Integration and Optimization of a 3D Scanning-Based Wound Coverage System with AI-Driven Boundary Detection

Szabina Molnár-Zékány[1,2]

[1] University of Miskolc, Hatvany József Doctoral School, Miskolc, Hungary
[2] Bay Zoltán Applied Research Nonprofit Ltd., Miskolc, Hungary
sz.zekany@gmail.com

Abstract. Accurate characterization of the geometric and surface properties of pressure ulcers is important for supporting therapeutic decisions and wound coverage planning. Three-dimensional scanning technologies provide new possibilities for capturing wound geometry in a digital form and enable more detailed spatial analysis than traditional two-dimensional approaches. Despite these advances, practical application remains challenging, particularly with respect to system integration, usability, and extensibility of the complete processing pipeline in real-world environments. This study presents the integration and optimization of a wound coverage system, building on processing components developed in earlier stages of the research. The pipeline integrates steps from 3D data capture to wound coverage planning and combines manual user interaction with automated algorithmic procedures. An essential aspect of the present work is the incorporation of artificial intelligence methods into the workflow, with a particular focus on automatic wound boundary detection for subsequent geometric evaluation and coverage planning. A modular system architecture forms a central design principle, allowing individual functional units to be developed, tested, replaced, or extended independently while maintaining consistent data flow. Compared with the initial wound coverage baseline version, the integrated system introduces consolidated functional units, increased automation, AI-supported boundary detection, and targeted improvements to the graphical user interface. The current implementation demonstrates the feasibility of combining these components within a functional research prototype. Further technical and clinical validation is required, with future development focusing on improved automation, expanded AI integration, and enhanced clinical applicability.

Keywords: 3D scanning, pressure ulcers, wound segmentation, artificial intelligence, wound assessment.

1 Introduction

Pressure ulcers represent a significant clinical and economic burden worldwide, reducing patients' quality of life and imposing costs on healthcare systems that are far from

© The Author(s) 2026
S. Árvai-Homolya et al. (eds.), *Proceedings of the 2nd Innovation in Engineering Education Conference (IEE 2026)*, Atlantis Highlights in Social Sciences, Education and Humanities 58,
https://doi.org/10.2991/978-94-6239-823-8_12

negligible. Planning appropriate treatment requires accurate geometric data including area, volume, depth, and surface characteristics, whether the question is about dressing selection, coverage strategy, or simply tracking how the wound is healing. Traditional methods such as ruler-based measurement or acetate tracing have been part of clinical practice for a long time, and in many settings they still are. Their two-dimensional nature and dependence on the individual examiner, however, impose fundamental limitations [1, 2]. Three-dimensional scanning aims to address these shortcomings, as using structured light projection, laser triangulation, or photogrammetry, wound geometry can be captured digitally and spatial metrics can be extracted that are simply not measurable with conventional tools [3, 4, 5, 6]. Of course, solving the measurement problem does not automatically solve clinical deployment — from reliable data acquisition through robust processing to a usable interface, a great many details remain. The present work describes the integration and optimization of such a system, with the not insignificant addition that alongside the earlier desktop application, a web-based version has now been developed. When the system runs in a browser it is no longer tied to a single machine, requires no installation, and a single server can serve multiple clinical sites, which is a meaningful practical advantage. The 3D rendering is handled by a WebGL-based JavaScript library optimized for this purpose, which can take advantage of hardware acceleration, something that matters because dedicated graphics cards are rarely available on clinical workstations. The system covers the entire workflow from 3D data capture to wound coverage planning, and the modular architecture discussed in more detail in Section 3 also means that individual parts, including AI-based boundary detection, can be developed independently [7, 8, 9].

