Security of Infocommunication Systems and Internet of Things

2025 Vol 3, No 2

https://doi.org/10.31861/sisiot2025.2.02011

Received 28 August 2025; revised 28 December 2025; accepted 29 December 2025; published 30 December 2025

Formation of a Three-dimensional Relief Model Based on a 2D Image

Vitalii Ariichuk* **and Yuliya Tanasyuk**

Computer Systems and Networks Department, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: ariichuk.vitalii@chnu.edu.ua)

ABSTRACT This paper introduces an automated pipeline for generating high-quality 3D bas-relief models directly from single 2D images. Our method replaces laborious manual height-map editing with ZoeDepth, a state-of-the-art neural network that outputs dense depth maps without camera calibration or manual annotations. Input photographs, whether of architectural facades, artwork reproductions, or industrial scenes, are first contrast-normalized and preprocessed prior to depth estimation. The raw depth output then undergoes metric-attractor correction, which refines depth-bin centers using multiple predicted attractor points per pixel to improve continuity and reduce quantization artifacts. A custom depth-aware triangulation algorithm subsequently converts the refined depth field into a surface mesh, with user-controlled parameters for real-world scale, spatial resolution, and triangulation density. The final mesh is exported as an STL file, enabling immediate compatibility with CAD software and 3D printers. The performance, robustness and fidelity of the elaborated pipeline were evaluated on a quad-core Intel i5 CPU and a variety of image domains. It turned out that depth inference for an 800 × 800 pixel image took around 120 seconds, while mesh generation and STL export took approximately 110 seconds. These times scale linearly with image resolution. Quantitative assessment yields a mean relative error below 7.7 % and threshold accuracy above 95.3 %, indicating that over 95 % of pixel depth estimates fall within 25 % of true values. A qualitative inspection has confirmed that the obtained reliefs preserve critical geometric details and maintain surface smoothness, even on previously unnoticeable inputs. Comparative analysis highlights significant reductions in manual effort and total modeling time versus traditional Blender-based sculpting workflows, without sacrificing mesh quality.

KEYWORDS bas-relief, 3D modeling, depth estimation, neural networks, STL.

I. INTRODUCTION

Three-dimensional modeling has become increasingly important in domains such as digital heritage preservation, 3D printing, and architectural design. A particularly challenging task within this subject area is the creation of bas-relief sculptures directly from flat 2D images, i. e. representations where objects subtly protrude from a background surface.

Accurate object recognition is a core challenge in computer vision, often tackled by modeling biological visual systems of humans and animals. Human vision processes vastly more information than other senses [1-8]. Perceptual processing unfolds in stages from initial detection to coarse and detailed analysis allowing trained observers to recognize familiar objects in as little as 0.06 seconds [7, 8]. Two leading cortical models – a detector approach extracting basic primitives and a spatial-frequency analysis that uses localized Fourier transformations – inspire modern computer-vision architectures aiming for human-level performance.

Over the past decades, 3D reconstruction from 2D imagery has followed distinct paradigms. Classical geometric pipelines-notably Structure-from-Motion [9] reconstruct camera positions and generate dense point clouds from multiple overlapping photographs, with open-source tools such as COLMAP [10], OpenMVG [11] and OpenMVS [12]. Silhouette-based visual-hull techniques [13] project object outlines across views to carve a maximal 3D volume commonly prototyped in OpenCV [14] and expanded by point-cloud libraries. However, the mentioned approaches often produce convex approximations and require multiple calibrated images. Shape-from-shading methods infer local relief from pixel-intensity gradients under known lighting, yet they are noise-sensitive and necessitate careful photometric calibration. The Deep learning based monocular depth-estimation models such as MiDaS [15] and ZoeDepth [16] predict dense depth maps from single RGB images, offering full automation without multi-view data.

