A Grammar for 3D Visualization and a Cloud-Distributed Implementation

Submitted June 2024, in partial fulfillment of
the conditions for the award of the degree **PhD Computer Science.**

by
Manu Antony

Supervised by Prof. Nick Holliman

School of Computing
University of Newcastle

Signature Manu

Date 27 / 06 / 2024

Abstract

In many real-time control situations, including traffic management centres, the quantity and surge of events can overwhelm the operator's cognitive ability to understand and react to event notifications. Research has shown that effective visualizations that provide context and personalization to the operator can improve their ability to interpret visualizations. This thesis investigates the efficacy of employing a 3D visualization grammar for prototyping 3D visualizations. It also tries to compare how geo-representation improves context for operators in understanding the underlying information in contrast to the traditional text based systems.

The literature proposes visualization design techniques for effective use in the traffic saturation visualization. Different declarative systems and their system architectures were also investigated for the design of the 3D visualization grammar. In this context, Aloka - a 3D visualization grammar, was designed and developed. It allows users to define 3D visualizations in a declarative specification in a JavaScript Object Notation (JSON) file. It also provides a novel way to customize 3D visualization types and offers a flexible approach towards different data types, display types(VR, HDTV etc) and data mapping to visual representations. The design and development of Aloka is explored, its system architecture dissected and it's philosophy explained. This is a novel approach to creating complex 3D visualizations for a wide range of systems by defining the specification of a 3D visualization by data-binding it to visual properties for high quality ray traced rendering. Experiments were carried out to find that operators using a custom designed 3D geo-visualization over the existing text based systems UTC (Urban Traffic Control) improved context and significantly speeds up response time in detecting traffic saturation. These 3D visualizations were prototyped using Aloka. This project is sponsored by an Engineering and Physical Sciences Research Council (EPSRC) iCASE award from Siemens PLC, Poole.

Publications

- N. S. Holliman, M. Antony, J. Charlton, S. Dowsland, P. James and M. Turner, "Petascale Cloud Supercomputing for Terapixel Visualization of a Digital Twin," in IEEE Transactions on Cloud Computing. doi: 10.1109/TCC.2019.2958087 keywords: Data Visualization;Internet of Things;Scalability;Supercomputers,
  URL:
  http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8931236&isnumber=6562694

- "Aloka a 3D Visualization Grammar" - Fourth Annual UK Systems Research Challenges Workshop 2019 @ Redworth Hall Hotel, Surtees Rd

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Nick Holliman for the continuous support of my Ph.D study and research, for his patience, motivation, and immense knowledge. His guidance has continually helped me in throughout the research and writing of this thesis. I also thank his complete support when I needed the most during my study. I am incredibly lucky to have him as my advisor and mentor for my Ph.D study. I would also like to thank Graham Street for his help when I was at Siemens to aid with the experiments and support.

I thank my furry friend Leo who provided me with emotional support and love during turbulent times. I would also like to thank my parents, Mr Mekkattuparambil Lona Antony and Dr Grace Maria Antony and also thank my sister Dr Anu Antony and my brother Akhil Jose for his support during my PhD. I also thank my friends for their support and help to maintain my sanity through the isolating lockdowns.

Finally, I acknowledge and thank both Siemens and EPSRC for funding the PhD.

Acronyms

AR Augmented Reality. 8

ASTRID Autonomous Tram in Depot. 101

AWS Amazon Web Services. 38

BNF Backus–Naur Form. 73

CAD Computer-aided Design. 33

CLI Command Line Interface. 102

CNN Convolutional Neural Networks. 42

CRT Cathode-Ray Tube. 17

CSS Cascading Style Sheets. 37

D3 Data Driven Documents. 14, 61

DCNN Deep Convolutional Neural Networks. 21

DOM Document Object Model. 55, 56, 61

DSL Domain-Specific Language. 7

E-FRP Event-Driven Functional Reactive Programming. 60

EBNF Extended Backus–Naur Form. xv, 73, 75–77, 79, 81, 83, 85, 87

EPSRC Engineering and Physical Sciences Research Council. i

GIS Geographic Information System. 33

GPU Graphics Processing Unit. 8

HCI Human-Computer Interaction. 7

HTML Hypertext Markup Language. 37

iCASE Industrial Cooperative Awards in Science & Technology. i

JSON JavaScript Object Notation. i, 67

LiDAR Light Detection and Ranging. 98

LOC Locus of Control. 7

LOD Level of Detail. xxi, xxii, 44, 45, 107, 108

LSTM Long Short-Term Memory. 41, 42

NFTD Node Fine Tuning. xxii, 102

OGC Open Geospatial Consortium. 33

PaaS Platform as a Service. 113

PICT UTC Application Function to Display Selected Picture. xxii, 103

PTSD Post Traumatic Stress Disorder. 24

RFTD Region Fine Tuning Display. 101

SCOOT Split Cycle Offset Optimisation Technique. 42

SOA Service Oriented Architecture. 39

SVG Scalable Vector Graphics. 37

UICE Urban Insight Cloud Engine. 50

USD Universal Scene Description. 67

UTC Urban Traffic Control. 9

UTMC Urban Traffic Management Control. 50

VNG Virtual Newcastle Gateshead. 50, 107

VR Virtual Reality. 8

VRML Virtual Reality Modelling Language. 5

Glossary

Aloka A 3D Visualization Grammar. 10

saliency The property of salience is defined by the variance in detecting an object in relation to its background. 103

Nomenclature

$CO2$ Carbon Di-Oxide

$NO2$ Nitrogen Di-Oxide

Contents

Abstract i

Publications ii

Acknowledgements iii

Acronyms iii

Glossary vii

Nomenclature ix

List of Tables xix

List of Figures xxi

1 Introduction 1

1.1 Background Summary 2

1.1.1 The Importance of 3D in visualisation 5

1.1.2 Barriers to design in 3D 6

1.1.3 Using Declarative Design 7

1.2 Research Questions 8

1.3 Contributions 9

1.4 Thesis Structure 10

2 Background and Related Work 17

2.1 Physiology of Human Vision 17

2.1.1 Limits of Human Vision . . . 17
2.1.2 Color and Light Sensitivity . . . 18
2.1.3 Color Blindness . . . 19
2.1.4 Visual Attention . . . 20
2.1.5 Object Recognition . . . 21
2.1.6 Perception Adaptations and Object Categorization . . . 22
2.1.7 Visual Clutter . . . 22
2.1.8 Memory . . . 23
2.2 Designing for Display Types . . . 24
2.2.1 2D Display Size . . . 25
2.2.2 Virtual Reality . . . 25
2.2.3 Augmented Reality . . . 27
2.3 Personalization . . . 28
2.3.1 Culture . . . 28
2.3.2 Language . . . 29
2.3.3 Individual Preference . . . 30
2.3.4 Adapting to Bias . . . 31
2.4 3D Visualization . . . 32
2.4.1 GeoVisualization . . . 33
2.4.2 Flow Visualization . . . 33
2.4.3 3D Parallel Coordinates . . . 34
2.4.4 Architectural Design . . . 35
2.5 Declarative Design Systems . . . 36
2.6 Declarative Grammars for Visualization . . . 36
2.6.1 Vega-lite . . . 37
2.7 Cloud Computing . . . 38
2.8 Petascale Cloud Supercomputing for Terapixel Visualization of a Digital Twin . . . 40
2.9 Traffic Visualization . . . 41

2.9.1 Visual Analytics System 41
2.9.2 Geographic and temporal visualization of New York taxis 43
2.9.3 High-Resolution and High-Dimensional visual analysis of traffic data 45

3 In Pursuit of a Grammar for 3D Visualization 49

3.1 3D Visualizations 49
3.1.1 Data Visualization in Blender 50
3.1.2 Immersive Visualizations 51
3.1.3 Gigapixel & Terapixel Visualization 53
3.2 Design Objectives for a new System 53
3.3 Exploring grammars for visualization 55
3.3.1 D3 - the Javascript library 55
3.3.2 Vega - a Visualization Grammar 56
3.4 Challenges of 3D Grammar Design 62
3.5 Aloka System Design 65
3.5.1 System Architecture 65
3.6 Aloka API (Application Programming Interface) 67
3.6.1 Client 68
3.6.2 Server 70

4 Aloka - a Grammar for 3D Visualization 73

4.1 Aloka in Extended Backus–Naur Form (EBNF) 73
4.1.1 Basic Components 74
4.1.2 EBNF Grammar Components 76
4.1.3 Common Variables 77
4.1.4 Aloka Specification 77
4.2 A 3D Visualization Grammar 89
4.2.1 The View Composition 89
4.2.2 Data Processing 91
4.2.3 Data Binding Design 92

4.2.4 Abstract Scene-Graph 93
4.3 Chapter Summary 94

5 Case Study of Visualizations using Aloka 97

5.1 Temperature Visualization using Aloka 97
5.1.1 Data Sources 97
5.1.2 Temperature Glyph 98
5.2 Traffic Saturation 99
5.2.1 The Research Problem 99
5.2.2 Existing UTC System 100
5.3 Design of Traffic Saturation Visualization 103
5.3.1 Saliency tests 103
5.3.2 Glyph Design 105
5.3.3 Design Proposals 107
5.4 Defining Traffic Saturation Visualization in Aloka 113
5.4.1 Data Sources 113
5.4.2 Design translation into Aloka 114

6 Experiments and Results 115

6.1 Experiment 115
6.1.1 Purpose 115
6.1.2 Experiment Design 115
6.1.3 Parameters 115
6.1.4 Benefits 117
6.1.5 Consent Form Presented to Subjects 117
6.2 Experiment Methodology 118
6.2.1 Testing The Design Effectiveness 118
6.2.2 PsychoPy 119
6.2.3 Statistical Analysis 119
6.3 Experiment results 121

6.3.1 Experts . . . 121
6.3.2 Non-Experts . . . 123
6.4 Analysed Results . . . 125
6.4.1 Confusion Matrices and Statistical Analysis of Results . . . 126

7 Discussion **131**
7.1 Key Findings . . . 132
7.2 Interpreting the results . . . 132
7.2.1 Exploring User Feedback . . . 133
7.3 Limitations . . . 135
7.3.1 Limitations in study . . . 135
7.3.2 Limitations of Aloka . . . 135

8 Summary and Conclusions **137**
8.1 Thesis Conclusion . . . 137
8.2 Future Work and Direction . . . 138

Bibliography **139**

Appendices **151**

A Grammar Definition **151**
A.1 Resource Class . . . 151
A.1.1 Data Class . . . 151
A.1.2 Camera Class . . . 152
A.1.3 Shader Class . . . 153
A.1.4 Material Class . . . 154
A.1.5 Lights Class . . . 155
A.1.6 Scales Class . . . 156
A.2 Renderer Class . . . 160
A.2.1 Name . . . 160
A.2.2 Platform . . . 160

A.2.3 Type . . . 160
A.2.4 Engine . . . 160
A.2.5 Settings . . . 161
A.2.6 Output . . . 161
A.3 Filter Class . . . 162
A.3.1 Data . . . 162
A.3.2 Scripts . . . 162
A.4 Transforms Class . . . 163
A.4.1 Name . . . 163
A.4.2 Type . . . 164
A.4.3 Functions . . . 164
A.5 Geometric transforms Class . . . 166
A.5.1 Name . . . 166
A.5.2 Type . . . 166
A.5.3 Position . . . 167
A.5.4 Rotation . . . 167
A.5.5 Scaled . . . 167
A.6 View Class . . . 167
A.6.1 Name . . . 167
A.6.2 Layers . . . 167
A.6.3 Settings . . . 169
A.7 Start Point . . . 170

List of Tables

4.1 The proposed ISO/IEC 14977 standard in [Scowen, 1998] page 7, table 1 . 74

6.1 Experiment data for experts 122

6.2 Experiment data for non-experts 124

List of Figures

1.1 The Thesis Structure 11

2.1 The visual analytics system developed by [Lee et al., 2020] © 2022 IEEE . 42
2.2 The Geographic and temporal visualization model developed by [Ferreira et al., 2013] © 2022 IEEE 43
2.3 Different spatial visualizations of taxi trips for the week 05/01/2011 to 05/07/2011 in the TaxiVis model used several LOD developed by [Ferreira et al., 2013] © 2022 IEEE 44
2.4 Visualization of several trips within a user-defined area elaborated by a dot-map. The temporal context is represented as an animation © 2022 IEEE 47
2.5 Clock like visualization used to represent number of data points, average speed and CO2 emissions done by [Häussler et al., 2018] © 2022 IEEE . . 48

3.1 Temperature Visualization 51
3.2 Stereoscopic image rendered on the left and the VR image rendered on the right 52
3.3 360 VR image rendered 52
3.4 Gigapixel visualization of Temperature 54
3.5 The system architecture and pipeline of Aloka 66
3.6 The Json Editor used to Edit Aloka Specification 69
3.7 The Server to Client Connection and Pipeline showing the Importing of Data 70
3.8 The Rendered Temperature Visualization 71

4.1 The View Structure . . . 89
4.2 Example of the view in Aloka . . . 90
4.3 The view's layer structure . . . 90
4.4 The filter grammar example . . . 91
4.5 The transform example . . . 92
4.6 The resource class . . . 92
4.7 The data example . . . 93
4.8 The scales example . . . 93
4.9 The geometric transforms example . . . 94

5.1 The 3D model of Newcastle Upon Tyne city . . . 98
5.2 A simple glyph design to visualize data by color in a complex scene . . . 99
5.3 3D rendered temperature visualization generated by Aloka API . . . 99
5.4 Queue length in a road link explained . . . 101
5.5 The UTC Node Fine Tuning (NFTD) System . . . 102
5.6 The bar colors shows the queue length . . . 102
5.7 The UTC Application Function to Display Selected Picture (PICT) Display system used in UTC . . . 103
5.8 The Arrow glyph design . . . 106
5.9 Level of Detail (LOD) Design 1 . . . 107
5.10 Level of Detail (LOD) design 2 . . . 108
5.11 The Level of Detail (LOD) 3 which provides a 360 VR image . . . 108
5.12 Node Star Design 1 . . . 109
5.13 Node Star Design 2 . . . 109
5.14 Node Star Design 3 . . . 110
5.15 A figure with Node link design with both a main and summary page . . . 110
5.16 A figure with Node view design with both a main and summary page . . . 111
5.17 A simple Grayscale design . . . 111
5.18 3D-arrow Design . . . 112
5.19 Queue Lengths Design . . . 112

5.20 Simple Node Design 113
5.21 The Traffic Saturation visualization generated by Aloka API 114

6.1 The visualisation is on left is a aloka rendered visualization, and the UTC system display is on right 116
6.2 The Visualization scale generated by Aloka - a closeup view. 116
6.3 The UTC NFTD system - a closeup view to show saturation. 116
6.4 The formulae for calculating the *t-statistic* is given. 121
6.5 The graph above is visualizing the comparison between response times for each user in this category 123
6.6 The graph above is visualizing the comparison between response times for each user in this category 124
6.7 Confusion Matrix generated for both rendered and UTC dataset for experts 126
6.8 T-test result for experts 126
6.9 Confusion Matrix generated for both rendered and UTC dataset for non-experts 127
6.10 T-test result for non-experts 128

Chapter 1

Introduction

Semiotics was defined as the study of signs and cultures where "signs" stands for something other than itself. The concept of "Grammar in Visualisation" can be traced to 1967 when Jacques Bertin first published the semiology of graphics in French. The objective of using signs, symbols or graphics in data visualisation is to communicate meaningfully. In order to create a visualisation, one needs to understand the system of related information and user tasks. Bertin focused on the core principles of how we can create visual representations that can be understood. The importance of this representation was to group these signs based on human perception, to understand the relationship between various symbols and to generate significant evidence that can be understood in a universal manner. By defining and exploring eight visual (retinal) variables (these are size, shape, texture, colour, value (brightness), orientation and the X and Y planar dimensions), he extended the scope of visual representations [Bertin, 1967]. Expanding on Bertin's work, "Grammar in Visualisation" made a comeback in 1999 when Leland Wilkinson published the "grammar of graphics" [Wilkinson, 1999]. The book explored grammatical rules for creating graphs and charts. It also focused on creating more complex graphical forms using a grammar. Universally, the grammar of any system is declarative in design. This is because grammar should define what the rules are and not how it is implemented. Data visualisation can thus be defined as the representation of information in graphical format that follows a set of rules called "grammar" or the principles used for data analysis and communication. Given that we are already in the era of big data, it has become even

more imperative to process large datasets efficiently and effectively using such rules

1.1 Background Summary

Visualisation of data has been an indispensable tool in research as it constitutes a powerful method for conveying results of scientific research to others in a comprehensible manner, where effective visualisation can aid users to analyse and understand evidence based data more effectively. To achieve this, firstly, data should be represented using visual elements that help the user to understand and perceive the structure of the data. Secondly, visualisation should follow a logical structure and in order to identify the structure of data, a theory of visualisation is vital for effectively mapping data (for example magnitude data is well represented using colour maps of monotone luminance). And finally the integration of these methodologies into the visualisation software with respect to the grammar can provide a detailed understanding of the data. Interactive methodologies allow user to understand and explore data in a way that was unfamiliar previously. Similar to the process in data mining where the right test or method is extracted to suit the user using assumptions of the algorithm, a similar guidance is required using evidence based on meta-data, visual task, user perception and the cognitive capacity of the user. The key elements thus identified to comprehend and analyse useful patterns in data are visualisation, semantics and interactive methods [Dhieb, 2019, Eastman, 1986]. Visualisation is a thus an informative technique used to create diagrams, images, or animations to relay on a message. The applied grammars can vary in the degree of expressivity where low-level grammars include Protovis, Vega, and D3 and high- level grammars consist of ggplot2, Tableau and Vega-Lite. This study[Satyanarayan et al., 2017] presents Vega-Lite, a high-level syntax that allows rapid specification by combining traditional and novel grammars of interactive visual interaction. It extends the foundations laid by other graphics grammars, automatically generating the necessary data flow with more succinct specifications compared to the more verbose Vega language. [Satyanarayan et al., 2017] provide an algebra to compose the grammar's specifications from a single-view to a multi-view display. In Vega-Lite, a scale typically derives its domain values directly from the encoded field of

a channel. However, users have the option to customize these domain values by specifying the scale's domain property. Consequently, they explored the possibility of whether merging Vega-Lite scale domains or maintaining their independence would be more beneficial. To develop a preeminent interaction grammar, the providers create several selections to act as scale domains which eventually construct a detailed interaction. Vega-Lite also has several operators that can transform these selections by adding, removing, or manipulating specific points [Satyanarayan et al., 2017].

Communication through visualisation can be extracted by encoding data as visual objects (e.g., lines, bars or scatter points) within graphics. In order to communicate these ideas effectively, functionality and aesthetic representation can provide important details in a more interactive and effective way. However, sometimes, achieving this level of intricacy can be lost in either functionality or appeal, which is why a balance is required to effectively communicate data through interactive visualisation. Alongside the technical aspects of visualisation, human perception is an important factor in understanding the data. Human perception and cognition (ability to process information through memory, learning, attention, problem solving and forming thought) can help designers to understand the ability of the users to comprehend data (quantitative or qualitative) through visualisation. This is represented through effective visualisation, which begins at understanding the implication of human perception and application of this understanding into insightful visualisations This can be done effectively through statistical graphics, plots and informative graphics for qualitative data, while quantitative data is encoded using scatter points, lines, or bars.

Effective visualisation can make complex data more easily accessible which is reasoned with evidence. There has been a recent explosion of data and has not been synced with the users' capabilities to integrate and implement this information and visualisation is a vital means of linking this gap [Ziemkiewicz et al., 2012]. Although most visualisation techniques neglect individual dissimilarities and use a one-size-fits-all method. One proposed solution is to use adaptive visualisation. However, in order to generate appropriate adaptive visualisation tools, there is a need to understand the user relationship between

the context and the visualisation using semantics [Oscar et al., 2017]. The theory of visualisation projects human perceptions of the meaning of data depending on the transformation of the data and what meaning can be interpreted by the user. To achieve this, a deep dive into high level user data like attention, aesthetics and decision-making are required. A study [Witzel et al., 2021] conducted by social scientists used online studies to measure health and monitor individual well-being in their day to day life. The users were questioned daily or several times in a day about their emotional well-being and their activity throughout the day. This resulted in a large data set, which can be a treasure chest for visualisation if done correctly. This can be applied to engage users and encourage certain behaviours while also being used as tool for researchers. A personalised dashboard was also created to engage with the users and designers can chose what data to display based on user preferences using an algorithm. The algorithm was informed using the data from [Witzel et al., 2021] project which was a 100-day self-regulatory study of older participants. A focus group was also conducted to make decisions about customised users preferences by the researchers [Pham et al., 2012].

Interactive visualization uncovers hidden data structures by transforming data for better understanding. For instance, a user might use color markers to highlight specific ranges, effectively isolating and revealing underlying structures by concealing unselected areas. Alternatively, data can be transformed through mathematical manipulations. This visualization approach grants data analysts greater dynamic control over the display using graphic concepts. Additionally, it allows for enhancements and refinements based on user interactions, offering a more intuitive and responsive analytical experience. Vega-Lite uses interactive methods like panning and zooming in order to extract high specifications of the data [Satyanarayan et al., 2017]. Vega-Lite's high-level approach streamlines the process of creating complex visualizations but may restrict the level of customization and detailed control that users have, especially when compared to lower-level visualization tools. This restriction could pose challenges for users in need of highly tailored or unconventional visualizations. A suggested future direction is to devise models that accommodate these advanced customization features for interactive techniques. Such visualizations become

crucial, particularly for data that is time-sensitive (either real-time or geospatial) since representing this multidimensional data in 2D formats can be challenging. Geospatial visualizations emphasize observing the impact of time on specific variables. However, deriving answers to certain questions remains problematic with static visualization designs, indicating a need for more dynamic and adaptable visualization strategies.

1.1.1 The Importance of 3D in visualisation

Jacques Bertin's concept of visual variables suggests that showing time effectively is tricky because time is ongoing and extensive. It's important to find good ways to show ongoing data visually. However, this need makes it hard to use certain visual elements like color, shape, texture, and orientation, because they may not work well with time-based data. These elements often don't vary enough to show differences clearly over time. Also, showing timelines of events, which might be based on specific data points or changes in data, can be challenging in animations. To use animations effectively, it's important to think carefully about how to clearly and fully show time changes and events [Bertin, 1967]. Event based animations are usually effective for monitoring real time data in simulations [Matković et al., 2010]. Geospatial visualisations although difficult in small scale visualisations are extremely important in crisis and resource management, diplomacy, urban planning and traffic management. Although there are several tools for analysis of such data, the visualisation of such data has not yet been mastered. Representing specific answers to complex questions using time-based geographical data is even more challenging. This clearly requires a multi-dimensional approach as the visualisation of 3D objects cannot be replicated by 2D representations. To simulate 3D visualizations, variables such as perspective, shading, movement, and stereo display are required.The majority of image synthesis software can render along with shading calculations but the pictures produced are static. While other tools like Virtual Reality Modelling Language (VRML) browser focus on the dynamic visualisation, the image rendering is weak.

1.1.2 Barriers to design in 3D

In order to design or develop 3D visualisation some major barriers were identified. Although detailed research on the individual factors describing the user exists in psychology, there is little to no agreement on which factors are more relevant to visualisation. The design and structural factors describing the visualisation has no real standard language to use when deconstructing a visualisation or there is no real understanding of the aspects of the individual user that are significant for visualisation. Perhaps a more difficult question is how to uncover information about which aspects of a visualisation are significant to the individual user, as currently there are no standardised set of dimensions on which to analyse, let alone synthesise visual designs. It is thus imperative to describe and define factors of the visual design itself. The visualisation theory has no language that can easily describe the differences between the visual designs in relation to the locus of control. Although the visualisations are similar in terms of basic visual mapping and the significant design differences are mostly at the structural or metaphorical level, there is still no systematic way to fit structural design differences into a visual mapping schema. [Pinker and Feedle, 1990] provided an abstract representation of a visual design produced by decomposition analysis and reported that it could be measured and analysed more quantitatively, producing metrics that researchers can correlate to individual personality factors [Pinker and Feedle, 1990]. While it is widely believed that a dataset with more detail is more valuable for spatial analyses, this assumption has not been well examined. Another question that entails along with this argument that has not been investigated is the cost of producing such detailed visualisation which has either too much or too little information with respect to the user's requirement for detail in three dimensions and if they vary with the individual personality traits like gender, age, visual cues etc.

The impact of the 3D visualisation delivery methods depends on each individuals understanding of the data and investigations need to be made to see if the personalisation of these visualisation actually enhance the personal comprehension of the actual data and if this will be reflected in any decision making. However, it is to be noted that although 3D visualisations varies across displays and devices, the impact of this depends on the indi-

vidual understanding of data and range of personalisation. This aids in decisions making in terms of speed/accuracy of knowledge gained through such customised visualisation. Cognitive-psychology research has also shown that differences in user's experiences, personality and cognitive abilities, can influence the performance of a task as it determines the user's dexterity to interact with the tool. However, the design factors that lead to such user behaviour based on cognitive factors like perceptive capability, spatial and verbal ability, memory, perceptual speed and reasoning across several users need to be examined as revealed by [Velez et al., 2005]. The effect of personality traits in complex tasks especially in visualisation as per this research [ping Zhang and Tsingan, 2014] was said to be impactful, using a 5-factor model (openness to experience, conscientiousness, extraversion, agreeableness, and neuroticism), as a correlation was observed with Human-Computer Interaction (HCI) in [ping Zhang and Tsingan, 2014] and these personality factors correlate significantly with respect to the user's penchant for visual-interface designs.

The study "How Visualization Layout Relates to Locus of Control (LOC) and Other Personality Factors" by Caroline Ziemkiewicz and colleagues explores the relationship between personality traits, specifically LOC, and the effectiveness of different visualization designs. The research found that individuals with an internal LOC perform better with linear, list-based visualizations, reflecting their preference for structure and direct control. In contrast, those with an external LOC were more adept at using complex, containment-based visualizations, which align with their perception of external influences shaping their environment. These findings suggest that visualization tools should be designed with consideration for the user's personality traits to enhance usability and effectiveness, potentially leading to the development of adaptive systems that tailor visual layouts according to individual differences in LOC [Ziemkiewicz et al., 2013].

1.1.3 Using Declarative Design

Declarative design is to programmatically train a system on what the user wants instead of how user wants it. However, in practice, this method uses a Domain-Specific Language (DSL) to provide what the user wants and protecting them from low-level structures like

(loops, conditionals, assignments) which provide the result. Declarative languages is used to extract facts and deduce rules distinctly from the control flow [Heer and Bostock, 2010]. The main advantage of using declarative language is that the problem and the process of solution is distinctive while providing readability (usability) due to being closer to English than to pseudo code which is much easier to read and learn by non-programmer, succinctness where the meaningless lines are abstracted by DSL, reusability where the DSL can be used for multiple functions rather than being restrictive. Although declarative systems such as Vega/Vega-Lite, ProtoVis, Lyra, D3 etc., exists, there is no current system that can be extended to multiple dimensions.

