Transforming Coordinate Function Points into 2D or 3D Graphics: An Algorithmic Approach

Omar A. Alsammarraie[1], Qusay H. Al-Salami[2], Noor Q. Al-Salami[3]

College of Engineering, Istanbul Gelişim University, Istanbul, Turkey.
Department of Business Administration, Cihan University-Erbil, Kurdistan Region, Iraq.
Department of Communication and Computer Engineering, Cihan University-Erbil, Kurdistan Region, Iraq.

***Abstract*— Our research demonstrates the process of converting paper graphs into corresponding coordinates on a computer screen, whether in two-dimensional or three-dimensional form, through simplified mathematical equations. These equations are designed to enable users to comprehend their structure and develop their own code, avoiding reliance on ambiguous pre-built programs. To accomplish this, the initial phase involved understanding graph reading and interpretation methods through an extensive literature review. The process of transforming coordinate function points into 2D or 3D graphics involves several essential stages (Function Definition; Point Generation; Coordinate Transformation; Projection (for 3D Graphics); Rendering). In 3D graphics, converting a 3D point to a 2D perspective projection is crucial for rendering realistic scenes on a 2D screen. This conversion creates the illusion of depth and distance in computer-generated images. The research presented here offers a solution to the challenge of converting two- and three-dimensional coordinates in diverse fields of computer graphics. It provides students with a straightforward method to comprehend and manage these transformations.**

***Keywords*—2D and 3D Graphics, Algorithms, Coordinate Transformation, Coordinate Transformation**

I. INTRODUCTION

Graphs are utilized in diverse domains, ranging from data structures to networks, software engineering to databases. Typically, small graphs are manually depicted to effectively represent underlying relationships. Although numerous algorithms exist for drawing specific graph classes such as trees and planar graphs, there are fewer general-purpose graph drawing algorithms available. Force-directed methods are preferred for drawing general graphs due to their conceptual simplicity, adaptability across various graph types, and capacity to generate visually appealing layouts [1].

Converting coordinate function points into 2D or 3D graphics constitutes a foundational aspect within the realm of computer graphics and visualization [2]. This pivotal process encompasses the translation of mathematical depictions of points, curves, or surfaces, as defined by functions, into visual representations suitable for display on screens [3]. This article delves into the algorithmic methodology employed for this transformation of coordinate function points into 2D or 3D graphics [4].

Extracting 3D info from 2D images is challenging. The goal of 2D to 3D reconstruction is to create a volumetric or surface model [5]. Previous research focused on specific graph types, such as interpreting graphs in scientific experiments. Hand-drawn graphs pose challenges due to variations in styles. Image processing is commonly used to detect features [6]. Automating graph creation benefits software scientists transitioning to computational analysis. Graphs provide quick insights compared to tables of measurements [7], [8].

Human capacity to retain structure mentally varies; hence, graph layout complexity should be adjustable. Partitioning graphs into clusters can offer varied detail levels, with dynamic resizing based on user requirements. Graph layout poses challenges, particularly balancing hierarchical structure depiction with minimizing edge crossings for user orientation during exploration [9]. [10] propose a new method for creating 3D character representations from turnaround concept art. 3D modeling is crucial in AR/VR and gaming, offering creative and practical benefits, despite its time-consuming nature and skill requirements.

The issue of 3D coordinate transformation is a common challenge across various disciplines such as computer graphics, robotics, aeronautics, computer vision, surveying, and lidar point cloud transformation [11], [12].

Graph-based 3D visualization of complex data doesn't demand extra cognitive skills. However, 2D displays lack depth cues. Users rely on scene transformations during navigation for depth perception, posing a challenge for static viewpoints. Developers strive to balance layout complexity and navigation aids for user orientation. Simplified layouts ease navigation, yet must convey sufficient data hierarchy for hypothesis formulation [9].

DOI:http://doi.org/10.24086/cocos2024/paper.1562

[11] introduces an innovative expanded dual quaternion method for conducting 3D coordinate transformation, which additionally enables the calculation of the variance-covariance (VCV) matrix for the transformation parameters. This report provides a straightforward presentation of the new dual quaternion algorithm (DQA), accompanied by two numerical examples for clarity and simplicity.