2 Background and Related Work

Clinical wound measurement has long relied on two-dimensional techniques such as ruler-based assessment, manual tracing on transparent film, or digital planimetry. Among these, the Visitrak system is one of the most widely used, where the wound contour is traced onto a film sheet, re-traced on a digital touch pad, and the software calculates the area. Its accuracy and reproducibility have generally been found acceptable [2], but there is a fundamental limitation in that the film must make physical contact with the wound surface, which introduces infection risk, potential distortion of the wound edges, and discomfort for the patient. Chang and colleagues showed in a study published in ePlasty that photographic measurement is statistically equivalent to Visitrak in area estimation (ICC = 0.971), while being completely non-contact — a meaningful advantage in cases involving clean wounds, burns, or infection-sensitive presentations [23]. Both methods share the same underlying constraint though, as they capture two-dimensional data and cannot measure depth, volume, or surface curvature, all of which matter clinically. Three-dimensional optical scanning is precisely what addresses these gaps. Using structured light, laser triangulation, or photogrammetry, the complete wound geometry can be captured without touching it, remotely and non-invasively [3, 4]. Comparative studies have generally found that 3D methods match or exceed the accuracy of traditional techniques, with substantially lower inter-operator variability

[5, 6]. Cost and ease of use were barriers for a long time, but the development of consumer mobile hardware has changed this, as a smartphone equipped with a LiDAR sensor is now capable of performing clinical-grade scans, and in principle displaying the results on the same device [7, 8]. This opens a realistic path toward a system that is accurate, portable, and actually usable by clinical staff and not just by engineers.

Automatic wound boundary detection is arguably the hardest part of the pipeline to get right, but AI-based approaches have been making steady progress. In the medical image segmentation literature, the U-Net architecture was specifically designed for biomedical images and has performed well on wound segmentation tasks [10, 16]. Its encoder-decoder structure with skip connections preserves fine boundary detail through the upsampling pathway — exactly what is needed for accurate wound delineation. More recent approaches — Mask R-CNN [11], the Segment Anything Model [12], Transformer-based architectures [17] — extend the possibilities further, and the field is moving quickly. The combination of web-based delivery and server-side processing means all of this can run in a platform-independent, installation-free environment, where a WebGL rendering library renders the 3D wound visualization interactively in the browser [20, 21].

3 System Architecture

One of the most consequential design decisions was to build the system around modularity — not as an abstract principle, but out of practical necessity. In a medical device development project, requirements shift, algorithms get replaced, and what seems like the right solution today may not be tomorrow. If the system is written as a single monolithic block, every change carries risk. So each major functional part operates as an independent module with its own well-defined input and output. The data cleaning module turns raw scan data into a usable mesh representation — this is covered in Section 4.1. The boundary detection module identifies the wound contour using AI-based segmentation; the geometric analysis module computes the clinical measurements; the coverage planning module generates the closure surface; and the visualization module displays everything. The system runs in both desktop and web form, and what matters is that both versions share the same processing logic and the same AI core — it is not two separate codebases, just the same pipeline with different interfaces. Whether it is worth maintaining both in the long run is an open question, one that clinical feedback will eventually answer.

3.1 Modules and Processing Flow

Processing starts with the raw scan — the data cleaning module takes the Scaniverse PLY or OBJ export and produces a mesh representation that the rest of the pipeline can actually work with. This step cannot be skipped since raw scans typically contain small isolated fragments, duplicate points, and geometrically invalid structures that downstream modules cannot handle. The boundary detection module then runs on the top-view photograph, identifies the wound contour using AI-based segmentation, and this

result feeds directly into both the geometric computations and the coverage algorithm. The coverage planning module generates the closure surface using RBF interpolation — on the web this step is not yet fully stable and is under active development. The geometric module computes the clinical metrics from the closed mesh, and the visualization module renders the result interactively in 3D on both platforms, with per-layer toggle controls. Rendering works differently in the two versions. On the desktop, Open3D's own GUI handles this. On the web, the mesh is converted to JSON and loaded by Three.js r128 with WebGL acceleration, so the entire 3D visualization runs in the browser without installation. Data transfer size will need attention as the system scales, and Draco (.drc), a compression format developed by Google, can reduce geometric data size by up to a factor of ten without losing the detail that matters clinically, and Three.js handles it natively via DRACOLoader. The glTF/GLB format is also an option if texture data ever needs to travel with the mesh, but for now Draco is sufficient.