Aloka, the declarative language designed and developed as demonstrated in this thesis, attempts to bridge this gap by providing a visualization grammar for 3D visualization. It also explores personalisation of visualisations by processing graphics for individual users. By adding support for multiple display types such as 3D Stereoscopic displays, Virtual Reality (VR) / Augmented Reality (AR) and targeting different render platforms, the system tries to expand its scope and reach.

1.2 Research Questions

Research Problem:

In many real-time control situations, including traffic management centres, the quantity and speed of events can outpace an operator's cognitive ability to comprehend and react to event notifications. The need for a complex visualization system to generate and deliver 3D geo-visualizations exists within the traffic management domain.

Cloud computing and the latest IaaS graphics hardware in the cloud (such as NVIDIA's virtual Graphics Processing Unit (GPU)) allow a new approach to 3D visualisation, where rather than producing a single visualisation we can chose to dynamically produce the right one for the current user, data or client computing capacity. This will allow us to personalise the visualisation to the clients needs for 3D visualizations being rendered on server or client.

In this research, the aim is to use traffic data, produce new visualisations using cloud

technology that are dynamic and adapt to the user and their context. We will test these visualisations on a range of devices and with a range of subjects (people with and without expertise using Urban Traffic Control (UTC) systems) and check that the differences in the presentations to different users do not change the understanding that these users have of the underlying key features of the traffic saturation, but do improve their personal comprehension in terms of speed or accuracy of knowledge gained and appropriateness of decisions made.

Research Question:

> **This research aims to understand if using grammar for 3D visualisation is useful in prototyping effective designs for decision making scenarios**

The following, is the list of research milestones that will be achieved through this research.

- Define a visualization grammar that enables encoding of 3D visualizations.
- Utilize the visualization grammar to prototype a 3D visualization design specifically for analyzing traffic saturation.
- Implement a visualization grammar that effectively supports 3D visualizations.
- Design a 3D visualization of traffic saturation that improves operators' ability to identify saturated road links, in comparison with existing Urban Traffic Control (UTC) systems.

1.3 Contributions

This study asserts that its principal contributions to the field are the enhancement of the scope of 3D Visualization Designs and the creation of an innovative declarative system for 3D visualizations. This research also contributes by introducing novel 3D visualisation designs for visualizing traffic saturation for operator use cases.

Expanding Scope for 3D Visualizations: The scope of 3D visualizations have been expanded by exploring data binding to shaders within Aloka (3D Visualization Grammar) and opening the avenue to many possibilities in design and visualization. This is discussed in more detail in Chapter 3

3D Visualization Designs for Traffic data: Multiple 3D visualization designs have pushed this research in exploring why certain designs are useful in certain circumstances and how effective they can be. This is explored in more detail with numerous designs and explanations in Chapter 5.

A new 3D Grammar: A new grammar system developed in this research can be used to extend and narrow the gap in dynamic 3D Visualizations where high quality visualizations can be defined using a specification. This opens up possibilities for API to use this grammar to define and map visualizations in JSON. The grammar design and development is explored in Chapter 4

3D Visualization System: A 3D visualization system named Aloka API implements Aloka and makes it available for use with many other systems such as SPA applications, PowerBI integration and custom API that can use Aloka's data pipeline to generate the visualization (these are implemented in the actual system). This system is described in more detail in Chapter 3.

1.4 Thesis Structure

As show in Figure 1.1 this section briefly expands on each chapter and what the details in each of the sub sections entails.

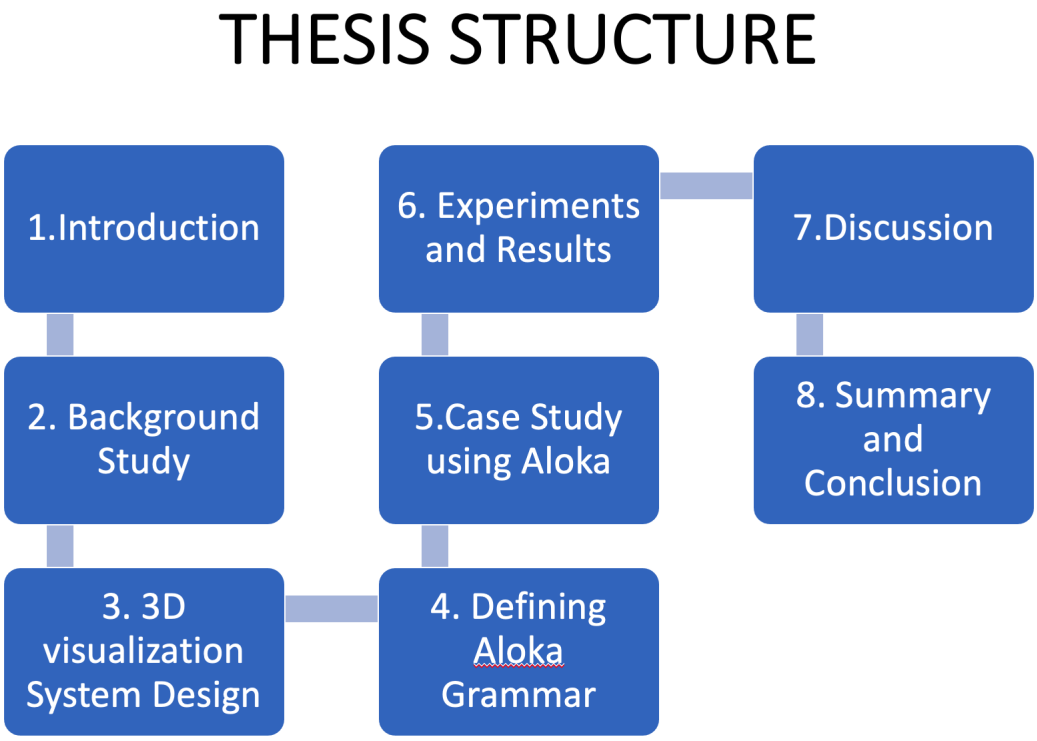


Figure 1.1: The Thesis Structure

Chapter 1 - Introduction: Chapter 1 introduces the thesis in 3 sections where the background summary gives a brief background into the field, the possible research gaps and intended areas of interest. The research questions and possible contribution to research is also explored here. The research problem is clearly defined, and the questions are outlined in detail, while the contributions of the thesis to the academic community are also highlighted.

Section 1.1 - Background Summary: The background summary gives a brief background into the field, the possible research gaps and intended areas of interest.

Section 1.2 - Research Questions: The research problem is defined and the questions are outlined in detail.

Section 1.3 - Contributions: The contributions by this research and thesis to the academic community.

Section 1.4 - Thesis Structure: The thesis structure explains the structure of the thesis.

Chapter 2 - Background and Related Work: Chapter 2 reviews literature in 5 major parts where a detailed 3D Data visualisation literature review entertains why big data is important and 3D visualisations can provide better context awareness and convey and pack more information than traditional styles of visualisations along with geo-visualisation, Flow Visualisation from multi-dimensional data. The architectural design section including the declarative systems and their architectures and system designs are explored. The grammars used in visualisation for exploring existing systems are investigated where petascale cloud super-computing for terapixel visualisation of a digital twin are examined, this explains how cloud super-computing is effective in delivering large scale visualisation renders along with evidence from a published paper. The philosophy of visual design is explored, alongside an examination of the physiology of human vision and its applications across various display types, aiming to tailor experiences to individual users.

Section 2.1 - Physiology of Human Vision: In this section, a model of perception, limit of the human visual system, colour and light sensitivity and colour blindness are all described.

Section 2.2 - Designing for Display Types: such as VR (Virtual Reality) and AR (Augmented Reality) are described.

Section 2.3 - Personalisation: Personalisation, Culture, Language, and individual preference are described. Visual Design and its philosophy is explored, while also trying to understand the physiology of human vision and its applications towards different display types and to personalize to individual users.

Section 2.4 - 3D Visualization: This section entertains why big data is important and 3D visualizations can provide better context awareness and convey and pack more information than traditional styles of visualizations. Specifically, it delves into various applications such as Geo Visualization, which maps and

analyzes geographic data; Flow Visualization, which helps in understanding movement within data, such as traffic or fluid dynamics; 3D Parallel Coordinates enhance traditional 2D data visualization by adding depth, transforming it into a three-dimensional space. This method allows for more dynamic and effective visualization of complex relationships between multiple data dimensions.

- Geo Visualization
- Flow Visualization
- 3D Parallel Coordinates
- Architectural Design

Section 2.5 - Declarative Design Systems: Declarative systems and their architectures and system designs are explored.

Section 2.6 - Declarative Grammars for Visualization: are described exploring existing systems and grammars in visualization.

Section 2.7 - Cloud Computing: software and details of portals are provided.

Section 2.8 - Petascale Cloud Supercomputing : Explores how cloud supercomputing is effective in delivering large scale visualization renders. With a published paper [Holliman et al., 2019].

Section 2.9 - Traffic Visualization: details such as Visual Analytics System, Geographic and Temporal Visualization are reviewed.

Chapter 3 - In Pursuit of a Grammar for 3D Visualization: Chapter 3 gives description of the system design, what guided the development and the motivations behind the objectives being pursued. The API is briefed in detail, from the system architecture to the client-side application to the server-side API.

Section 3.1 - 3D Visualization: This section is explaining Data Visualization in Blender, Gigapixel and Terapixel Visualization are detailed. An example of Aloka by visualizing the live temperature of Newcastle Upon Tyne is provided and briefed on the grammar specification.

Section 3.2 - Design objectives for a new system: objectives for a new system are described.

Section 3.3 - Exploring Grammars for visualisation: such as Data Driven Documents (D3) - the JavaScript library and Vega are described in detail.

Section 3.4 Challenges of 3D Grammar Design: Details the list of challenges when designing a 3D visualization grammar.

Section 3.5 Aloka System Design: Explains how Aloka - a visualization grammar is detailed through its data transformations, visual encoding rules, views and format and structure.

Section 3.6 - Aloka API: The Aloka API is briefed in detail, from the system architecture to the client-side application to the server-side API which is used to render the visualizations.

Chapter 4 - Aloka a grammar for 3D visualization: the grammar design and structure is explored.

Section 4.1 - Aloka in EBNF: Aloka is described in EBNF form.

Section 4.2 - A 3D Visualization Grammar: Future prospects of further research and scope are provided.

Section 4.3 - Chapter Summary: This summarises Aloka's design.

Chapter 5 - Case Study of Visualizations using Aloka: a case study of Aloka.

Section 5.1 - Temperature Visualization: Temperature visualization is defined in Aloka.

Section 5.2 - Traffic Saturation: The Traffic saturation problem is described in detail.

Section 5.3 - Design of Traffic Saturation Visualization: The design of traffic saturation visualization is explored. Glyph Design and Saliency Tests are also detailed.

Section 5.4 - Defining Traffic Saturation Visualization: The above design is modelled in a specification and rendered with the Aloka API

Chapter 6 - Experiments and Results: the experiments and results using the visualization from the case study.

Section 6.1 – Experiment: Purpose and experiment design, benefits are explained.

Section 6.2 – Experiment Methodology: The details of methodology, testing the design effectiveness studied on experts and non-experts, PsychoPy is described. The statistical analysis and the outcome are explained. Confusion matrix, sensitivity and efficiency of the experiment are detailed.

Section 6.3 – Experiment Results: Experiment results of experts and non-experts, time taken to identify images in both groups are detailed. The experiments results are detailed and explained why there is an improvement in user response time.

Section 6.4 – Analysed Results: The analysed results are explored with confusion matrices and sensitivity specificity.

Chapter 7 - Discussion: Discussion of the experiment's results.

Section 7.1 - Key Findings: The results of using Aloka as a grammar and its effectiveness in prototyping visualizations are provided and also explored in the evaluation process.

Section 7.2 – Interpretation of the results: The effectiveness of the traffic saturation design and the results are discussed.

Section 7.3 Limitations of the study and Aloka are provided.

Chapter 8 - Summary and Conclusions: The thesis concludes with the goal of the research question being answered along with summary of the results.

Section 8.1 - Thesis conclusion: Thesis conclusion are summarised.

Section 8.2 Future Work and Direction: Future prospects of further research and scope are provided.

Chapter 2

Background and Related Work

2.1 Physiology of Human Vision

Understanding human vision is key for creating effective 3D visualizations. Knowing the limits of human sight, including how we see color and light, helps in designing visuals that are clear and easy to understand. Factors like color blindness are important in making designs accessible. How we focus on and recognize objects influences the layout and importance of visual elements. It's also crucial to consider how people interpret complex data and manage visual overload and memory, which helps in creating user-friendly interfaces.

2.1.1 Limits of Human Vision

[Deering, 2000], in his document titled "The Limits of Human Vision" presented a model of perception limit of the human visual systems and estimated the pixels of an eye at 15 million variable resolutions per eye. The author predicted that rendering sufficient for human eye saturation at the rate of ten billion triangles in a second, assuming that the 60Hz stereo display at the six complexity depth. The author also analyzed 17 different physically realizable computer display configurations to help understand the limits of visual perception. Cathode-Ray Tube (CRT), Stereo-projection displays, head-mounted monitors, standard televisions, movie display, and multi-walled immersive stereo projec-

tion displays were used for comparison. The model is also suggesting the human eye can discern approximately 15 million variable resolution pixels. This leads to the conclusion that to fully engage the human visual perception capabilities, a rendering rate of around ten billion triangles per second in a 60 Hz stereo display with a depth complexity of six is required. [Oyster, 1999], in the human eye, describes the human eye as a remarkable biological invention and an evolution's shining triumph. The author also terms the eye as the detector that enables planet and cosmos exploration and lists a number of its shortcomings that limit the further investigation of life. The author names limited size and light-gathering power, a limited frequency response that can only see electromagnetic radiation in visible wavelengths, and distinguishes a new image many times per second. He argues that it cannot be used in light accumulation over a long period to intensify faint image, and inability to store images for future references as compared to photographic plates as limits of the human eye. The author also acknowledges that many instruments and techniques have been developed to supplement the human eye to alleviate shortcomings.

2.1.2 Color and Light Sensitivity

[Blake et al., 2020] look at light and color sensitivity. He argued that the human eye, like the compound eyes of arthropods, is sensitive to polarization across its structure. The term "taxa" means groups or categories used in biological classification to describe species or organism groups. According to the author, the human eye's sensitivity to polarization increases with the diversity of its photoreceptive cells, similar to arthropod taxa. However, the process by which the eye interprets polarized reflections from objects (how light changes when it bounces off surfaces) is not well understood. "Foliar reflections" describe how light reflects off leaf surfaces, varying in polarization and playing a significant role in plant-light interactions. By comparing human and small white butterfly (Pieris rapae) sensitivity to light and color, the author shows how the butterfly uses polarized light from leaves to identify host plants, relying on the color and intensity of the reflected light. This is vital for recognizing where to land and feed. The author narrows explicitly down on the linear polarization degree as to how the little butterfly recognizes the host plant. The

butterfly's visual system consists of three basic types of photoreceptors that are polarized light-sensitive, namely, blue, green, and red. The interaction of photoreceptors and their roles in behavioral responses to different stimuli that have different degrees of linear polarization is still unexplored and unknown. The author used several ways to investigate the potential neurological mechanisms, designed a few two-choice behavioral assays, and displayed plants' images on paired LCD monitors.

2.1.3 Color Blindness

In the study "Gene Therapy for Color Blindness" by [Hassall et al., 2017], the focus is on Achromatopsia, a rare eye disorder mainly caused by certain gene mutations that affect the function of the eye's cone cells. The study looks at early clinical trials in Germany, the UK, and the USA that test gene therapy using Adeno-Associated Virus (AAV) to treat this disorder. Researchers used a specific type of AAV vector in mice and saw promising results that suggest it could help restore the functioning of cone cells. These findings highlight the importance of treating the disorder early in life, ideally during childhood, to effectively repair the visual pathways. This research has important implications for designing visual systems and 3D visualizations that are accessible to people with color vision problems.

In another study titled "Assessment of color vision among health science students" by [Pandit and Dhakal, 2020], researchers tested color vision in healthcare students to understand how different parts of the eye and brain contribute to seeing colors. Conducted over two months in 2018 with 300 students aged 18 to 25, the study found that 2.3% of the male students had some form of color vision deficiency, but none were found in female students. The specific issues identified included one case of protanomaly, two of deuteranomaly, and four of deuteranopia. These findings are vital for the field of 3D visualization design, as they help in developing color schemes for visualizations that are effective and inclusive for all viewers, making sure that informational graphics are practical and accessible. This aligns with insights from [Wang et al., 2008] study, "Color design for illustrative visualization" which emphasizes the importance of understanding visual impairments to create

suitable visual tools.

2.1.4 Visual Attention

[Shahrbabaki, 2015] did a study on the role of color in visual attention that aimed to shed light on the influence of color information on the part of eye movements during video observations to incorporate the color information into a visual saliency model that included color saliency maps. The researcher compared the central regions in color videos concerning areas of grayscale and observed that color information moderately affected characteristics of eye movements, including gaze positions and fixation durations. [Lavoué et al., 2018] conducted a study on visual attention for rendered 3D shapes and argued that it is essential to understand the visual attention behavior of the human optical system in the visualization of rendered 3D shapes for use in the application of computer graphics appropriately. The researchers believe that the solution to exploring the complex cognitive mechanism in eye tracking and recognize that a large number of studies have been dedicated to the study of images and videos, and only a few eye-tracking experiments have been done using the 3D shapes. The authors also argue that another part that needs research is the understanding of potential factors that affect the human gaze in specific 3D rendering settings. They carried out an eye-tracking study with 3D shapes, using both static and dynamic camera angles, and introduced a methodology for mapping eye fixations onto 3D shapes. This process generated a benchmark for 3D meshes, complete with fixation density maps that highlight where humans tend to look. The experiments collected data on the influence of camera position, shapes, illumination, and material on visual attention. The study found that material and lighting affected visual attention and camera path in the dynamic scenes. The study used two metrics to compare the performance of four representative state-of-the-art mesh saliency models in the prediction of ground-truth fixation. It showed that even if the 3D saliency algorithm is combined with a center-biased model, it remained lacking in the prediction of human fixation. The authors also provided a qualitative analysis of the main factors that attracted the human attention and showed their weaknesses and gave out a comparison of the focus of the

human eye and Schelling points and showed a weak correlation.

2.1.5 Object Recognition

[van Dyck and Gruber, 2020] conducted a study to see how well humans and Deep Convolutional Neural Networks (DCNN) recognize objects when pictures are altered. They noted that even though DCNNs perform almost as well as humans in recognizing objects, there are still differences between how humans and machines see. The study's goal was to explore these differences by comparing how humans and neural networks recognize objects using the ImageNet database. For this, 65 people took part in an online experiment where they matched up against various DCNNs. The experiment started with a training and validation phase using natural images, followed by a testing phase that introduced new color and shape changes. The results showed that humans did better than DCNNs in all conditions, showing they were more adaptable to changes in shape and color. Additionally, an analysis of the mistakes each group made showed different kinds of errors between humans and machines. These findings suggest that adding recurrent circuits, similar to those found in the primate brain, to artificial vision models might improve their ability to recognize objects under new conditions. [Hafi et al., 2017] devised a method of identifying the focused objects in eye images captured using a single camera to facilitate innate eye-based relations using wearable devices. The authors posit that eye images allow and obtain natural user responses from cornea scene reflections and movements of the eye without the help of additional sensors, for instance, the frontal camera, a fact that contributes to its social acceptance. The method proposed by the researchers is reliant on a 3D eye representation rebuilding in evaluating the eye image gaze directions. The gaze course was combined with a deep learning algorithm in the classification of focused objects the cornea reflects. The experimental results used a wearable prototype to express the latency of the process based on eye images captured using a single camera.

2.1.6 Perception Adaptations and Object Categorization

[Witzel and Gegenfurtner, 2018] used scientific methods to investigate eye color and linked color appearance linking to colorimetric measurements of light entering the eye. The color perception's primary purpose was to help in visual; perception of objects and materials in the environment rather than determining the incident light properties. The study reviewed the state of the art colors on items, the constancy of colors, and categories of colors to study the functional spacers of color perception. The research area's common ground was that the appearance of color was loosely connected to identifying objects and materials and communication among observers. The researchers concluded that attention should be focused on how color processing is adapted to the object surface properties in the natural environment to bridge the space between the known color perception early stages and the subjective color appearance. [Kaiser et al., 2019] argue that objects appear in typical locations in natural vision. Object appearance is based on the visual space. The authors recognize that recent research has significantly advanced the field, demonstrating how object vision is linked to positional regularities. In the study, the authors delve into the developments and highlight the adaptations for position regularities that facilitate the detection and recognition of objects and the sharpening of object representation in the visual cortex. The researchers believe that the effects are pervasive across a range of high-level content types and argue that the adaptations to real-world structure collectively underpin the optimal limited cortical processing resources usage. Considering position irregularities is crucial for the understanding of the efficient vision of an object in the real world.

2.1.7 Visual Clutter

[Medwetz, 2019] is a research that looked at how color affects our perception of visual clutter, and how it influences our emotions and behavior in indoor spaces. She found that too much visual stimulation in a space can become overwhelming and no longer looks good to the user. Her thesis suggests that interior design should be done carefully, especially

when it comes to using color. The right amount and application of color can help create spaces that boost productivity, create a festive atmosphere, and increase satisfaction. Medwetz aimed to show how clutter and color affect each other and argued that more research is needed in this area. Her findings underline how much our surroundings can affect our personal and professional lives, our mood, and how well we perform tasks. She pointed out that controlling visual clutter is essential for effective interior design. [Wibble et al., 2020] carried out a study to see how rotating visual clutter affects eye rotation, body movement, vertical distortion, pupil reactions, and the discomfort these conditions can cause. They focused on how people with dizziness often react more to visual motion and clutter. In their experiment, 16 healthy participants were exposed to 20 seconds of rotating visuals that varied in how cluttered they were, using black lines on a white background with either 19 or 38 lines to represent low and high clutter levels, respectively. They measured eye rotation and vertical distortion using the Chronos Eye Tracker, which also tracked pupil size to gauge automatic body responses. Body movement was monitored using the Wii Balance Board, which measured how much participants swayed. They compared these findings to data from a 20-second period where participants looked at a still scene before the moving visuals began. The results showed that more intense stimuli led to faster eye rotation, larger pupils, and more body sway, especially in more cluttered conditions. These were linked to the speed of eye rotation. The authors believe these findings are important for assessing patients who are very sensitive to visual motion and could help explain why some healthy people feel uncomfortable in visually cluttered spaces.

2.1.8 Memory

[Damiano and Walther, 2019] looked closely at how seeing and remembering are connected, focusing on how moving our eyes can improve memory. They conducted two experiments to find out if the helpful effects of eye movements on memory mainly happen when we first see something (encoding) or when we try to remember it (recognition). The results showed that people who moved their eyes a lot when they first saw something had a

much better chance of remembering it correctly later, no matter what they did during the memory test. Also, making fewer mistakes in remembering things incorrectly was linked to moving the eyes during the memory test, especially if participants were encouraged to look around the scene. Overall, the study highlighted that eye movements play a key role both when we first store memories and when we try to recall them, helping us form memories and judge if they are true. [Leer et al., 2017] conducted experimental studies and discovered that dual tasks are inversely proportional to self-reported emotionality and the vividness of memory. The data are affected by demand, and due to this, little can be inferred about the underlying observed effects mechanism. This study was aimed at filling the literature lacuna by providing memory performance objective tests. The researcher posits that desensitization and reprocessing eye movement therapy for Post Traumatic Stress Disorder (PTSD) encompass making eye movements when patients recall traumatic images. The findings of the researcher included the fact that eye movements during stimulus recall reduced the vividness of self-reported memory and, on the other hand, slowed the time of reaction in a task that required the subject's discrimination of the stimulus from perceptually similar stimuli. The authors also demonstrated that eye movements can exacerbate responses of fear to both similar and non-threatening stimuli in the presence of a threatening stimulus. The variation is shown by danger expectancy, and skin conductance response increases. The researcher concludes that eye movement manipulation renders results in less available stimulation in future recalls.

2.2 Designing for Display Types

Different display types like 2D, VR, and AR each have their own benefits and challenges for 3D visualization. The size of 2D screens affects how much information can be displayed at once. VR and AR offer immersive experiences that can change how we interact with and understand data in space. Designing visuals specifically for each type of display can make them more effective and engaging.

2.2.1 2D Display Size

[Figueroa et al., 2018] conducted a classical visual search and used the artificial stimuli in the identification of factors that affect accuracy and search times. The authors argue that people perform visual search tasks in their daily lives that range from petty jobs to emergency tasks. Recent studies have simulated real scene visual search in 2D displays to investigate visual search in natural scenes as the scientific community strives to incorporate new technology in better method and practice formulation. Among the technologies used in virtual reality, a novel technology offering users immersive and evokes real responses. The researchers aimed to compare search efficiencies in real scenes on virtual reality and 2D displays. The virtual search experiment that measured the time and accuracy of reaction was conducted in the evaluation of both methods. The results of the study indicate that real scene virtual search is faster and more accurate in virtual reality than in 2D Displays. The researchers argue that the findings of their study could be a new opportunity for researching virtual search on life scenarios and real scenes. [Gao et al., 2016] reviewed recent progress in 2D and 3D display technology field and present the current display materials and their uses such as in organic light-emitting diodes (OLEDs), curved displays, holographic 3D displays, light field 3D displays, stereoscopic 3D displays, volumetric 3D displays, electronic paper (E-paper), flexible OLEDs quantum dot light-emitting diodes and active-matrix organic light-emitting diodes. The current applications of 2D displays include liquid crystal devices that, due to true depth information deficiency, result in ambiguity in high dimensional data images—the study details description of 3D display technologies.

2.2.2 Virtual Reality

[Roettl and Terlutter, 2018] reviewed virtual reality graphics use in 3D video game technology. They analyzed the similarities and experience among video games played in 2D, stereoscopic 3D, and Head-mounted Virtual Reality versions and how they affect the brands placed in the video games. The authors also analyzed game-related variables in-

cluding presence, and video gamed attitudes and arousal when playing the game. The authors also examined the Brand placement-related variables, including attitude and recall and recognition memory employed towards the brands in the study that involved 237 players. The results of the survey indicated that HMD virtual reality registered a higher presence than stereoscopic 3D that was also higher than in 2D video games. All the three video games recorded the same arousal and attitudes which were not affected. In contrast, memory for the brands was higher in 2D video games as compared to the Head-mounted Virtual reality. The authors also conducted a post hoc study on 53 participants. They concluded that cognitive load was lowest in 3D games and highest in Virtual reality games as dizziness levels and motion sickness were higher in Virtual reality than in both 2D and 3D games [Brychtová and Çöltekin, 2017]. The researchers addressed limitations and outlined implications for developers, marketers, and researchers. [Li et al., 2017] conducted a review of virtual reality technology application in the field of clinical medicine, with a focus on surgical training, management of pain, and treatments of mental illnesses, and introduced the common Virtual Reality simulator types and their principles of operation in the above fields. The authors discussed the clinical effects and argued that both patients and doctors have a lot to reap from the benefits held by the application of virtual reality as a novel technology in the medical field. In the review, the authors also discussed further advantages and disadvantages of the application of virtual reality technology in the area and posit that benefits outweigh drawbacks. In a paper that proposed a new approach for mapping in 3D for indoor environments, [Du et al., 2016] aimed to suggest ways that would allow robot avatars to collaborate with human beings. Collaboration is possible through virtual reality devices and argues that map building is an essential task in the application of robotics. The authors posit that 3D data obtained from an RGB-D camera mounted on a robot, transmitted to a remote server, and rendered to a Virtual Reality device is used in the creation of 3D maps. Conversely, user intentions are inferred through head movement motions on hidden Markov models (HMMs) and interpreted into the robot command controls.