Using a 3D graph enhances data exploration by providing extra layout space, accommodating larger datasets, and facilitating visualization of complex structures. It aids in identifying data relationships, clusters, and significant nodes, enabling hypothesis formulation. Despite limited 3D visualization software, advancements in hardware and high-level languages offer improved performance [9].

However, 3D representations may complicate additional plot overlays and obscure details, like those in mountainous terrain. Despite a focus on 2D layouts, specialized algorithms and tools cater to 3D graph representation, offering more intuitive depictions supported by research on enhanced comprehension in 3D space [1].

[5] introduced a technique that merges homograph estimation and voxel mapping to enhance 3D reconstruction accuracy. This method efficiently reconstructs diverse objects, enhancing quality in a multi-camera environment.

Currently, there is a lack of software or algorithms explaining the process of converting paper graphs into 2D or 3D digital formats. Consequently, students heavily depend on existing applications for this task. Typically, companies developing drawing programs do not disclose the underlying code to users, leading to our research problem.

The aim of this research is achieved through the creation of algorithms for transforming 2D drawings into 3D representations and vice versa. This process utilizes a simplified code, enhancing student comprehension of the underlying drawing methods instead of solely relying on pre-existing applications.

II. 2D & 3D Applications in Social Environment

In a social environment, the below algorithms can be used to enable various features such as augmented reality effects, virtual avatar customization, or interactive 3D experiences [13], [14]. Transforming 2D coordinates into 3D coordinates in a social environment can be achieved using various algorithms and techniques, like:

A. Depth Mapping

We map each point (x, y) in 2D space to a corresponding depth value (z) in 3D space to get (x, y, z). This depth value represents how far the point is from the viewer's perspective. We can have used it such as in text, logos, or user-generated content.

B. Triangulation Algorithms

Can be used to convert a set of 2D points into a corresponding set of 3D triangles, which can then be rendered in 3D space, such as terrain elevation maps or 2D images with depth information.

C. Machine Learning-Based Approaches

Such as neural networks, can also be used to learn the mapping between 2D and 3D coordinates from training data.

III. Material and Methods

A. Digitizing Paper Graphs

As computers become central to experiments, converting paper-based data to digital form is essential. This research focuses on a mathematical algorithm for converting paper graphs into a function for creating and modifying 2D and 3D graphs easily. Digitizing hand-drawn graphs is crucial for effective scientific data analysis. Without it, the utility of data analysis programs is limited, hindering data interpretation and confirmation [15]. Simplifying analysis by directly digitizing hand-drawn graphs is invaluable, aiding students in understanding and communicating research findings.

B. Coordinate Systems

Prior to exploring the algorithm, it is imperative to comprehend the coordinate systems utilized in 2D and 3D graphics. Within a 2D space, points are conventionally expressed using Cartesian coordinates (x, y), with x denoting the horizontal position and y denoting the vertical position. Conversely, within a 3D space, points are represented utilizing Cartesian coordinates (x, y, z), where x, y, and z denote positions along the x, y, and z axes respectively, as shown in figures (1 & 2) [16], [17].