The creation of a high-quality bas-relief in Blender [17] requires advanced 3D modelling skills, encompassing the crafting and editing of height maps, mastering sculpting brushes, modifier stacks, and mesh topology. Furthermore, this procedure involves multiple iterative steps that must be finely tuned by a skilled user, making the process both time-consuming and inaccessible to non-experts.

To overcome these limitations, our research proposes a fully automated pipeline for generating 3D bas-reliefs from 2D images using deep learning-based depth estimation. The proposed solution is based on the ZoeDepth architecture [16], which is a neural network capable of using relative and metric depth. This enables the accurate reconstruction of scene geometry.

The objective of this paper is to create a fully automated Python-based [18] pipeline that transforms a single 2D image into a printable STL file of its 3D bas-relief. This will be achieved by integrating ZoeDepth [16] for high-fidelity depth estimation, applying metric-attractor

smoothing to eliminate artifacts, and implementing a triangulation algorithm with real-world scaling in centimeters. The pipeline offers configurable spatial resolution, and multiple processing modes. To ensure reproducibility, all functionality will be exposed via a command-line interface. The generated model must match specified size, exhibit no visible surface artifacts and be immediately loadable and 3D-printable in standard CAD environments without manual adjustment. The system's performance and geometric fidelity will be benchmarked against traditional manual modeling workflows in modern Blender, with positive results defined by low processing times, high-quality STL outputs, and integration into STL-based 3D printing workflows.

II. PREREQUISITES AND RELATED TOOLS

A depth map [19] is an image in which each pixel encodes the distance from a virtual camera to the scene's surface. It's usually represented as a grayscale image where lighter pixels indicate nearer points and darker pixels indicate farther ones. By extracting this per-pixel depth information, we gain a height field that can be directly translated into geometric displacement.

The traditional pipeline for generating a bas-relief from a single 2D image includes image preprocessing to produce a grayscale height map, mesh generation through triangulation and displacement modifiers, and final STL export for printing. In Blender, users can generate depth maps natively by enabling the Z pass under Render Layers and normalizing raw depth values in the compositor. The one-sample-per-pixel Z pass often exhibits aliasing and lacks anti-aliasing in motion-blur or depth-of-field scenes. Converting to 8-bit can white-out distant geometry if near and far clip planes are not carefully set. Wrong camera settings produce noise and depth inaccuracies in the final relief.

Online lithophane converters such as 3DP-Rocks Lithophane [20] allow users to upload a photo and download a light-transmissive relief optimized for backlit display. A lithophane is a thin, translucent 3D print. Its varying thickness modulates the amount of light that passes through it, revealing an image when it is illuminated from behind. Unlike models intended for architectural or artistic bas-reliefs, which require opaque geometry and robust thickness, these models prioritise precise light transmission. As a result, lithophanes lack the solid form, material compatibility, and dimensional accuracy needed for traditional relief generation.

III. PROPOSED METHOD

Depth estimation is handled by ZoeDepth [16], a two-stage neural model combining Relative Depth Estimation (RDE) and Metric Depth Estimation (MDE). In stage one, a MiDaS-based [15] encoder extracts multi-level scene features and predicts a scale-invariant relative depth map. In stage two, a domain classifier directs those features to one of two lightweight metric heads, namely indoor or outdoor, each trained on metric datasets (NYU Depth v2, KITTI). These heads use Metric Bins, which predict a fixed set of depth – bin centers and then refine them via attractor offsets at each decoder level (Fig. 1).

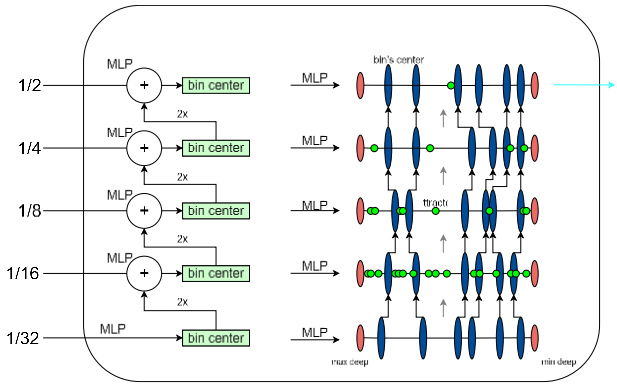

FIG. 1. ZoeDepth's Metric Bin Module Architecture [16].