2.2.3 Augmented Reality

[Yuan, 2016] argues that the recent years in display technologies have been marked by the growth of the use of virtual reality and augmented reality even though the concepts were invented decades ago. The author attributes this growth to the fact that people are starting to enjoy their specialized experience as opposed to tolerating them as in the early days. [Yuan, 2016] believes that from this point, more people believe that virtual reality and augmented reality hold the potential to disrupt and change the world in many aspects. [Sage, 2016] posits that creating augmented reality experiences for enterprise use cases requires a lot of conditions met as it is a very complex undertaking. In the case of industrial environments, the author argues that developers and design teams of user experiences can meet many restrictions and unforeseen obstacles. The author also describes designing and developing augmented reality experiences as an interactive process marked by a lot of critical decision making, that if correctly completed, result in a successful project. The author also stresses in making the right decisions in the correct order as a privilege for a small developer and experience designer numbers that make more significant achievements from the expertise. [Saputro et al., 2018], in a physics journal, reviewed technology use in a museum for historical education, keeping historical objects, and preservation of national culture. Technologies, including augmented reality, are increasingly gaining relevance and application in the face of the ever technologically advancing world, smart cities, and the internet of things. The author posits that current developments in Information and Communication Technology are used as the transmission medium, arguing that the advent of augmented reality technology has helped create virtual objects in the real world with the use of images and markers. The researcher, in this study, used signals to make virtual objects appear utilizing IEEE 802.14.5 protocol to replace the augmented reality marker. The author also used Triangulation and RSSI as micro-location substitutes for augmented reality objects and concluded that wireless sensor network performance could be used for transmission of data in the museum. The LOS study results showed that the 15-meter distance with 1000ms delay found a 1.4% error rate. In contrast, the NLOS error rate was 2.3%, leading to the conclusion that

the use of technology using signal wireless sensor networks can be used to replace market augmented reality and be used in museums.

2.3 Personalization

Personalizing visualizations means adjusting the content and style to meet the needs of various users, considering their cultural background, language, personal tastes, and biases. This makes the visualization not just accessible but also more meaningful to a wider audience, helping different groups better understand the data.

2.3.1 Culture

[Neuliep, 2017] argues that a better part of the world is still cultural in their perception of technology, including 3D visualization and hence the need for cultural integration in the development of the technology. 3D visualization opens opportunities and the need for intercultural communication as an essential skill in the face of globalization as communicating with people from other cultures is on the rise in such areas as business, education, and community. The author posits that the art of knowing how to communicate with different cultures is increasingly becoming a workplace skill, boosting its incorporation in the development of technologies. It is, therefore, essential to consider this in 3D visualization development, in the face of globalization, which allows for effective interaction between users and the technology. Effective interaction will help avoid intercultural conflicts, communication frustrations, and misapprehensions, and in all these helps one improve the acceptance of the technology. [Neuliep, 2017] also maintains that effective and successful communication is essential for a significant transfer of knowledge in schools. The development of the technology will have a rich repertoire of nonverbal and verbal capabilities to enable them users hailing from different cultures from all over the globe accept and welcome the technology and get rid of barriers such as attitudes and dispositions, stereotyping, and ethnocentrism. Intercultural technology communication is, therefore, an essential skill in the 21st century with the advancement in years, technology and scope

of education and doing business and many other areas. [Doersch et al., 2015] claim that 3D visualization helps in extracting insight from data for decision making, especially when large amounts of data are involved. Visualization enables users to understand and perceive data virtually through transparent user-data effective communication interaction methods. It offers a range of advantages such as understanding vast amounts of data, perceiving unexpected properties, extracting problems such as outliers and anomalies, understanding both large and small scale data features, and creating hypotheses related to data. The authors also claim that in communication, the use of graphics to present quantities, processes, and spatial relations in a simple to complex visual language enhances the effectiveness of message conveyance. Visual representations are used to convey messages metaphorically or directly, offering core messages and other qualities embedded. When markers are embedded in theses visual representations, they generate interest in the topics and issues under communication, create the urge to seek more information on them, and allow the users to explore more on the topics. Therefore, the visual representation can not only be considered attractive and pleasing but also are designed to tell stories effectively to help in the better carriage of the communication messages and meet the needs of the users for multiple usage modes. Visual communication, therefore, is not only used in communication as a way of telling stories but also is a way of presenting underlying meanings. It also helps present concepts and events across time as well as disseminating information widely through various media. In the communication reports, visual depictions are used to communicate channels, modes, and applications in different spheres of life and to observe the cases and effects of viewers' decision making.

2.3.2 Language

In their study [Frank et al., 2019], Frank, Dalenogare, and Ayala examined the implementation patterns of Industry 4.0 technologies, addressing the convergence of digital solutions in a new industrial age. The research aimed to enhance understanding of how companies adopt these technologies, thereby facilitating their integration into manufacturing. The authors proposed a conceptual framework that divides the technologies into

base and front-end categories. Base technologies include the Internet of Things, Big Data, Analytics, and Cloud Services, while front-end technologies encompass Smart Working, Smart Manufacturing, Smart Supply Chain, and Smart Products. Their survey of 92 manufacturing companies revealed a systematic adoption of front-end technologies, with Smart Manufacturing playing a pivotal role. However, the study identified significant challenges in implementing base technologies, particularly Big Data and Analytics, which were poorly executed among the surveyed companies. The researchers also discussed the levels of technology adoption and their implications for the manufacturing sector, providing a diagnostic tool for companies to assess and enhance their integration of these technologies. The findings highlight the importance of Industry 4.0 technologies in improving operational efficiencies and fostering innovation within the manufacturing sector.

2.3.3 Individual Preference

Change, planned or unexpected, is inevitable in life and careers like in the medical field, in the face of evolution. Efforts to maintain and improve the quality of patient care often encounter resistance and conflicting factions committed to the status quo, disrupting the balance or homeostasis of a group or individual. [McDonald et al., 2011] argues that implementation of technological change within a group must be done to recognize beliefs, social and cultural backgrounds, education, and past experiences, both negative and positive, of an individual member of a group. These factors determine the resistance impact amount to the proposed or implemented change as a natural or expected response as this will help minimize or manage the resistance. They further posit that inter-professional leadership influences the resistance to change. [Noordegraaf, 2016], in 'Change is Constant,' opines that setting a change agent is necessary to the facilitation and effecting of positive change. Also, growth can be fast due to the advancing technology and evidence-based research steering it to constant policy regeneration to ensure users get quality. The researcher also contends that an inter-professional collaborative effort is needed among stakeholders and users for change to be effectively effected as they inspire others for change for the improvement of outcomes. The researcher also claims

that even though change improves quality and safety, a section of users may not appreciate it as they view it as expensive, time-consuming, troublesome, and job-threatening, such as in the case of technology integration due to limited knowledge. For change to happen effectively and efficiently in a healthcare organization, managers and leaders should plan for it to avoid resistance through staff education on new processes and the importance of change [Yost and North, 2006].

2.3.4 Adapting to Bias

[Jordan, 2016] reports that, like any other market, the technology market and investment are affected by demand and supply forces implying that there are factors that affect the buying and selling actions among the participants in the market. Among these factors are non-economic factors such as behavioral biases, religious, political events such as the infamous Brexit, wars, natural disasters such as earthquakes, currency heading and investments, investor speculation, HFT, and Algorithm trading and business sentiment (Laxar and CT, 1998). These factors have economic significance as they can affect investments in areas that they occur in, either by reducing or increasing investment opportunities and outcomes depending on the type of investment. Investing in technology requires psychology-based strategies offering mental fortitude to stick to the plan and develop traits such as patience and discipline to attain consistency. These strategies help eliminate self-inclinations to seek shortcuts, getting distracted, or making decisions informed by fear or greed. These kinds of traits lead to irrational and biased decisions despite being told in [Holliman and Watson, 2015]. [Frankel and Laby, 2015] opine that behaviors of users can be logical or reasonable, and biases in them may lead to the making of lousy technology investment decisions attributes to emotional processes. Mental mistakes and individual personality traits inform technology investment decisions made by investors. Making investment decisions is not only based on analyzing numbers and performance table to sell and buy various assets and securities. Users must use reviews and managers to eliminate these biases that might lead them to make uninformed decisions that increase risks. These behavioral biases can be of many types, including cognitive and

emotional biases, and lead to irrational decision making based on attitudes and behavior. Due to these, a user can be either classified as status-quo or overconfident. The best way to eliminating these biases is by making investors aware of them or using investment managers in making informed financial decisions. [Carnall, 2019] claims that biased users have very minimal chances of survival since their investment decisions are not entirely based on financial facts and prognosis. Their investments are based on biases that are in no way related to the condition of the market, new opportunities, and the running of the companies that they are poised to invest. The CEOs of companies to be invested in are professionals that are capable of making wise decisions that will affect the performance of companies. Performance trends of companies should be assessed by independent advisors and managers to ensure investors' monies are in the growth path by avoiding mispricing and correction patterns that could be managed by CEO's to create an inaccurate impression. Elimination of the human element in making technology investment decision making is essential. It can be achieved by the use of third party management firms of individuals that advise and make an informed decision since they have sizeable real-time information on the state of the market and possess superior market watch and analysis technologies [Demian and Fruchter, 2009].

2.4 3D Visualization

Certain techniques in 3D visualization, like GeoVisualization, Flow Visualization, and Multidimensional data representation, are vital for showing complex datasets effectively in three-dimensional space. For example, architectural designs and traffic visualizations benefit greatly from 3D to better depict structures and data points. Similarly, 3D parallel coordinates are useful for displaying data with many dimensions in a way that's easy to grasp.

2.4.1 GeoVisualization

[Neuville et al., 2018] reviewed Virtual 3D city models and opined that they are vital information centers that support many city aspects such as planning, management, and simulation. The authors note that the models have not been fully exploited owing to ineffective visual communication routes across developers and end-users of 3D models. The authors aimed to efficiently and usefully underpin visual identification and recognition of an object in 3D. The researchers presented a solution based on a 3D geospatial data knowledge network that collects and links rendering and mapping techniques. The authors propose using second-order logic in formalizing the knowledge base and making it usable for decision-making and providing a set of efficient graphic design guidelines to avoid creating graphical conflicts and improving visuals communication. The review implements an apt resolution to support the 3D geospatial model visualization process using Computer-aided Design (CAD) and Geographic Information System (GIS) software. The authors also propose extending to Open Geospatial Consortium (OGC) symbology encoding in the provision of appropriate graphic guidelines to web mapping services [Bariic et al., 2012]. [Gardony et al., 2018] is convinced that independent Augmented Reality (AR) systems offer significant promise for interactive 3D geo-visualization. He points to advanced technologies like head-worn AR devices, which can display images of extensive environments and enable intuitive interactions through gesture and voice commands. These technologies allow users to engage with geo-visualizations through interfaces, and strategies for interaction provide superior spatial memory and navigation experiences.

2.4.2 Flow Visualization

[Piri, 2017] developed a three-dimensional shipping information system to integrate dynamic flow field and 3D virtual geographical environment. The study conducted by the author formulated a 2D hydrodynamic model to determine the waterway wavelength depth and distribution of velocity using the real-time hydrologic monitoring data. The study compared the traditional 2D navigation map with a combined flow simulation and

visualization module to integrate flow velocity magnitude and distribution to enhance navigation to enhance the safety of navigation. The comparison helped display integrated scientific information with the potential for use within 3D information systems in several domains.

2.4.3 3D Parallel Coordinates

3D Parallel Coordinates expand on the traditional 2D method for visualizing multidimensional data. In this setup, each axis represents a different data dimension, and data points are shown as lines intersecting these axes at corresponding values. While the 2D system uses multiple vertical axes lined up side by side with lines connecting data points across dimensions, 3D parallel coordinates add a third dimension, often by adding depth. This transformation creates a 3D space that allows for a more dynamic visualization of complex relationships in the data.

[Forsell and Johansson Westberg, 2007] evaluates 3D and 2D parallel coordinates by analyzing how they perform in exploring data with multiple variables. It examines whether these visual tools improve users' ability to identify complex connections within the data. The study suggests that 3D parallel coordinates could be more beneficial for complex datasets because they display multiple relationships at the same time. It emphasizes that 3D visualizations can better manage and clarify the complexities of multidimensional datasets, especially where traditional 2D displays fall short in showing detailed relationships effectively.

[Johansson et al., 2014] closely examines how 3D parallel coordinates work for analyzing 2D relationships in data with many dimensions. It explores if 3D visualizations offer any advantages over traditional 2D parallel coordinates in detecting relationships between variables. The results show that 2D parallel coordinates are often better than 3D in terms of speed and accuracy. They assess the effectiveness of using a 3D approach to improve data visualization. It suggests that although 3D parallel coordinates are visually appealing, they may not always be the best option for certain tasks, like identifying 2D relationships. However identifying 3D relationships have not been explored here.

[Akram Hassan et al., 2019] examines how well 2D and 3D parallel coordinates can identify trends in data that changes over time. It finds that 3D setups are particularly good at making patterns easier and more accurate to see and understand, making them very useful. The study notes that 3D parallel coordinates are especially effective at displaying complex and changing data in a way that's easier to interact with and understand, something that's difficult with 2D methods. Hassan also discusses how each piece of research contributes to our knowledge of using 3D methods to visualize data with many dimensions. It points out that 3D coordinates work better in certain situations, especially in making complicated and dynamic data simpler to grasp. These findings are important for improving how data is visualized in data science and analytics, helping to make better decisions and explore data more effectively.

[Zhonghua and Lingda, 2016] argue that parallel coordinates can be efficiently used to visualize multidimensional data when there is a shortage of displaying time-varying data. The authors propose a new method to add a time dimension to extend the parallel coordinates in 3D consisting of an attribute, time dimension, and range. Time-varying multidimensional data can be shown in the form of a polygon line cluster for recording and analyzing. Dataset highlighting around current time and results is done using a technique referred to as a clipping scale to reveal that 3D parallel Coordinates to analyze the time-saving character of an attribute effectively.

2.4.4 Architectural Design

[Entezari et al., 2018] posits that the challenges in successfully regenerating functional bone tissue in critical-size defects represent a significant hurdle in the medical field. Synthetic scaffolds were developed as a response but were not successful due to poor performance. The authors demonstrate that architectural designs of scaffolds printed in 3D can help improve outcomes of regeneration. The author showed that manipulation of sizes of pores and permeability in scaffolds printed in 3D helped create a useful strategy for enhancement of results of bone regeneration.

2.5 Declarative Design Systems

Declarative design systems let designers focus on the desired results instead of how to achieve them. This method is great for 3D visualization because it allows for more flexibility and ease in creating complex visuals, making it simpler to try new ideas and make adjustments based on overall goals. [Satyanarayan and Heer, 2014] reviewed Vega and D3. They stated that there are already some tools for web-based visualization, including D3 for data-driven documents, and says that Vega does not replace D3 as it is of the higher library. The author refers to D3 as a visualization kernel designed to a layer that supports higher-level visualization tools. [Sicat et al., 2019] reviewed DXR as a toolkit used to build immersive data visualization utilizing the unity development platform. The author argued that immersive data visualization in virtual reality and augmented reality merged in recent years as a promising medium for making sense of data beyond the desktop. The creation of immersive visualizations is still a challenge that requires the intervention of complex low-level programming and tedious manual encoding of data attributes, visual and geometric properties. According to the author, these can impede the process of idea to prototype for developers that have no experience with 3D graphics, augmented reality, and virtual reality programming. Developers can efficiently specify visualization designs with DXR using concise declaration grammar based on Vega-Lite. DXR provides GUI for easy access and quick edits and visualization design previews in-situ while in the virtual world. It also provides reusable templates and customizable graphical marks, and this enables unique and engaging visualizations (Bostock and Heer, n.d). The authors demonstrate the flexibility of the DXR using several examples from many applications.

2.6 Declarative Grammars for Visualization

Declarative visualization grammars help the acceleration of development and facilitation of retargeting of visualization designs in many platforms. They also allow for optimizations at the language level. Most declarative visualization languages are only applicable in

visual encoding and rely on imperative handlers for interaction behaviors [Carnall, 2019]. DirectX Raytracing [Sandy, 2018] is Microsoft's DirectX feature for real-time tracking. Protovis as defined in [Bostock and Heer, 2009] is used in composing data views with simple marks, including dots and bars. It differs from low-level graphic libraries that become tedious quickly for visualization. Protovis defines marks through dynamic properties encoding data, allowing for an inheritance, scales, and layouts to simplify construction. Lyra is an environment enabling custom visualization design without writing any code [Satyanarayan and Heer, 2014]. Vega is a high-level grammar of interactive graphics [Satyanarayan et al., 2015] providing concise, declarative JSON syntax to create a range of visualization data. D3.js, on the other hand, is a Javascript library for visualization data with Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), and Scalable Vector Graphics (SVG) [Bostock et al., 2011]. [Schiffer et al., 2010] argue there is a gamut of data visualization tools used in the field. However, there is still a gap to be explored in the field between high-level visualization system efficiency and low-level graphical system expressiveness and accessibility. Power visualization systems are, at times, inflexible and can impose abstractions that are new to visual thinking. Conversely, graphic designs, such as those created with rendering APIs and vector-based programs, are often cumbersome and complex. The author contends that there is a need for graphic design tools that are user-friendly and specifically tailored for visualization purposes. The researchers present Protovis as an extensible toolkit applied in constructing visualization by composing graphical and straightforward primitives. The designers of Protovis regard visualization as a hierarchy of marks that have visual properties defined as data functions.

2.6.1 Vega-lite

Visualization is a technique used to create diagrams, images, or animations to pass on a message. The applied grammars vary in expressivity; low-level grammars include Protovis, Vega, and D3, while an example of a high-level grammar is ggplot2 [Satyanarayan et al., 2015]. This report by Satyanarayan presents Vega-Lite, a type of high-level syntax that allows rapid specification by combining traditional and novel grammars of interaction. It builds

on the work done by other grammars of graphics and automatically synthesizes the requisite data flow. This particular version has more concise specifications, unlike the lower Vega language. First and foremost, [Satyanarayan et al., 2015] provide an algebra to compose the grammar's specifications from a single-view to a multi-view display. As a result, they can examine whether Vega-Lite scale domains could be better off merged or independent. To develop a preeminent interaction grammar, the providers create several selections to act as scale domains which eventually construct an overview-detail interaction. Vega-Lite also has several operators that can transform these selections by adding, removing, or manipulating specific points. Interactive visualization focuses on graphic concepts and how they can be improved to interact with information. Therefore, this report demonstrates how Vega-Lite utilizes several methods, such as panning and zooming, to bring high specifications to this field [Satyanarayan et al., 2017]. Nevertheless, Vega-Lite presents two primary limitations. First, it favors conciseness over expressivity. Secondly, the components that are determined before run-time cannot be manipulated. The authors suggest that future work could develop models that support productive, interactive methods for specific visualizations and their underlying data types. Overall, Vega-Lite is an excellent addition to the programming world as it evokes rapid exploration in design variations and may assist designers in modifying interactive graphics effectively.

2.7 Cloud Computing

Cloud computing provides the strong processing power and scalable resources needed for the intensive computational demands of 3D visualization. It supports handling large datasets and complex calculations necessary for real-time data visualization, making 3D visuals more dynamic and accessible. Cloud computing services can be offered in the ways of software as Service, Platform as a Service, and Infrastructure as a service. Also, such offerings an as Microsoft Azure and Amazon Web Services (AWS) offered by some cloud service providers to provider platform and infrastructure services. Software As A Service SaaS is a service capable of mitigating obstacles in the growth of cloud computing. It provides opportunities such as the fact that users get accounts used in accessing applications main-

tained and hosted by the service provider. It is also primarily used in the replacement of end-user applications, and its examples include Concur, Flickr, Google Apps, Citrix GoTo Meeting, Facebook, Cisco Webex, and Adobe Marketing Cloud. The common security threat in this service category includes the stealing of user passwords and logins. Service Oriented Architecture (SOA) is a software design style in which application components are used to provide services to other components through the network communication protocol. It was first referred to as service-based architecture in 1998 by the foundational management services developing team and then as business process-type services founded on the work units that use CORBA in inter-process communication [Goode et al., 2015]. Cloud computing in SOA is a model used to access a usage-based network to a shared pool of configurable computing resources such as storage systems, services, network servers, and services. Affordable bandwidth and capacity optimization technologies are making cloud backup popular with such options as public crowd direct data backup, including Microsoft Azure, AWS, or Google cloud platform. Another is by using a service provider in hosting and managing backups at their clouds and cloud-to-cloud data backup created using SaaS apps, including sales force and Microsoft 365 [Seethamraju, 2015]. Advantages include efficiency and reliability in data backup as it uses the latest technologies such as compression encryption, data duplication, disk-based backups, and others. Also, capital savings scalability to low-cost consumers and small businesses that do not have a lot of data to protect are advantages of cloud data backup. Others are easy accessibility, improved recovery time for some datasets, and border protection. The cons include seeding data and full recovery based on the data capacity, size limitations, service discontinuation, and lack of efficient monitoring and managing tools. Clouds have to be realistically rendered in 3D applications. The advancement of GPU technology has made raymarching a viable and widespread technique to render clouds. [Bittner, 2020] gave a description and an overview of the method based on the game engine and recent games. The author explained cloud modeling, position, and lighting founded on physical reality and posits that pseudocode can be used to illustrate the concepts that interact in the raymarching renderer. The author describes potential optimization techniques and performance con-

siderations to demonstrate the efficiency of highly realistic cloudscapes in real-time. The author discusses the limits of present-day technology and advancements in the future. Ray tracing has been used for a long time in non-real-time rendering and provides lighting as it simulates the physical behavior of light. The technique has been used to calculate the color of pixels when tracing the light path that was to be taken of it were to travel from the viewer's eye through a virtual 3D scene. Light is reflected from one object to another as light travels traverse the scene resulting in reflections. Light is then blocked, but items and results in the formation of shadows or pass through semi-transparent or transparent objects to cause refractions. All the above interactions are combined to form the final pixel color that's then displayed on the screen. Ray tracing has always been considered the future of computer rendering. Still, the advent of consumer GPU is taking this position as it has enough computing capability for ray tracing workloads in real-time. The incorporation of any use case is expected will use hybrid renders that combine ray tracing and rasterization [Terence, 2010]. Technology stack is one of the essential things in the development of top-notch applications. The Tech stack can determine the success of a software product and is a collection of all technology services used in building, running, and managing an application. They are combining programming languages, frameworks, and tools that developers need to interface with the application [Borgo et al., 2013]. A tech stack helps reflect the weaknesses and strengths as well as limitations and areas of improvement for a coding language.

2.8 Petascale Cloud Supercomputing for Terapixel Visualization of a Digital Twin

In order to create photo realistic terapixel scale visualizations, it is very computationally intensive and so far there have not been any such visualizations of urban digital twins. The few terapixel scale visualizations that exist have been focused on space rather than earth. [Holliman et al., 2019] focuses on creating a scalable cloud supercomputer software architecture for large scale 3D visualization involving photo realistic terapixel 3D visu-

alization of urban IoT data which can also render it's daily updates . It explored using blender's ray trace engine to a public cloud service such as Azure to construct a scene with data binding scripts to be able to scale to peta-flop level performance. It was able to demonstrate that one can compute a terapixel 3D visualization in under one hour with the system scaling at nearly 98% efficiency to use 1024 public cloud GPU nodes delivering nearly 14 peta-flops. The GPU compute resource available in modern day cloud infrastructure is far greater than anything available on national level supercomputers. This opens a gateway for many users who want to deploy their custom supercomputer infrastructure but don't have access to one. The direct financial costs of access, procuring and running these systems is low. The indirect costs relating to cloud development and maintenance etc would reduce and be optimized over time making this a feasible way to run large scale computations [Holliman et al., 2019]

2.9 Traffic Visualization

Visualizing traffic data in 3D can give clear insights into traffic patterns, densities, and flows that might not be obvious in 2D views. This can be incredibly helpful in urban planning and traffic management by offering a better understanding of how traffic moves over time and space. In this section, different approaches to traffic visualization for congestion exploration and surveillance are reviewed.

2.9.1 Visual Analytics System

[Lee et al., 2020] present the details of an interactive visual analytics system that enables traffic congestion exploration, surveillance, and forecasting based on vehicle detector data. Through domain expert collaboration they extracted task requirements and incorporated the Long Short-Term Memory (LSTM) model (these are a type of recurrent neural network capable of a learning order dependence in sequence prediction problems) for congestion forecasting, and designed a weighting method for detecting the causes of congestion and congestion propagation directions. This visual analytics system they designed enables

users to explore congestion causes, directions, and severity.

Several government organizations having invested considerable resources towards the installation of vehicle detectors and Intelligent Transportation Systems (ITS) including Siemens Split Cycle Offset Optimisation Technique (SCOOT) systems want to analyze and monitor road congestion. But analyzing and surveilling congestion is rather difficult due to the high volume and complexity of the data, bringing need for systems that can monitor a variety of streaming data from multiple sources and integrate this data for real-time congestion monitoring, prediction, and retrospective analysis. In [Lee et al., 2020], the authors present a visual analytics system as shown in Figure 2.1 to support analysts and reporters in the congestion management and traffic broadcast domains. Two views namely StreamView and PropagationView to view the congestion in different perspectives are presented. Two primary modeling options are available to predict road congestion: Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). The LSTM model was used for predicting both road speeds and congestion conditions.