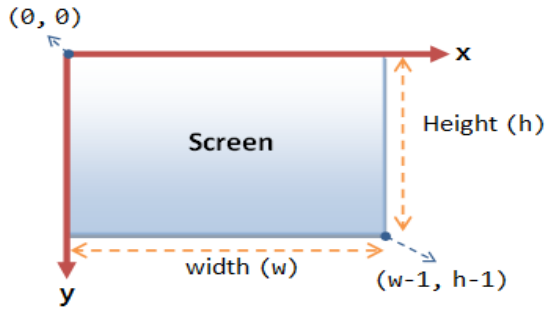

Fig. 1. The 2D screen coordinates

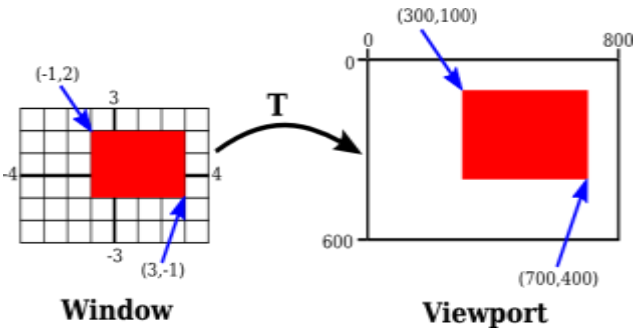

Fig. 2. An example of 2D screen coordinates

DOI:http://doi.org/10.24086/cocos2024/paper.1562

C. Algorithms for Transforming Coordinate Function Points

The Transformation section outlines the transformation process. It starts when the visualization matches the Initial Visualization State and the user provides inputs. The transformation continuously updates the visualization to an Intermediary Visualization State, changing its Schema and/or Geometric State. This ongoing modification forms the transformation's output before reaching a Final Visualization State (or returning to the initial state) [4], [18].

- The process of converting coordinate function points into 2D or 3D graphics encompasses several pivotal stages (as shown in figure 3):

  1. Function Definition: Initiate by establishing the mathematical function that encapsulates the points, curves, or surfaces intended for visualization. This function may manifest as a linear equation, polynomial, trigonometric function, or any other mathematical expression.

  2. Point Generation: Proceed by computing the function at regular intervals or specific points to generate a series of coordinate function points. These points serve as the foundational elements for constructing the graphical representation.

  3. Coordinate Transformation: In the realm of 2D graphics, map the generated points to screen coordinates based on the dimensions of the display window. This transformation entails operations such as scaling, translating, and potentially rotating the points to conform to the screen space.

  4. Projection (for 3D Graphics): For 3D graphics, projection becomes necessary to convert 3D coordinates into 2D screen coordinates. Common projection techniques encompass perspective projection and orthographic projection.

  5. Rendering: Following the transformation of points into screen coordinates, proceed with rendering using graphics primitives such as lines, polygons, or curves. This step involves linking the points to construct the desired visual depiction.

We need to transform the x-axis, that is:
(-a , 0) to (0 , 475) & (a , 0) to (1900, 475)
Let a = 10, so the equation is going to be:

$$Tx = 95x + 950 \qquad (1)$$

The transform of y-axis, that is:
(0 , b) to (950 , 0) & (0 , -b) to (950, 950)
Let b = 10, so the equation is going to be:

$$Ty = 475 - \frac{95y}{2} \qquad (2)$$

In general, for (a , b):
(1) become as:

$$Tx = \frac{950}{a}(x + a) \qquad (3)$$

(2) become as:

$$Ty = \frac{475}{b}(b - y) \qquad (4)$$

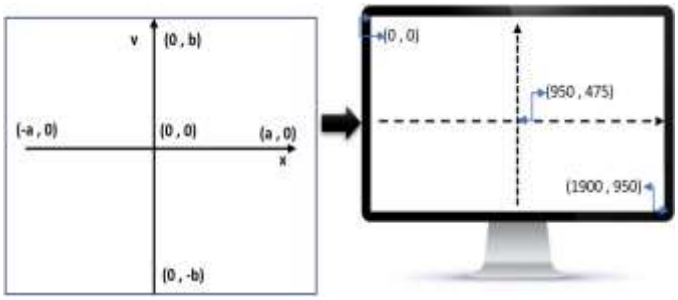

Fig. 3. Transform 2D paper coordinates to 2D screen coordinates

The algorithm for paper to 2D as follow:

Input a (x-axis limit)
Input b (y-axis limit)
Input p (steps)
Input $f(x)$ (the function to be graphed)
m=1
For x := -a to a step p
Begin
m = m - p
$y := f(x)$
if (y is unknown) then end;
$Tx := 950 * (x + a)/a$
$Ty := 475 * (b - y)/b$
Plot (Tx , Ty, m)
End.

- Transforming a 3D Point to 2D Perspective Projection

In 3D graphics, converting a 3D point to a 2D perspective projection is vital for rendering lifelike scenes on a 2D display. This conversion creates the illusion of depth and distance in computer-generated images (as shown in figure 4) [19].

Perspective projection represents 3D objects in a 2D space while maintaining depth perception. Points in 3D space are mapped onto a 2D plane, typically the screen, using geometric and optical principles. The resulting image mimics real-world object appearances. To convert a 3D point (x, y, z) to a 2D perspective projection, we employ various transformations considering the camera's position (viewpoint) and projection plane properties. The process is outlined as follows:

1. Translation: Initially, the 3D point is translated to match the camera's viewpoint, adjusting its position relative to the camera.

