International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at :www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET

Performance Optimization Techniques : Improving Software Responsiveness

Santosh Panendra Bandaru
Independent Researcher, USA

Article Info
Volume 8, Issue 2
Page Number : 486-495

Publication Issue :
March-April-2021

Article History
Accepted : 10 April 2021
Published: 30 April 2021

ABSTRACT

Due to the fact that object-oriented programming promotes folding code into tiny, reusable components, which increases the frequency of these costly activities, dynamically dispatched calls often restrict the performance of object-oriented applications. One aspect of more recent health or functional status measures that hasn't gotten much attention is how sensitive they are to clinical changes over time. Recent advancements in dynamic window systems, particularly in different attachment methods, have greatly enhanced windows' energy, thermal, and optical performance. The number of power grid devices in the national power system is growing, the power grid's structure is getting more complicated, and the distribution network's information construction is getting better, but real-time visualisation is still comparatively poor. In terms of information ionisation, the way information is now presented and interacted with is unable to keep up with the daily operations and maintenance management of the distribution network and the fast growth of the power supply scale. In this research, we examine responsive visualisation solutions for multiple terminals and grid panoramic visualisation display technologies based on geographic information systems. These technologies not only increase the distribution network's monitoring effectiveness but also shorten the time needed to identify and fix issues as they arise. Additionally, it makes it possible to visualise topological data and quickly get the information that is needed. In order to optimise both the visualisation components and the GIS rendering, the studies in this study use clustering techniques, which significantly increases the visualisation efficiency.

Keywords: - Visualization Efficiency, GIS-Based, Dynamic Window Structures, Clustering Methods, Monitoring Efficiency, Factoring Code, Energy Performances, Object-Oriented Programs, Power Grid

I. INTRODUCTION

The construction industry's substantial energy use draws attention to the sustainability problem in this sector. While some studies have examined sustainability throughout building operations, others have concentrated on sustainability during construction [1]. Because of its enormous impact on energy consumption, the building façade has been recognised in several publications as an important aspect of building operation [1, 2]. Numerous articles have examined the importance of the façade in relation to building energy consumption and occupant comfort, and methods for enhancing the overall performance of the façade have been proposed [2].

The main reason why traditional façades don't work well for building energy efficiency and comfort is that they are static and don't adapt to changes in the external environment. In light of this, the dynamic façade is one suggested remedy that adapts the façade to both interior demand and exterior conditions [2, 3]. The dynamic façade's methodology is mostly determined by the issue it is intended to address. The design and development of this dynamic glazing system was motivated by the need to reduce solar heat intake while increasing window thermal efficiency in response to seasonal and diurnal variations [3, 4]. Consequently, the ideal situation is one in which solar heat gain is high during the heating-demand phase and low during the cooling-demand phase. To control solar heat gain, a variety of glazing technologies were established, including coatings, glazing systems, automated dynamic shading systems, automatic blind systems, and chromic-based films [3, 4]. Certain commercial buildings have used chromatic-based dynamic systems, such as electrochromic and thermochromic dynamic glazing's, but demand has been significantly reduced due to restricted supply and expensive product and installation costs [5].

For modern distributed systems to be both responsive and economical, they must have strong methods for handling performance variations. This is mostly because the performance of the underlying infrastructure is stochastic [5, 6], the demand varies, and potential faults are frequent in today's sophisticated computer systems. In order to detect environmental changes and respond appropriately, contemporary distributed systems must include self-adaptive capabilities [5, 6]. Although several approaches have been put forward and put into practice, relatively little study has been done on the use of control theory-based solutions.

Because of its efficacy, predictability, and mathematical assurances, control theory has been the preferred method for many adaptive systems, particularly in physical systems [6, 7]. Accurate performance models of the computer software system are necessary in order to use control-theory based methodologies to efficiently manage the performance of large-scale software systems. However, for today's large distributed systems, creating models with a sufficient level of accuracy has proved to be time-consuming and prone to errors [5, 8]. As a result, the existing theoretical methods for ad hoc control are application-specific and have not been shown to be transferable to other systems.