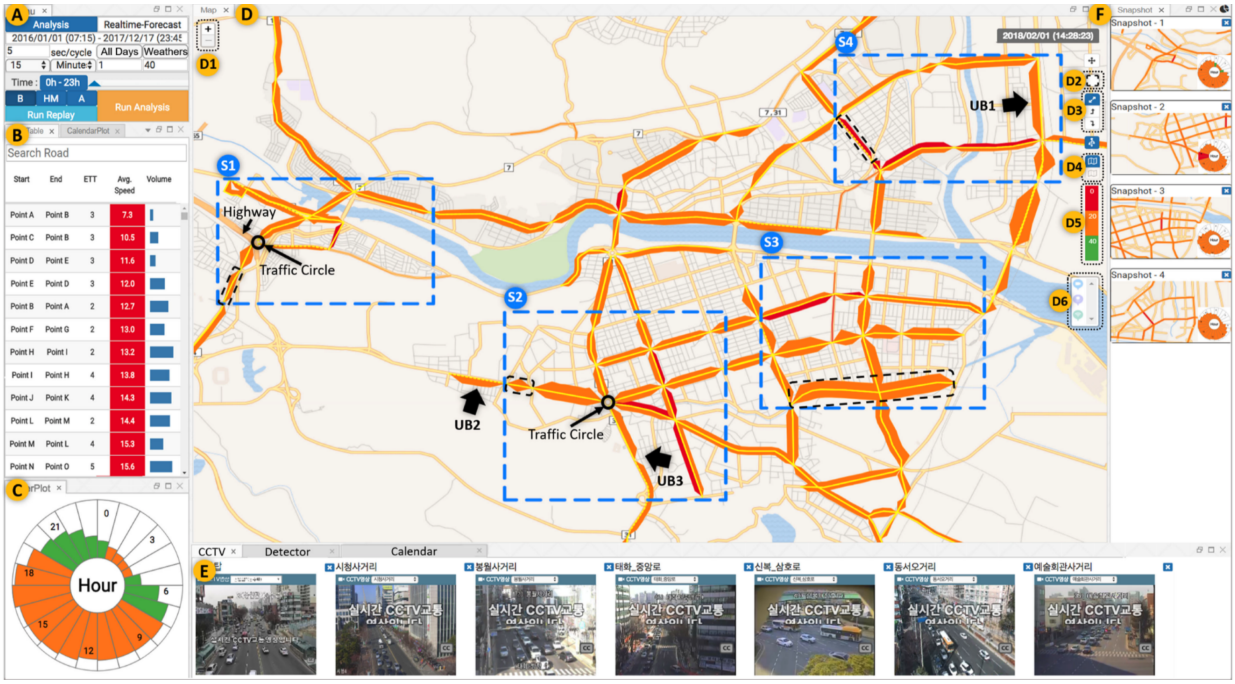


Figure 2.1: The visual analytics system developed by [Lee et al., 2020] © 2022 IEEE

Three case studies are presented here. In the first study, authors illustrate how an analyst used the system to analyze congestion characteristics of the city. Secondly, they have demonstrated the usefulness of their system for the city's congestion improvement pro-

jects. The final study illustrates how congestion reporters in a radio station utilize their system during their broadcasts and the expert feedback is collected. The visual analytic system enables domain experts to have overviews of city-wide congestion conditions by utilizing multiple coordinate views and allowing many filtering options, such as dates, time, speed and volume [Lee et al., 2020].

2.9.2 Geographic and temporal visualization of New York taxis

[Ferreira et al., 2013] propose a new model that allows users to visually query taxi trips across New York. This model supports origin-destination queries that enable the study of population mobility across the city. They present a model which enables interactive response times; makes use of an adaptive level-of-detail rendering strategy to generate clutter-free visualization for large results as shown in Figure 2.2. Ferreira explores a series of case studies motivated by traffic engineers and economists to show how the model and system enable domain experts to perform tasks that were previously unrealistic for them.

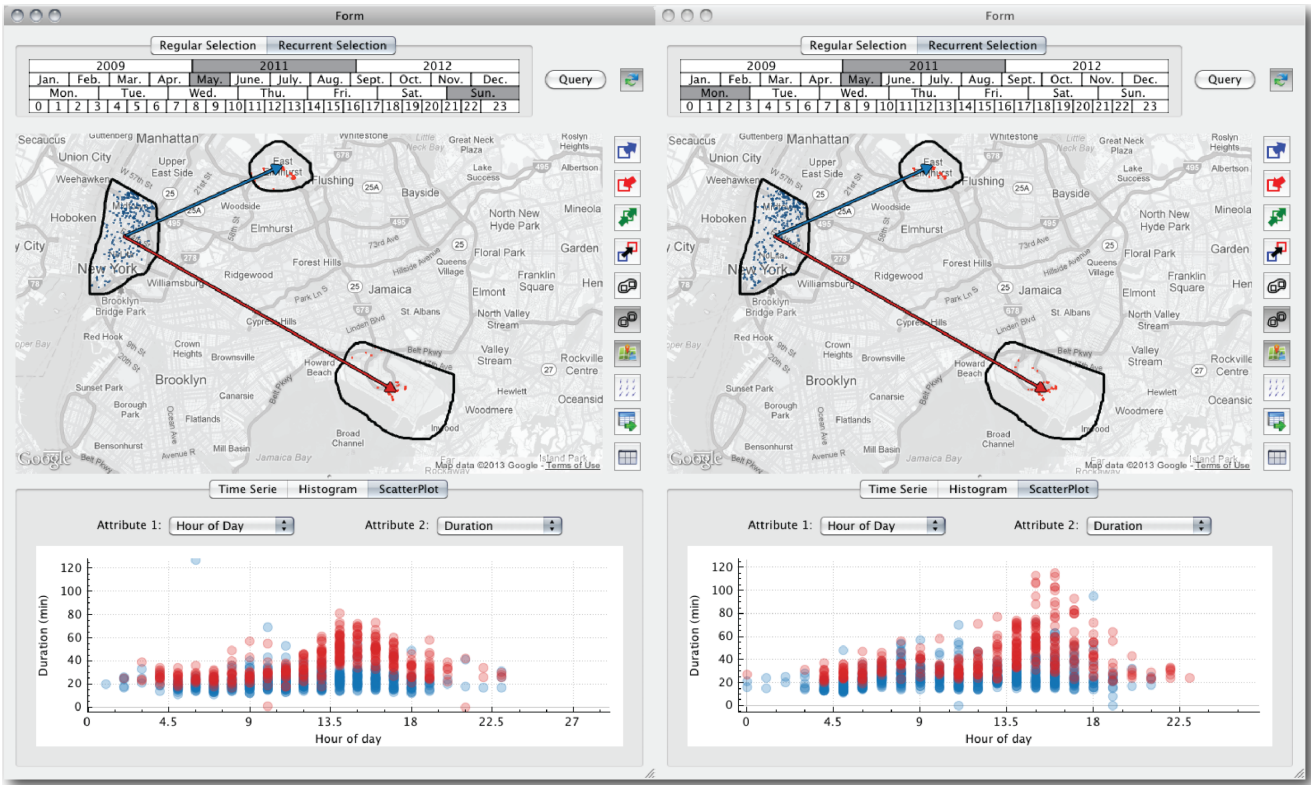


Figure 2.2: The Geographic and temporal visualization model developed by [Ferreira et al., 2013] © 2022 IEEE

[Ferreira et al., 2013] states that in New York city, each day 13,000 taxis carry over one million passengers and make, on average, 500,000 trips totaling over 170 million trips a year. They carried out interviews with social scientists and engineers that have used this data-set in their research to better understand their needs. [Ferreira et al., 2013] found that existing tools that are commonly used, including R, MatLab, Stata, ArcGIS and Excel, could not handle the large data sets for geographic visualizations. This prevented scientists from analyzing the whole data. Instead, they first select small slices and then load them into these tools for analysis.

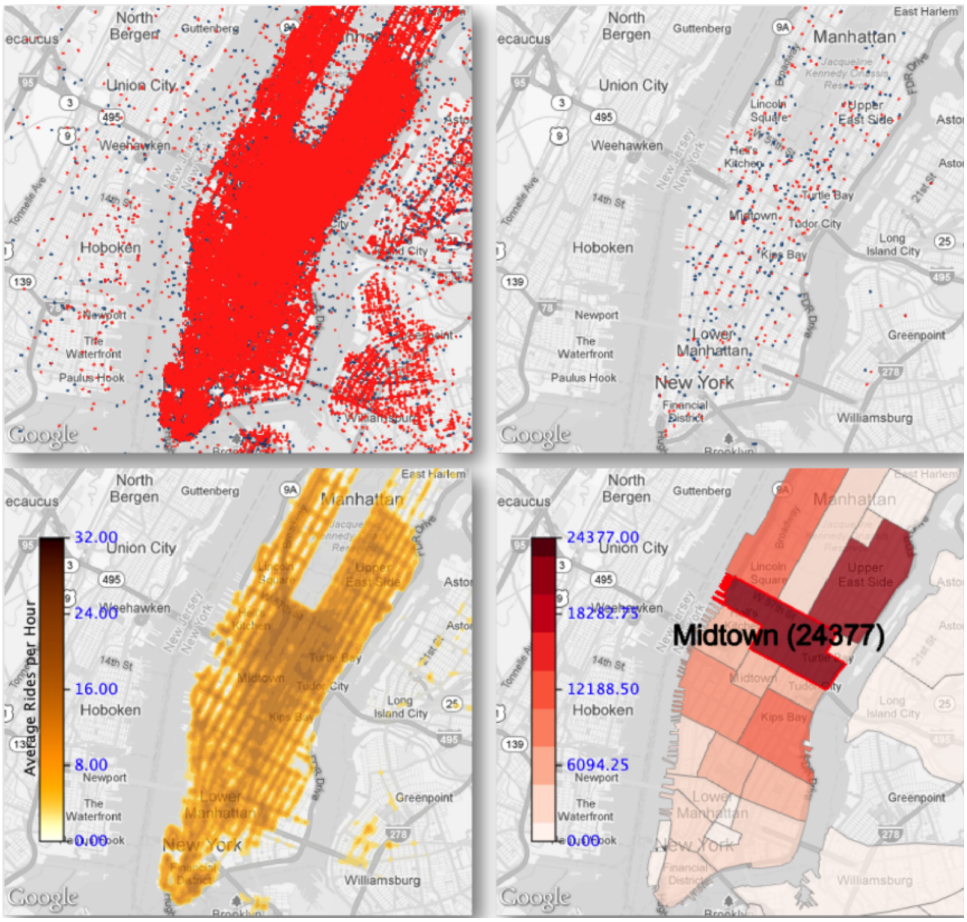


Figure 2.3: Different spatial visualizations of taxi trips for the week 05/01/2011 to 05/07/2011 in the TaxiVis model used several LOD developed by [Ferreira et al., 2013] © 2022 IEEE

[Ferreira et al., 2013] propose a new visual query model TaxiVis that supports complex spatio-temporal queries over origin-destination (OD) data. With no requirement for users to be experts in any textual query language. In summary they can directly query the data using visual operations in the tool as shown in Figure 2.2. In visualizing the taxi data to

better improve clutter when dealing with large data-sets, different spatial visualizations of taxi trips were used. In Figure 2.3 data of taxi trips from 05/01/2011 to 05/07/2011 was used to render with different Level of Detail (LOD), heat maps and density organized by well-defined regions and neighborhoods.

After getting information from engineers and economists from New York University they better understood the questions they wanted to investigate They were interested in understanding the dynamics of the city, how different aspects of the data vary over space and time. For example, "What is the average trip time from Midtown to the airports during weekdays?" or "How does the taxi fleet activity vary during weekdays?". The limitations of doing complex analysis by researchers using the existing tools is that they can only consider slices of data to test their hypothesis and it gets very tedious to get this from raw data. To address these limitations, TaxiVis was an important step forward to unify the two phases of exploration of data, selection and visual analysis. Different case studies were also presented to aid the above.

It was evident that using visual query model that allows users to quickly select data slices and explore them helps attain a good balance between simplicity and expressiveness. Another important contribution of this work is that the system design, which not only combines the visual query model with other visualization primitives, but also addresses performance challenges that arise due to the scale of the data. In particular, to support interactivity, authors designed an efficient storage manager as well as a rendering subsystem [Ferreira et al., 2013].

2.9.3 High-Resolution and High-Dimensional visual analysis of traffic data

[Häussler et al., 2018] propose a visual analytics system for the exploration of environmental factors using high-resolution and high-dimensional mobility sensor data. Additionally the authors introduce an interactive visual logging approach to enable experts to cope with complex interactive analysis processes and the problem of the reproducibility of results when sharing analytics.

[Häussler et al., 2018] argues that global warming and climate change are related to industrial pollution and urbanisation and tracking and visualizing CO2 emissions by motor vehicles can help urban and transport planners satisfy both mobility and health demands. The paper explores several tasks such as

T1 Detection of current pollution situations in a city for policy actions (mid-term)

T2 Short term or real-time regulation/adaption of traffic flows to minimize congestion and reduce pollution.

Häussler focused on T1 and T2 by having a holistic view to the analysis. The mobility data was helping experts to develop and realize sustainable mobility concepts. Using data from the citizen-science platform enviroCar, authors develop a visual analytics system allowing analysts to leverage their background knowledge in the analysis process. The system provides multiple interactive views linked via state-of-the-art techniques such as Brushing and Linking. Experts can include environmental factors such as CO2 emissions in the visualization

T1 – Detection of Current Pollution Situations Focusing on characteristics of individual data points which are nearby to specific infrastructures rather than whole trips helps determine the current pollution situation. Infrastructure such as road sections, road network crossings etc are the ones which are interesting. The visual analytics system [Häussler et al., 2018] developed provides spatial filtering by switching into a draw-mode and selecting areas of interest allowing to filter unwanted roads as shown in 2.4.

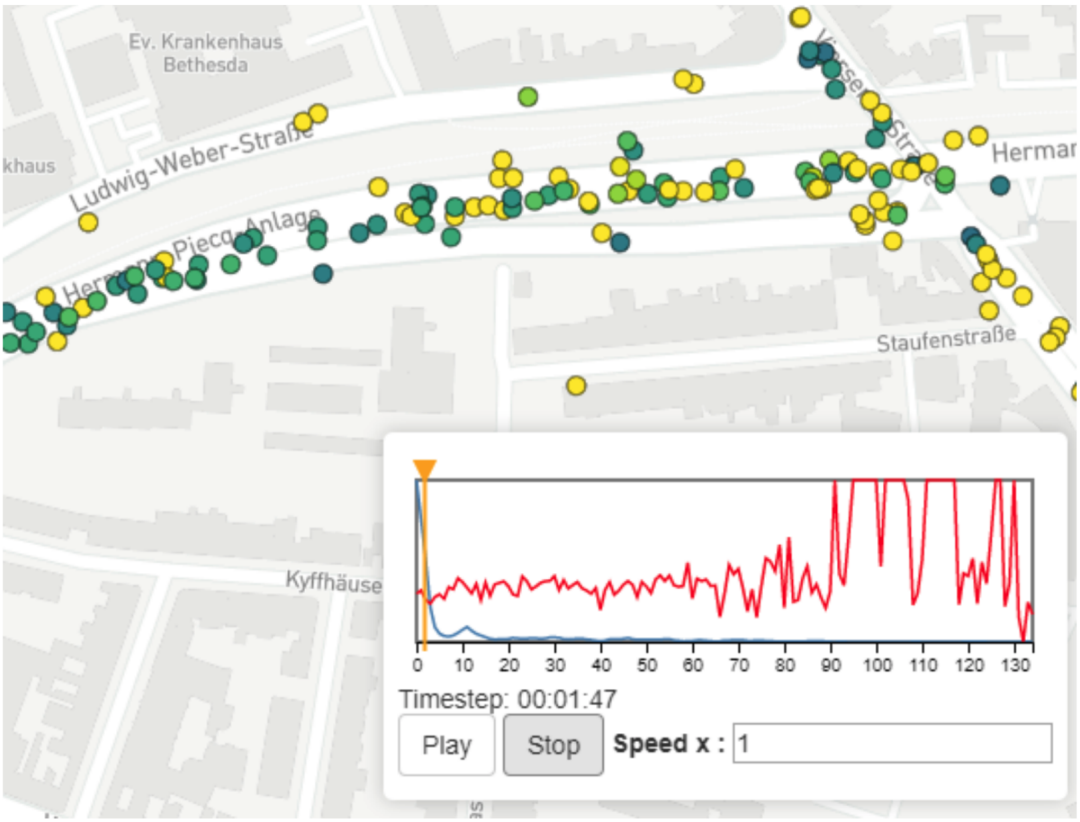


Figure 2.4: Visualization of several trips within a user-defined area elaborated by a dot-map. The temporal context is represented as an animation © 2022 IEEE

T2 – Regulation of Traffic Flows Traffic flows have high sensitivity to time, and re-occurring patterns and anomalies concerning day time such as the hour of day, day of week, festivals or special occasions are of special interest to traffic operators. [Häussler et al., 2018] developed an intuitive overview of the temporal distribution of the available traffic data such as the number of data points, average speed, and CO2 value using a radial visualization. The Clock View visualizes and orders the appropriate data values aggregated by the average of samples which are taken a minute per day as shown in 2.5.

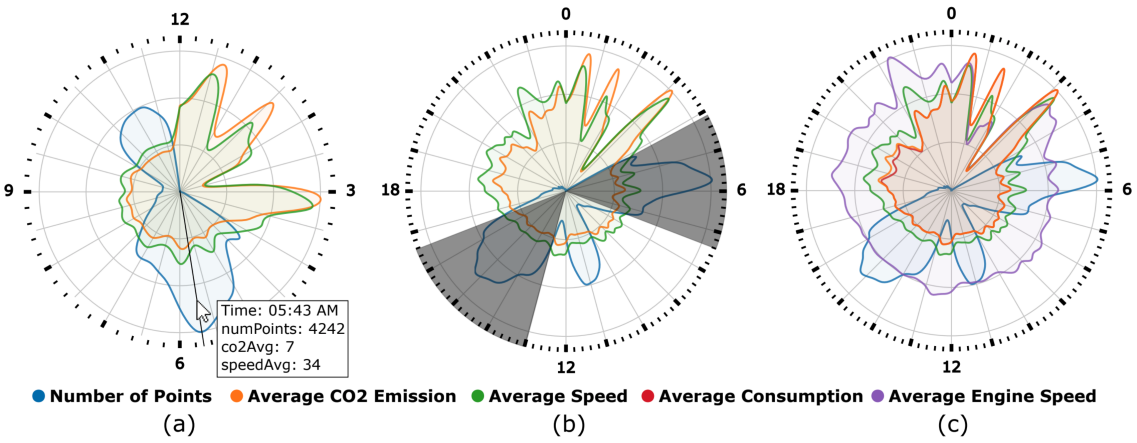


Figure 2.5: Clock like visualization used to represent number of data points, average speed and CO2 emissions done by [Häussler et al., 2018] © 2022 IEEE

Visual Interactive Logging [Häussler et al., 2018] say that performing a visual analysis of urban traffic data is a complex task that requires different visualizations using different perspective and aspect of the data. Both T1 and T2 visualizations provide interaction techniques for performing visual data analysis such as navigation, zooming for the map and filtering for the clock-based glyph visualization. They have used brushing and linking to keep all visualizations consistent. The experiment was demonstrated by approaching both experts who have used the system before and the experts who have not.

Evaluation The authors have done evaluation by experts and non experts using both interactive visualizations to allow dynamic and versatile data exploration. They have also done qualitative evaluation by describing each of the expert evaluation results. The results of the qualitative evaluation are very positive. Both sets of experts used color to map various attributes such as CO2 emission on the road network. They also used visual variables, such as size, to visualize the traffic volume. Both experts mentioned that a purely map-based approach is not sufficient to visualize all environmental factors. They also note that the used enviroCar data-set only reflects a small sample of the whole traffic flow data-set.

Chapter 3

In Pursuit of a Grammar for 3D Visualization

To understand the motivation behind Aloka, it would be best to describe the needs that arose organically during the research study. It was also part of the research goal to create a system capable of describing 3D visualizations

3.1 3D Visualizations

This section discusses several 3D visualizations designed and developed for exploration and presentation at conferences and in published papers. It examines the initial design and development process conducted manually in Blender and explores various visualization types for different display formats, including stereoscopic and VR visualizations. Throughout this process, the design and development were manual and time-intensive. Following this, the Petascale [Holliman et al., 2019] project demonstrates cloud computing's capacity to create terapixel-scale visualizations. Among these smaller projects, a significant issue emerged: the ability to quickly prototype new visualization designs in response to changes in data and design. Although the Petascale project could produce highly detailed and large-scale 3D data visualizations, any changes in data or design necessitated major modifications to the original model. To address this, the exploration of a 3D grammar is presented, justifying its use in facilitating more efficient design and

exploration processes.

3.1.1 Data Visualization in Blender

The Urban Observatory offers an API that delivers data on weather, air quality, temperature, CO2, NO2, humidity, and wind speeds through a RESTful endpoint. The Alphasense sensors that are used to collect these readings provide high spatial and temporal resolution of air quality measurements and noise data which is representative of population exposure. They are not however certified MCERTS sensors and provide indicative rather than precise calibrated readings. Open Data Service is a service from Newcastle city council on behalf of north east combined authority, to provide travel and transport related data-sets. Specific travel and transport data from this API is sourced from Urban Traffic Management Control (UTMC). This also uses a RESTful API, which is useful for quick integration. [The Urban Observatory Link]

Virtual Newcastle Gateshead (VNG) is a joint venture between Northumbria University, Newcastle City Council and Gateshead Council to create a three-dimensional digital model of Newcastle and Gateshead. The model to date is based upon aerial photogrammetry and laser scanning survey techniques. [VNG Project Venture Link]

Urban Insight Cloud Engine (UICE) is a system designed at the University's Digital Institute to visualize readings from environments sensors in a high-quality 3D simulation of Newcastle The scripts are run on the VNG model in a private cloud to produce the visualizations and displayed on the client side.

3.1.1.1 Temperature & NO2 Visualization

Live Temperature and NO2 visualizations designed and deployed as shown in Figure 3.1. Quality can be switched between HD and 4K within the Web app. The glyphs represent sensors which are taking live data from Urban Observatory and displaying their value via a colour code depicted in the bottom right (this uses Met office colour standard). An

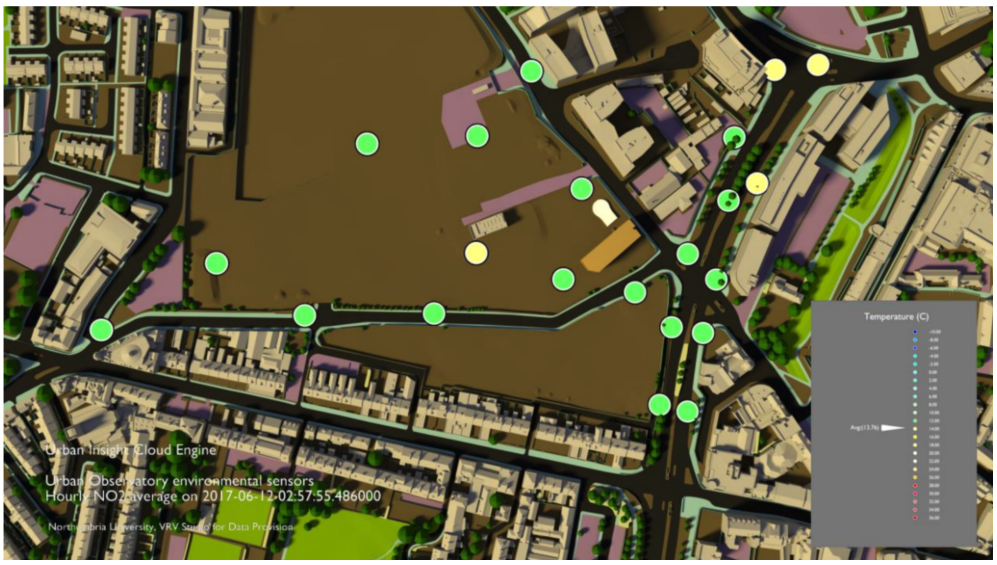


Figure 3.1: Temperature Visualization

average value is also displayed for readings that day. The units used for temperature are in degrees Celsius and NO2 is (parts per billion) ppb.

3.1.2 Immersive Visualizations

Further experimentation with the same visualizations to make them more immersive required to change the display type, this caused a challenge and required a custom script to design these visualizations.

3.1.2.1 Stereoscopic Visualization

A top-bottom stereo image as shown in Figure 3.2 is also rendered for use in stereoscopic displays active/passive. This is also the display to view the visualization in relative 3D. This gives a depth perception to the viewer and can convey information in a more immersive way than compared to the HD and 4K visualizations.

3.1.2.2 VR 360 Visualization

To take the next step in immersion, the visualizations were rendered as 360 images. To avoid complications of building a library from scratch to view the 360 VR image in a browser a [KRPano] license was bought. Krpano viewer is a small and very flexible high-performance viewer for all kind of panoramic images and interactive virtual tours.

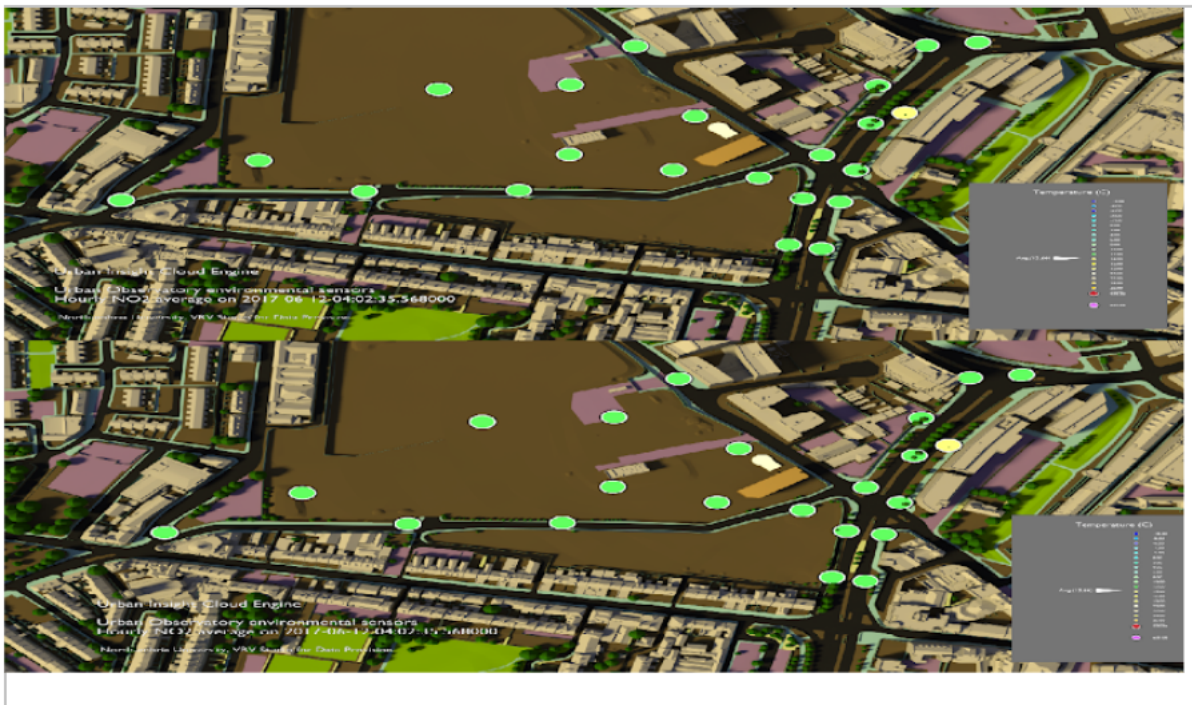


Figure 3.2: Stereoscopic image rendered on the left and the VR image rendered on the right

It was successfully integrated into the angular app in the front-end. Virtual tours was also constructed such as the user can view said visualization from 6 predefined hotspots and can navigate between them. Hot spots are points on the visualization where one can warp to and see the environment from that point of view. This was tested to be cross platform and cross browser compatible. Flash support is purposely dropped due to security concerns of flash even though Krpano supports it, the VR images are rendered in HTML 5 canvas exclusively. This also allows for integration of Google cardboard type VR headsets to view the entire visualization in VR and interact with the hotspots. Navigation

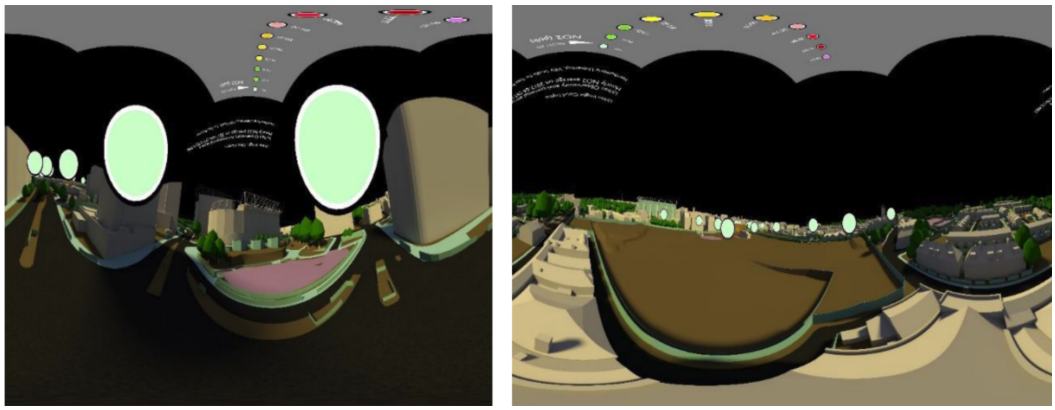


Figure 3.3: 360 VR image rendered

in the VR environment is by looking in a direction or at a hotspot. The viewer also allows for gyro based input in the browser so it works on all mobile and tables to look around using gyroscopic navigation. All the 6 images are rendered live in the private cloud and updates are loaded seamlessly via the front-end to the client's browser.