The lowest level forms the metric bin centers, and the remaining levels create attractors that shift the bin centers according to the decoder level.

Final per-pixel depth $d(i)$ is computed by a weighted sum over N_{total} bins:

$$d(i) = \sum_{k=1}^{N_{total}} p_i(k) c_i(k), \quad (1)$$

where N_{total} is the total number of discrete depth intervals (e.g., 0–1 m, 1–2.5 m, etc.), $p_i(k)$ is the probability that pixel i's true depth falls into interval k, $c_i(k)$ is the depth value predicted for bin k (the bin center), and $d(i)$ is the final metric depth (in meters) assigned to pixel i.

At the l^{th} decoder level, the Multi-Layer Perceptron (MLP) takes the pixel's feature vector and predicts n_a attractor points $\{a_k : k = 1, \ldots, n_a\}$ for that pixel. The corrected bin center is then given by:

$$c_i' = c_i + \Delta c_i, \quad (2)$$

where the correction is computed as:

$$\Delta c_i = \sum_{k=1}^{n_a} \frac{a_k - c_i}{1 + \alpha |a_k - c_i|^{\gamma}}, \quad (3)$$

α and γ are hyperparameters that control the strength of each attractor.

For final depth estimation, the ZoeDepth [16] uses a binomial distribution, determining which bin center best matches the attractor outputs. The probability of selecting the k-th bin is given by:

$$p(k; N, q) = \binom{N}{k} q^k (1-q)^{N-k}, \quad (4)$$

where q is the most likely attractor – predicted probability, $N = N_{total}$ is the total number of depth bins (see Eq. 1), and k is the bin index.

Triangle Soup is a collection of triangles used for storing the geometric structure of a 3D model in a simple, unstructured format.

Although each triangle is defined by three points, the triangles themselves are not strictly connected to one another. This structure has two key advantages: it is simple to implement, and it works with many graphics editors, including Blender.

Before constructing the 3D model of a bar-relief, the depth values must be normalized to account for the image dimensions and the scene's maximum depth. It helps to correctly represent height differences on the 3D model, matching height values with real-world measurements:

$$h = \frac{sar}{m}, \quad (5)$$

where h is the normalized vertex height (z coordinate), s is the model's width, a is the user-specified scale factor, r is the red-channel intensity of the pixel, and m is the maximum brightness value in the image.

Once the metric depth map is obtained and normalized, we construct a surface mesh by treating each adjacent 2 × 2 pixel block as two triangles. For pixels at (i,j), (i+1,j), (i,j+1) and (i+1,j+1), the first triangle connects vertices at (i+1,j), (i,j+1), (i,j), and the second at (i+1,j+1), (i,j+1), (i+1,j) (Fig. 2).

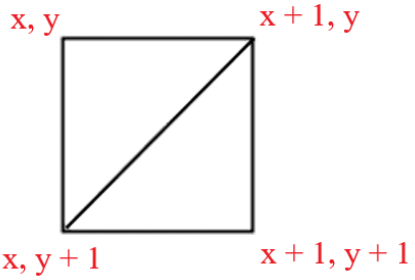

FIG. 2. Pixel block as two triangles.

Each vertex's X,Y coordinates are scaled to centimeters by the real_width/image_width and real_height/image_height ratios, and Z is taken from the normalized depth map. The full mesh, a "Triangle Soup" of independent faces, is then written to an STL file.