The Reliability, validity, and responsiveness to clinical changes over time are important considerations in the development and testing of scales used to measure function or "health status." Reactivity is important when using functional scales as outcome measures in clinical trials because it influences a trial's statistical power, or its capacity to identify a difference between treatments when one exists [5, 6]. Assessing responsiveness to change is not as standardised as methodologies, terminology, and statistics for evaluating validity and reliability [5, 8]. Consequently, this attribute has not been given enough consideration in the creation of functional scales. Moreover, there is almost no comparison evidence to suggest which scale may be more responsive when there are competing scales for a certain goal [8, 9]. We contend that evaluating a functional scale's responsiveness is comparable to evaluating a diagnostic test's discriminating capabilities. Whether or whether a clinically significant change has taken place is the condition that has to be "diagnosed" in this instance. Like other

diagnostic tests, functional scale scores are subject to random fluctuation over time and are never entirely accurate [8, 9]. Functional ratings will thus fluctuate over time in both "true positive" and "false positive" ways. Therefore, sensitivity and specificity in identifying improvement or deterioration, as determined by other criteria, may be used to characterise a scale's responsiveness [8, 9]. The problem lies not only in being sensitive to change but also in being able to distinguish between those who become better and those who don't.

The success of today's software systems depends heavily on performance. Nonetheless, a lot of software products fall short of their performance goals when they are first developed. Resolving these issues is expensive and leads to a number of challenges, including missed market windows, lost revenues, lost productivity, cost overruns, schedule delays, and strained customer relations [8, 9]. In severe situations, performance issues may not be resolved short of a comprehensive redesign and re-implementation. In certain situations, the project is either gratefully cancelled or turns into an endless waste of time and money. Performance has to be built into software from the start and cannot be retrofitted [9, 10]. The strategy of "make it run, make it run right, make it run fast" is risky.

Recent research on software architectures has shown how crucial architecture is to meeting software quality goals, such as performance. Although choices are crucial at every stage of the development process, architectural choices have the most influence on quality traits including performance, dependability, reusability, and modifiability [11]. According to Clements and Northrup:

> *"The choice of architecture has a significant impact on whether a system will be able to display the intended (or necessary) quality qualities."*

In our experience, poor design decisions—rather than ineffective coding—are often the cause of performance issues. It could be too late to tune the architecture to obtain sufficient performance by the time it is corrected [11,12]. A bad design might hinder the accomplishment of performance goals, but a good architecture cannot ensure that they be met [13].

1.1 Overview of Software Performance Engineering

In order to identify issues with the system architecture, design, or implementation plans, SPE is a model-based method that leverages purposefully simplistic models of software processing [14, 15]. To get feedback on whether the suggested software is likely to satisfy performance requirements, these models are simple to create and solve. The models are improved during the course of the software development process to better reflect the performance of the final product. The accuracy of the resource needs estimations determines how accurate the model findings are [16].

SPE employs adaptive techniques, such upper- and lower-bounds estimations and best- and worst-case analysis, to manage uncertainty since they are hard to predict for software structures. For instance, analysts utilise estimations of the upper and lower limits of these values when there is a significant degree of uncertainty about resource needs. Analysts forecast both the best-case and worst-case performance using these projections [15, 16]. They look for workable alternatives if the anticipated best-case performance is not adequate. They go on to the next stage of the development process if the worst-case scenario is acceptable. In the event that the outcomes fall somewhere in the middle, evaluations pinpoint the crucial elements whose resource estimations have the most impact and concentrate on gathering more accurate data for them [16, 17].

More accuracy may be achieved by a number of methods, such as improving the design and creating more intricate models or building performance prototypes and calculating the resource needs of important parts. The software execution model and the system execution model are the two sorts of models that provide data for architectural evaluation. The UML models of the software are the source of the software execution model. It

stands for important facets of the behaviour of program execution [17]. Workload situations are represented via the use of execution graphs in its construction. Arcs show the control flow, whereas nodes show the software's functional components. The graphs are hierarchical, offering comprehensive information on predicted resource needs at the lowest level.

A static study of the mean, [18], best-, and worst-case reaction times is obtained by solving the program execution model. It describes the resource needs of the suggested program just, without the presence of additional workloads, many users, or delays brought on by resource competition. It is not necessary to build more complex models if the expected performance without these extra performance deciding elements is inadequate. In most cases, software execution models are enough to detect performance issues brought on by bad architectural choices [19].

One crucial component of the smart grid is the distribution grid, which is directly linked to every consumer and the electricity system. Gathering and compiling distribution grid data, user data, grid structure, and geographic information is the goal of the intelligent distribution operation and maintenance management system [11]. In order to offer early warning in the event of accidents or distribution network failures, this system is utilised to monitor the operational state of the distribution system. The information presentation of the existing distribution network is straightforward, the data source is comparatively single, the visualisation is not very high in real time, and the conventional distribution network is still monitored and managed by routine human inspection [1, 11]. The safety, dependability, and efficiency of the distribution cluster are all negatively impacted when a power equipment malfunction occurs since it is difficult to promptly locate the appropriate problem location for repair [18].