3.1.3 Gigapixel & Terapixel Visualization

When exploring large data-sets there became a possibility to explore and to integrate large data in visualizations such as multiple data types on a 3D visualization. This gave requirement for higher resolution and higher quality visualizations and led to the gigapixel project and eventually the Terapixel project as shown in Figure 3.4.

The terapixel project as described in [Holliman et al., 2019] was a cloud super-computing project capable of delivery petaFLOPs of compute for workloads. It was a synthetic trillion pixel panoramic telescope looking deep into Newcastle's smart city IoT data. The scale of this render was over 1 trillion pixels nearly 530,243 full HD TV images and all this done as a live render as in it updates the visualization with live IoT city data at least once a day. The system scaling at 98% efficiency to use 1024 public cloud GPU nodes delivering 14 petaFLOPS. The resulting terapixel image supported interactive browsing of the visualization in the city and its data at a wide range of scales. [Holliman et al., 2019]

3.2 Design Objectives for a new System

Constantly having to redesign and redeploy for each kind of visualization, setting and display environment made the idea of having declarative systems to define visualizations very attractive. This was also part of the research goal to try and encode 3D visualization in a DSL like format. Thus Aloka was the next project to achieve this.

While designing Aloka the key things to achieve were:

A 3D Grammar: To design a grammar/language for 3D visualizations

Declarative Schema: To have a declarative approach in its design

Scalable: Flexibility of having to use the grammar in a cloud environment.

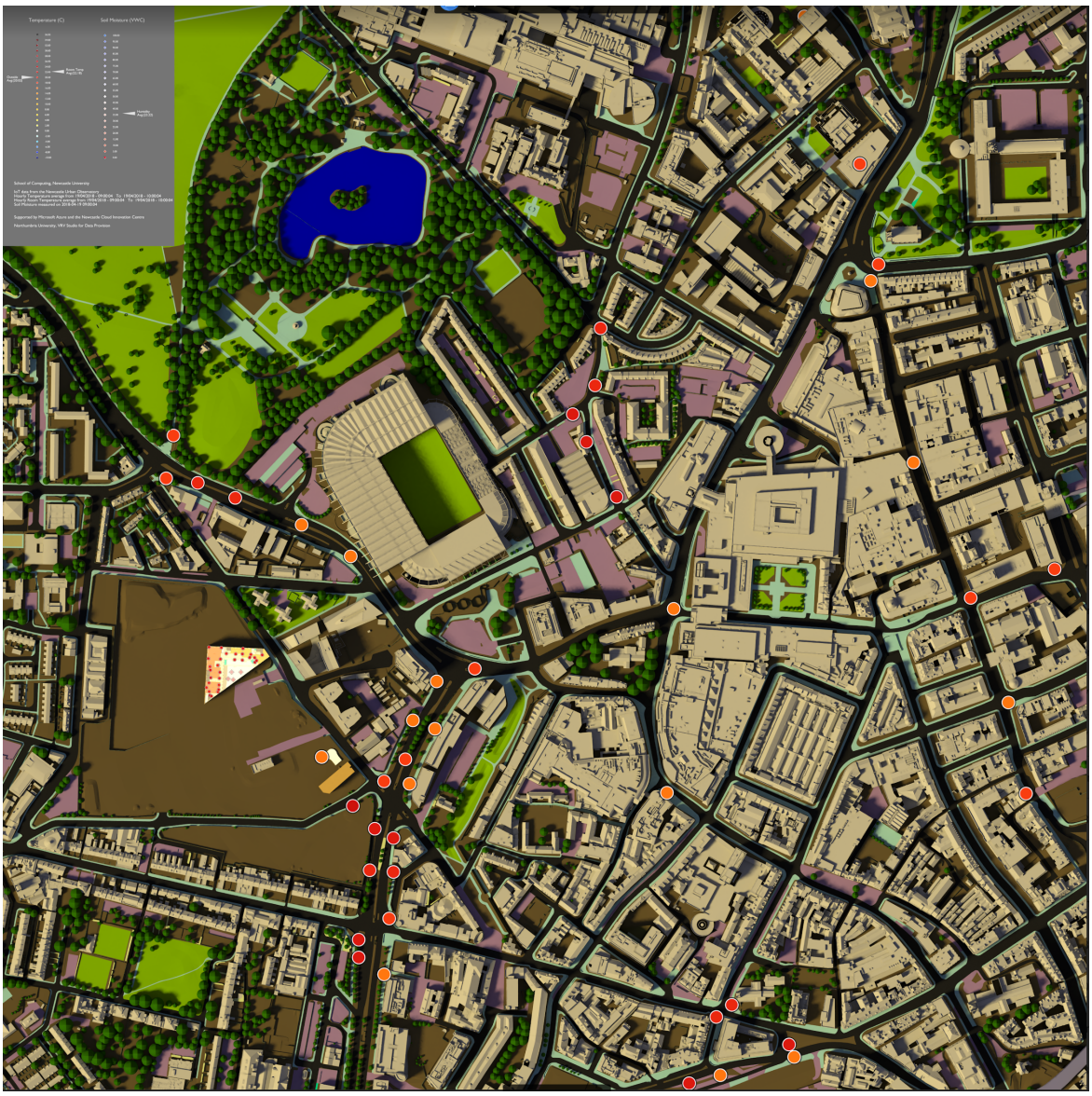


Figure 3.4: Gigapixel visualization of Temperature

Platform Agnostic: Platform and application independent

Data Binding: Data-binding from a visual and interactivity perspective, along with the expressivity in the variations and scope of visual elements used with data-binding

3.3 Exploring grammars for visualization

In the pursuit for designing a grammar for 3D Visualization, it was important to scope out the existing grammars that are prominent in the 2D space as there are no 3D grammars for visualization as of yet. The most important ones were D3 and Vega (Vega-lite is a derivative of Vega). The core technologies that lie below these grammars/libraries are crucial in understanding approaches to designing grammar to visual elements. This will be explored the following sections.

3.3.1 D3 - the Javascript library

This Javascript library (also known as an embedded domain-specific language) is quite popular both in the industry and academia and used extensively to create incredibly complex visualizations. D3 library relies on manipulating the Document Object Model (DOM) which is a document model for web documents. It represents the web page so that programs such as Javascript can change the document structure, style, and content. The Document Object Model (DOM) represents the document as nodes and objects and for ease of use with programming languages. D3 was a improvised from Protovis and focused on the following objectives: **compatibility, debugging and performance**.

3.3.1.1 D3 - How it Works

D3 demonstrates exceptional utility by treating the Document Object Model (DOM) as a scenegraph for the manipulation and organization of visual elements. The primary atomic unit in D3 is the selection, which consists of a filtered set of elements queried from the current DOM. D3's operators act on these selections, modifying their content and positioning within the DOM. The data joins mechanism binds input data to DOM elements,

facilitating functional operators that depend on this data and generating enter and exit sub-selections for the dynamic creation and removal of DOM elements. Operators apply changes instantaneously, while animated transitions interpolate attributes and styles smoothly over time within the DOM. Special operators, such as event handlers, respond to user input and enhance interaction. Additionally, D3 includes numerous helper modules, such as layouts and scales, which streamline common visualization tasks.

3.3.1.2 D3 - Limitations

This library however is limited to within the Document Object Model (DOM), although there are WebGL extensions they do not exactly provide a declarative way to define a 3D visualization as the structure for each visualization is different. This provides ease of use but lacks consistency. Not to mention the client overhead as WebGL is fairly taxing on the client computer.

3.3.2 Vega - a Visualization Grammar

Vega is a visualization grammar. It is a declarative language for creating, saving, and sharing interactive visualizations via the web. In Vega, you describe the visual appearance and interactive behaviour of a visualization in a JSON format and generate Document Object Model (DOM) based views using Canvas or SVG. Vega is built on D3.js and concepts from D3 translate quite well. Vega provides basic building blocks for a wide variety of visualization designs. It supports data loading, transformation, scales, map projections, axes, legends, and graphical marks such as rectangles, lines, plotting symbols, etc. Interaction techniques can be specified using reactive signals that can dynamically modify a visualization in response to input event streams via the DOM. A Vega specification are parsed by Vega.js library to generate both static images or interactive web-based views. It also provides a representation for procedural generation of visualizations.

3.3.2.1 Vega - A Visualization Grammar

Vega consists of a unified data model in which input data, scene graph elements, and interaction events are all treated as primary streaming data sources. Vega specification is organized as shown below:

Specification: This provides comprehensive definitions, encompassing an appropriate subset of the data, scales, axes, markers, etc. It also outlines interactive visualization in JSON format.

Config: Set settings for the visual encoding options. The default settings for a number of visual encoding options are defined in a JSON configuration file that the Vega parser accepts. The style and feel of charts can be altered by using various configuration files. Simply said, a configuration file is a JSON object with a collection of named properties arranged by type.

Data: To visualise, define, load, and parse the data. Tabular data, which resembles a spreadsheet or database table, makes up the foundation of the Vega data model. It is assumed that each individual data set consists of a set of records (or "rows") that may have any number of named data attributes (fields, or "columns") The modelling of records uses common JavaScript objects.

Transforms: Before visualising the data, it applies data transformations (filter, sort, aggregate, layout). A data stream is processed through transformations in order to filter data, create new fields, or create new data streams. Usually, transforms are defined in a data definition's transform array. Additionally, post-encoding transformations can be specified in the transform array of a mark definition using transforms that do not filter or create new data objects.

Triggers: Data sets are to be altered or characteristics to be marked in response to signal values. When particular circumstances are satisfied, triggers allow for dynamic modifications to data sets or the marking of items. One or more data updates (insert, remove, toggle, and/or alter) are applied when a trigger expression, which

often refers to one or more signals, evaluates to true (boolean). The attributes (modify only) of mark items inside a mark definition can be updated using triggers, or data objects within a data collection can be updated using triggers. To be noted that derived data sets do not allow triggers, so any triggers defined on a derived data set will be useless.

Projections: projections used in cartography to map (longitude, latitude) data Longitude and latitude pairings are translated into projected (x, y) coordinates using cartographic projections. Both geographic points (such as spots on a map) for which longitude and latitude coordinates are included in the input data and geographic areas (such as countries and states) expressed using the GeoJSON format are laid out using projections in Vega.

Scales: Map data values (numbers, texts) to visual attributes (coordinates, colors, sizes). Scales map data values (numbers, dates, categories, etc.) to visual values (pixels, colours, sizes). Scales, which govern the type of visual encodings, are a crucial component of data visualisation. Vega provides mappings for position, shape, size, and colour encodings and offers a variety of scales for both continuous and discontinuous input data. Axes or legends may be included in Vega specs to help visualise scales.

Schemes: Scale ranges that can be represented by colour schemes. For both discrete and continuous colour encodings, colour schemes offer a collection of designated colour palettes. Scales having discrete domains, such ordinal, quantize, and quantile scales, can employ discrete colour schemes directly. Continuous colour schemes can be used directly with continuous scales (including linear, log, and sqrt scales) and can also be used to create discrete colour schemes by supplying a scheme count attribute.

Axes: Use coordinate axes to visualise scale mappings for spatial encodings. The use of ticks, gridlines, and labels by Axes helps to visualise spatial scale mappings. Axes for Cartesian (rectangular) coordinates are now supported by Vega. Axes can be declared at the top level of the specification, like scales, or as a component of a group mark.

Legends: Create a visual representation of the scale mappings used to encode colour, shape, and size. Legends display scale mappings for visual elements including colour, shape, and size. Legends can be given at the level of a group mark or at the top-level representation, just like scales and axes.

Title: Gives the visualisation a title for the chart. A chart's title is added using the title directive. A title can be specified at the top level of a specification or as a component of a group mark, just as scales, axes, and legends.

Marks: Use graphical symbols like rectangles, lines, and symbols to visually encode data. Using geometric primitives like rectangles, lines, and plotting symbols, graphic marks visually encode data. Marks are the fundamental visual building blocks of a visualisation, offering simple forms with adjustable attributes. Simple constants, data fields, or scales can be used as the specification of a mark attribute. Data values can also be mapped to visual values using scales.

Signals: these are dynamic parameters capable of driving interactive updates. A visualization's parameters are dynamic variables called signals, and they can control interactive behaviours. Signals can be used to define a mark property or a data transform parameter across a Vega specification. Signal values can update in reaction to input event streams, external API requests, or modifications to upstream signals since they are reactive. Event streams record and order input events like touchmove or mousedown. Signals with associated event handlers are reevaluated in the order specified when an event occurs. The visualisation is then automatically re-rendered after updated signal values have been propagated to the remainder of the specification.

Event Streams: Event streaming to define interactions, define input event streams. The main technique for representing user input to enable dynamic, interactive visualisations is event streams. A series of input events, such as mouse clicks, touch movements, timer ticks, or signal changes, are captured in event streams. Events that meet a stream definition trigger the evaluation of any associated signal event

handlers, possibly modifying a signal value.

Expressions: Custom calculations over data and signals can be expressed. Vega contains its own expression language for defining fundamental formulas, enabling unique calculations. These expressions, for instance, are used in signal definitions to update values in response to user input and in filter and formula transforms to modify data. A limited version of JavaScript is used in the expression language. Boolean, number, text, object (), and array ([]) literals are all supported, as are all fundamental arithmetic, logical, and property access expressions. There is support for ternary operators (ifTest? thenValue : elseValue) and a unique if(test, thenValue, elseValue) function.

Layout: Layouts are a collection of group marks on a grid. The composition of small multiples and coordinated multiple view displays is made simpler by a layout, which places a group of group markings within a grid. All immediate children group markings will be gathered and placed in accordance with the layout specification when applied at the top level of a specification or within a group mark. The layout engine offers column, row, and grid-aligned layouts in addition to flow layout.

Types: This is a recurring parameter type documentation. The Vega specification properties demand certain parameter types.

3.3.2.2 Vega - The System Design

Architecture: The Vega system architecture integrates streaming database techniques with Event-Driven Functional Reactive Programming (E-FRP). It has data-flow operators and methods to model both raw data and user interactions events as streaming input evenly. Data-flow operators are initiated and connected by the parser which traverses a declarative specification containing definitions for input data-sets, visual encoding rules, and interaction primitives. When data tuples are read or when interaction events occur signals are propagated through the graph with each operator being evaluated in turn. Propagation ends at the renderer.

Constructing the Scene Graph: Vega follows Protovis when constructing a scene-graph it uses bind-build-evaluate pipeline [Bostock and Heer, 2009]. When a specification is parsed, Vega traverses the mark hierarchy to bind property definitions. The property sets are compiled into encoding functions and stored with the specification. Then at run-time build the evaluation operators are created for each bound mark. This build operator then performs a data join to generate one scene-graph element also known as a mark per each tuple in the data-set and the evaluation operator runs the encoding functions. The downstream bound operator calculates the bounding boxes of generated marks. There is an order to generate these marks and the parent elements are generated before the children and rendered on the Document Object Model (DOM).

Signals: Signals are how Vega manages interactive events, an event listener node in the data-flow graph for each low-level event type are required by the visualization example: mouseup or touchstart. These nodes are directly connected to dependent signals as specified by event selectors.

Interactive Performance: Vega library performs both compile and run-time optimizations to increase throughput and reduce memory footprint, which includes tracking metadata to prune unnecessary computation and also optimizing scheduling by inlining it's linear chains of operators. In addition, benchmark evaluations of streaming and interactive visualizations found that Vega meets or exceeds the performance of Data Driven Documents (D3) [Satyanarayan et al., 2015].

3.3.2.3 Vega - Limitations

Though Vega is used as an inspiration for Aloka It cannot be extended to handle 3D visualizations as the concepts of projections, graphical marks etc in 2D do not translate well to 3D. 3D requires a whole new way of looking at visualizations as you are dealing with different parameters. In 3D materials, ray-tracing, global coordinate systems and many other challenges cannot be handled by extending Vega. This needs a custom solution.

3.4 Challenges of 3D Grammar Design

Developing a 3D visualization grammar involves numerous challenges due to the increased complexity and additional dimensions compared to 2D visualization. Here are some of the key challenges:

- Complexity of 3D Transformations:
  - Mathematical Complexity: Implementing rotations, translations, and scaling in three dimensions requires complex mathematical operations and understanding of coordinate systems.
  - Precision Issues: Ensuring accurate and precise transformations, especially when dealing with floating-point calculations, can be challenging.
- Rendering and Performance:
  - Performance Optimization: Rendering 3D scenes can be computationally intensive. Balancing visual quality and performance is critical, particularly for real-time applications.
  - Resource Management: Efficiently managing resources such as textures, shaders, and buffers to prevent performance bottlenecks and ensure smooth rendering especially when considering multiple clients.
- Camera and Perspective Handling:
  - Camera Configuration: Defining and managing different camera settings (e.g., perspective vs. orthographic) and transitions between them can be complex and tricky to implement.
  - Depth Perception: Ensuring that objects are correctly perceived in a 3D space, considering aspects like depth of field and occlusion.
- Lighting and Shading:

  - Realistic Lighting: Simulating realistic lighting, including shadows, reflections, and refractions, involves advanced techniques such as ray tracing and global illumination.
  - Shader Complexity: Writing and optimizing shaders to achieve desired visual effects while maintaining performance can be difficult.

- User Interaction and Navigation:
  - Interaction Design: Designing intuitive interaction mechanisms for navigating and manipulating objects in a 3D space, including zooming, panning, and rotating.
  - Feedback and Control: Providing users with adequate feedback and control to understand and interact with the 3D scene effectively.

- Data Integration and Representation:
  - Data Mapping: Mapping multidimensional data to 3D visual properties (e.g., position, color, size) in a meaningful way.
  - Data Formats: Supporting various data formats and ensuring compatibility with the 3D grammar.

- Complexity of Geometric Shapes and Models:
  - Modeling: Defining and handling complex geometric shapes and models, including their transformations and interactions.
  - Animation: Incorporating animations and dynamic changes within the 3D scene, which requires additional considerations for timing and interpolation.

- Error Handling and Debugging:
  - Error Detection: Identifying and resolving errors in the grammar definitions and their implementation.
  - Debugging Tools: Developing tools to visualize and debug 3D scenes and transformations, which are inherently more complex than their 2D counterparts.

- Compatibility and Standards:
  - Interoperability: Ensuring the 3D grammar can work with existing standards and tools in the field of 3D graphics and visualization.
  - Standardization: Balancing the need for a comprehensive grammar with the adoption of standard practices and conventions in the industry.
- User Experience:
  - Ease of Use: Designing the grammar to be user-friendly and accessible to both novice and experienced users.
  - Documentation and Support: Providing clear documentation and support to help users understand and effectively use the 3D grammar.

Addressing these challenges requires a multidisciplinary approach, combining expertise in computer graphics, mathematics, human-computer interaction, and software engineering.

3.5 Aloka System Design

Aloka is very much inspired from Vega/Vega-lite. Vega-lite is a high-level interactive visualization grammar built on top of Vega and used for rapidly generating visualizations to support analysis. Vega-Lite specifications describe visualizations as mappings from data to properties of graphical marks (e.g., points or bars) [Satyanarayan et al., 2017].
Based on some carefully designed rules, the Vega-Lite compiler creates visualization components such as axes, legends and scales and determines its properties. It gives the specifications expressiveness and clarity while maintaining user control. It also supports data transformations such as binning, filtering, sorting, aggregation and visual transforms. It can also be layered and have multi-view displays while also providing interactive selections.
Vega's system defines the Vega grammar from a large schema file which defines the grammar in its entirety, the advantages are that the API/library never needs to be updated just the schema file when upgrades are being made, the downside is not being able to debug and identify issues in the system as the entire grammar is defined through a large schema file. This can hamper development and debugging process. The complexity of 3D Visualization adds another depth to this problem. Aloka does not have a schema file defining the entire grammar like Vega, as debugging such a system would present great challenges as discussed in the previous section.

3.5.1 System Architecture

The system architecture delineates the various components of the Aloka API as an operational system and its abstract layers. The architecture can be divided into several key modules. The state module manages the visualization state necessary for interactions, such as zoom levels and view navigation. This module then transmits the required state and visualization schema to the pipeline module, which handles both filtering and mapping (as illustrated in the figure below 3.5). The pipeline module organizes visualizations as pipelines to be rendered and comprises multiple layers to generate an abstract

scenegraph. This scenegraph is subsequently processed by the render module, which determines the appropriate render engine or platform to use for specific screen types. This modular division facilitates the creation of a highly scalable system capable of managing multiple interactive 3D visualizations without system bottlenecks.

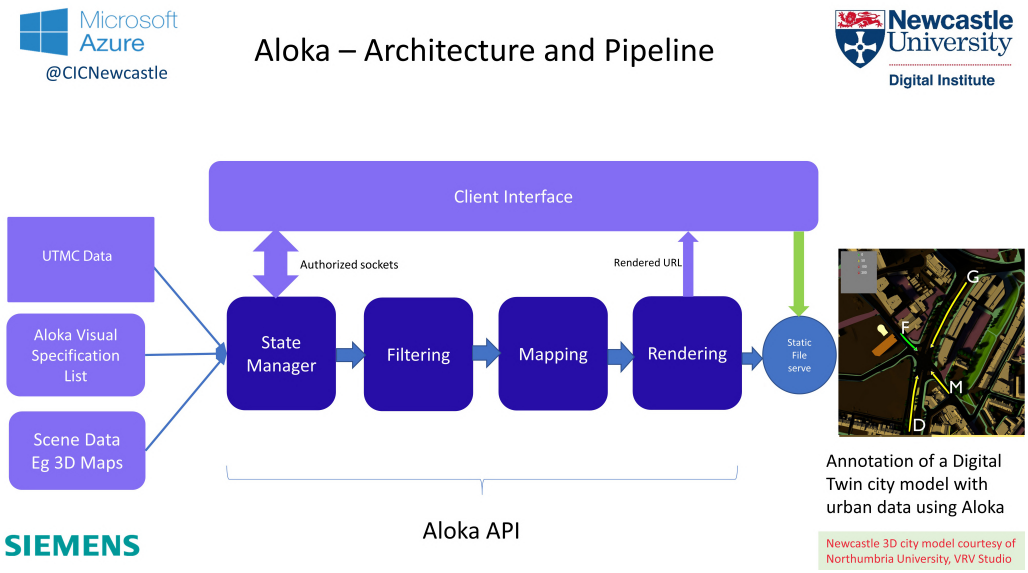


Figure 3.5: The system architecture and pipeline of Aloka

3.5.1.1 State Manager module

The API Architecture consists of a state manager module which manages pipeline processes for each client and ensures resource allocation for each pipeline process. It is also responsible for the process execution, error management, expanding and deploying cloud nodes programmatically based on client demand.

3.5.1.2 Pipeline structure

Each pipeline process is a visualization pipeline and represents a client demand. It receives a grammar specification to visualize. The pipeline class acquires data required for the modules and manages the data sources and updates them accordingly throughout the data transformation process.

The pipeline first runs the filter module as described in the Figure above by providing data

and the grammar specifications. The data is filtered based on JSONPath and transforms are carried out on data which are defined by functions which are in the python libraries such as numpy and pandas). It then maps the data representations to an abstract scenegraph (there is a consideration to export the scene in Universal Scene Description (USD) format for future updates) in the mapping module and stores them in a TinyDB database JSON format. If the scenegraph requires frequent updating then an in-memory DB (TinyDB has two types of storage: JSON and in-memory) option is used for real time visualizations.

3.5.1.3 Render module aka platform selector

The render module activates the required settings for the selected platform or render engine and deploys the render job as a separate process on the cloud which takes available resource it needs. As more processes (each representing a client request) join the stage the state manager must allocate more resources to keep the show running. Once the render job process is complete the process chains return upwards until the client gets the rendered resource (this is usually a link to the location of the rendered resource). This process continues as long there is a client on the other end requesting a visualization.

3.6 Aloka API (Application Programming Interface)

The Aloka API implements the Aloka Grammar. However, several requirements had to be met before this API could be hosted on a server or Mindsphere. Given that the API exposes the system to expensive compute resources such as GPUs, it was prudent to design the system as a front-end client web application that authenticates itself and communicates with a back-end server application. This approach mitigates the risk of unauthorized access to the system. Additionally, this design allows multiple authorized clients to use the system concurrently without being hindered by sequential processes. The system architecture demonstrates the necessity of dividing the modules in this manner to prevent blocking issues between clients. Moreover, the API had to be designed with

parameterization to ensure compatibility with multiple platforms and render engines. The following sections provide a detailed description of the client and server-side designs and the rationale behind these decisions.

3.6.1 Client

3.6.1.1 Frameworks and structure

The client side application is a single page app developed using Angular 5 with Angular materials. It uses a basic login page for security purposes. The primary reason for securing the connection between client and server is to avoid malicious third party clients from overloading the server resources.

3.6.1.2 Protocols

The client uses the http protocol and REST architectural style to gain login access to the Aloka API. Once access is established the API transfers the client to a socket connection which is secured using tokens. This was by design for future real time interactive changes on the API to reflect the visualization. The client is presented with a list of available visualizations and is provided provision to also modify the JSON (uses the angular version of JSON editor library) and send a custom grammar to the API for render via the established secure socket connection.

3.6.1.3 Json editor

The client uses a Json editor library to edit the specification of Aloka before sending it to the server for rendering. Users have the freedom to make edits to the specification here.