2. Projection: Subsequently, the translated point is projected onto the projection plane, determining its intersection with the plane based on its distance from the camera.

DOI:http://doi.org/10.24086/cocos2024/paper.1562

3. Normalization: Screen coordinates are obtained by normalizing the projected point, dividing its x and y coordinates by its z-coordinate. This step ensures proper scaling for objects at varying distances from the camera.

4. Viewport Transformation: Lastly, viewport transformations are applied to map the normalized coordinates to screen space. This includes scaling and translating the coordinates to fit within the screen boundaries.

5. In computer graphics, perspective projection is commonly implemented using matrices and transformations within rendering pipelines.

So the transform point of the x-axis will be:
(a , -b) to (0 , 950) and (-a , b) to (1900 , 0)
We need to transform the **x part** of x-axis, that is:
(a , 0) to (-a , 1900)
Let a = 10, so the equation is going to be:

$$Tx = -95x + 950 \quad (5)$$

We need to transform the **y part** of x-axis, that is:
(-b , 950) to (b , 0)
Let b = 10, so the equation is going to be:

$$Ty = 475 - \frac{95x}{2} \quad (6)$$

In general, for (a , b):
(5) become as:

$$Tx = \frac{950}{a}(x - a) \quad (7)$$

(6) become as:

$$Ty = \frac{475}{b}(b - x) \quad (8)$$

We will get the point (Tx , Ty), now we need to add the y & z coordinate, that is:
For y:

$$Ry = \frac{1900}{b}$$

$$Cy = Ry \times y$$

$$Txy = Tx + Cy \quad (9)$$

For z:

$$Rz = \frac{950}{c}$$

$$Cz = Rz \times z$$

$$Tyz = Ty + Cz \quad (10)$$

That is (x, y, z) is going to be (Txy, Tyz)

The algorithm for paper to 3D as follow:

Input a (x-axis limit)
Input b (y-axis limit)
Input c (z-axis limit)
Input p (steps)
Input $f(x, y, z)$ (the function to be graphed)
m=1
For x := -a to a step p
Begin
m = m - p
For y := -b to b step p
Begin
$z := f(x, y)$
if (z is unknown) then end;
$Tx := 950 * (x - a)/a$
$Ty := 475 * (b - x)/b$
$Ry = 1900/b$
$Txy = Tx + Ry \times y$
$Rz = 1900/c$
$Tyz = Ty + Rz \times z$
Plot (Txy, Tyz, m)
End.

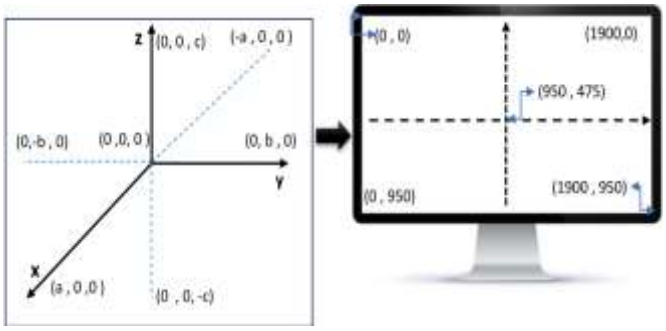

Fig. 4. Transform 3D paper coordinates to 2D screen coordinates

IV. Results and Discussion

To assess the potential impact of the proposed algorithms on user (student) interaction with 2D and 3D drawing, a simulation was initiated. Initially, a hypothetical function was drawn in 2D using two existing applications (Desmos [20], and GeoGebra [21]), followed by a rendition of the same function using the proposed 2D algorithm by using the MATLAB R2021b [22]. Subsequently, the process was repeated with a virtual function rendered in 3D (Desmos [23], and GeoGebra [24]). These comparisons aim to underscore the motivations driving the current research.

A. MATLAB Applications

- MATLAB is a powerful programming environment widely used in various fields, including engineering, science, and mathematics.

- When it comes to creating 2D and 3D drawings, MATLAB offers a range of functions and tools that make the process efficient.

- Here we used "plot (Tx, Ty, '0')", "figure" and "meshgrid (x, y)" functions. For 3D drawings, MATLAB provides functions like "graph3D()", "plot3 (Tx, Ty, Tz, '0')", and "scatter3" to create three-dimensional visualizations of data or mathematical functions.