In order to visualise responsive distribution network panorama monitoring, this article evaluates the present distribution network using GIS and topology data. It makes possible the full presentation of equipment ledger data, topology data, and distribution network operation data, as well as the panoramic monitoring of the operating state of the network. Layer customisation, multi-layer overlay, and a variety of display techniques, including hover boxes, micro-charts, and unique features, are all realised [19]. Use the logical and commercial linkages of the distribution network automation panoramic data to realise interactive distribution network automation.

II. TOPOLOGY MODEL AND GIS-BASED VISUALIZATION

2.1 Topological model construction

Using the concepts of topological geometry, topological data structures are a means of arranging spatial data. Topological data structures for distribution network graphics [18, 19], only take into account the abstract concept of the relationships between the distribution network representation elements (points, lines, and surfaces) and ignore the coordinates and positions of nodes and line segments, focussing instead on their adjacency and linkage relationships. A significant amount of data is produced by every link in the distribution network, including power plant generation, line transmission, substation transformation, electricity distribution in distribution stations, customer level consumption, and dispatching in operational and upkeep classes [19, 20].

During the power generating process, the power plant creates data on power production, operation monitoring, and equipment overhaul; logs, telematics, telemetry, remote control, and telecontrol are among the various data created during the operation of power equipment. Grid data is a carrier of data information that is visually conveyed via data visualisation [19, 20]. The primary topological connections and features of typical transportation networks are analysed in this study, constructed as diagrams, and stored in the form of diagram

databases. The substation, transformer, pole tower, switch, and line relationships, as well as the topological relationships they establish, are saved to the schematic database while other data is saved to the relational database [2, 21]. Define the nodes of the sibling and single-data parent-child relationships based on the relationships between distribution transformer that has been pole tower, switch, and line for later display and computation. Figure 1 [22] displays the topological model.

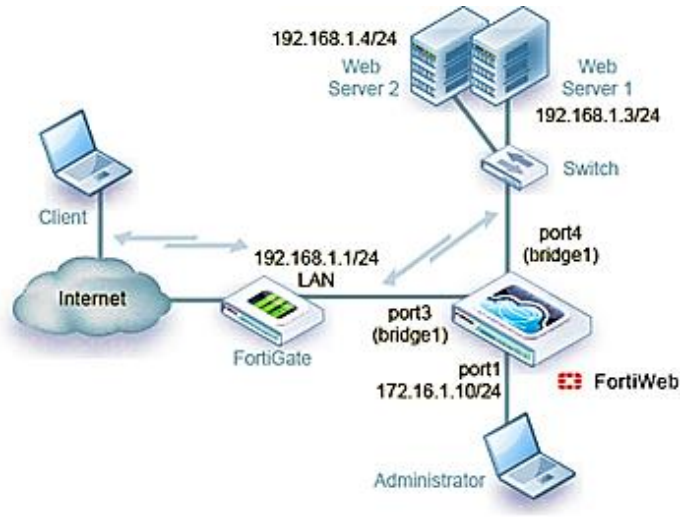

Fig. 1 Topology mode. [23]

The node contains the object's ID, name, voltage level, and coordinates for its latitude and longitude. A line segment connecting nodes is represented by the connection relationship using the line ID, line name, and other information [22]. In the real service, a line is a connection relationship with the same line ID, and the line is installed underneath a number of distribution substations, towers, switches, and other node objects [22].

The open-source program Neo4j is selected as the graph database to record topological connection relations in this research after a thorough evaluation of the commercial and open-source graph databases in terms of performance, [23], openness, and security.

2.2 Application of GIS

GIS is an all-encompassing high-tech field that combines computer science, geometry, geography, and a variety of application models. GIS offers great assistance for the administration, analysis, and maintenance of complex power grid data in the power system due to its robust spatial data management and analytical capabilities [23, 24]. It really helps in the decision-making process for applications. The finest GIS development platform and information visualisation environment is the power system's complicated structure and vast geographic distribution [22], which makes it ideal for power system monitoring, including transmission, distribution, and power-using systems.

2.3 Visualization Platform Construction

As shown in Figure 2, the visualisation is separated into three layers: [29], map visualisation, topology visualisation, and data visualisation.