3.2 Technology and Rendering

The scanning hardware is an iPhone 13 Pro with a built-in LiDAR sensor, which is capable of capturing dense meshes under clinical conditions. The phone sits in a hand-designed 3D-printed holder that can be placed at the bedside — no cables, fully portable. A circular LED ring around the holder provides even illumination, which noticeably improves the consistency of wound photographs used for AI segmentation. Scanning is done with the Scaniverse app, which exports PLY and OBJ files that the processing pipeline reads directly. The processing logic is Python-based, using Open3D [14] for mesh manipulation, OpenCV [18] for image processing, and SciPy [19] for numerical computation. The web layer consists of a Flask server [21] and a Three.js browser-side renderer [20].

4 Processing Pipeline

4.1 Wound Scanning Protocol

Wound scanning should be performed during dressing changes. Following removal of the dressing and cleaning of the wound according to the routine wound-care procedure, the scan is acquired before the application of a new dressing, ointment, or any other topical wound-care product. Scanning is a non-contact procedure; therefore, the device did not come into contact with either the patient or the wound surface. Where possible, the patient should be positioned or repositioned so that they are not lying on the wound being examined and the entire wound area is accessible from multiple directions. Bedridden patients were turned onto their side when necessary to ensure adequate visibility of the wound. The scanning device should typically be held at an approximate distance of 20–50 cm from the wound. The working distance should be adjusted according to the size, depth, and anatomical location of the wound. When recording deeper areas or regions extending beneath the skin, the device may be moved closer to the wound as required. During scanning, the device should be moved relatively slowly and at a steady

speed around the wound, while its surface is recorded from multiple viewpoints and different angles. The purpose of this movement is to ensure that the wound edges, surface, and optically accessible deeper regions are included in the three-dimensional reconstruction. The circular LED light was used during every scan. It provides more uniform illumination, reduces the formation of shadows, and limits the effect of variations in ambient lighting. Adequate general room or hospital ward lighting is also required. The portable wound-scanning setup and its main components are shown in Fig. 1. After brief instruction and minimal practice, the scanning of a readily accessible wound can typically be completed in approximately one minute. However, the required time depends on the size, depth, shape, and accessibility of the wound; larger, deeper, or less accessible wounds may require a longer scanning time. The resulting three-dimensional model can be inspected directly on the smartphone. Under normal circumstances, one scan of adequate quality is sufficient; however, the entire scan should be repeated if the reconstruction is incomplete or contains errors. The method can only be applied when the wound is uncovered, the patient is not lying on it, and the wound can be adequately accessed with the device from multiple directions. In cases of severe bleeding, scanning should be postponed to a more appropriate time.



Fig. 1. Portable wound-scanning setup including the smartphone, circular LED illumination, and 3D-printed holder

4.2 Data Acquisition and Preprocessing

Processing begins with the raw LiDAR scan data. From the Scaniverse PLY or OBJ export, the preprocessing module first retains the largest connected component — smaller fragments and isolated islands produced during scanning are discarded. Duplicate vertices, degenerate triangles, and non-manifold edges are then removed, vertex normals are recomputed, and quadric decimation is applied if needed to bring the triangle count to a manageable level. The result is a consistent, manifold mesh on which subsequent processing steps can run reliably.

4.3 AI-Based Wound Boundary Detection

Identifying the wound boundary accurately is the most critical step in the pipeline — and the one most likely to fail in practice. The boundary detection module operates on the top-view photograph taken at the time of scanning. The segmentation architecture follows the U-Net encoder-decoder design [10], using MobileNetV2 as the encoder backbone [22]. The encoder extracts hierarchical feature representations through successive downsampling stages, while skip connections carry spatial detail to the corresponding decoder levels — this is what allows the predicted boundary to preserve fine geometric structure. The output is a binary segmentation mask separating the wound region from intact skin. When the trained model is temporarily unavailable, the system falls back to HSV-space thresholding combined with Otsu binarization — an automatic threshold selection method that separates wound from skin based on intensity distribution. This fallback is not a replacement for neural network-based segmentation; its accuracy is considerably lower, particularly under difficult lighting or with atypical wound morphology. For this reason, the output should clearly indicate when processing ran on the fallback rather than the trained model. A systematic comparison of the two approaches is planned as future work. After segmentation, morphological closing and opening operations clean the mask, and the largest connected component is retained as the final wound region.