- MATLAB provides thorough documentation and a supportive user community, aiding students in learning and problem-solving for their drawings, and allowing students to quickly understand how to use different plotting functions.

DOI:http://doi.org/10.24086/cocos2024/paper.1562

- Its user-friendly interface and robust plotting functions make it a top choice for students and researchers for creating both 2D and 3D drawings.

B. Simulation of 2D graph:

The simulation of the behavior of the basic trigonometric function (y = sin(x)) was conducted using the aforementioned suite of application programs, depicted in figures (5 and 6). Subsequently, a comparison was made with the proposed 2D algorithm, illustrated in figure 7.

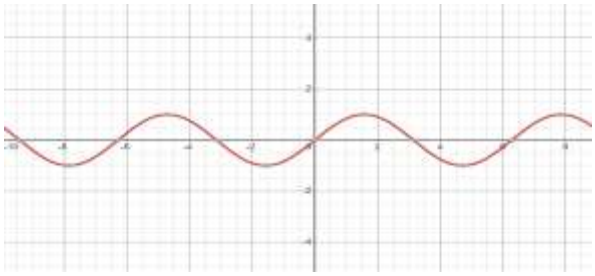

Fig. 5. Analysis spreadsheet: 2D simulation using Desmos application

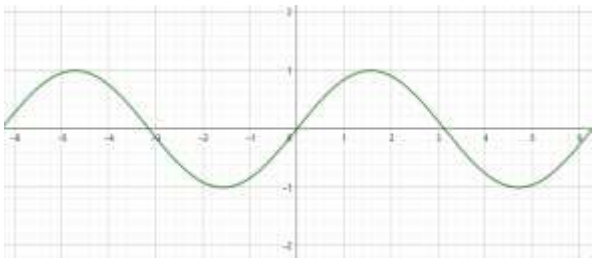

Fig. 6. Analysis spreadsheet: 2D simulation using 2D Calculator – GeoGebra application

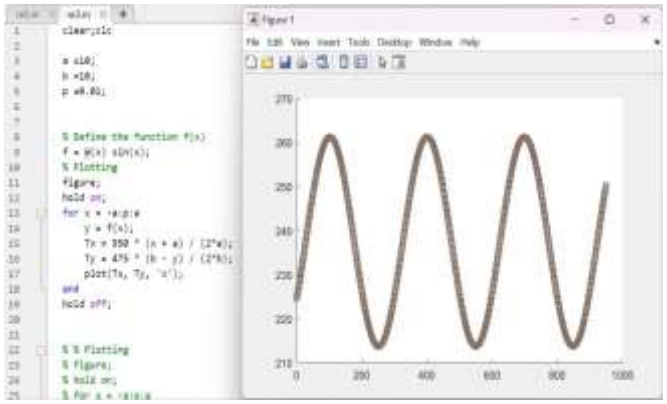

Fig. 7. Analysis spreadsheet: 2D simulation using MATLAB

C. Simulation of 3D graph:

The behavior simulation of the hypothetical function ($Z = x^2 + 2y$) was carried out using the aforementioned suite of application programs, depicted in figures (8, and 9). Following this, a comparison was made with the proposed 3D algorithm by using MATLAB R2021b, as illustrated in figure (10.a & 10.b).