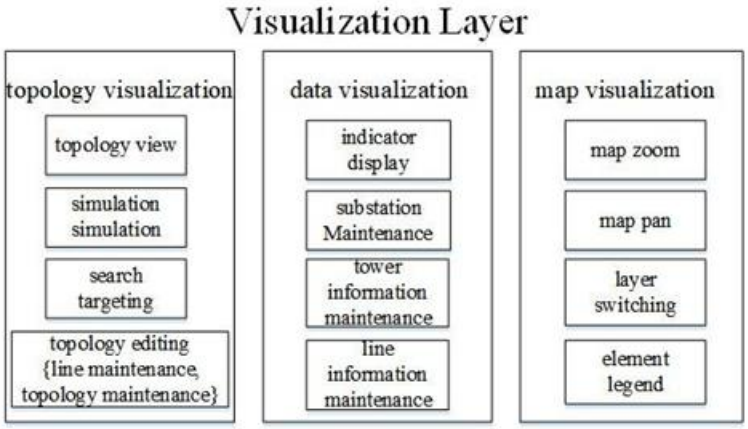

Fig. 2 Visualization layer. [16]

You can zoom in and out and switch between various visuals by lowering the map by column to load and show the map visualisation. In order to swiftly and accurately find the relevant place and examine on-site information based on your needs, it may move between levels. In order to visually depict the topology of the power system's equipment, including substations, towers, distribution substations, cable wells, branch boxes, lines, and other items, topology visualisation adds topology layers based on maps and topology data [19]. It creates the distribution network topology based on their geographic position, including their latitude, longitude, and connection to one another. Main lines, branch lines, feeders, and [19, 20] are all distinguished differently, and the equipment that has to be interrogated is identified and concealed.

In order to examine or manage associated equipment ledger, power, fault, and outage data, data visualisation employs topology components that are tied to ledger, power, fault, and outage data [20]. This enables direct interaction with the elements on the topology diagram. First, the geographic information engine's building is finished in order to create an area map. The distribution network topology data is retrieved from the map, and we overlay the substation, tower, and line topologies on the geographic map.

In order to facilitate human-computer interaction, things like towers, substations, and lines are simultaneously data-bound with the model [22], and the ledger data of the item is seen. Corresponding pages are offered in the interim to show company statistics and indications.

III. OPTIMISATION

This study optimises the visualisation picture once the topological model and visualisation platform have been constructed. Latitude and longitude data are used to superimpose the element points on the map [23], and the size and colour of the points indicate the sum of the index value. This value is the result of extensive processing and computation.

The quantity of data for points within the map's viewable range will increase as the map is zoomed in and scaled down to smaller layers; tens of thousands or even hundreds of thousands of points may need to be shown [23]. Optimising the presentation of enormous data volumes is more important since the existing general browsers are having trouble presenting points of this magnitude. Those that are geographically near to one another within a certain range on the map are combined into a single point in this study using a technique called clustering. In order to arrive at a total score, the points' values are determined by subjectively and objectively weighing many indicators [24]. The k-means clustering technique determines the points' locations. This lowers the data level of pointers by a single degree, which may significantly enhance the system's performance.

3.1 Graph element fusion optimization

For a certain zoom level S of the map, [24, 25], each element that needs to be rendered is $(\boldsymbol{P_0}, \boldsymbol{P_1}, \boldsymbol{P_2},) \ldots\ldots\ldots \boldsymbol{P_n}$. The latitude and longitude coordinates of each point are $((\boldsymbol{X_0}, \boldsymbol{y_0}), (\boldsymbol{X_1}, \boldsymbol{X_1}), (\boldsymbol{X_2}, \boldsymbol{y_2}), \ldots\ldots\ldots (\boldsymbol{X_n}, \boldsymbol{y_n})$, precise stages for clustering using the k-means method based on each point's distance are as follows::

1. Using the scaled hierarchy to determine the number of clusters K.
2. Choose K locations at random from each item to serve as the starting centre of mass.
3. Allocating the remaining items to the closest mass centre [25].
4. Use techniques like the mean to update the class's core value.
5. Repeat $\boldsymbol{S1}, \boldsymbol{S2}, \boldsymbol{S3}$ until the K center points no longer change.