The wound segmentation component was implemented using a U-Net architecture with a MobileNetV2 encoder. The network received RGB images resized to 224 × 224 pixels as input and produced a single-channel wound probability mask through an output layer with sigmoid activation. The MobileNetV2 encoder was initialized with weights pre-trained on the ImageNet dataset, as well as compatible encoder weights from a previously trained wound segmentation model [13]. The publicly available Foot Ulcer Segmentation Challenge dataset [24] contained a total of 1,010 labelled image–mask pairs. Of these, 810 pairs were used for model training, while the 200 labelled validation images were further divided into subsets: 100 images were allocated to the validation set and an additional 100 images to the internal test set. The 200 official test images of the challenge were excluded from quantitative evaluation, as the corresponding reference masks are not publicly available. In addition to the public data, 50 wound images collected in-house with manually created segmentation masks were also included. The in-house data were split at the wound or patient level into 30 training, 10

validation, and 10 test images. The complete labelled dataset thus comprised 1,060 image–mask pairs. In the first training phase, the network was trained on the public dataset using a batch size of 8 and a maximum of 100 epochs. The Adam optimizer was applied with an initial learning rate of 1×10^{-4}. The MobileNetV2 encoder was kept frozen during the first five epochs, after which it was unfrozen and the entire network was fine-tuned at a learning rate of 1×10^{-5}. In the second phase, the model was fine-tuned on the in-house wound images using a batch size of 4, a maximum of 50 epochs, and a learning rate of 1×10^{-5}. During training, the Dice coefficient, intersection over union, precision, and recall were monitored. Early stopping with a patience of 15 epochs was applied based on the validation Dice score, and the model parameters yielding the best validation Dice were restored. If the validation Dice did not improve for five consecutive epochs, the learning rate was reduced by a factor of 0.5, down to a minimum of 1×10^{-6}. Model performance was evaluated using the Dice coefficient, IoU, precision, and recall. The final values will be inserted after completion of the predefined independent evaluation of the in-house test set: Dice 0.7568, IoU 0.6235, precision 0.7314, and recall 0.9126.

4.4 Contour Projection and 3D Metric Computation

The wound contour, which is initially represented in pixel coordinates of the top-view mask, needs to be projected onto the 3D mesh surface to produce a geometrically meaningful boundary representation. The projection uses a linear mapping between the mask image coordinate space and the mesh bounding box in the XY plane, where each contour pixel (u, v) is mapped to a 3D world coordinate, and a vertical ray is then cast downward to find the intersection with the mesh surface using Open3D's ray casting scene. Ordering the projected contour points is where the earlier desktop version often went wrong. That version used polar coordinate sorting, ordering points by angle around the centroid, which works for convex wound shapes but produces crossing polygons for concave geometries, which is not uncommon. The web version therefore switched to nearest-neighbor ordering, where the algorithm always picks the closest next point, which handles concave shapes correctly, though with very noisy point sets it can still occasionally produce small errors. From the ordered 3D contour, wound perimeter follows from the sum of consecutive Euclidean distances, area from the shoelace formula applied to the XY projection, and depth from the difference between the mean Z of the contour and the minimum Z among mesh vertices inside the contour polygon. Volume is computed using signed tetrahedral decomposition when the mesh is watertight, and falls back to an area-depth approximation otherwise, which is less accurate and improving this remains one of the main development priorities.

