Yet, constraints persist, restricting play space to specific areas. These hindrances are cited as factors that have precluded the diversification of mobile AR games, leading to the failure of titles emulating 'Pokemon Go'.

Recent innovations by Unity, including the 'AR Foundation', enabling the convergence of ARKit and ARCore within a unified framework, have offered prospects for further enhancements. Specifically, the introduction of 'Meshing,' which generates meshes corresponding to physical space using sensor data, has opened avenues to transcend play spaces confined to planes. However, this approach has been challenged by the potential generation of holes where meshes are unattainable by the sensor, giving rise to serious gameplay implications. Such holes can induce abrupt disappearances or escape of game objects, disrupting player engagement [8, 9]. Without addressing this issue, the utilization of 'Meshing' in game development may lead to negative gameplay experiences and constrict the expressive potential of developers.

The objective of this study is the exploration of methods to mitigate the hole problem through the generation of supplemental meshes capable of filling holes. A centroid-based method, originally conceived for additive manufacturing repair, is introduced, contrasted, and analyzed against existing 'Meshing' approaches. The suitability of the proposed method, with an emphasis on maintaining real-time performance within AR, is validated. The remainder of this paper is organized as follows: Section 2 provides an overview of related works; Section 3 elaborates on the ‘Hole-Filling algorithm’; Section 4 offers comparisons between the results obtained through 'Meshing' and those achieved via the proposed ‘Hole-Filling algorithm’, examining the impact on game performance; and Section 5 concludes the paper.

2. RELATED WORK

With the progression of 3D scanning technology, significant investigation into Hole-Filling algorithms has been conducted to remedy the occurrence of holes generated by diverse causes within 3D models. These algorithms are primarily bifurcated into two major categories, namely 'Point Cloud-based methods' and 'Mesh-based methods' as shown in Figure 1 [12].

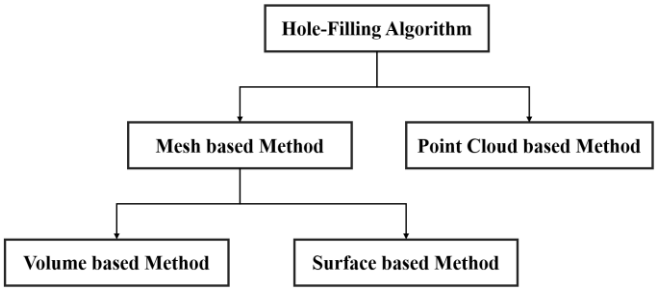


Figure 1. Classification of 'Hole-Filling algorithm'

Initially, the focus was directed towards 'Point Cloud-based methods', where raw point data of scanned objects were utilized as input. Subsequent advancements in computer graphics instigated a shift towards 'Mesh-based methods', which became predominant. These methods largely accept meshes, comprised of vertices and triangles, as input and are further categorized into 'Volume-based methods' and 'Surface-based methods'.

Within 'Volume-based methods', the input mesh is transformed into volume representation, followed by the application of algorithms such as Volumetric Diffusion, Octree Grid, and Oriented Voxel Diffusion, to ensure reconstruction without defects [12, 13]. In contrast, 'Surface-based methods' operate solely on the input hole's information or its surrounding information to fashion patches for the hole. The trajectory of research in this area has witnessed multiple developments.

Initially, Peter [14] devised a technique to fill the holes by creating new triangle meshes and modifying them to align with the surrounding density. This was followed by smoothing to produce a more even surface. Xia and Zhang [15] introduced an innovative method to fill holes through curve and corner splitting. Through this method, feature points were extracted via Euler spirals and quadratic equation optimization from the segmented holes. The recovery of feature lines with these points and subsequent hole-filling was achieved using Advancing Front Methods (AFM), characterized by the application of triangular meshes within specific spaces.

Feng et al. [16] proposed a classification of holes based on their size (small, medium, large) determined by the number of vertices, with each category being addressed through tailored techniques. An alternative approach was introduced by Park et al. [17], which focused on the creation of patches that precisely fit various holes. By utilizing a process involving triangulation, refinement, fairing, and smoothing, source and target patches were generated, followed by a measurement of shape differences. The blending process facilitated the creation of patches with pronounced features, enabling the seamless filling of holes.

It has been observed that 'Surface-based methods' are typically less noisy and faster in comparison to 'Volume-based methods' that necessitate comprehensive reconstruction. Considering the emphasis on high performance within AR, 'Surface-based methods' were selected for this study. Despite the efficiency of surface-based methods, their real-time application in dynamic environments such as games remains a challenge, indicating a need for a more lightweight approach. This necessity spurred the design of a method tailored for small-sized holes.