For each cluster point after clustering, $\boldsymbol{Pa}, \boldsymbol{Pb}, \boldsymbol{Pc} \ldots, \boldsymbol{Pn}$. According to the distance as the weight value, calculate the coordinates of the center point $(\boldsymbol{Xc}, \boldsymbol{Xc})$ is $\boldsymbol{Vc},$. The processes involved in the computation are listed in Table 1 [25, 26].

Table 1 Experimental Parameters. [26]

Samples Point	*Initials Distance*	*Correction Factors*	*Weight*
(X_{c1}, X_{c1})	Y_{i1}	$1/Y_{i1}$	λ_1
(X_{c2}, X_{c2})	Y_{i2}	$1/Y_{i2}$	λ_2
---	---	---	---
(X_{ck}, X_{ck})	Y_{ik}	$1/Y_{ik}$	λ_k

3.2 Determine the sample sequence

Determine the amount of distance between each test point and the centre of the cluster in each category based on the clustering findings of the previous procedures [26, 27], then arrange them from big to small, shown as:

$Y_i = [Y_{i1}, Y_{i2,\ldots\ldots} Y_{ik}]$ …….1

$Y_i^* = \left[\frac{1}{Y_{i1}}, \frac{1}{Y_{i2}}, \ldots\ldots \frac{1}{Y_{ik}}, \right]$ ……2

$\lambda_1 = 1/Y_{ij} / \sum_{j=1}^{k} 1/y_{ij}$ ……3

$Y_i^* = \sum_{j=1}^{k} \lambda_j V_{ij},$ ……4

S1. Calculate the above Table 2 [28, 29] to get the location of each category index value Vi∗ in all index values.

Table 2 Characteristics of the algorithm. [29, 30]

Class Indicator Class	*R*	*G*	*B*
Y_i^*	$\boldsymbol{255\times(1-\mu_1)}$	$\boldsymbol{255\times\mu_1}$	**0**
Y_2^*	$\boldsymbol{255\times(1-\mu_2)}$	$\boldsymbol{255\times\mu_2}$	**0**
Y_k^*	$\boldsymbol{255\times(1-\mu_k)}$	$\boldsymbol{255\times\mu_k}$	**0**

$\mu_i = \frac{Y_i^* - V}{V_{max} - V_{min}} (i = 1,2,..,K)$ ……5

S2. Determine the RGB values that correspond to each category's index values using Table 2 [22] above.

$R: 255 \times (1 - \mu_i), i = 1,2, \ldots, K, G: 255 \times \mu_i, i = 1,2, \ldots, K$ ……..6

The technique presented in this work increases the number of primitives that may be shown in a single window from 10,000 to 100,000 in real applications. The rendering time for showing 10,000 primitive points from 0.91 to 0.13 seconds Figure 3, 4 [28].

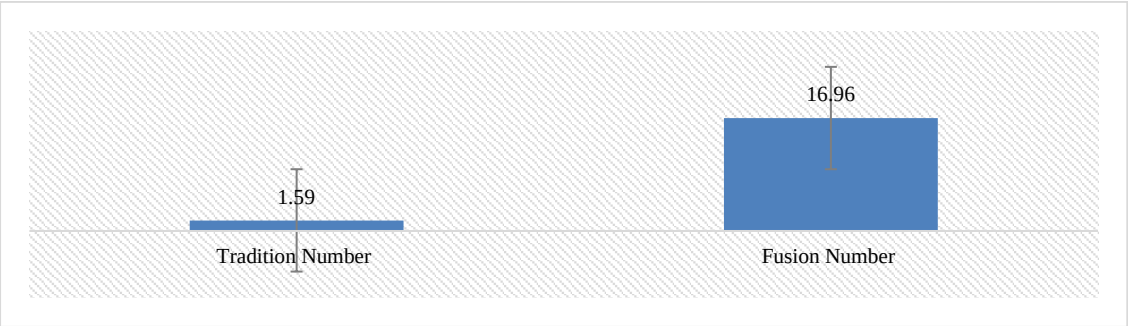

Fig. 3 Prior to and during optimisation, the number of primitives. [22]