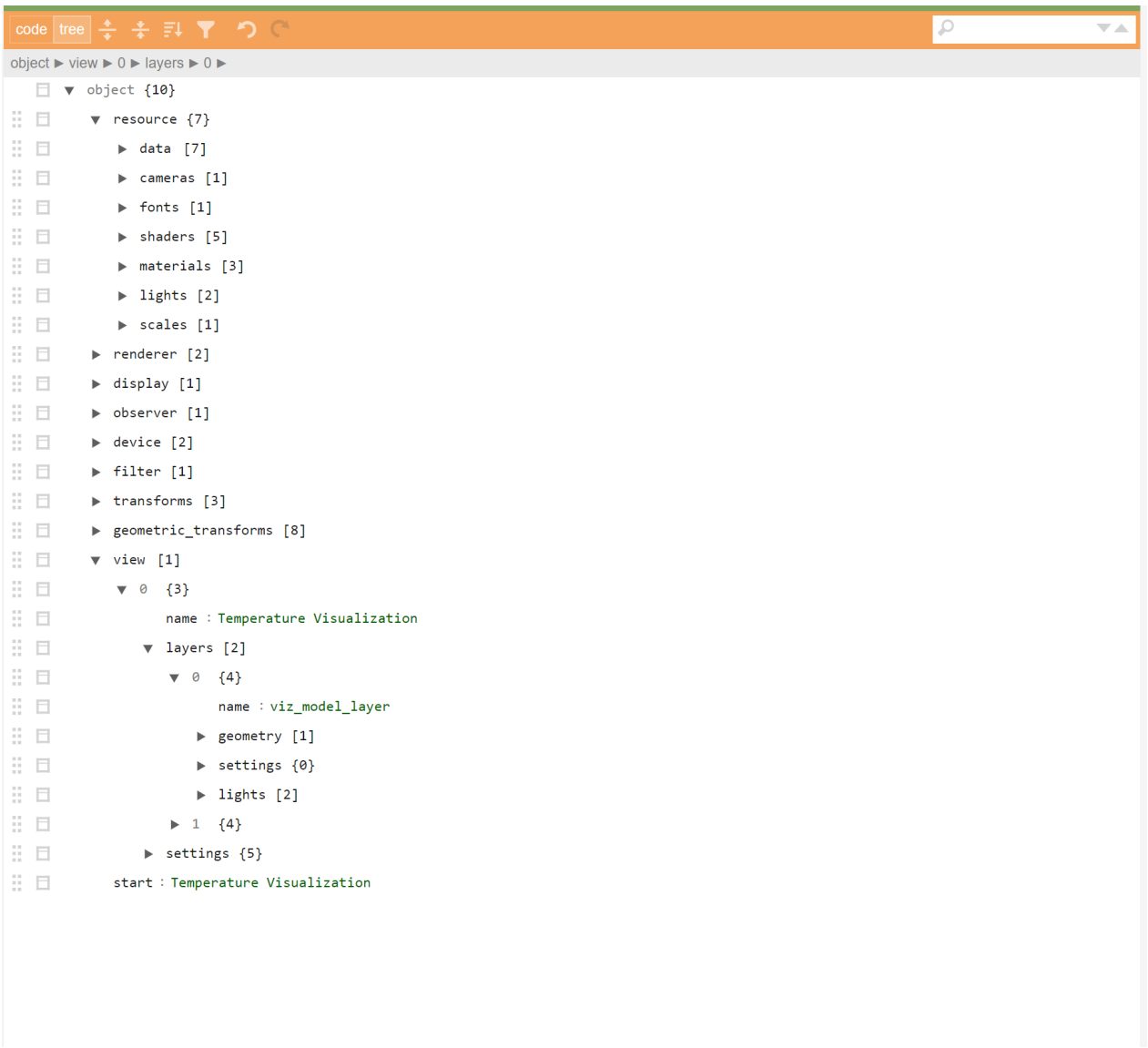


Figure 3.6: The Json Editor used to Edit Aloka Specification

3.6.1.4 Render canvas

Once the render is complete the image is retrieved and displayed on the canvas.

3.6.2 Server

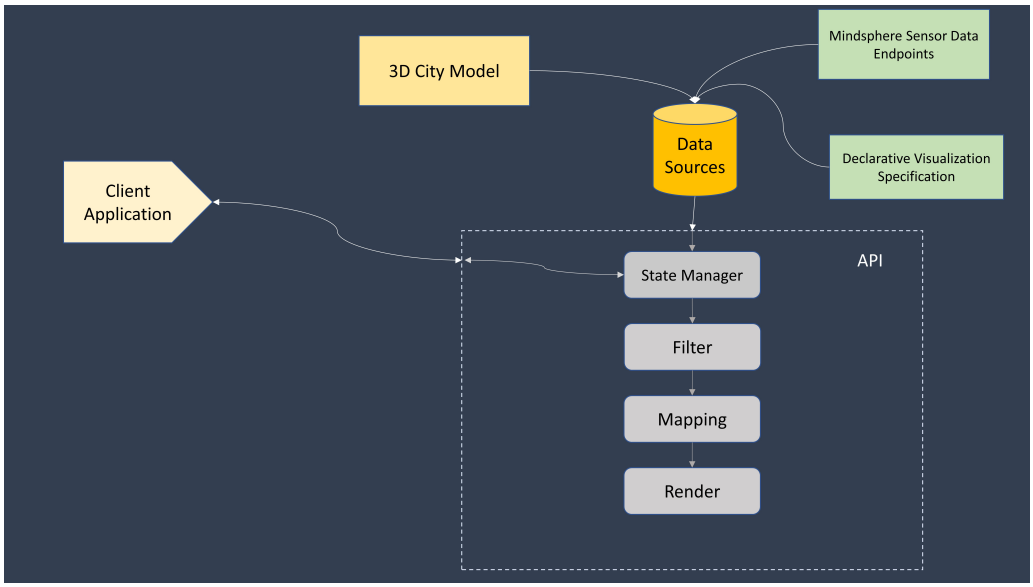


Figure 3.7: The Server to Client Connection and Pipeline showing the Importing of Data

The server API processes Aloka grammar it receives from the client once the secure connection has been established. The server API has a manager module which manages pipeline processes for each request. The request is handled in stages.

3.6.2.1 Data Fetching

The API checks the grammar for any data sources that need to be fetched, and brings the data for further processing. In some cases the data is defined locally. In others the data is live or streamed and has to be fetched from a remote server elsewhere or streamed as a source.

3.6.2.2 Filtering and transforming Data

Filtering data is done using an existing query language for JSON based data called JSON-Path. The API uses a python library which supports this schema called Python JSONPath RW. This library differs from other JSONPath implementations in that it is a full language implementation, meaning the JSONPath expressions are first class objects, easy to analyze, transform, parse, print, and extend. Further transformations on the data are

carried out eg: analytics using the pandas library.

3.6.2.3 Creating Mappable Data points

The system then creates data points from the filtered and transformed data-sets. These data points are mapped accordingly to geometries, materials and lights as defined in the client requested grammar. These mapped data points are stored in a JSON format file used in TinyDB (a compact in-memory database library for python).

3.6.2.4 Construction and Rendering

The mapped objects are retrieved and positioned and placed in the scene as defined. Depending on the render engine and software settings the render process is run in a separate sub-process and managed by the pipeline sub-process. Once the render is complete the image is stored in a ftp location and the URL is sent to the client to view the image.

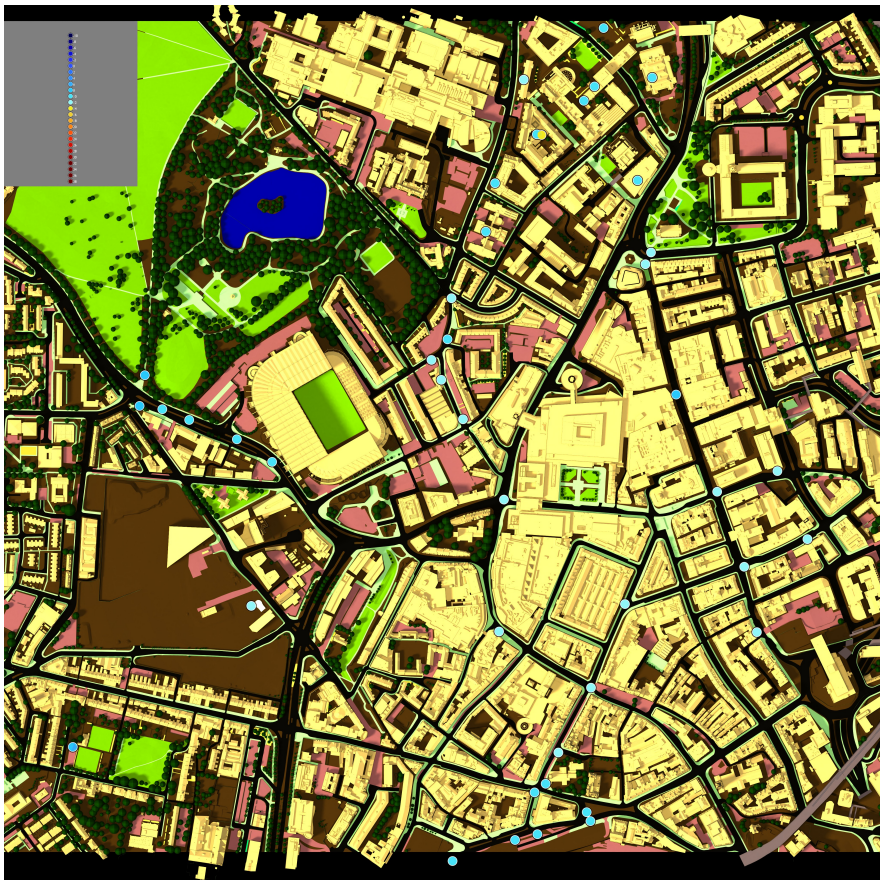


Figure 3.8: The Rendered Temperature Visualization

Chapter 4

Aloka - a Grammar for 3D Visualization

Aloka is a 3D visualization grammar designed for 3D visualization, it uses JSON format for its specification much like Vega. Custom visualizations can be scripted by writing a specification in Aloka.

4.1 Aloka in Extended Backus–Naur Form (EBNF)

Backus–Naur Form (BNF) Backus–Naur form also known as *Backus normal form* is a meta-syntax notation for representing context-free grammars. It is quite often used to describe the syntax of computer programming languages, document formats, instruction sets and communication protocols used in computing. [McCracken and Reilly, 2003]

Extended Backus–Naur Form (EBNF) EBNF is also a meta-syntax notation that expresses the grammar of a language. However it is far more readable and intuitive than Backus–Naur Form (BNF). It is an extension of Backus–Naur Form (BNF). A formal language is defined as a language with a precise structure much like programming languages, data languages, or Domain Specific Languages (DSL). There are multiple variants of EBNF. However, the format used to describe Aloka is : ISO/IEC 14977 EBNF. As this is a ISO standard and well recognized. [Scowen, 1998]

A brief syntax definition of how to structure the meta-syntax notation is shown in the Table 4.1

Usage	Notation
definition	=
concatenation	,
termination	;
alternation	—
optional	[...]
repetition	{ ... }
grouping	(...)
terminal string	” ... ”
terminal string	’ ... ’
comment	(* ... *)
special sequence	? ... ?
exception	-

Table 4.1: The proposed ISO/IEC 14977 standard in [Scowen, 1998] page 7, table 1

4.1.1 Basic Components

These are the fundamental core components and data types. The data types and alias used in Aloka are given below as represented in Listing 1

In this Listing 1, it specifies that a *letter* can be any uppercase or lowercase English alphabet character, while a *digit* can be any numeral from 0 to 9. The *symbol* set includes various punctuation marks and operators. A character is defined as a *letter*, *digit*, *symbol*, or an underscore. The *string* rule denotes a sequence starting with a *character* followed by zero or more characters. Boolean values are represented by *true* and *false*, corresponding to 1 and 0, respectively. A *number* comprises one or more digits, and a *floating_point* number includes a decimal point separating two numbers. The *number_sign* can be either a plus or minus sign, allowing the definition of integer and float values as signed numbers and floating-point numbers, respectively. The *coords* rule defines a tuple of three floats, and *variables* are denoted as a set of strings.

```
        letter = "A" | "B" | "C" | "D" | "E" | "F" | "G"
               | "H" | "I" | "J" | "K" | "L" | "M" | "N"
               | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
               | "V" | "W" | "X" | "Y" | "Z" | "a" | "b"
               | "c" | "d" | "e" | "f" | "g" | "h" | "i"
               | "j" | "k" | "l" | "m" | "n" | "o" | "p"
               | "q" | "r" | "s" | "t" | "u" | "v" | "w"
               | "x" | "y" | "z" ;
        digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;
        symbol = "[" | "]" | "{" | "}" | "(" | ")" | "<" | ">"
               | "'" | '"' | "=" | "|" | "." | "," | ";" ;
        character = letter | digit | symbol | "_" ;
        string = character , { character | character } ;
        true = 1;
        false = 0;
        number = digit , { digit | digit} ;
        floating_point = number , "." , number ;
        number_sign = "+" | "-";

        integer = number_sign , number ;
        float = number_sign ,  floating_point ;
        coords = (float, float, float);
        variables = {string};
```

Listing 1: Basic components - These are low level components and data types

4.1.2 EBNF Grammar Components

These are the grammar components which are defining the rules regarding the meta-syntax. This is similar to the EBNF default as shown in Listing 2

```
1
2     identifier = letter , { letter | digit | "_" } ;
3     terminal = "'" , character , { character } , "'"
4              | '"' , character , { character } , '"' ;
5
6     lhs = identifier ;
7     rhs = identifier
8          | terminal
9          | "[" , rhs , "]"
10         | "{" , rhs , "}"
11         | "(" , rhs , ")"
12         | rhs , "|" , rhs
13         | rhs , "," , rhs ;
14
15    rule = lhs , "=" , rhs , ";" ;
16    grammar = { rule } ;
```

Listing 2: EBNF Grammar Components - These are defining how the meta-syntax is being defined.

The given EBNF code in Listing 2 defines the syntax for a simple grammar specification language. An *identifier* consists of a *letter* followed by zero or more combinations of letters, digits, or underscores. A *terminal* is a sequence of characters enclosed in either single or double quotes. The *lhs* (left-hand side) of a rule is defined as an *identifier*, while the *rhs* (right-hand side) can be an *identifier*, a *terminal*, or nested structures using square brackets, curly braces, or parentheses. The *rhs* can also include alternation (denoted by the pipe symbol —) and concatenation (denoted by a comma). A *rule* is formed by an *lhs* followed by an equals sign, an *rhs*, and a semicolon. The *grammar* itself is composed of one or more such rules. This grammar effectively captures the essential components for defining syntactic rules in a formal language specification.

4.1.3 Common Variables

Common variables used across the Aloka grammar among rules and properties are defined to their very data types or refer to the basic component.

```
1     abrv = string;
2     units = string;
3     value = string;
4     name = string;
5     source = string;
6     format = string;
7     type = string;
8     api = string;
9     client_secret = string;
10    client_id = string;
11    tenant = string;
12    word=string;
13    line=string;
14    regular = string;
15    bold=string;
16    italic=string;
17    underline=string;
```

Listing 3: Common Variables used in the Aloka Specification.

The provided EBNF code in Listing 3 extends the previously defined grammar with additional rules for various string-based identifiers commonly used in configuration or specification contexts. Each rule specifies that an *abrv* (abbreviation), *units*, *value*, *name*, *source*, *format*, *type*, *api*, *client_secret*, *client_id*, *tenant*, *word*, *line*, *regular*, *bold*, *italic*, and *underline* are all defined as **string** types. This denotes that these elements are expected to be sequences of characters, encapsulating various identifiers and configuration elements typically required in application settings or API interactions.

4.1.4 Aloka Specification

The Aloka Specification as shown in Listing 4. Each class is further explored and detailed below.

```
aloka = { visualization };

visualization = { [resource] } , {view}, {display}, {observer}, {device}, {filter},
{transforms}, {geometric_transforms}, {renderer}, {start};

start = view;
```

Listing 4: The Aloka Specification - core classes

The provided EBNF code in Listing 4 defines additional grammar rules related to visualization components within the context of the previously established grammar. The *aloka* rule consists of one or more *visualization* elements. A *visualization* includes optional sequences of *resource*, *view*, *display*, *observer*, *device*, *filter*, *transforms*, *geometric_transforms*, *renderer*, and *start*. The *start* rule specifically begins with a *view*. This grammar outlines the structure and composition of visualization elements, detailing the relationships and optional sequences of various components involved in a visualization setup.

```
resource = { data } , { shaders }, {materials}, {lights}, {cameras}, {fonts}, {scales};

data =  (name, type, source, [params], format);
params = (auth, variables);
auth =  (type, api, client_secret, client_id, [tenant]);

shaders = (name, type, [roughness], [strength], color);
roughness = number;
strength = number;
color = (floating_point, floating_point, floating_point);

materials = (name, surface, [volume]);
surface = (node, shader_a, shader_b, fac);
node = string;
shader_a = shaders;
shader_b = shaders;
fac = floating_point;
volume = (node, shader_a, shader_b, fac);

cameras = (name, type, sensor_size, orthographic_scale, resolution);
resolution = (width, height);
sensor_size = number;
orthographic_scale = floating_point;
width = number;
height = number;

lights = (name, type, size, strength, color);
size = floating_point;

fonts = (name, family, [geometry], [spacing], [fill]);
family = (regular, bold, italic, underline);
geometry = (extrude, offset, depth);
spacing = (letter, word, line);
fill=string;

scales = (name, type, range, data);
range = (minimum, maximum, iteration);
minimum = integer;
maximum = integer;
iteration = integer;
```

Listing 5: The Resource Class

The EBNF code in Listing 5 extends the grammar with detailed rules for defining resources within a visualization context. The *resource* rule consists of one or more of the following components: *data*, *shaders*, *materials*, *lights*, *cameras*, *fonts*, and *scales*.

- Data: Defined by a *name*, *type*, *source*, optional *params*, and a *format*. The *params* include *auth* and *variables*, with *auth* comprising a *type*, *api*, *client_secret*, *client_id*, and an optional *tenant*.

- Shaders: Defined by a *name*, *type*, optional *roughness* and *strength*, and a *color*. The *roughness* and *strength* are numbers, and *color* consists of three floating-point numbers.

- Materials: Defined by a *name*, a *surface*, and an optional *volume*. The *surface* includes a *node*, *shader_a*, *shader_b*, and a *fac*, while the *volume* has similar structure. The *node* is a string, *shader_a* and *shader_b* are shaders, and *fac* is a floating-point number.

- Cameras: Defined by a *name*, *type*, *sensor_size*, *orthographic_scale*, and *resolution*. The *resolution* is described by *width* and *height*, both numbers, with *sensor_size* as a number and *orthographic_scale* as a floating-point number.

- Lights: Defined by a *name*, *type*, *size*, *strength*, and *color*. The *size* and *strength* are floating-point numbers, and *color* consists of three floating-point numbers.

- Fonts: Defined by a *name*, *family*, and optional *geometry*, *spacing*, and *fill*. The *family* includes *regular*, *bold*, *italic*, and *underline*. *Geometry* includes *extrude*, *offset*, and *depth*, while *spacing* includes *letter*, *word*, and *line*. *Fill* is a string.

- Scales: Defined by a *name*, *type*, *range*, and *data*. The *range* includes *minimum*, *maximum*, and *iteration*, all of which are integers.

This grammar specifies the structure and interrelations of various resources used in a visualization, ensuring each component is clearly defined and consistently structured.

```
display = {(name, type, resolution, [display_size], [profile])};

display_size = (value, units, abrv);
profile = (brightness, contrast, rgb);
rgb = color;
brightness = number;
contrast = number;
```

Listing 6: The Display Class

The provided EBNF code in Listing 6 further elaborates the grammar for the *display* component within the context of visualization resources.

- Display: Defined by one or more instances of a tuple containing *name*, *type*, *resolution*, and optionally *display_size* and *profile*.
  - Display Size: Defined by a *value*, *units*, and an abbreviation (*abrv*).
  - Profile: Includes *brightness*, *contrast*, and *rgb*, where *rgb* is a color.
  - Color: Defined as a tuple of three floating-point numbers (as previously specified).
  - Brightness and Contrast: Both are numbers.

The *display* grammar encapsulates detailed specifications for display attributes, ensuring that each display's characteristics such as size and profile are clearly defined and consistently represented.

```
renderer = {(name, platform, engine, settings, type, output)};

platform = string;
engine = string;
output = (type, format, compressed);
compressed = true | false;
settings = (samples, denoising);
samples = number;
denoising = floating_point;
```

Listing 7: The Renderer Class

The EBNF code in Listing 7 further refines the grammar by defining the structure for the *renderer* component within the context of visualization resources.

- Renderer: Defined by one or more instances of a tuple containing *name*, *platform*, *engine*, *settings*, *type*, and *output*.
  - Platform: Defined as a string.
  - Engine: Defined as a string.
  - Output: Defined by a tuple containing *type*, *format*, and *compressed*, where *compressed* can be either *true* or *false*.
  - Settings: Defined by a tuple containing *samples* and *denoising*, where *samples* is a number and *denoising* is a floating-point number.

This grammar specifies the detailed structure of the renderer component, ensuring the attributes related to platform, engine, settings, and output are clearly defined and consistently represented.

```
filter = { (data, scripts) };

scripts = { jsonpath };
jsonpath = rule;
```

Listing 8: The Filter Class

The EBNF code in Listing 8 further defines the structure for the *filter* component within the context of visualization resources.

- Filter: Defined by one or more instances of a tuple containing *data* and *scripts*.
  - Scripts: Defined by one or more instances of *jsonpath*.
  - Jsonpath: Defined by a *rule*.

This grammar specifies the structure for filters, ensuring that each filter includes data and a set of scripts, where each script is defined by a JSONPath rule.

```
transforms = {( name, type, functions)};
functions = rule;
```

Listing 9: The Transforms Class

The EBNF code in Listing 9 further extends the grammar by defining the structure for the *transforms* component within the context of visualization resources.

- Transforms: Defined by one or more instances of a tuple containing *name*, *type*, and *functions*.
  - Functions: Defined by a *rule*.

This grammar specifies the structure for transforms, ensuring that each transform includes a name, type, and a set of functions defined by a rule.

```
geometric_transforms = {(name, type, position, rotation, scaled, coordinate_system)};

position = coords | data;
rotation = coords | data;
scaled = coords | data;
coordinate_system = string;
```

Listing 10: The Geometric Transforms Class

The EBNF code in Listing 10 defines the structure for the *geometric_transforms* component within the context of visualization resources.

- Geometric Transforms: Defined by one or more instances of a tuple containing *name*, *type*, *position*, *rotation*, *scaled*, and *coordinate_system*.
  - Position: Can be either *coords* or *data*.
  - Rotation: Can be either *coords* or *data*.
  - Scaled: Can be either *coords* or *data*.
  - Coordinate System: Defined as a string.

This grammar specifies the structure for geometric transformations, ensuring that each transformation includes attributes such as name, type, position, rotation, scaling, and coordinate system, with positions, rotations, and scales being either coordinates or data.

```
observer = {(name, [contrast_sensitivity], [color_blindness])};

color_blindness = (ishihara);
ishihara = (pass, max);
pass = number;
max = number;
```

Listing 11: The Observer Class

The EBNF code in Listing 11 defines the structure for the *observer* component within the context of visualization resources.

- Observer: Defined by one or more instances of a tuple containing *name*, and optionally *contrast_sensitivity* and *color_blindness*.
  - Color Blindness: Defined by an *ishihara* test result.
  - Ishihara: Defined by a *pass* and a *max* value, both of which are numbers.
  - Pass: A number representing the pass value for the Ishihara test.
  - Max: A number representing the maximum score for the Ishihara test.

This grammar specifies the structure for observers, ensuring that each observer includes attributes such as name, and optionally, contrast sensitivity and color blindness information based on the Ishihara test.

```
device = {(name, type, [event_types], [device_sources])};

event_types = {string};
device_sources = {string};
```

Listing 12: The Device Class

The EBNF code in Listing 12 defines the structure for the *device* component within the context of visualization resources.

- Device: Defined by one or more instances of a tuple containing *name*, *type*, and optionally *event_types* and *device_sources*.
  - Event Types: Defined by one or more instances of **string**.
  - Device Sources: Defined by one or more instances of **string**.

This grammar specifies the structure for devices, ensuring that each device includes attributes such as name, type, and optionally lists of event types and device sources, all represented as strings.

```
view = {(name, layers, view_settings)};

layers = {(name, geometry, [settings])};
geometry = {(name, type, source, geometric_transforms, materials, [settings])};
view_settings = (renderer, display, optimization, camera, post_processing);
camera = (cameras, geometric_transforms);
optimization = string;
post_processing = (style_sheet, compositing);
style_sheet = string;
compositing = variables;
```

Listing 13: The View Class

The EBNF code in Listing 13 defines the structure for the *view* component within the context of visualization resources.

- View: Defined by one or more instances of a tuple containing *name*, *layers*, and *view_settings*.
  - Layers: Defined by one or more instances of a tuple containing *name*, *geometry*, and optionally *settings*.
  - Geometry: Defined by one or more instances of a tuple containing *name*, *type*, *source*, *geometric_transforms*, *materials*, and optionally *settings*.
  - View Settings: Defined by a tuple containing *renderer*, *display*, *optimization*, *camera*, and *post_processing*.
  - Camera: Defined by a tuple containing *cameras* and *geometric_transforms*.
  - Optimization: Defined as a string.
  - Post Processing: Defined by a tuple containing *style_sheet* and *compositing*.
  - Style Sheet: Defined as a string.
  - Compositing: Defined as *variables*.

This grammar specifies the structure for views, ensuring that each view includes attributes such as name, layers, and view settings. Each layer and geometry is further defined by specific attributes, while view settings include details related to the renderer, display, optimization, camera, and post-processing.

4.2 A 3D Visualization Grammar

This section provides information on how the visualization is composited from the specification. It details the view being composited and the how the scenegraph is created. The 3D scenegraph creation process involves acquiring data sources, filtering, transforming and then mapping them on to visual attributes onto glyphs or objects in the visualization.

4.2.1 The View Composition

In Aloka every visualization is a view. The view can be described as a scene with various objects with various properties. The view can also have properties such as the display width/height or the angle with which it is viewed or the display type such as a stereoscopic display/VR(Virtual Reality)/AR(Augmented Reality). The View defines the scene. It can be divided into layers each of which can be inter-dependant as shown in figure 4.1. This helps in organizing portions of the view with the data binding for visualization and also separating scene interactions. It can also be used for separate scene rendering pipelines for real time performance. Most importantly it provides structure to the grammar.