Our solution implements a fully automated Python pipeline (Fig. 3) that converts a single 2D image into a printable 3D bas-relief. The process doesn't require manual preparation of the image. Users can import an image and specify the model's parameters by selecting the relevant options: a path to an input image; a path for saving the depth map; a path for saving the output STL file; a ratio of thickness to width in cm for the STL; a real-world width of the model in cm; a real-world height of the model in cm, by using only the width option the height is automatically calculated based on the real-world width. However, it is important to ensure that the imported image is of good quality and not overly smooth, as this could result in artifacts in the obtained bas-relief. The pipeline begins by generating a depth map using the ZoeDepth [16] neural network model. Based on the depth map, vertexes are created, and a 3D bas-relief model is generated, which is then exported as an STL file. The program also offers features such as exporting depth maps for analysis or visualization, exporting STL 3D models with defined dimensions in centimeters, and supporting different image processing modes, including performance, balance, and quality. This comprehensive process transforms a flat image into a detailed 3D representation, suitable for printing and artistic applications.

The codebase is organized into three core Python [18] modules (Fig. 4). The main script parses arguments, ensures the ZoeDepth [16] model is cloned or cached, and orchestrates processing. The DepthProcessor module wraps the ZoeDepth inference API and handles depth-map normalization. The StlGenerator module reads the 16-bit depth image, applies real-world scaling, executes triangulation and writes the STL. This separation of concerns simplifies maintenance, testing and future extension (e.g., replacing ZoeDepth or output formats).

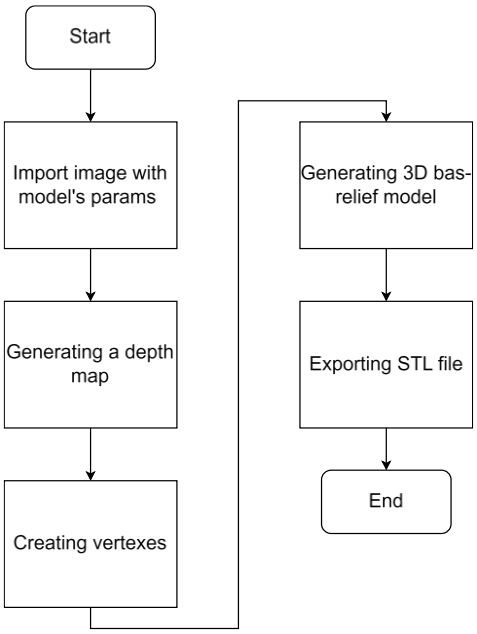

FIG. 3. Pipeline of formation of a three-dimensional relief model.

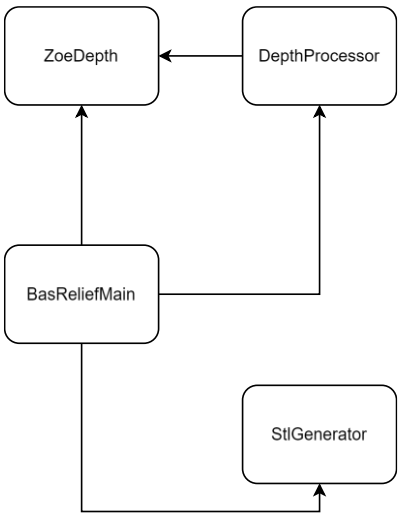

FIG. 4. Class diagram of the developed Python solution.

IV. RESULTS AND DISCUSSION

Testing of the program solution was conducted manually using different types of images. Each 3D bas-relief model was verified in Blender to ensure that its size equals the specified values in centimeters.

Fig. 6 shows the 700 × 700-px crop of a decorative tile from the Chernivtsi University facade (Fig. 5). The ZoeDepth [8] generated the 16-bit depth map in 120 s on the Intel i5 CPU (Fig. 7). Building and exporting the STL using the mesh module took 110 seconds (Fig. 8).

FIG. 5. Original picture of the facade.

FIG. 6. Cropped picture to the tile.