3. PROPOSED ALGORITHM

As delineated in Section 2, the prevailing studies predominantly assess the fidelity of object restoration to the original form and commonly require hundreds of milliseconds to fill most holes. Given the imperative need for high performance in Hole-Filling within the context of 'AR Foundation' Meshing, employed within games or simulations, the design of a real-time structure was deemed necessary. The proposed method, as illustrated in Figure 2, was conceived to address these demands.

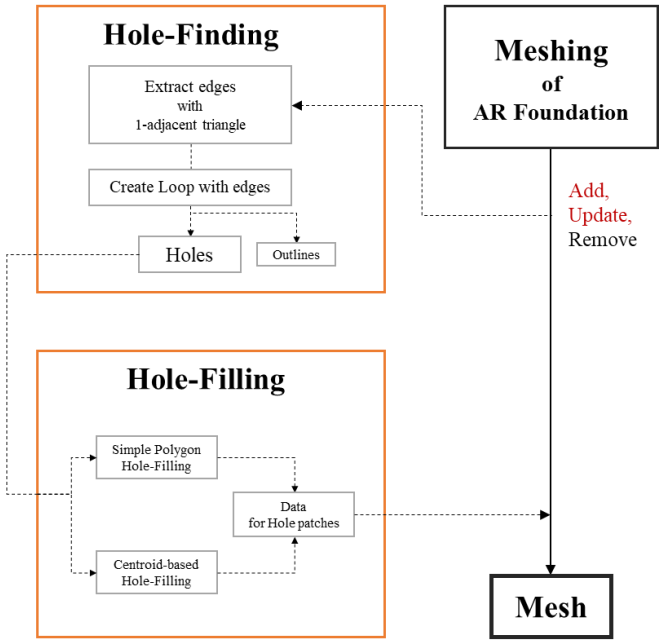


Figure 2. The structure of the proposed method

Within the framework outlined in Figure 2, the 'Meshing' of 'AR Foundation' is responsible for adding, updating, and removing meshes in accordance with data transmitted from sensors. The proposed method is activated when 'Meshing' either adds or updates meshes. Initially, 'Hole-Finding' identifies holes via the corresponding mesh's vertices and triangles and forwards the hole data to 'Hole-Filling'. Subsequently, 'Hole-Filling' determines the vertices and triangles of patches required to fill holes, supplements the original data, and updates the mesh. The resultant structure, depicted in Figure 2, facilitates the generation of new vertices and triangles to fill the holes.

3.1 Hole-Finding

The utilization of depth sensors can lead to the formation of holes, exemplified in Figure 3.

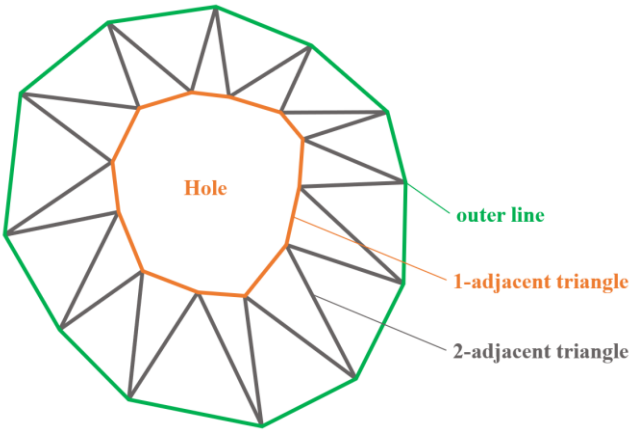
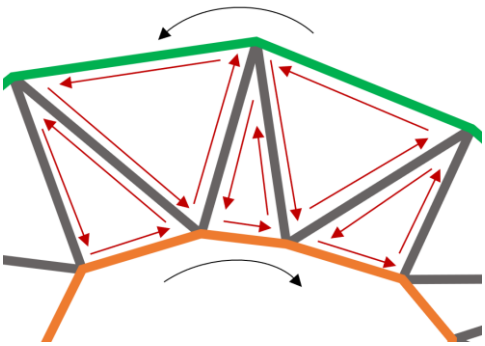


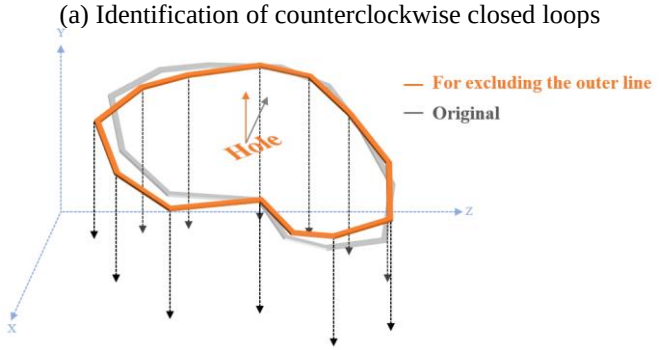
Figure 3. Generated mesh with a hole

As illustrated in Figure 3, the perimeter of the hole is defined by the presence of only a single adjacent triangle. The execution of 'Hole-Finding' is accomplished by identifying the edges associated with a singular triangle and subsequently recognizing holes by connecting these edges, leveraging the inherent closed-loop structure of the holes.