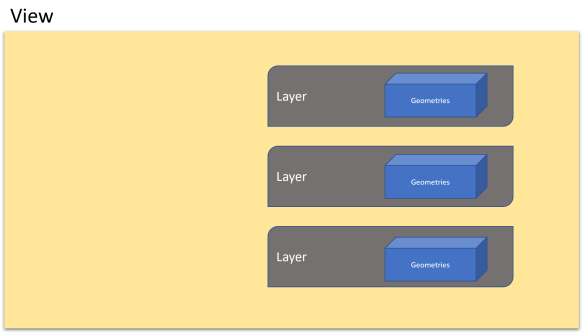


Figure 4.1: The View Structure

Each Aloka specification can hold several views as shown in figure 4.2. This is for scenarios where the user might interact with the current view which leads to another view. The start variable is how Aloka knows where to start looking for the initialization of the first view.

```
"view": [
    {
        "name": "Temperature Visualization",
        "layers": [...
        ],
        "settings": {...
        }
    }
],
"start": "Temperature Visualization"
```

Figure 4.2: Example of the view in Aloka

4.2.1.1 View

Each View is divided into layers. In figure 4.3 below each layer holds a list of geometries and each geometry object is defined by a data source or geometry object. In the example below in figure 4.3 the first layer's first geometry refers to a data source of a 3D model, which is the 3D scene of Newcastle. The second refers to a mapped source of temperature data. It explicitly states that a scale should be used to map to the shader of the material and data should be dynamically binded to its color.

```
"layers": [
    {
        "name": "viz_model_layer",
        "geometry": [
            {
                "name": "newcastle model layer",
                "type": "MODEL",
                "source": "3D Virtual Newcastle Gateshead Model",
                "geometric_transforms": "primary_base_model",
                "material": "",
                "settings": {}
            }
        ],
        "settings": {},
        "lights": [
            {
                "source": "sun_light",
                "geometric_transforms": "sun_light_projection"
            },
            {
                "source": "ambient_light",
                "geometric_transforms": "ambient_light_projection"
            }
        ]
    },
    {
        "name": "viz_data_layer",
        "geometry": [
            {
                "name": "data points",
                "type": "DATA",
                "source": "GLYPH_POINT_CONTRAST_01",
                "geometric_transforms": "data_projection",
                "material": {
                    "source": "emissive shade",
                    "mapping": {
                        "shader_a": {
                            "color": {
                                "scale": "temperature",
                                "map": "data"
                            }
                        }
                    }
                },
```

Figure 4.3: The view's layer structure

4.2.2 Data Processing

Data needs processing before being used in a visualization pipeline. This is to help in mapping data to the right values. Aloka uses JsonPath, a Json query language to filter and query data. Similar to SQL, JsonPath allows users to filter data using the query language. Using statistical functions from numpy and pandas allows users to visualize the output of the data analysis. This requirement scales the possibilities of visualization for the grammar.

4.2.2.1 Filtering

The section defines queries in JSONPath which are executed on the data sources. This facilitates the creating of new data sources from the results of the query execution, serving as an effective filter mechanism. An example is shown in figure 4.4

```
"filter": [
    {
        "data": "Newcastle Temperature Data",
        "scripts": {
            "name": "$[0:].name",
            "data": "$[0:].data.Temperature.data.*",
            "coordinates": "$[0:].geom.coordinates",
            "latitude": "$[0:].geom.coordinates[1]",
            "longitude": "$[0:].geom.coordinates[0]",
            "elevation": "$[0:].base_height"
        }
    }
```

Figure 4.4: The filter grammar example

4.2.2.2 Transforms

Transforms allow one to perform statistical functions on the data. Aloka allows users to run array operations on the data using numpy. An example of this is shown below in figure 4.5

```
"transforms": [
    {
        "name": "temperature average",
        "type": "STATIC",
        "functions": {
            "average_data": "average: data",
            "product": "multiply: data, data",
            "easting": "WGS84toOSGB36: latitude, longitude",
            "northing": "WGS84toOSGB36: latitude, longitude",
            "elevation_offset": "set_value: elevation, 2"
        }
    },
    {
        "name": "E_N adjustments",
        "type": "STATIC",
        "functions": {
            "northing_adjusted": "subtract: northing, 564179",
            "easting_adjusted": "subtract: easting, 424132"
        }
    },
```

Figure 4.5: The transform example

4.2.3 Data Binding Design

Binding data to visual properties such as color in materials and the properties of the material themselves allows for a wide range of possibilities in visualizing data. Data is also allowed to be bound to positions, rotations and scale of objects. This creates a very large collection of ways to visualize data based of its values and context. Aloka uses a resources class to organize its resources as shown in figure 4.6. It also uses data from the data class as shown in figure 4.7 to be used as a source in the specification.

```
"resource": {
    "data": [ ···
    ],
    "cameras": [ ···
    ],
    "fonts": [ ···
    ],
    "shaders": [ ···
    ],
    "materials": [ ···
    ],
    "lights": [ ···
    ],
    "scales": [ ···
    ]
},
```

Figure 4.6: The resource class

```
"resource": {
    "data": [
        {
            "name": "Newcastle Temperature Data",
            "source": "http://[redacted].json",
            "type": "LIVE",
            "params": { },
            "format": "JSON"
        },
        {
            "name": "3D Virtual Newcastle Gateshead Model",
            "source": "/home/ubuntu/aloka-api/city_3d_vng_newcastle.blend",
            "type": "LOCAL_STORAGE",
            "format": "BLEND"
        },
```

Figure 4.7: The data example

The scales are also defined in the grammar as shown in the figure 4.8 below.

```
"scales": [
    {
        "name": "temperature",
        "type": "INTERVAL",
        "data": "temperature_colormap",
        "range": {
            "minimum": -10,
            "maximum": 38,
            "iteration": 2
        }
    }
]
```

Figure 4.8: The scales example

4.2.4 Abstract Scene-Graph

The scene is represented in an abstract definition of the scene-graph. This is generated as a JSON file. It contains the positions, rotations and scale of geometric objects in a scene, this is defined in the specification from which it is generated (eg. in figure 4.9).

```
"geometric_transforms": [
    {...
    },
    {...
    },
    {
        "name": "data_projection",
        "type": "DYNAMIC",
        "position": {
            "x": "easting_offset",
            "y": "northing_offset",
            "z": "elevation_offset"
        },
        "rotation": {
            "x": 0,
            "y": 0,
            "z": 0
        },
        "scaled": {
            "x": 1,
            "y": 1,
            "z": 1
        },
        "coordinate_system": "BLENDER"
    },
```


Figure 4.9: The geometric transforms example

It also defines materials of geometric objects in the scene. Definitions of lights and its values are also defined. The abstract scene-graph is very simplistic and contains the information needed to recreate a scene programmatically. At this point the visualization scene is in JSON form. The next step would be to read the list of objects and create a scene from the desired render engine or application. This design helps in cross application and cross platform compatibility .

4.3 Chapter Summary

The 3D visualization grammar defined by the provided EBNF code in this chapter includes specific elements and structures that cater to the complexities and additional dimensions involved in 3D visualizations. Here are the key differences between this 3D grammar and the 2D visualization grammars like Vega-Lite and D3:

- Dimensionality: 3D visualization grammars require handling an additional spatial dimension, leading to complexity in transformations, camera perspectives, and ren-

dering techniques, which are not present in 2D grammars.

- Components: 3D grammars include additional components like cameras, materials, shaders, and lights to support realistic rendering of 3D objects.
- Transformations: 3D transformations involve more complex mathematical operations to handle rotation, scaling, and positioning in three dimensions.
- Rendering: 3D rendering involves considerations for lighting and shading that are not necessary for 2D visualizations.
- Viewer Interactions: Observers and their characteristics, such as handling color blindness, can be more intricate due to the depth and perspective involved in 3D visualizations.

While both 3D and 2D grammars aim to represent data visually, the additional dimension and complexity of 3D visualizations necessitate a more detailed and nuanced grammar.
To summarize Aloka's design is explicitly meant to be used in a declarative way for definitions of 3D visualizations. The design layout of the schema aims to define a list of views through which users can navigate from one to another, as evidenced by its structure. This is essentially like a movie running frame by frame where the user's actions lead to a certain frame being selected. Since computational load is transferred onto the cloud it provides high quality visualizations on demand and is able to be flexible due to the architecture design. The cross platform nature of the application also makes it scalable to different cloud platforms.
The design and development of Aloka was explored, its system architecture explained and the design philosophy is detailed. This is a novel approach in creating complex 3D visualizations for a wide range of applications and users for different systems.

Chapter 5

Case Study of Visualizations using Aloka

5.1 Temperature Visualization using Aloka

To understand Aloka's structural design. It follows a very different approach and does not have a schema file like Vega. The temperature visualization of Newcastle Upon Tyne, UK will be used as an example in this section to explain and justify the design decisions behind Aloka.

5.1.1 Data Sources

The Urban Observatory provides a range of data services, they have numerous sensors across the city of Newcastle that cature a wide variety of data that includes Weather, Temperature, Air Quality etc. The data we are interested in is live temperature data within the bounding box of our 3D VNG model (Virtual Newcastle Gateshead is a 3D digital city model of Newcastle and Gateshead, developed in partnership with Northumbria University, Newcastle City Council and Gateshead Council). This comes down to the about 21 sensors for this visualization [Data Source API link].

5.1.1.1 The Urban Observatory - Temperature Data

The Urban Observatory of Newcastle holds the largest set of publicly available real time urban data in the UK. Accessing the data via the Urban Observatory API to get live temperature data is as simple as requesting a get call to its API, with required parameters (eg: bounds and data types).The temperature visualization of Newcastle Upon Tyne uses live data from the sensors across the city. This data is provided via an API from [The Urban Observatory]

5.1.1.2 The 3D City Model

The Newcastle Upon Tyne 3D city model is provided by [Virtual Newcastle Gateshead Project] as shown in figure 5.1. This is a joint venture between Northumbria University, Newcastle City Council and Gateshead Council to create a 3D digital model of the urban core areas of both Newcastle and Gateshead, UK. They have used aircraft with Light Detection and Ranging (LiDAR) sensors to map out the geographical terrain and accurately map a model of Newcastle and Gateshead. The LiDAR mapped model is then sent to 3D modelers to refine and polish the 3D model before distribution.

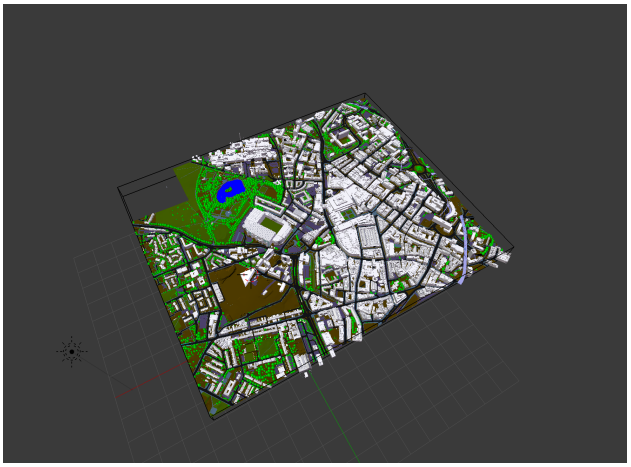


Figure 5.1: The 3D model of Newcastle Upon Tyne city

5.1.2 Temperature Glyph

The idea is to visually represent temperature data in glyphs as shown in figure 5.2:

Figure 5.2: A simple glyph design to visualize data by color in a complex scene

The visualization generated by Aloka for temperature data is shown below in figure 5.3

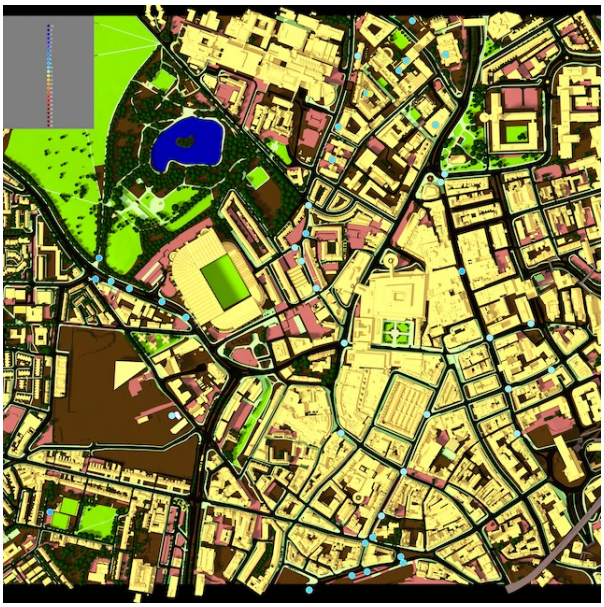


Figure 5.3: 3D rendered temperature visualization generated by Aloka API

5.2 Traffic Saturation

5.2.1 The Research Problem

How can the representation of traffic saturation using custom-designed glyphs in a 3D visualization context improve the response times and situational awareness of traffic operators compared to traditional tabular formats?

- Current System Analysis: Analyze the existing tabular representation of traffic saturation data in UTC display systems and identify their limitations and challenges.
- Design and Development: Design custom glyphs tailored to represent traffic saturation data and then develop a 3D visualization using Aloka integrating these custom glyphs.

- Usability Testing: Conduct usability tests with traffic operators using both the traditional tabular system and the new 3D visualization prototype to collect quantitative and qualitative data on response times, accuracy, and user satisfaction.

- Data Analysis: Analyze the collected data to compare the effectiveness of the 3D visualization with custom glyphs against the tabular format, using statistical methods to determine the significance of any observed differences.

5.2.2 Existing UTC System

Operators oversee a node (a specific traffic junction) and assess whether its saturation percentage remains unchanged for a set period, typically 15 minutes or more, which may vary based on factors such as rush hour (when saturation is expected) or other unusual conditions. If saturation does not decrease during this time, they will extend the cycle time for the region's traffic lights to alleviate congestion in the nodes within that area. Note: In a region every node must have the same cycle time. The only time manipulation that can be done per node is changing it from single cycle to double cycle. If a single cycle time is 96 seconds, double would be single/2 = 48 seconds.
How does this work? In every cycle some time is registered as wasted capacity [Wood et al., 1994] eg: cycle time of 96 may have 20 seconds wasted capacity due to late changing of reds, other signalling issues, slow moving queue etc. When operator increases the cycle time for the whole region which includes all nodes (this is for synchronization purposes, so the network remains synchronized with its signalling) the wasted capacity goes down as SCOOT allocates more time for each stage in the cycle; essentially cycle time of 105 seconds may have a wasted capacity of 17 seconds and cycle time of 120 may have 15 seconds. Note these are purely examples and do not correlate to real world values. Queue clear time is the time it takes to clear the queue completely. As the queue clears the saturation percentage goes down. The best way to affect queue clear time is by increasing cycle time for the region. SCOOT system will try its best to keep saturation percentage. A node's saturation is calculated as a sum of saturation of links it contains. The data to be recorded from SCOOT can be every interval of 40 seconds or depending on the

system, variable. The Existing Autonomous Tram in Depot (ASTRID) stores data as aggregated for every 15 minutes or 5 minutes essentially an average of all those messages. For diagnostics operators use the Region Fine Tuning Display (RFTD) tool in the UTC system.

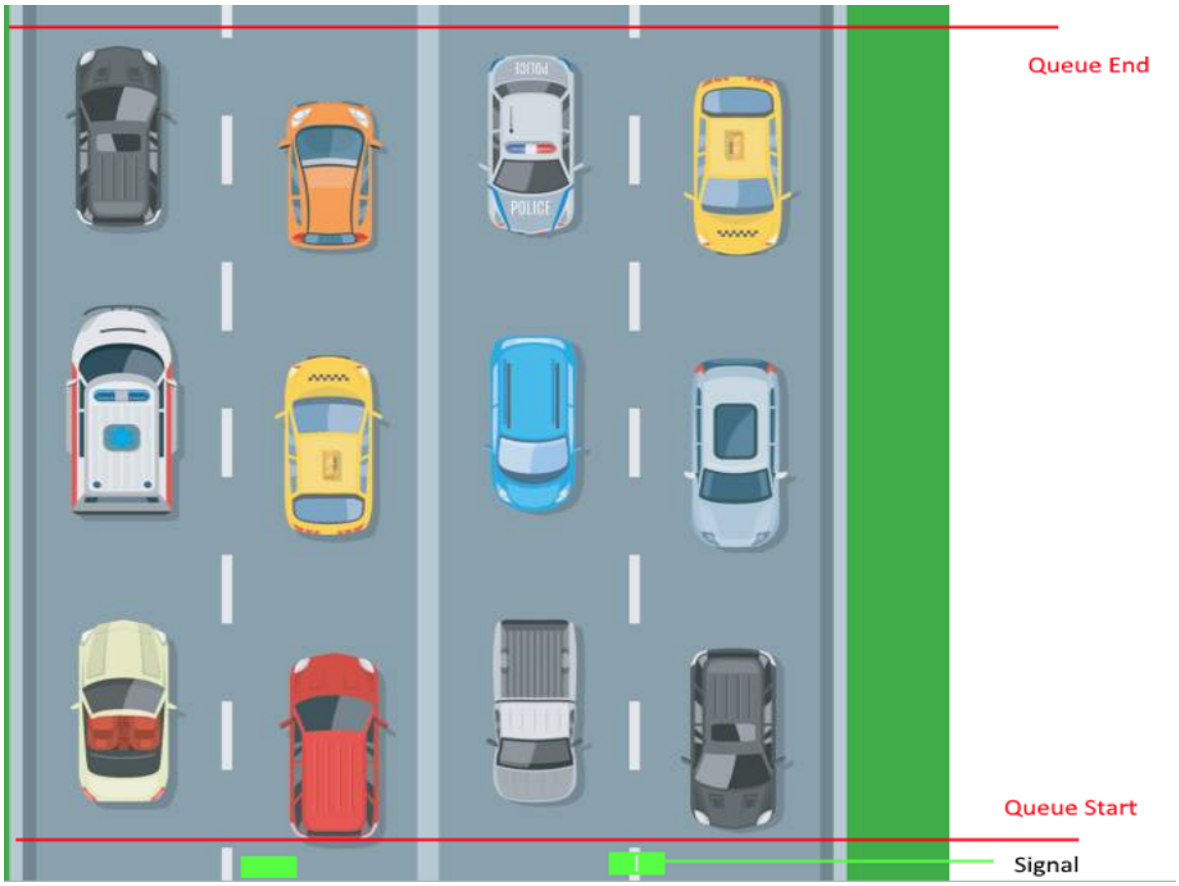


Figure 5.4: Queue length in a road link explained

5.2.2.1 NFTD - Node Fine Tuning

This is a display mode in the UTC system that displays traffic data collected by sensors on the traffic node and link. This is a tabular looking visualization that shows traffic data as shown in Figure 5.5. This is mostly used by experienced operators. They use this with a combination of DIPM to observe live stage timings and may also use multiple instances of the same window type but targeting different nodes in a region. If they feel the issue of saturation percentage persists longer than usual, then they make the decision to increase the region cycle time.

```
N10141  REGION 10      IMPL=OFF REPL=OFF TREND=OFF FDOWN=NO  14-Aug-18  11:14:24
RCYT=096   NCYT=096   MPCY=120   FORCE=RELEASE   MINCYT=044 PERIOD=150 TIMER=052
BUS ACTIVE=0           ISAT=090    TSAT=080   OPNI=NO     INDEP=NO
Maximum Lengths: S1: 116  S2: 120  S3: 120
Current Lengths: S1: 017  S2: 015  S3: 064
Minimum Lengths: S1: 017  S2: 013  S3: 013
LINK TYP  GRN  QUE  Qback  QatGN  QclrT %SAT %CONG  OFFSET JNYT  QCMQ  STOC CGIP
 Z   NO    13    7    7     10     -1    138    0      15     8    18    10    3
 P   NO    11    3    3      8     -1    136    0     -16     5    15    10    3
 B   NO    13   10   10     13     -1    169    0       0     8    22    11    3
 Q   NO    11    3    3      7     -1    136    0      17     6    15     9    3
 A+  EN    60   -1    0      3      6     24    0             9    22    11    4
```

Figure 5.5: The UTC Node Fine Tuning (NFTD) System

5.2.2.2 PICT to Generate semi-graphic pictures

PICT is a display system feature available in the UTC software that shows a pictograph visualization of a traffic node or multiple nodes. PICT is mostly used by more inexperienced operators The inexperienced operators also observe the same as the experienced operators, but they prefer to use the Graphical interface as the Command Line Interface (CLI) interface is just not easy towards beginners. The queue lengths are animated live, and saturation is deduced from the amount the red has covered the green eg: once red has fully covered green it means the queue lengths has reached max and saturation is at 100%

Figure 5.6: The bar colors shows the queue length

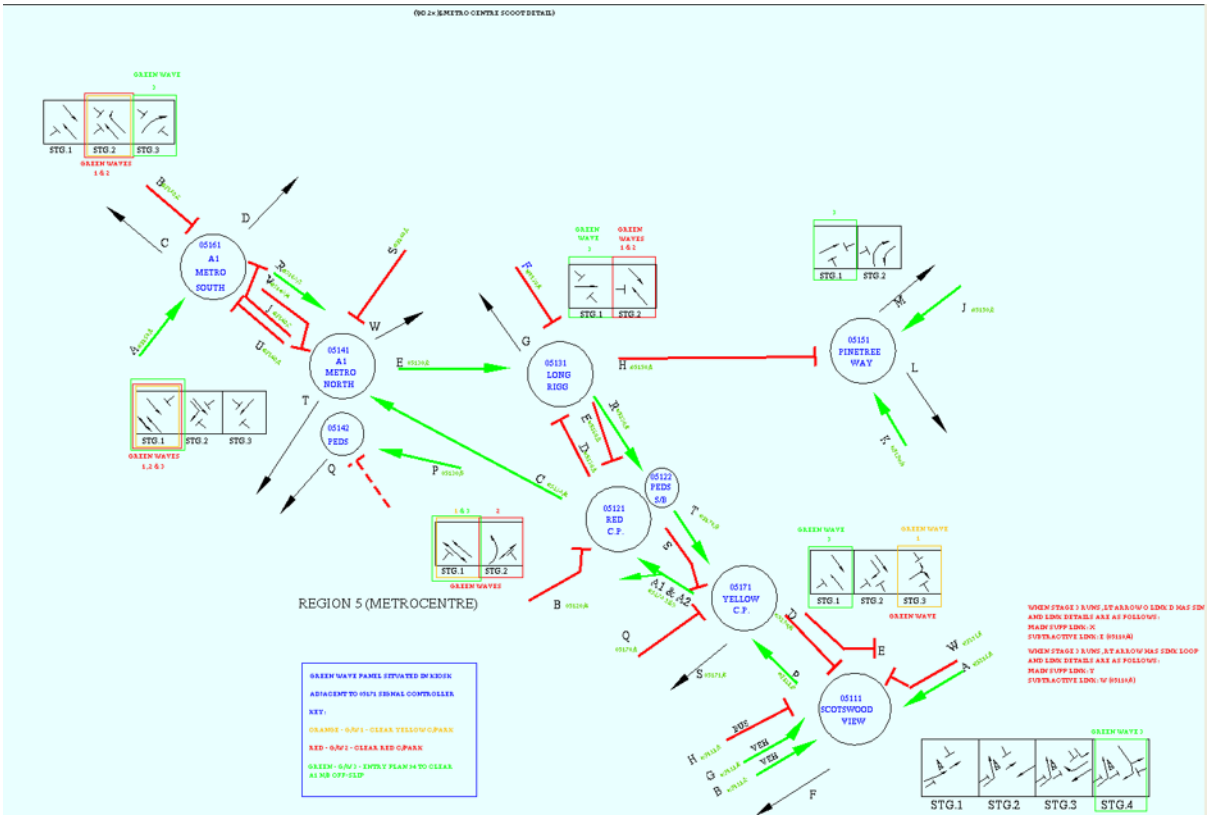


Figure 5.7: The UTC Application Function to Display Selected Picture (PICT) Display system used in UTC

5.3 Design of Traffic Saturation Visualization

There were a few iterations when designing traffic saturation visualizations. The designs were initially creative way to find a solution. However, during this process we realized we have to use as much knowledge in the industry as in academia to be able to come up with a sound design that will be effective. Each design went through some **saliency** tests using opencv library in Python.

5.3.1 Saliency tests

The three primary forms of saliency detection algorithms used in opencv are

Static saliency: These saliency detection techniques use statistics and picture attributes to pinpoint the most salient areas of an image. This is the prominent element of an image that draws our attention to it. The salient area is the part of the image that

naturally draws our attention.

Spectral Residual: This algorithm examines an input image's log-spectrum, extracts the spectral residual from the image in the spectral domain, and recommends a quick way to build a saliency map that predicts the locations of proto-objects. Redundancies result from similarities. A system must be aware of the statistical similarity of the input stimuli if it is to minimise redundant visual information. Therefore, the information that pops out of the smooth curves in various log spectra where significant shape similarities may be seen merits our attention. We think that aberrant areas in the image, where proto-objects are sprung up, may be caused by statistical singularities in the spectrum.[Hou and Zhang, 2007]

Fine Grained: Ganglion cells, of which there are two varieties—on-center and off-center—make up the retina of human eyes. The on-center ganglion cells react to patches of brightness surrounded by darkness. The ganglion cells that are off-center react to areas that are dark with a light background. Based on the variances between the on-center and off-center points, this method determines the saliency.[Wang and Dudek, 2014]

Threshold - Fine Grained: A binary threshold of the saliency map of fine grained saliency which when computed is useful as a part of a contour detection pipeline and clearer understanding of saliency in a visualization. [Hao and Wang, 2013]

Motion saliency: This type of algorithm often uses video or frame-by-frame inputs. The frames are processed by the motion saliency algorithms, which maintain track of moving objects.

Objectness: In order to determine "objectness," saliency detection algorithms provide a set of "proposals," or more simply bounding boxes, of potential locations for objects in an image.

5.3.2 Glyph Design

The Design requires simplicity and given the data sources the glyph has to be defined within those parameters. The data sources used are:

UTC: The UTC export dump from the live system for 7 days recorded by the UTMC in Newcastle Upon Tyne St James Boulevard.

GEOJSON: The GeoJSON of the road network of St James Boulevard, helps create a detailed map of the road links(also provided by UTMC).

VNG: The Virtual Newcastle Gateshead 3D Model of the City.

To design a glyph that represents the road link the requirements were

1. It shows direction of the road link (point to the node itself and also shows the direction of traffic)
2. It shows the saturation of the traffic in that particular road link
3. It should also be accurately shown in the 3D map to provide context of the data in relation to the geography.