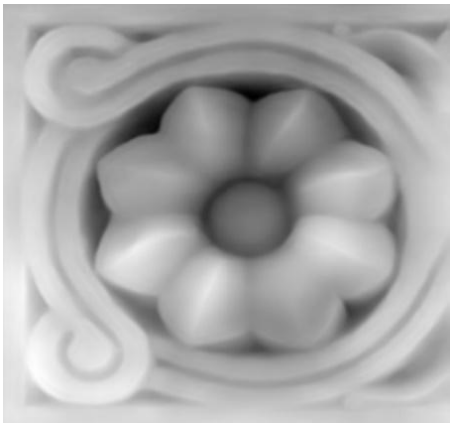

FIG. 7. The processed depth map of the tile.

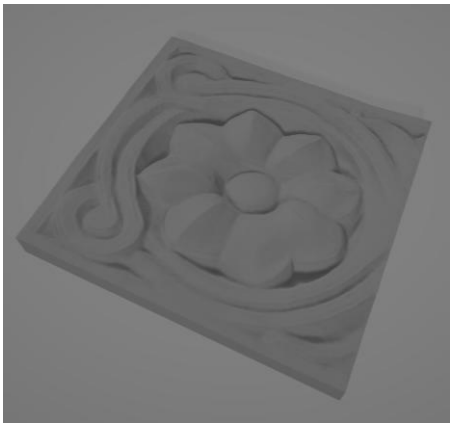

FIG. 8. The processed STL mesh of the tile.

The resulting model contains 1 027 114 triangles. The output file size is 48.9 MB. When opened in Blender the dimensions of the obtained image were exactly as specified: 15 cm × 14 cm × 1.5 cm. Using the Blender to manually create a three-dimensional bas-relief model with the same depth map will result in the same number of triangles and a file of the same size.

Another image used to verify the effectiveness of the developed approach was the photo of Vincent van Gogh's painting "Starry Night". The resolution of the image was 960 × 760 px (Fig. 9). The pipeline produced a 71.4 MB STL file with about 1.1 million triangles (Fig. 11). Although the runtime increased to around 160 s, the resulting bar-relief accurately reproduces the details of the painting.

FIG. 9. The input "Starry Night" picture.

FIG. 10. The processed depth map of the "Starry Night" picture.

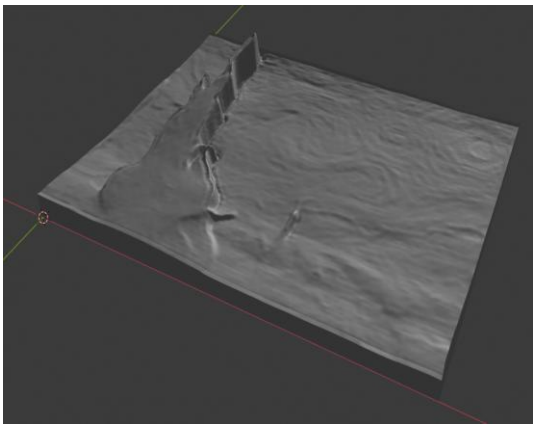

FIG. 11. The processed STL mesh of the image in Fig. 9.

On a quad-core Intel i5 CPU, it takes ZoeDepth about 120 seconds to perform depth inference on an 800 × 800 pixel image, and about 110 seconds to perform mesh construction and STL export. Processing time grows linearly with pixel count, so higher-resolution inputs require proportionally more time. To achieve the best results, the source images should be high-contrast with clearly defined features, as blurred or low-detail photos result in depth-map artefacts and corresponding mesh defects.

V. CONCLUSION

This paper presents an end-to-end pipeline for the automated generation of 3D bas-reliefs from 2D images, using a deep learning–based depth estimation model and a custom STL export mechanism. By integrating the

ZoeDepth [16] architecture, the system achieves strong generalization and accuracy across diverse scenes without requiring manual intervention or camera calibration. Importantly, the quality of bas-reliefs produced with the use of our method is comparable to those sculpted manually in Blender. The authors of ZoeDepth assert that predictions exhibit a mean relative error of less than 7.7 % and a threshold accuracy of 95.3 %. This means that over 95 % of pixel depth estimates fall within 25 % of the true value [16].