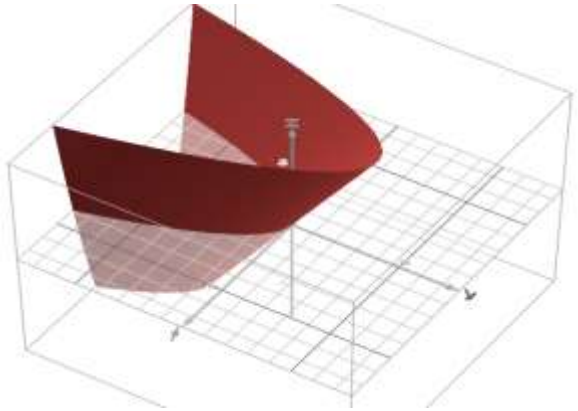

Fig. 8. Analysis spreadsheet: 3D simulation using Desmos3D application

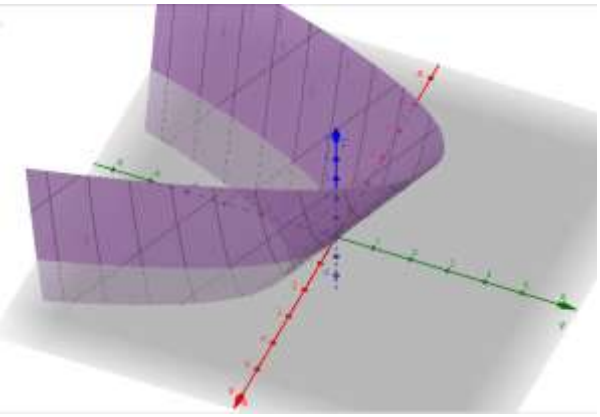

Fig. 9. Analysis spreadsheet: 3D simulation using 3D Calculator – GeoGebra application

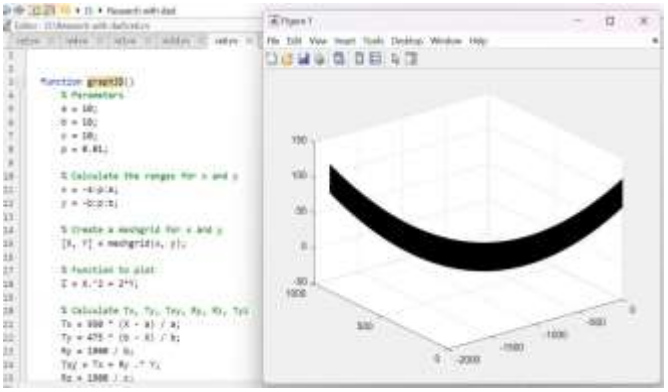

Fig. 10. Analysis spreadsheet: 3D simulation using MATLAB (a)

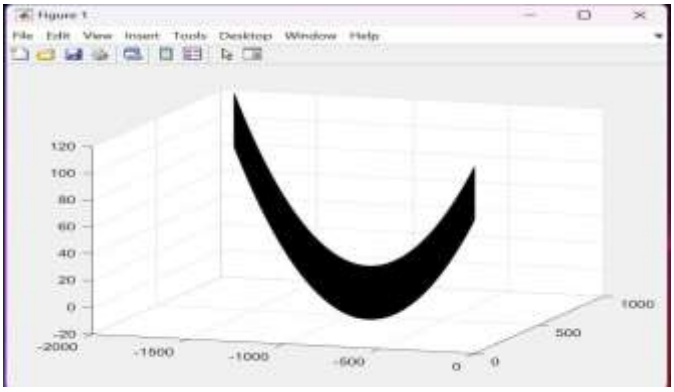

Fig. 11. Analysis spreadsheet: 3D simulation using MATLAB (b)

DOI:http://doi.org/10.24086/cocos2024/paper.1562

V. Conclusion

Converting a 3D point into a 2D perspective projection is essential for creating realistic images in computer graphics. By understanding the mathematical principles behind this transformation and implementing it effectively, developers can render immersive scenes with depth and perspective. Mastering perspective projection is key to producing visually compelling graphics in games, simulations, virtual reality, and other interactive applications. The proposed algorithms offer a streamlined approach to convert paper diagrams into 2D or 3D digital formats, potentially augmenting students' capacity to independently develop drawing programs.

VI. Future Works

A. Providing a starting point for student researchers to enhance existing software and websites, particularly after detecting algorithms and codes used here in a simplified manner, which large companies typically refuse to disclose.

B. Motivating students and researchers to uncover algorithms for other programs.

References

[1] P. Gajer, M. T. Goodrich, and S. G. Kobourov, "A fast multi-dimensional algorithm for drawing large graphs," in *Graph Drawing'00 Conference Proceedings*, 2000, pp. 211–221.

[2] B. N. Mohammed, F. H. Al-Mukhtar, R. Z. Yousif, and Y. S. Almashhadani, "Automatic classification of COVID-19 chest X-ray images using local binary pattern and binary particle swarm optimization for feature selection," *Cihan Univ. Sci. J.*, vol. 5, no. 2, pp. 46–51, 2021.