5 Graphical User Interface

The interface is designed to work for both clinical and engineering users, which in practice means different things to different people. The left panel presents a step-by-step

workflow. The mesh upload zone accepts PLY or OBJ files by drag-and-drop, triggering automatic cleaning and 3D rendering. The photo upload zone accepts PNG or JPG images and launches AI segmentation — the resulting mask and contour overlay appear immediately as thumbnail previews. The analysis button runs the full 3D pipeline, and a timestamped log tracks what is happening at each stage. The integrated web-based user interface and its main workflow components are shown in Fig. 2. The wound contour appears as a green line directly on the mesh surface, the reconstructed closure surface as a semi-transparent layer beneath it — each can be toggled independently. Top-down and side-view presets support clinical review. The right panel displays the computed metrics — perimeter, area, depth, volume — in color-coded cards. Below that, mesh information, segmentation statistics, and export controls let the user download results as PLY, JSON, or CSV.

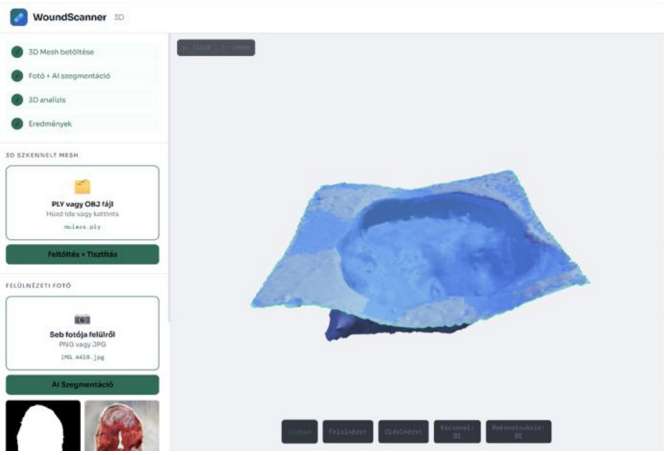


Fig. 2. Graphical user interface of the WoundScanner 3D system

6 Optimization

Raw Scaniverse scans typically contain tens of thousands of triangles, sometimes over a hundred thousand, and a large fraction of these are redundant from a clinical standpoint. The system applies quadric decimation to bring this down to a manageable level, using an algorithm that iteratively collapses adjacent vertices while minimizing surface deviation, so the wound edge shape, depth profile, and surface curvature are preserved while model size and rendering load drop substantially. Beyond the triangle count, geometric stability also needs attention. Open3D's built-in validation functions handle the most common issues [14], starting with duplicate vertices that are removed because the same spatial point appearing twice causes numerical instability further down the pipeline, degenerate triangles where the three vertices are collinear and the face has zero area are discarded since they carry no geometric information, and non-manifold edges are cleaned up because a valid mesh requires exactly two triangles per edge and any-

thing more is physically meaningless. Normal orientation is made consistent with orient_triangles(), which matters for both rendering and volume computation. None of this is particularly exciting, but skipping it tends to produce hard-to-diagnose failures later. The modular structure has an optimization benefit that is easy to miss. Because each functional unit has a well-defined interface, individual modules can be replaced, for instance swapping in a better-trained segmentation model, without touching anything else. In-memory data passing between modules avoids unnecessary disk writes, and the session-based output directory keeps intermediate results accessible if something needs to be traced back. These are small things individually, but they add up over the course of a longer development cycle.

7 Results and Future Work

At the current stage, the system should be regarded as a functional research prototype that demonstrates the technical feasibility of integrating portable 3D wound acquisition, AI-assisted boundary detection, geometric analysis, and visualization within a unified workflow. The iPhone 13 Pro with its LiDAR sensor, the 3D-printed bedside holder, and the circular LED ring form a portable experimental acquisition setup, while the desktop and web-based implementations support the subsequent processing and visualization steps. The system has not yet undergone prospective clinical validation; therefore, the presented results should be interpreted as a demonstration of technical feasibility rather than evidence of clinical performance. Development continues in several directions. The current bounding-box-based contour projection can produce errors when the scanning geometry deviates from vertical, and replacing it with a more principled camera-based or depth-map-based approach is one of the most important next steps. The AI segmentation component's performance depends heavily on the training dataset, and with a larger, more varied collection of annotated wound photographs and systematic testing across different wound types and coverage situations, the model should generalize considerably better. A feedback loop is also conceivable, where the AI not only segments but also evaluates and refines the generated contour, which would directly improve mesh accuracy. The closure algorithm is another candidate for improvement, where AI-based surface reconstruction methods could produce a more anatomically accurate estimate of the original skin geometry, which would in turn make volume computation more reliable.