An inherent challenge in this method is the possible generation of an outer line that is uncharacteristic of standard 3D models. This outer line of the mesh is also identified as a closed loop, as depicted in Figure 3. In this study, the counterclockwise closed loop was excluded from the compilation of closed loops to eliminate the outer line from the holes, as demonstrated in Figure 4.



(a) Identification of counterclockwise closed loops



(b) Modification of the vertices of the boundary

Figure 4. Methods for finding outer lines

As indicated in Figure 4(a), the holes are oriented clockwise, whereas the outer lines are counterclockwise due to the partial shaping of the generated 3D models. When calculating the direction of closed loops, the vectors of the boundaries were rotated to align the normal vectors with the Y axis, as portrayed in Figure 4(b). Following the projection of the rotated vectors onto the XZ plane, the direction of the closed loop was ascertained using a vertex with an angle exceeding 180 degrees but falling short of 360 degrees.

3.2 Hole-Filling

The architecture of 'Hole-Filling', delineated in Figure 5, is bifurcated into two fundamental components: 'Simple Hole-Filling' and 'Centroid-based Hole-Filling'.

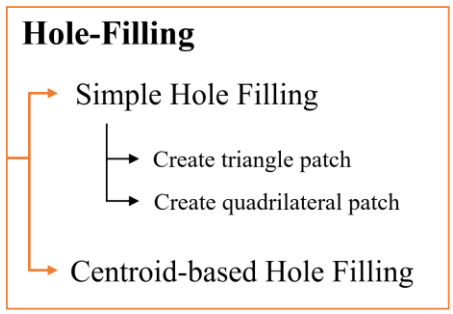


Figure 5. Hole-Filling structure

In the process of filling the detected holes, as characterized in Subsection 3.1, the distinction between the two methods becomes salient.

3.2.1 Simple Hole-Filling

The operation of 'Simple Hole-Filling' is invoked when the boundary of a hole is constituted by either three or four vertices. In these instances, the method entails connecting the vertices of the boundary to one another, circumventing the need to calculate the central vertex. However, if the process involves the creation of a quadrilateral patch and a reflex angle is encountered, the vertex associated with the reflex angle is first connected to the opposing vertex to form a triangle, as depicted in Figure 6.

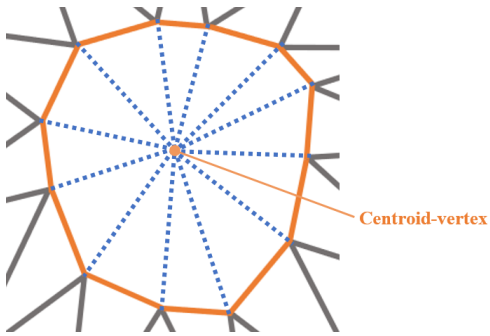


Figure 6. Centroid-based Hole-Filling

3.2.2 Centroid-based Hole-Filling

When the boundary of a hole consists of more than four vertices, the centroid-based algorithm is implemented. This algorithm is esteemed for its compatibility with real-time games, owing to its capability to function in real-time [16].

As shown in Figure 6, the process of Centroid-based Hole-Filling involves the calculation of the centroid vertex of the vertices of each hole. Subsequently, the vertices for the new triangles are constructed, and both the centroid vertex and the new triangles are updated in the original meshes. Though multiple approaches exist for calculating the centroid vertex, in this study, the mean value of the vertices defining each hole was utilized, a decision motivated by the desire to minimize

computational cost.

3.3 Algorithm details

The synthesis of the 'Hole-Finding' and 'Hole-Filling' techniques culminated in the development of Algorithm 1, a distinct pseudo-code representing the proposed method. This algorithm encapsulates a procedural sequence that delineates the mechanism for restoring object integrity within the mesh structure.

Algorithm 1. Hole-Filling algorithm

Input: vertices, triangles of original mesh
Output: vertices, triangles of updated mesh
1. foreach triangle in triangles of original mesh
2. Extraction of edges with one adjacent triangle
3. end
4. foreach edge in edges with one adjacent triangle
5. Reconfigure to be closed loops (v_0, v_1, … , v_n)
6. end
7. foreach loop in closed loops
8. Rotate & Project the loop
9. Calculate the direction of the loop
10. end
11. Remove the opposite loop
12. if the number of found vertices > 4
13. Compute the centroid vertex $v_{centroid}$ from (v_0, v_1, … , v_n)
14. for (v_0, v_1, … , v_n) do
15. Create new triangle including $v_{centroid}$
16. end
17. else
18. if the number of found vertices == 3
19. Create new triangle
20. if the number of found vertices == 4
21. for (v_0, v_1, … , v_n) do
22. Check the reflex angle
23. end
24. if reflex angle
25. Create new triangles (v_{r-1}, v_r, v_{r+2}, v_{r+2}, v_r , v_{r+1})
26. if not reflex angle
27. Create new triangles (v_0, v_1, v_2, v_2, v_1, v_3)