[3] B. N. Mohammed, F. H. Al-Mukhtar, R. Z. Yousif, and Y. S. Almashhadani, "Automatic Classification of Coronavirus Disease-19 Chest X-Ray Images Using Local Binary Pattern and Binary Particle Swarm Optimization for Feature Selection," *Cihan Univ. Sci. J.*, 2021.

[4] B. Lee, M. Cordeil, A. Prouzeau, B. Jenny, and T. Dwyer, "A design space for data visualisation transformations between 2d and 3d in mixed-reality environments," in *Proceedings of the 2022 CHI conference on human factors in computing systems*, 2022, pp. 1–14.

[5] T. Jadhav, K. Singh, and A. Abhyankar, "Volumetric 3D reconstruction of real objects using voxel mapping approach in a multiple-camera environment," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 26, no. 2, pp. 755–767, 2018.

[6] R. I. Yousif and N. H. Salman, "Image compression based on arithmetic coding algorithm," *Iraqi J. Sci.*, pp. 329–334, 2021.

[7] R. Freedle, *Artificial intelligence and the future of testing*. Psychology Press, 2014.

[8] W. T Kahwachi and H. Q Birdawod, "A New Hybridization of Bilateral and Wavelet Filters for Noisy De-Noisy Images," *Eurasian J. Sci. Eng.*, vol. 9, no. 1, 2023.

[9] D. Danilov, "3D Graph Exploration." Master thesis, Tartu, 2010.

[10] E. Chu, Y. Chen, C. Raissi, and A. Bhojan, "CharNeRF: 3D Character Generation from Concept Art," in *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, 2024, pp. 185–194.

[11] S. Bektas, "An expanded dual quaternion algorithm for 3D Helmert transformation and determination of the VCV matrix of the transformation's parameters," *J. Spat. Sci.*, pp. 1–16, 2023.

[12] A. Abu-Monshar, A. Al-Bazi, and Q. H. Al-Salami, "On the Development of a Multilayered Agent-based Heuristic System for Vehicle Routing Problem under Random Vehicle Breakdown," *Cihan Univ. Sci. J.*, vol. 5, no. 1, pp. 1–10, 2021.

[13] W. A. Hashim, J. H. Hameed, R. J. Ismail, and M. Q. Al-Obaidi, "Optimizing Power Controller Performance using Genetic Algorithm: A Step-by-Step Approach for Finding Optimal Values and Improving Control Performance," in *2023 16th International Conference on Developments in eSystems Engineering (DeSE)*, 2023, pp. 87–93.

[14] I. T. Abbas and Q. H. Al-Salami, "Inverted Generational Distance Bat Algorithm for Many-Objective Optimization Problems".

[15] V. Agrawal, J. Jagtap, and M. V. V. P. Kantipudi, "An Overview of Hand-Drawn Diagram Recognition Methods and Applications," *IEEE Access*, 2024.

[16] R. Shih, *Learning SOLIDWORKS 2024: Modeling, Assembly and Analysis*. SDC Publications, 2024.

[17] R. J. Ismail, "Mining Tutors' Interesting Areas to Develop Researched Papers Using a Proposed Educational Data Mining System," *Eng. Technol. J.*, vol. 30, no. 10, 2010.

[18] R. Nemirovsky and T. Noble, "On mathematical visualization and the place where we live," *Educ. Stud. Math.*, vol. 33, no. 2, pp. 99–131, 1997.

[19] F. Steinicke, G. Bruder, and S. Kuhl, "Realistic perspective projections for virtual objects and environments," *ACM Trans. Graph.*, vol. 30, no. 5, pp. 1–10, 2011.

[20] "Desmos2D." https://www.desmos.com/calculator (accessed Mar. 14, 2024).

[21] "GeoGebra2D." https://www.geogebra.org/classic#2d (accessed Mar. 14, 2024).

[22] "MATLAB R2021b." https://www.mathworks.com/products/matlab-online.html (accessed Mar. 10, 2024).

[23] "Desmos3D." https://www.desmos.com/3d (accessed Mar. 16, 2024).

[24] "GeoGebra3D." https://www.geogebra.org/3d (accessed Mar. 16, 2024).

DOI:http://doi.org/10.24086/cocos2024/paper.1562