References

1. Langemo, D.K., Melland, H., Hanson, D., Olson, B., Hunter, S.: Two-dimensional wound measurement: comparison of 4 techniques. American Journal of Wound Care 7(1), 29–34 (1998).
2. Gethin, G., Cowman, S.: Wound measurement comparing the use of acetate tracings and Visitrak digital planimetry. Journal of Clinical Nursing 15(4), 422–427 (2006).
3. Plassmann, P., Jones, T.D.: MAVIS: a non-invasive instrument to measure area and volume of wounds. Medical Engineering & Physics 20(5), 332–338 (1998).

4. Pavlovcic, U., Diaci, J., Mozina, J., Jezersek, M.: Wound perimeter, area, and volume measurement based on laser 3D and color acquisition. BioMedical Engineering OnLine 14(39), 1–15 (2015).
5. Shah, A., Wollak, C., Shah, J.B.: Wound measurement techniques: Comparing the use of ruler method, 2D imaging and 3D scanner. Journal of the American College of Clinical Wound Specialists 5(3), 52–57 (2013).
6. Bowling, F.L., King, L., Fadavi, H., Paterson, J.A., Boulton, A.J.M.: An assessment of the accuracy and usability of a novel optical wound measurement system. Diabetic Medicine 26(1), 93–96 (2009).
7. Wang, S.C., et al.: Point-of-care wound visioning technology: Reproducibility and accuracy of a wound measurement app. PLoS ONE 12(8), e0183139 (2017).
8. Foltynski, P.: Ways to increase precision and accuracy of wound area measurement using smart devices. PLoS ONE 13(3), e0192485 (2018).
9. Wannous, H., Lucas, Y., Treuillet, S.: Enhanced assessment of the wound-healing process by accurate multiview tissue classification. IEEE Trans. Medical Imaging 30(2), 315–326 (2011).
10. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. MICCAI, pp. 234–241. Springer (2015).
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. ICCV, pp. 2961–2969 (2017).
12. Kirillov, A., et al.: Segment Anything. ICCV, pp. 4015–4026 (2023).
13. Wang, C., Anisuzzaman, D.M., Williamson, V., et al.: Fully automatic wound segmentation with deep convolutional neural networks. Scientific Reports 10, 21897 (2020). https://doi.org/10.1038/s41598-020-78799-w
14. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018).
15. Zahia, S., et al.: Tissue classification and segmentation of pressure injuries using convolutional neural networks. Computer Methods and Programs in Biomedicine 159, 51–58 (2018).
16. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CVPR, pp. 3431–3440 (2015).
17. Dosovitskiy, A., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021).
18. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal (2000).
19. Virtanen, P., et al.: SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods 17(3), 261–272 (2020).
20. Cabello, R., et al.: Three.js – JavaScript 3D Library. https://threejs.org (2010–2024).
21. Grinberg, M.: Flask Web Development: Developing Web Applications with Python. O'Reilly Media (2018).
22. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks. CVPR, pp. 4510–4520 (2018).
23. Chang, A.C., Dearman, B., Greenwood, J.E.: A comparison of wound area measurement techniques: Visitrak versus photography. ePlasty 11, e18 (2011).
24. Wang, C., Mahbod, A., Ellinger, I., Galdran, A., Gopalakrishnan, S., Niezgoda, J., Yu, Z.: FUSeg: The Foot Ulcer Segmentation Challenge. Information 15(3), 140 (2024). https://doi.org/10.3390/info15030140

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