Algorithm 1 represents the pseudo-code of the proposed method. In this construct, lines 1 to 3 are dedicated to the identification of boundary edges, lines 4 to 6 function to connect these edges in order to detect holes, lines 7 to 11 operate to exclude outer lines from the detected holes, lines 12 to 16 facilitate the implementation of 'Centroid-based Hole-Filling', and lines 17 to 27 govern the application of 'Simple Hole-Filling'.

4. RESULT

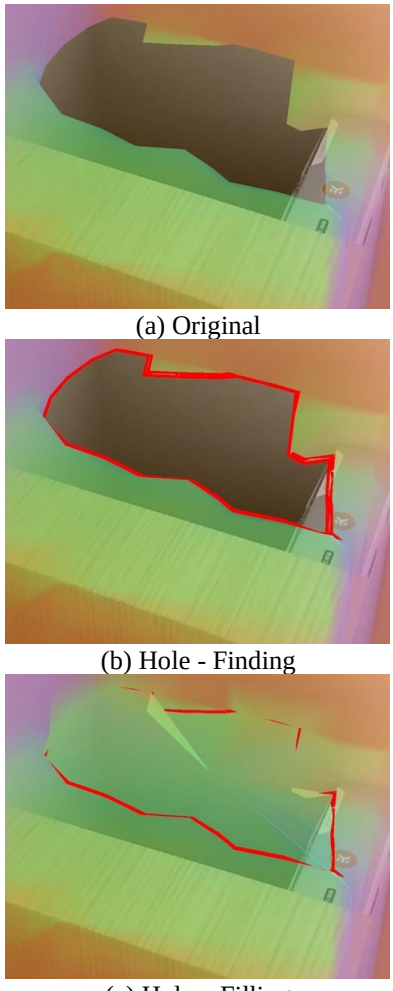
This section elucidates the outcomes, inclusive of the depiction of holes and the application of the Hole-Filling algorithm to rectify them. The efficacy of the procedure is assessed both in terms of performance metrics and its applicability within a gaming context.

4.1 Performance metrics

The validation of Algorithm 1 was conducted utilizing Unity 2022.2.0b8 and AR Foundation 5.0.2. An 'iPad Pro 4th generation' equipped with 'Meshing' support was selected as the mobile device for this assessment.

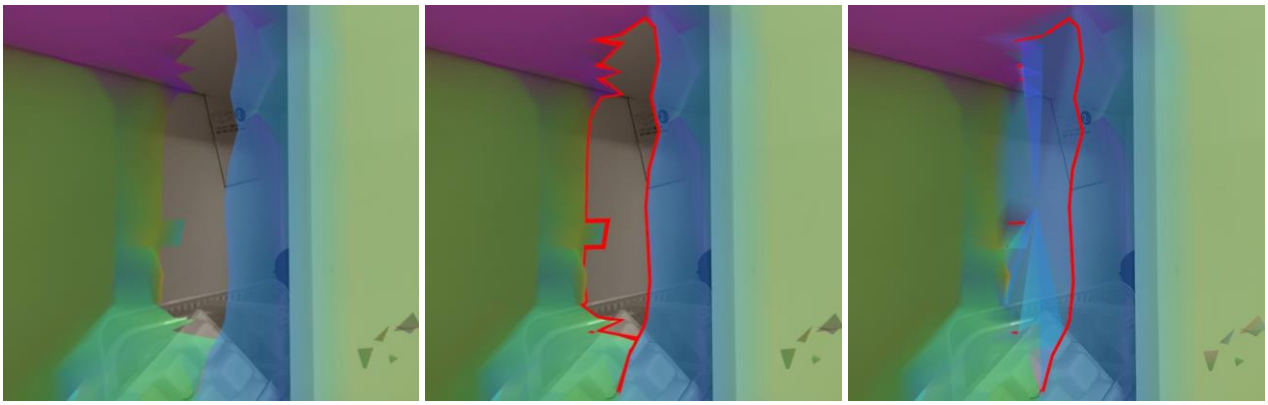
Initially, a comparative analysis was executed between the generated results and the outcomes obtained post 'Hole-Filling'. A locale exhibiting the sensor's constraints was selected for this comparison. Figure 7(a) displays a blind spot, signifying a sensor limitation when the native 'Meshing' is utilized against the space. This spot is noteworthy as it inhibits mesh generation. In contrast, as seen in Figure 7(b), the edges of the holes are identifiable (highlighted within the red box) and subsequently filled, as indicated by the red box in Figure 7(c). Figure 8 furnishes further examples, with the original meshes portrayed on the left, the 'Hole-Finding' outcomes in the center, and the 'Hole-Filling' results on the right.