The proposed Python-based solution addresses key limitations in traditional 3D modeling workflows, such as time-consuming mesh sculpting and limited scalability. The use of command line parameters enables flexible model customization, including the consistency with real-world parameters and resolution control. Experimental validation confirms the system's usability, STL compatibility, and applicability across artistic, architectural, and heritage restoration tasks.

Combining metric-attractor depth correction with depth-aware triangulation allows one to produce high-resolution, printable bas-reliefs. The practical value of the research consists in the pipeline's ability to streamline digital modeling workflows for educational, design, and industrial use.

ACKNOWLEDGMENT

The authors gratefully acknowledge Doctor of Technical Sciences, Associate Professor Serhii Balovsyak for his expertise in computer vision, his guidance on selecting the most suitable technologies, and his insights into potential limitations and challenges encountered during the research.

AUTHOR CONTRIBUTIONS

V.A. – conceptualization, methodology, software, validation, investigation, writing-original draft preparation, visualization; Y.T. – supervision, writing-review and editing.

COMPETING INTERESTS

The authors declare no conflict of interest.

REFERENCES

[1] M. M. Lybovets and O. V. Oletsky, *Artificial intelligence: a textbook [for students of higher education]*. Kyiv: KM Academy, 2002, 336 p.

[2] N. B. Shakhovska, R. M. Kaminsky, and O. B. Vovk, *Artificial intelligence systems: a textbook*. Lviv: Lviv Polytechnic Publishing House, 2018, 392 p.

[3] DaeEun Kim and Dosik Hwang, Eds., *Intelligent Imaging and Analysis*. Switzerland, Basel: MDPI, 2020, 492 p. [Online]. Available: https://mdpi.com/books/pdfview/book/2059. DOI: 10.3390/books978-3-03921-921-6.

[4] L. Zhu and P. Spachos, "Towards Image Classification with Machine Learning Methodologies for Smartphones," *Machine Learning and Knowledge Extraction*, 2019, No. 1(4), pp. 1039–1057.

[5] S. D. Shtovba and V. V. Mazurenko, *Intelligent technologies for identifying dependencies. Laboratory practical: electronic textbook*. Vinnytsia: VNTU, 2014, 113 p.

[6] R. Gonzalez and R. Woods, *Digital image processing*, 4th edidion. NY: Pearson/ Prentice Hall, 2018, 1192 p.

[7] S. Krigg, *Computer Vision Metrics. Survey, Taxonomy, and Analysis*. Spredd Open, 2014, 498 p.

[8] J. C. Russ, *The Image Processing. Handbook [Sixth Edition]*. Taylor & Francis Group, LLC, 2011, 853 p.

[9] J. L. Schonberger and J. M. Frahm, "Structure-from-Motion Revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, 27–30 June 2016, pp. 4104–4113.

[10] "COLMAP Documentation." [Online]. Available: https://colmap.github.io/.

[11] "OpenMVG Documentation." [Online]. Available: https://github.com/openMVG/openMVG/wiki.

[12] "OpenMVS Documentation." [Online]. Available: https://github.com/cdcseacave/openMVS/wiki.

[13] K. N. Kutulakos and S. M. Seitz, "A Theory of Shape by Space Carving. A theory of shape by space carving," *International Journal of Computer Vision*, 38 (3), 2000, pp. 199–218.

[14] "OpenCV Documentation." [Online]. Available: https://docs.opencv.org/4.x/.

[15] "MiDaS: Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer." [Online]. Available: https://arxiv.org/pdf/1907.01341v3.

[16] "ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth." [Online]. Available: https://arxiv.org/pdf/2302.12288.