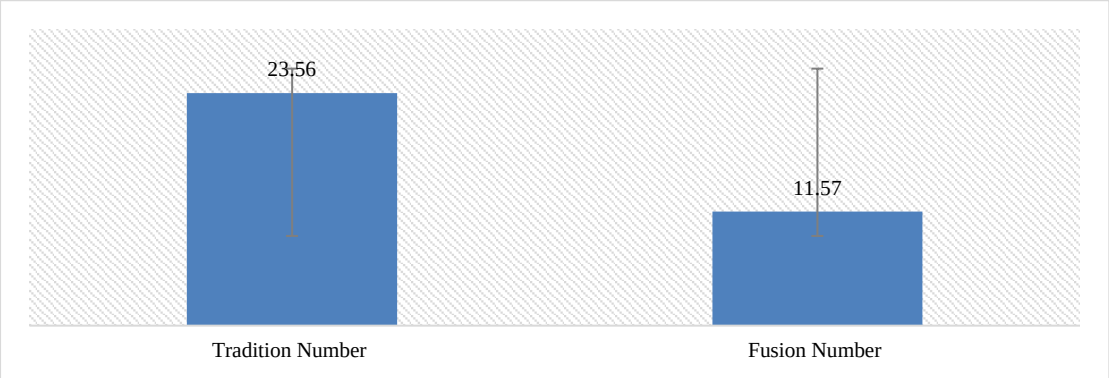

Fig. 4 Rendering duration before to and after optimisation. [19]

3.3 3D GIS rendering optimization

Support for 3D simulation models is introduced to GIS to more accurately and efficiently illustrate how real lines, substations, etc. operate [19, 20]. The Levels of Detail (LOD) technology, which describes how GIS processes models ranging from basic to sophisticated, is the primary means of representing models in GIS [22]. Experiments were carried out to evaluate the model's rendering time and frame rate in the scene, and the page's performance was assessed using Chrome developer tools [19, 20]. The original scenario without data hierarchical and the mixed model scene with LOD hierarchy were used for the performance testing. After that, a software was written to render the scene and read out the frame rate. The original model and the LOD1toLOD4 models each experienced a frame rate fluctuation of 100 seconds during this test trial. The resulting data was then shown as a matching frame rate variations graph, as seen in Figure 5 [23].

The softness of the rendering procedure is indicated by the frame rate; the smoother the image, the greater the frame rate. The figures show that LOD1-LOD2 ends up at 60 frames per second [22, 23], LOD3 at 40 frames per second, and LOD4 at 35 frames per second. The ultimate stable values of the LOD1-LOD4 frame rate curves are greater than those of the original model in all instances of direct renderings of the original model, which ends up at 20 pfs [23].

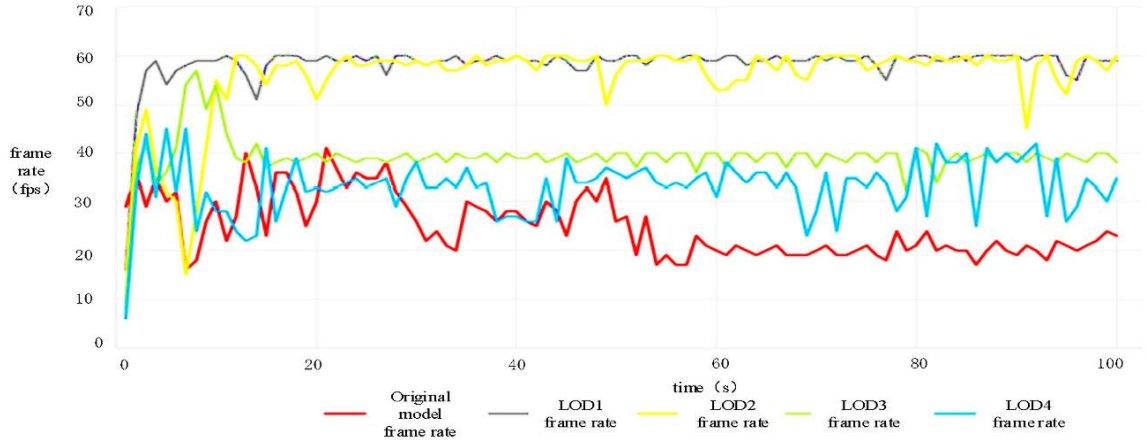

Fig. 5 Rate of frame curve comparison. [24]

Based on the graph analysis, scene browsing will be more fluid if the view is at LOD1-LOD2 level when the huge volume of model interior and model details are not drawn if the user is not interested in model details [26, 27]. Only one facility is displayed, regardless of whether the user takes an interest in a model and navigates to the LOD4 level to examine its features. In addition to avoiding the resource-wasting drawback of the system rendering all entire models regardless of the perspective, this significantly reduces the workload on the system [19]. It enhances the systematic scheduling and administration of geographical data and may significantly boost visualisation effectiveness.