Subsequently, the proposed Hole-Filling algorithm's performance was juxtaposed against existing Hole-Filling algorithms [14, 17], building upon the recommended 'Hole-Finding'. Each of the compared algorithms incorporated a refinement function to enhance quality post hole-filling. The same space was scanned 10 times over a duration of 5 minutes to gauge the minimum elapsed time, maximum elapsed time, and average elapsed time. The term "elapsed time" refers to the duration required by the Hole-Filling algorithm to mend the holes present within the meshes, as updated by 'Meshing' in real-time. The results are delineated in Table 1. Although the minimum method time was observed to be relatively constant across methods, the proposed method's maximum elapsed time was discerned to be up to 1100 times swifter.

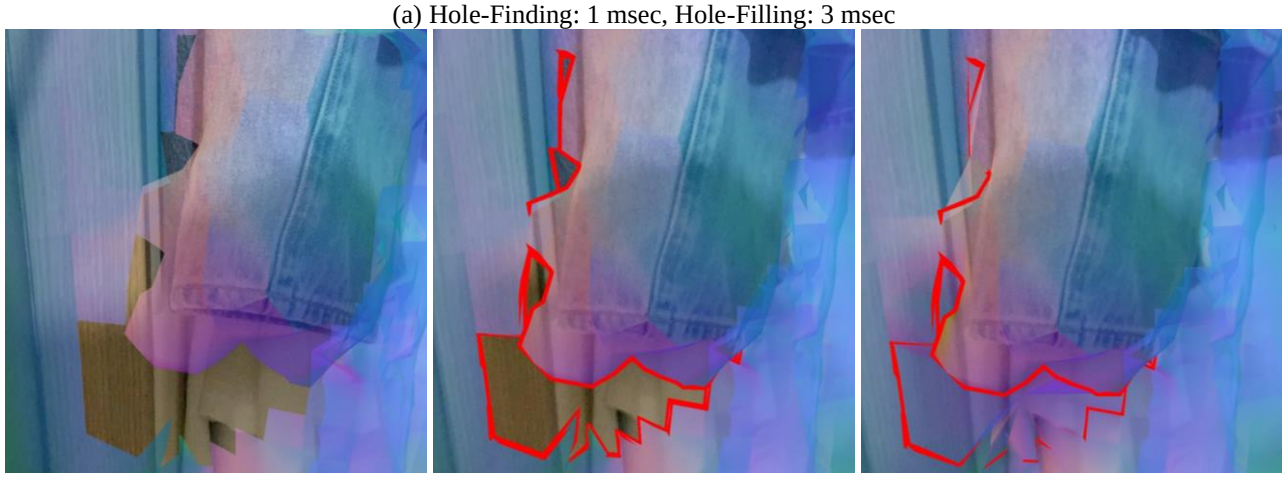


(a) Original
(b) Hole - Finding
(c) Hole – Filling

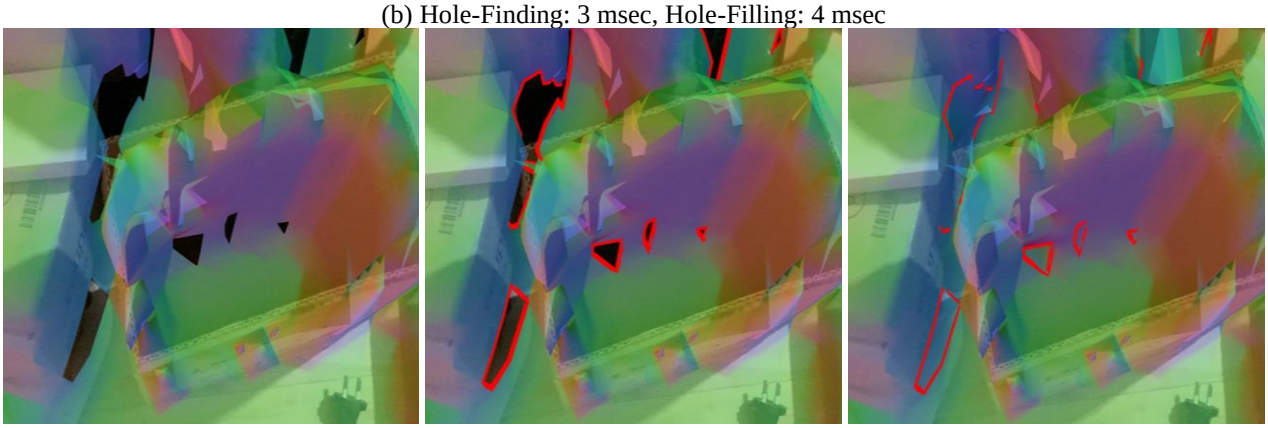
Figure 7. Depiction of a blind spot unreachable within the mobile device