[17] Blender Foundation, *Blender – Open Source 3D Creation Software*. [Online]. Available: https://www.blender.org/.

[18] "Python Documentation." [Online]. Available: https://docs.python.org/3/.

[19] "Depth Maps: How Software Encodes 3D Space." [Online]. Available: https://blog.lookingglassfactory.com/depth-maps-how-software-encodes-3d-space/.

[20] "Image to Lithophane." [Online]. Available: https://3dp.rocks/lithophane/.

Vitalii Ariichuk

PhD student in Computer Engineering at the Department of Computer Systems and Networks, Yurii Fedkovych Chernivtsi National University, Ukraine. Software Development Engineer. Interests and expertise: applied geometry, computational geometry, feature recognition, CAD systems. His work focuses on 3D bas-relief modeling with neural networks and Python.

ORCID ID: 0009-0006-0383-9137

Yuliya Tanasyuk

PhD, Associate Professor at the Department of Computer Systems and Networks of Physical, Technical and Computer Sciences Institute, Yuriy Fedkovych Chernivtsi National University, Ukraine. Research interests and academic activities are as follows: programming, network information technologies, cybersecurity, IoT, software engineering.

ORCID ID: 0000-0001-8650-0521

Формування тривимірної моделі барельєфа на основі 2D-зображення

Віталій Арійчук*, Юлія Танасюк

Кафедра комп'ютерних систем та мереж, Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: ariichuk.vitalii@chnu.edu.ua)

АНОТАЦІЯ У статті запропоновано повністю автоматизовану методику створення тривимірних моделей барельєфів на основі звичайних двовимірних зображень. Розроблений підхід передбачає використання глибинної нейронної мережі ZoeDepth для автоматичного формування точних карт глибини без необхідності ручного редагування або калібрування камери. Отримані карти глибини проходять послідовну обробку в Python: значення глибини нормалізуються та масштабуються відповідно до реальних фізичних розмірів зображення, після чого за методом тріангуляції формується полігональна сітка. У результаті згенерована модель у форматі STL придатна для швидкого 3D-друку або подальшого редагування. Ключові переваги запропонованої системи – це гнучкість налаштування розмірів через інтерфейс командного рядка, відсутність потреби ручного втручання в процес побудови карти глибини та універсальність застосування мережі для найрізноманітніших сцен, як-от архітектурні деталі, художні твори або промислові об'єкти. Проведені експерименти демонструють, що час формування карти глибини на центральному процесорі для зображення розміром 800 × 800 пікселів становить близько 120 секунд, а тріангуляція та збереження моделі у файл – близько 110 секунд. При збільшенні роздільної здатності вхідного зображення відповідно зростає й загальний час обробки. Для досягнення найвищої якості карти глибини рекомендовано застосовувати метод до контрастних сцен із виразними деталями, оскільки нечіткі та розмиті зображення породжують артефакти в карті глибини та фінальній моделі. У порівнянні з Blender наше рішення також досягає якісного результату у створенні барельєфу. Скорочення обсягу ручних налаштувань та автоматизація процесів формування барельєфу дають змогу швидко та якісно отримати модель із потрібними параметрами. Кількісна оцінка виявила, що середня відносна похибка становить менше ніж 7.7 %, а порогова точність – понад 95.3 %, що вказує на те, що понад 95 % оцінювань глибини пікселів лежать у межах 25 % від справжніх значень. Якісна перевірка підтвердила, що отримані рельєфи зберігають важливі геометричні деталі та вирівнювання поверхні навіть на раніше непомітних вхідних даних. Порівняльний аналіз показує значне зменшення обсягу ручного оброблення та загального часу моделювання порівняно з традиційними робочими процесами скульптування на основі Blender без втрати якості отриманої тріангуляційної сітки.

КЛЮЧОВІ СЛОВА барельєф, тривимірне моделювання, оцінка глибини, нейронні мережі, формат STL.