3.4 Experiment analysis

The optimisation approach based on graph component fusion and LOD hierarchical rendering, as well as the distribution system topology model and visual monitoring architectural suggested in this article [19, 30]. Large-scale graphical components and 2-3D fusion are shown in a panoramic manner. First, the topological model is built, as shown in Figure 6; second, a GIS-based platform for visualising panoramic monitoring is built; and last, experiments are conducted to optimise graph fusion and visualise the model using multi-layer LOD rendering, as shown in Figure 6 [19].

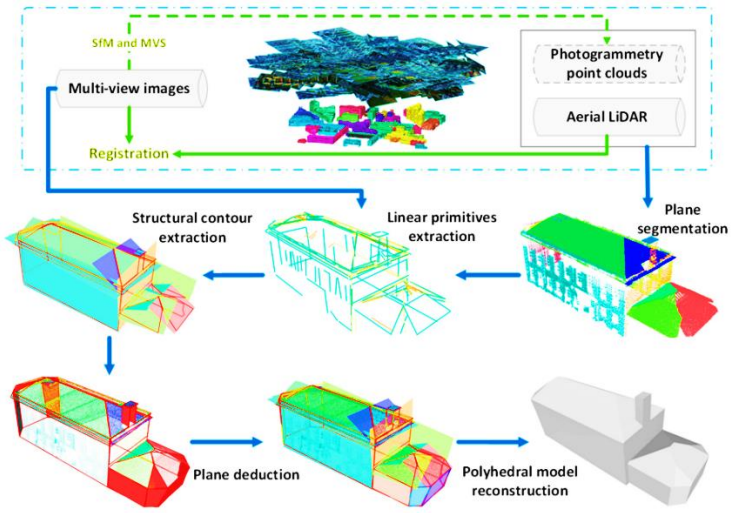

Fig. 6 Primitive presentation and topology. [11]

IV. CONCLUSION

This article examines a GIS-based panoramic monitoring system for distribution networks that combines geographic data, network frame equipment status metrics, and distribution network topology information. It

reveals the underlying rules and values behind the power data by realising a panoramic three-dimensional multi-dimensional picture of line and geographic information from the views of time, space, topological dimension, geography, and business dimension. The distribution network's operating condition may be easily seen via the large-screen visualisation display, which also enables quick access to regular information about the network's quality and safety, energy efficiency, and other aspects. Future distribution grid panoramic monitoring visualisation design and development projects may benefit greatly from mastering the GIS-based distribution grid panoramic monitoring technique presented in this research.

REFERENCES

[1] Deyo RA: Measuring functional outcomes in therapeutic trials for chronic disease. Controlled Clin Trials 5: 223-240, 1984.

[2] Deyo RA, Inui TS: Toward clinical applications of health status measures: sensitivity of scales to clinically important changes. Health Serv Res 19: 275-289, 1984.

[3] Bergner M, Bobbitt RA, Carter WB, Gilson BS: The Sickness Impact Profile: development and final revision of a health status measure. Med Care 19: 787-805, 1981.

[4] Roland M, Morris R: A study of the natural history of back pain. Part I: Development of a reliable and sensitive measure of disability in low-back pain. Spine 8: 141-144, 1983.

[5] Deyo RA: Pitfalls in measuring the health status of Mexican Americans: comparative validity of the English and Spanish Sickness Impact Profile. Am J Public Health 74: 569-573, 1984.

[6] Carter WB, Bobbitt RA, Bergner M et al: Validation of an interval scaling: the Sickness Impact Profile. Health Serv Res II: 516-528, 1976.

[7] Barone G, Zacharopoulos A, Buonomano A, et al. (2021). Concentrating PhotoVoltaic glazing (CoPVG) system: modelling and simulation of smart building façade. Energy, 238: 121597.

[8] Bui DK, Nguyen TN, Ghazlan A, et al. (2021). Biomimetic adaptive electrochromic windows for enhancing building energy efficiency. Applied Energy, 300: 117341.

[9] Cedeno Laurent JG, Samuelson HW, Chen Y (2017). The impact of window opening and other occupant behavior on simulated energy performance in residence halls. Building Simulation, 10: 963–976.

[10] Chel A, Kaushik G (2018). Renewable energy technologies for sustainable development of energy efficient building. Alexandria Engineering Journal, 57: 655–669.

[11] Curcija DC, Zhu L, Czarnecki S, et al. (2015). Berkeley Lab WINDOW. Berkeley, CA, USA.