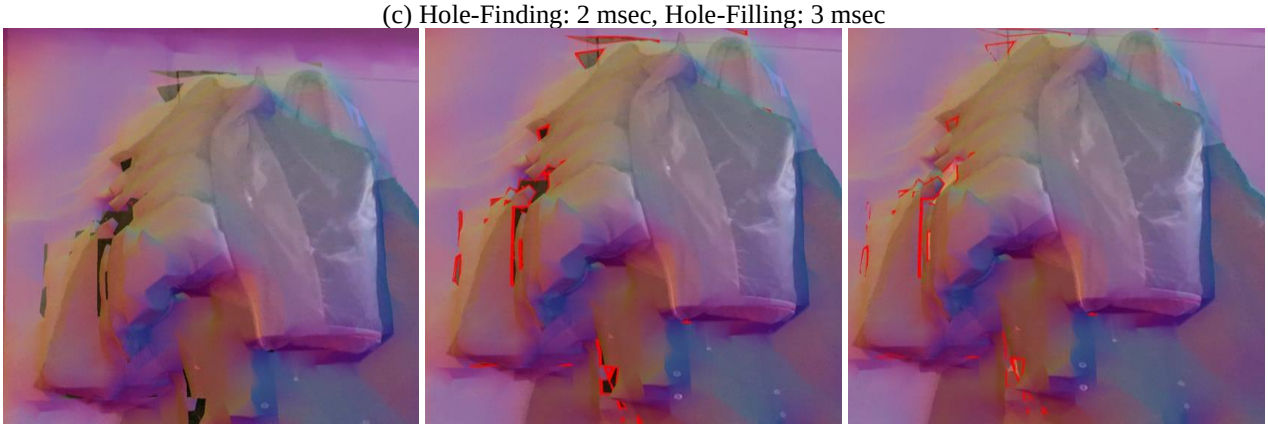
(a) Hole-Finding: 1 msec, Hole-Filling: 3 msec



(b) Hole-Finding: 3 msec, Hole-Filling: 4 msec



(c) Hole-Finding: 2 msec, Hole-Filling: 3 msec



(d) Hole-Finding: 2 msec, Hole-Filling: 3 msec

Figure 8. 'Hole-Finding' and 'Hole-Filling' in different situations

The heightened performance exhibited by the proposed method is attributed to the avoidance of existing algorithms known for their cubic-time complexity, which are conventionally employed to attain elevated quality. This deviation from conventional methodology underpins the accelerated operation of the proposed algorithm, highlighting its viability for real-time applications.

Table 1. Performance metrics of the proposed method

Unit: µsec	Minimum	Maximum	Average
Proposed Method	1	112	7
Peter [14]	1	123205	6879
Park et al. [17]	1	10507	507

4.2 Application in a game environment using the proposed method

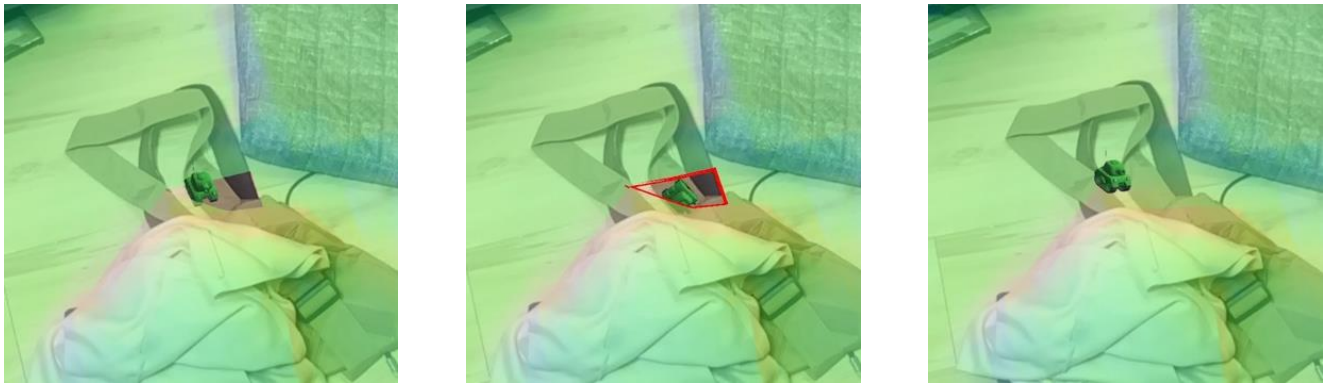
This subsection elucidates an application of the proposed method within a gaming context.

As demonstrated in Figure 9, the rapid hole-filling enabled by the proposed algorithm effectively addresses a critical problem in game development: the escape of game objects through undetected holes. A visual representation of this issue, as well as the method's resolution, is documented in a video available at https://youtu.be/eZw8CK49MUQ.

Figure 10 further delineates this process. The sub-figures on the left side display the game environment with original meshes, while the center sub-figures emphasize the hole boundaries with red lines. The right sub-figures portray a game environment that has been enhanced by the implementation of the proposed algorithm.

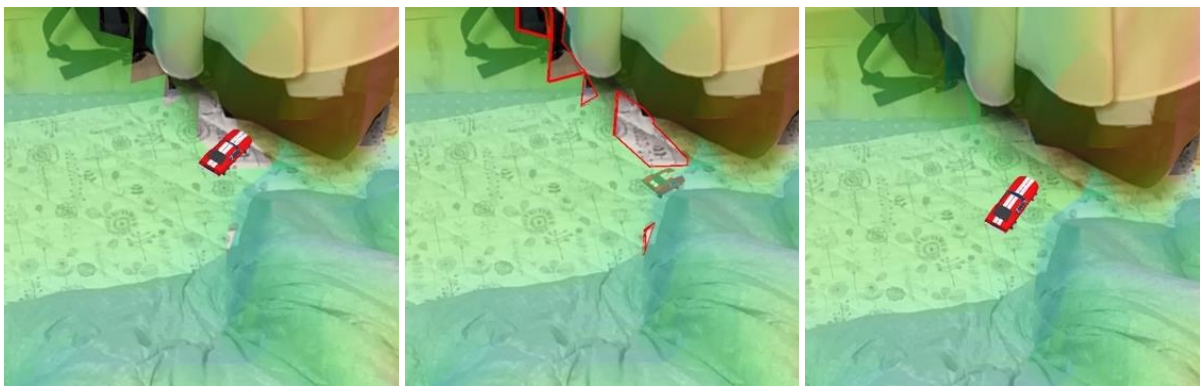
The ingenuity of the proposed algorithm lies in its capacity to fill holes swiftly, thereby preventing potential disruptions within the game environment. Its application not only rectifies visual inaccuracies but also ensures the integrity of the gaming experience. The rapid identification and subsequent rectification of holes are imperative for maintaining a seamless gaming interface.

The practicality of the proposed algorithm in real-world gaming scenarios illustrates its robustness and adaptability. By mitigating a common challenge in gaming development, this method showcases potential for further integration into various virtual environments.



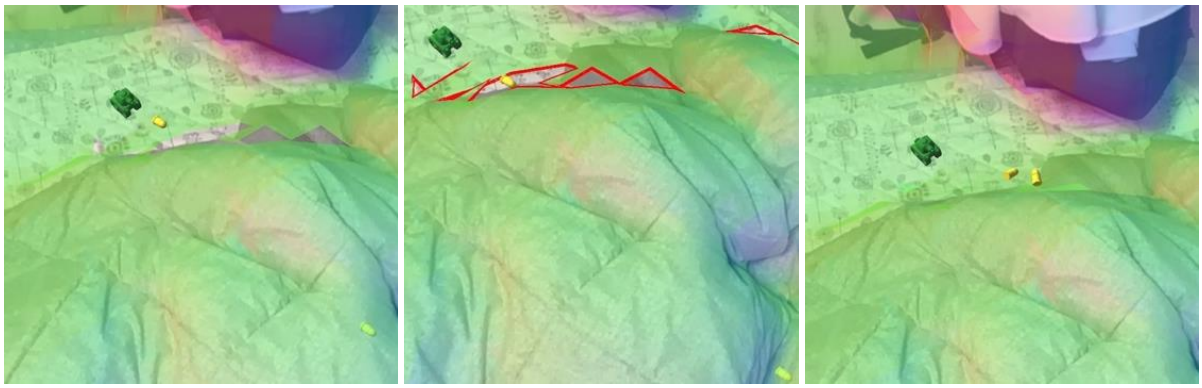
(a) Original (b) Original marked with holes (c) Hole – Filling

Figure 9. Implementation of the proposed algorithm within a gaming scenario



(a) Characters controlled by the player

(b) Projections of the player



(c) Projections of characters controlled by the player

Figure 10. 'Hole-Filling' in different scenarios

5. CONCLUSIONS

In the present study, an algorithm was introduced to address the prevalent issue of holes within AR content. Comprised of two main components—Hole-Finding, for the discernment of real holes from candidate holes by excluding outer lines, and Hole-Filling, for the rectification of the identified holes—the proposed method was systematically evaluated.

The evaluation process included scanning spaces where the limitations of conventional AR Foundation could be distinctly observed. A comparison was made between the performance of the proposed method and the original meshing mechanism, which often failed to generate meshes in blind spots, resulting in a multitude of large and small holes. In stark contrast, it was found that the proposed method was adept at recovering these holes, efficiently supporting real-time requirements. Notably, the proposed method outperformed existing studies by approximately 1,100 times, highlighting its suitability for performance-critical applications such as gaming.