[12] Dabbagh M, Krarti M (2021). Energy performance of switchable window insulated shades for US residential buildings. Journal of Building Engineering, 43: 102584.

[13] de Vries SB, Loonen RCGM, Hensen JLM (2021). Multi-state verticalblinds solar shading—Performance assessment and recommended development directions. Journal of Building Engineering, 40: 102743.

[14] D. Arcelli, V. Cortellessa, A. Filieri, and A. Leva. 2015. Control theory for model-based performance-driven software adaptation. In 2015 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA). 11–20.

[15] M. Arlitt and T. Jin. 2000. A Workload Characterization Study of the 1998 World Cup Web Site. Netwrk. Mag. of Global Internetwkg 14, 3 (2000), 30–37.

[16] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. 2010. A view of cloud computing. Commun. ACM 53, 4 (2010), 50–58.

[17] L. Baresi and S. Guinea. 2013. Event-Based Multi-level Service Monitoring. In 2013 IEEE 20th International Conference on Web Services. 83–90.

[18] Luciano Baresi, Sam Guinea, Alberto Leva, and Giovanni Quattrocchi. 2016. A discrete-time feedback controller for containerized cloud applications. (2016), 217–228.

[19] C. Barna, M. Fokaefs, M. Litoiu, M. Shtern, and J. Wigglesworth. 2016. Cloud Adaptation with Control Theory in Industrial Clouds. In 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW). 231–238.

[20] Cornel Barna, Hamzeh Khazaei, Marios Fokaefs, and Marin Litoiu. 2017. Delivering Elastic Containerized Cloud Applications to Enable DevOps. IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2017 (2017), 65–75.

[21] Cornel Barna, Hamzeh Khazaei, Marios Fokaefs, and Marin Litoiu. 2017. A DevOps Architecture for Continuous Delivery of Containerized Big Data Applications. In International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). IEEE.

[22] A. Valerio, “Visualization system integrated for electric power distribution networks,” IEEE Latin America Transactions, vol. 8, no. 6, pp. 728–733, 2010.

[23] P. C. Wong, K. Schneider, P. Mackey, H. Foote, G. Chin Jr, R. Guttromson, and J. Thomas, “Anovel visualization technique for electric power grid analytics,” IEEE Transactions onVisualization and Computer Graphics, vol. 15, no. 3, pp. 410–423, 2009.

[24] B. Sun, H. Li, H. Guo, and T. Li, “A study of smart system of power utilization safety management based on a cloud platform,” in 2019 IEEE 4th Advanced Information Technology, Electronicand Automation Control Conference (IAEAC), vol. 1. IEEE, 2019, pp. 1446–1450.

[25] L. Shuo, Q. Liwen, Y. Xiaoyong, Z. Yangjun, and L. Shan, “Research on guangxi multi- dimensional visualization platform construction of distribution network based on bigdataarchitecture,” in 2019 2nd International Conference on Safety Produce Informatization (IICSPI). IEEE, 2019, pp. 56–60.

[26] S. S. S. R. Depuru, L. Wang, V. Devabhaktuni, and N. Gudi, “Smart meters for power grid—challenges, issues, advantages and status,” in 2011 IEEE/PES Power Systems Conferenceand Exposition. IEEE, 2011, pp. 1–7.

[27] D. Wei, M. Zhaoyong, C. Yaomin, and Z. Boxi, “Automatic generation for single-line diagramof distribution network,” in 2016 IEEE 11th Conference on Industrial Electronics andApplications (ICIEA). IEEE, 2016, pp. 348–353.

[28] Cornel Barna, Marin Litoiu, Marios Fokaefs, Mark Shtern, and Joe Wigglesworth. 2018. Runtime Performance Management for Cloud Applications with Adaptive Controllers. In Proceedings of the 2018 ACM/SPEC International Conference on Performance Engineering (Berlin, Germany). ACM, New York, NY, USA, 176–183.

[29] Xiaolin Chang, Ruofan Xia, Jogesh K Muppala, Kishor S Trivedi, and Jiqiang Liu. 2016. Effective modeling approach for IaaS data center performance analysis under heterogeneous workload. IEEE Transactions on Cloud Computing 6, 4 (2016), 991–1003.

[30] HÖLZLE, U. AND AGESEN, O. 1994. Dynamic vs. static optimization techniques for objectoriented languages. Theor. Prac. Obj. Syst. 1, 3.