This research represents a pioneering effort to address the hole problem in AR content on real mobile devices. While the proposed method effectively fills individual mesh holes, a recognized limitation is its inability to fill the empty spaces between different meshes. Future work aims to refine this method, leveraging extracted outer lines from the Hole-Finding process to connect meshes, thus enhancing the overall quality of the AR experience.

The implications of this study extend beyond gaming, offering potential applications in diverse AR content. By enabling the creation of more dynamic and varied content, the proposed method contributes significantly to the advancement of AR technology. This investigation thus serves as a foundation for further exploration and development in the field, providing insights that can be applied to a wide range of virtual environments.

ACKNOWLEDGMENT

To increase the understandability of the hole problem and our solution, the authors have created a video and uploaded it to https://youtu.be/eZw8CK49MUQ. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (NRF-2023R1A2C1005950).

REFERENCES

[1] Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, pp. 590-596. https://doi.org/10.1109/CVPR.2005.74

[2] Zhang, B. (2017). Design of mobile augmented reality game based on image recognition. EURASIP Journal on Image and Video Processing, 2017(1): 1-20. https://doi.org/10.1186/s13640-017-0238-6

[3] Kim, H.H., Jung, H.W. (2017). Markerless augmented reality game development method utilizing the Unity engine and KUDAN engine -In the center of the development case of 'Our neighborhood hero'. Journal of Digital Convergence, 15(4): 421-426. https://doi.org/10.14400/JDC.2017.15.4.421

[4] Bang, J.S., Lee, D.C., Seo, S.H., Kim, Y.J., Lee, H.J., Son, W.H. (2016). Trends of VR/AR game technology. Electronics and Telecommunications Trends, 31(1): 146-156.

[5] Joo, E.R., Chung, J.H. (2019). A study on the mobile augmented reality of game Pokemon Go. Journal of Digital Convergence, 17(12): 473-480. https://doi.org/10.14400/JDC.2019.17.12.473

[6] Oufqir, Z., Abderrahmani A. El, Satori K. (2020). ARKit and ARCore in serve to augmented reality. International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, Morocco, pp. 1-7. https://doi.org/10.1109/ISCV49265.2020.9204243

[7] Feigl, T., Porada, A., Steiner, S., Loeffler, C., Mutschler, C., Philippsen, M. (2020). Localization Limitations of ARCore, ARKit, and Hololens in Dynamic Large-scale Industry Environments. VISIGRAPP.

[8] Han, T.W. (2018). A study on the development direction of augmented reality games. The Treatise on The Plastic Media, 21(4): 283-292.

[9] Cho, N.J., Wang, Y.R., Jung, E.J., Yu, G.S. (2021). Factors influencing the intention for continuous use of augmented reality games: Immersion as a mediating variable. Journal of Information Technology Applications & Management, 28(6): 1-21.

[10] Kim, K.S., Park, J.S. (2021). Constructing AR game space through cuboid detection in indoor environment. Journal of Korea Game Society, 21(5): 3-15. https://doi.org/10.7583/jkgs.2021.21.5.3

[11] Lee, W.J., Park, J.S. (2019). Improvement of plane tracking accuracy in AR game using magnetic field sensor. Journal of Korea Game Society, 19(5): 91-101.
[12] Guo, X., Xiao, J., Wang, Y. (2018). A survey on algorithms of hole filling in 3D surface reconstruction. The Visual Computer: International Journal of Computer Graphics, 34(1): 93-103. https://doi.org/10.1007/s00371-016-1316-y
[13] Sobhiyeh, S., Dechenaud, M., Dunkel, A., LaBorde, M., Kennedy, S., Shepherd, J.A., Heymsfield, S.B., Wolenski, P.R. (2019). Hole filling in 3D scans for digital anthropometric applications. 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Berlin, Germany, pp. 2752-2757. https://doi.org/10.1109/EMBC.2019.8856713
[14] Peter, L. (2003). Filling holes in meshes. SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp. 200-205. https://doi.org/10.2312/SGP/SGP03/200-206
[15] Xia, C., Zhang, H. (2017). A fast and automatic hole-filling method based on feature line recovery. Computer-Aided Design and Applications, 14(6): 751-759. https://doi.org/10.1080/16864360.2017.1287677
[16] Feng, C., Liang, J., Ren, M., Qiao, G., Lu, W., Liu, S. (2020). A fast hole-filling method for triangular mesh in additive repair. Applied Sciences, 10(3): 969. https://doi.org/10.3390/app10030969
[17] Park, J.H., Park, S., Yoon, S.H. (2020). Parametric blending of hole patches based on shape difference. Journal of the Korea Computer Graphics Society, 26(3): 39-48. https://doi.org/10.15701/kcgs.2020.26.3.39