The road link designed is shown in the figure 5.8

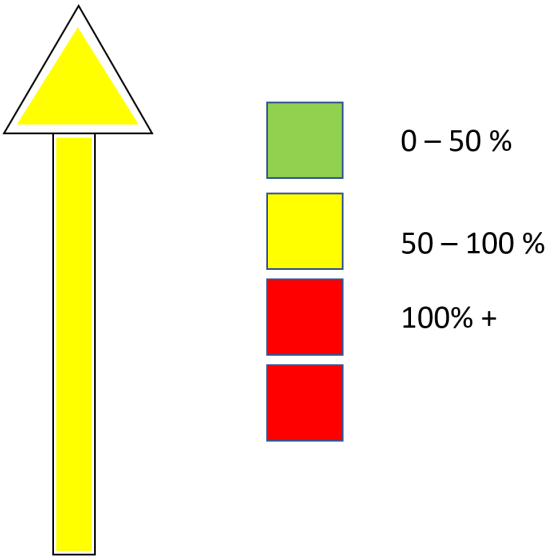


Figure 5.8: The Arrow glyph design

5.3.3 Design Proposals

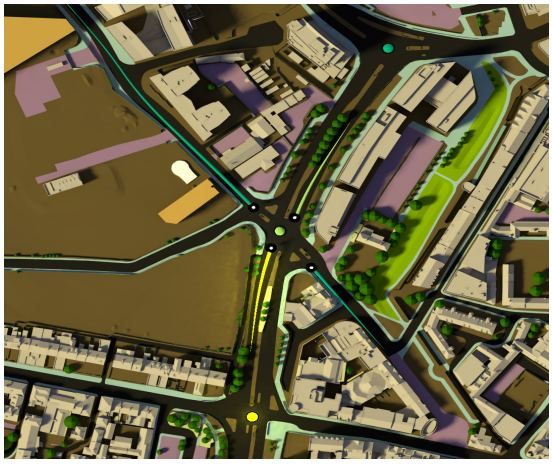
We are interested in visual attention hence using fine grain algorithm to understand which designs help better in visual detection. We are also going to perform a threshold to demonstrate a binary map that can be processed for contours to extract each salient region in the designs. By exploring a variety of designs we progressively explore and analyse to find out what works in each design and improve upon them.

Level of Detail (LOD) : The initial design proposal was to have a multi-level Level of Detail (LOD) visualizations so user can navigate each LOD as required. This design focuses on creating multiple Level of Detail (LOD) to give operators context of the region and overview of traffic data in the digital twin Virtual Newcastle Gateshead (VNG) model. However this made little sense when comparing to a singular UTC tabular visualization. So in terms of saliency and clarity the LOD 2 as shown in Figure 5.10b seemed the best out of the 3 LOD in terms of camera perspective.

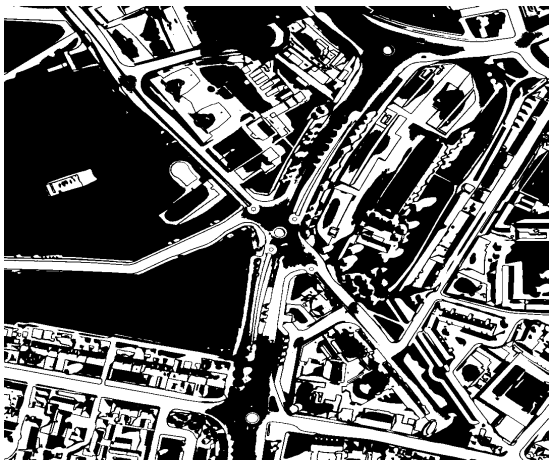
(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

Figure 5.9: Level of Detail (LOD) Design 1



(a) The Original Image



(b) Threshold performed on fine-grained saliency map of image a

Figure 5.10: Level of Detail (LOD) design 2

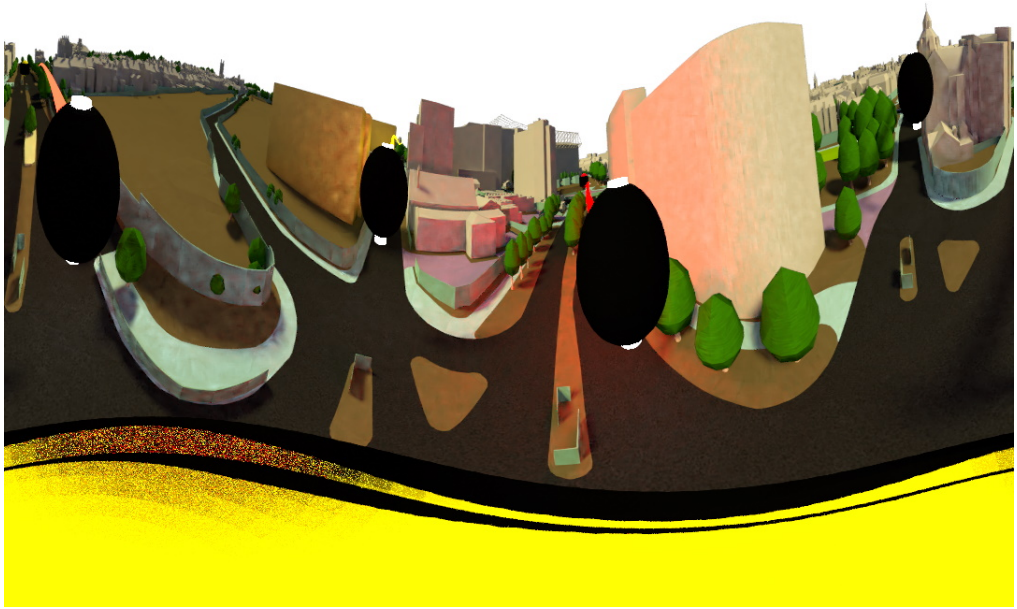


Figure 5.11: The Level of Detail (LOD) 3 which provides a 360 VR image

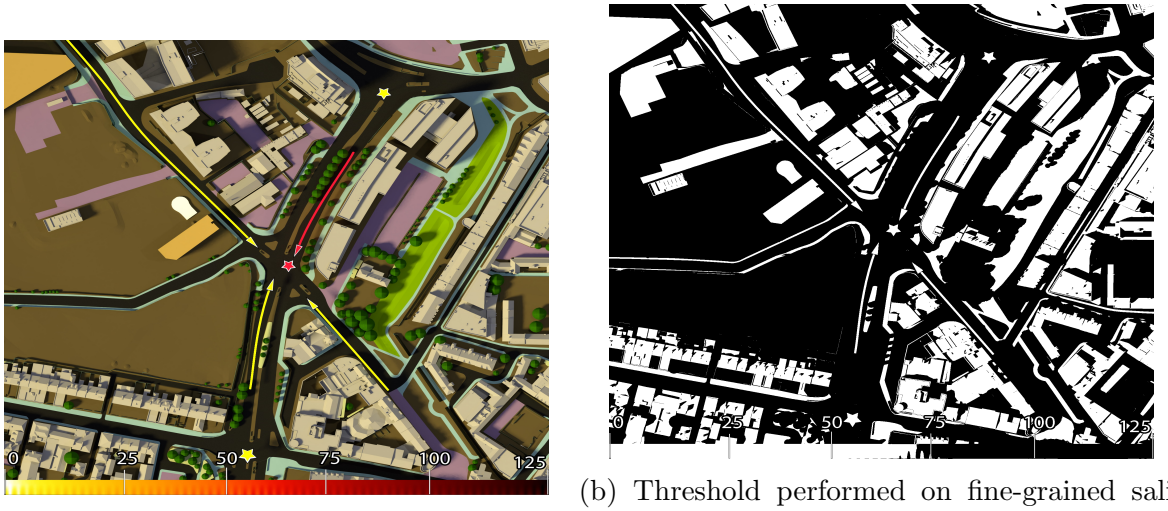
Node Star Design: These designs as shown in Figure 5.12, Figure 5.13 and Figure 5.14

were an experimental design to see if different glyph shapes prove more salient [Borgo et al., 2013]. The glyph design in Figure 5.13 was noticeably more salient than the other two.

(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

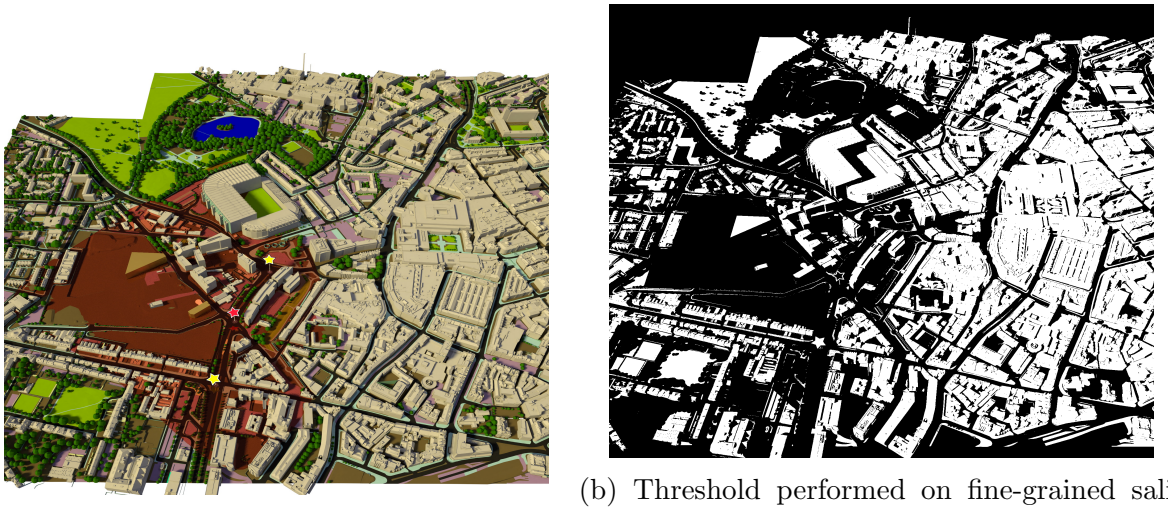
Figure 5.12: Node Star Design 1



(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

Figure 5.13: Node Star Design 2

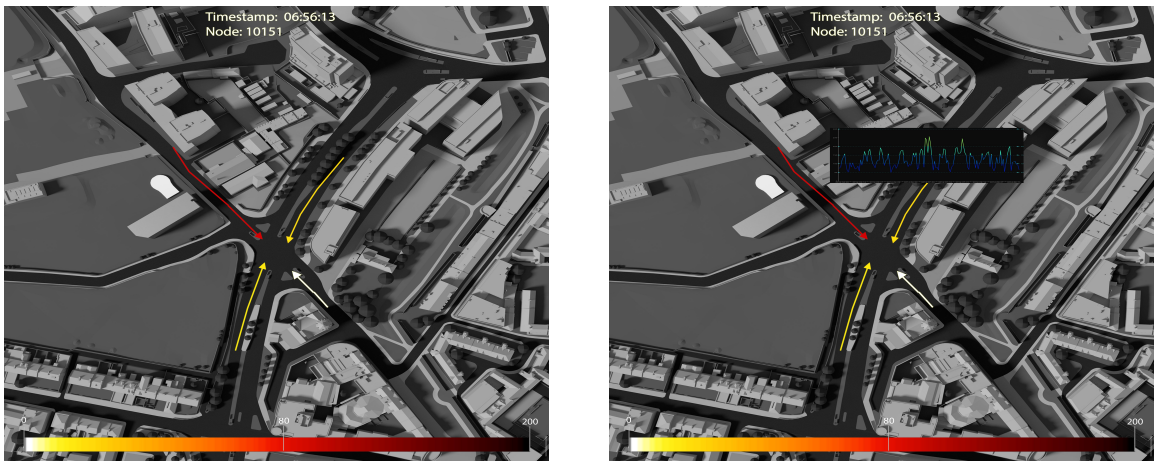


(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

Figure 5.14: Node Star Design 3

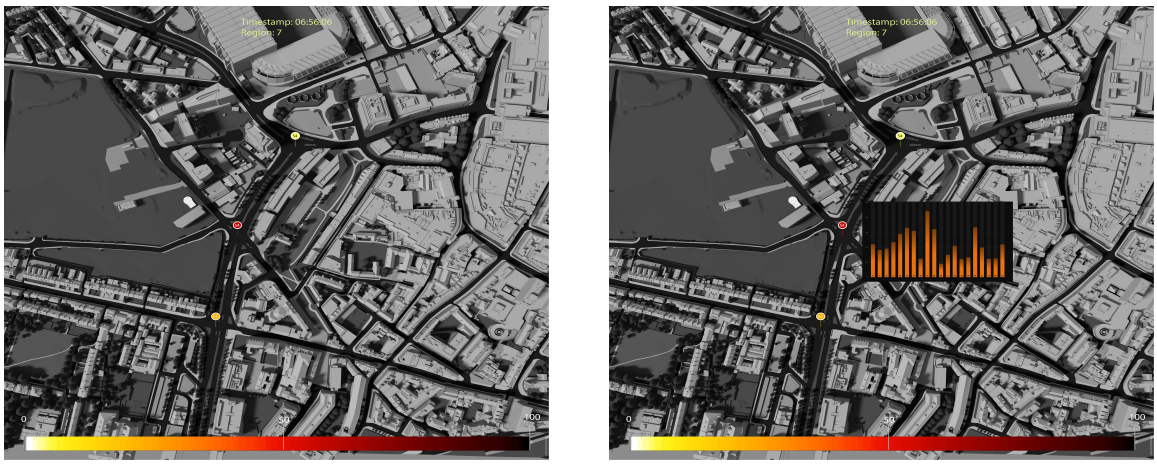
Node Link Design: These designs as shown in Figure 5.15, Figure 5.16 and Figure 5.17 were less about identifying the salient features and trying to experiment color isolation by using background greyscale. But all these designs in greyscale lose information about background details considerably. The background details and context are important for operators in identifying the node location or surroundings.



(a) Node Link Main design

(b) Node Link Summary design

Figure 5.15: A figure with Node link design with both a main and summary page



(a) Node View Main design (b) Node View Summary design

Figure 5.16: A figure with Node view design with both a main and summary page

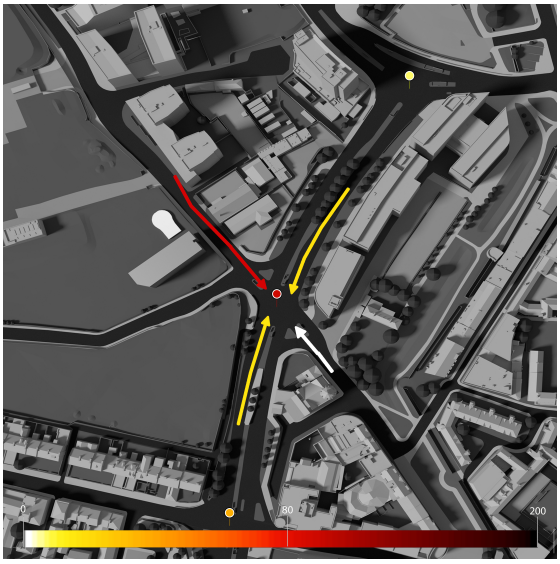


Figure 5.17: A simple Grayscale design

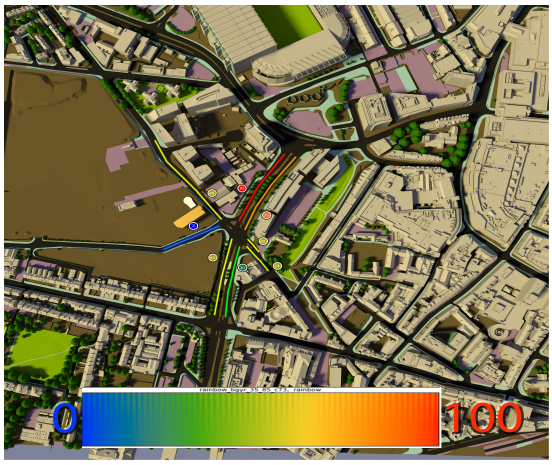
3D Arrow Design : These designs as shown in Figure 5.18 were experimenting to see how 3D arrows stack up against the other designs and they were pretty salient features in the visualizations as shown in their respective thresholds.

(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

Figure 5.18: 3D-arrow Design

Queue Length Design: This design as shown in Figure 5.19 was a more simplified version of Figure 5.18 but without arrow heads and having separate lines for each way road link. This design was dropped as it is not specific enough and can confuse traffic direction of road links.



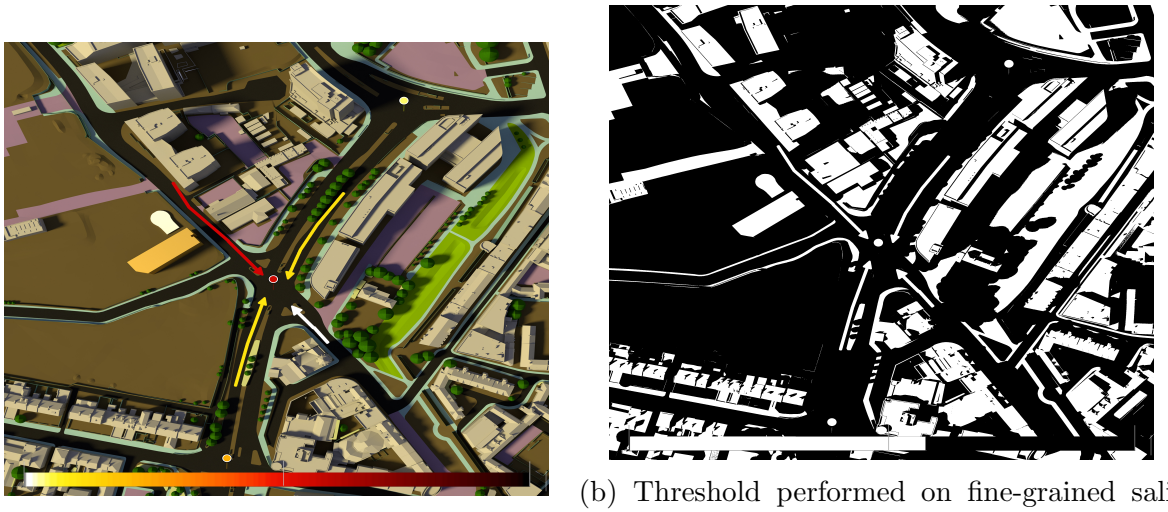
(a) The Original Image

(b) Threshold performed on fine-grained saliency map of image a

Figure 5.19: Queue Lengths Design

Simple Node Design: This design is a combination of both Queue lengths Figure 5.19 and 3D arrow 5.18. Its more specific easier to observe and also its much easier to construct the glyph in a schema, compared to a 3D arrow or queue lengths. Hence the simple node design as shown in Figure 5.20 was the best reference to use in

traffic saturation visualization.



(a) The Original image

(b) Threshold performed on fine-grained saliency map of image (a)

Figure 5.20: Simple Node Design

5.4 Defining Traffic Saturation Visualization in Aloka

The gylph design was defined as a standard type (for now) within Aloka, The gylph design required parameters such as the GeoJSON points of the link and mapping of the 3D model to the geographic coordinates. The combining of this will enable the road link to be displayed correctly inside the map

5.4.1 Data Sources

The Aloka specification of data sources are first defined to point to the UTMC export of the saturation data (processed). This is hosted on MindSphere service as a data serve app Platform as a Service (PaaS). The MindSphere application is a simple node RESTful server that allows access to the data.

5.4.2 Design translation into Aloka

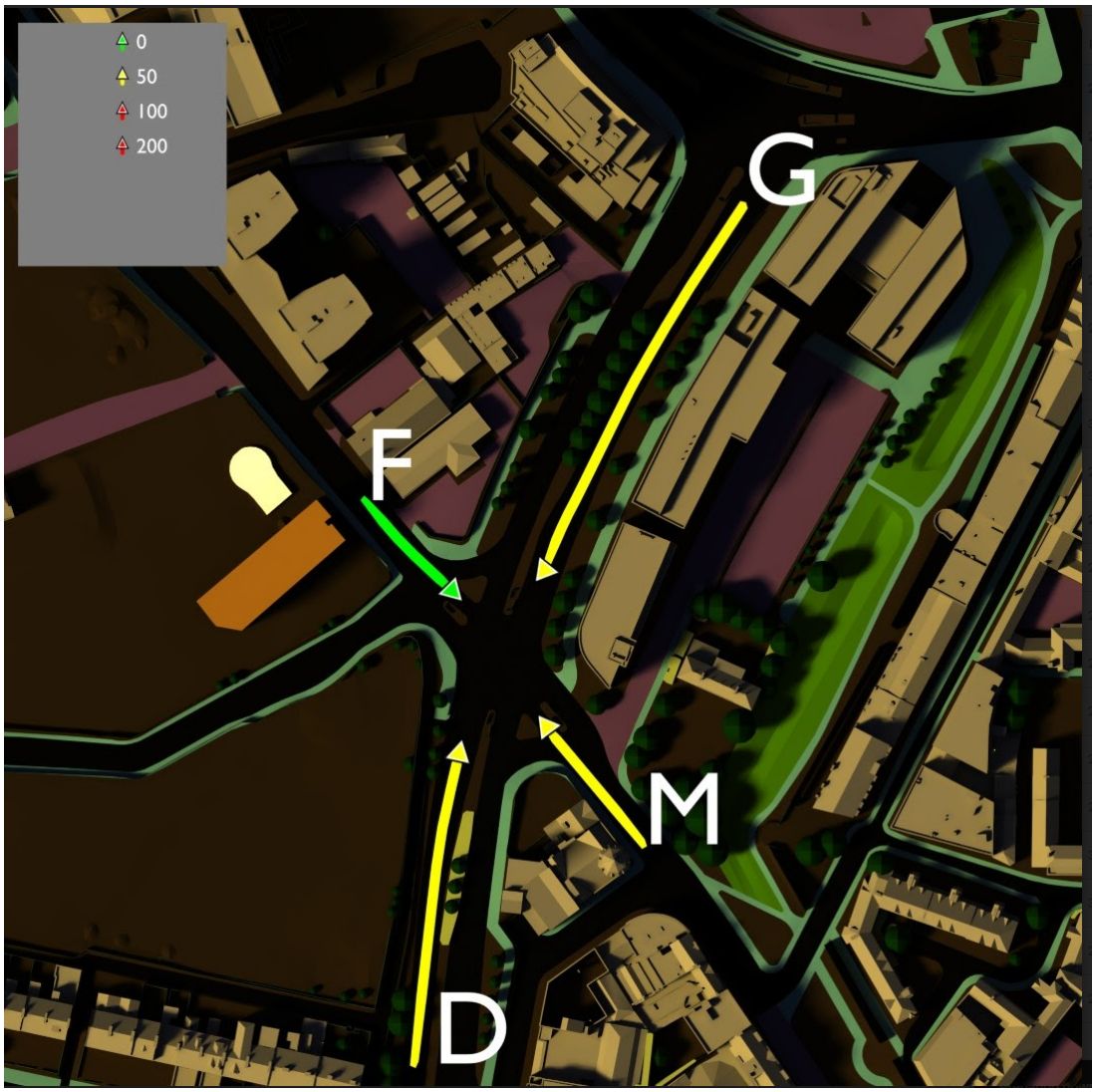


Figure 5.21: The Traffic Saturation visualization generated by Aloka API

Chapter 6

Experiments and Results

6.1 Experiment

6.1.1 Purpose

This research study aims to understand how traffic saturation can be represented visually. Traffic saturation is usually displayed in a Tabled format in a UTC (Urban Traffic Control) display system. The intent is to investigate if representing the data in a 3D geo-visualization helps with context and improves response times to detecting traffic saturation.

6.1.2 Experiment Design

6.1.3 Parameters

The user participation in the evaluation, which will last up to 10 minutes, should include completing the required forms and tasks. The forms include a standard consent form and user evaluation questionnaire.

The user will be requested to examine a collection of pictures and determine whether the road node is saturated by examining the roads segments (links) or by inferring the same from the table. There are two different types of images. The displayed visualisation is at the bottom left, and the UTC system display is on the bottom right as shown in Figure

6.1.

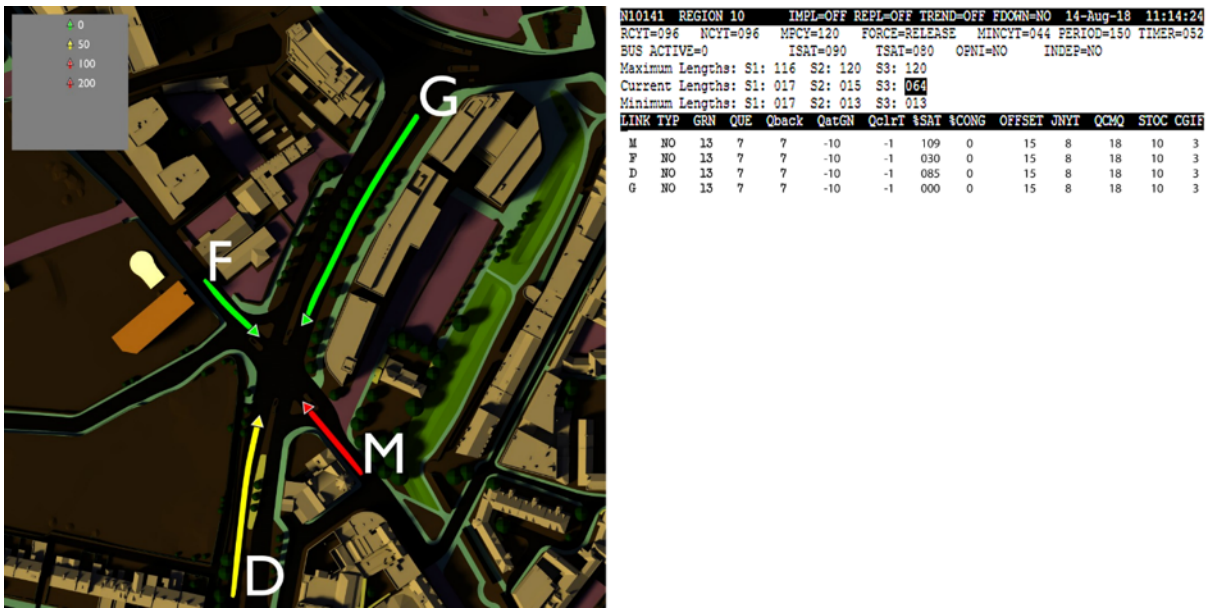


Figure 6.1: The visualisation is on left is a aloka rendered visualization, and the UTC system display is on right

The scales for the rendered visualization (as shown on the bottom) left to infer if it is saturated and %SAT column (as shown on bottom right) on the UTC system display.

Figure 6.2: The Visualization scale generated by Aloka - a closeup view.

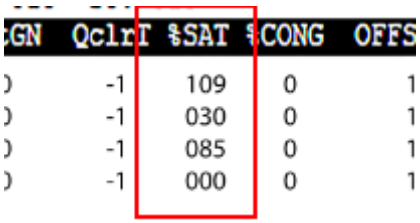


Figure 6.3: The UTC NFTD system - a closeup view to show saturation.

6.1.3.1 Privacy

This trial is a pilot study to begin to understand if there are benefits to this type of visual representation of traffic saturation. We are selecting a small local group of participants to understand if a larger study could be worthwhile in the future.

6.1.4 Benefits

There is a direct benefit to this research in guiding its future direction. There are also many external parties who will be interested in the results of the research if it proves successful, these include government, charities and industry.

6.1.5 Consent Form Presented to Subjects

"This is a simple low risk study as such it brings no risks beyond those normally found in the workplace. Siemens and EPSRC are the sponsors for this study based in the United Kingdom. Newcastle University will be using information from the users in order to undertake this study and will act as the data controller for this study. This means that Newcastle University is responsible for looking after the users information and using it properly. It is also the intention of the study to publish in summary form the results of the study and your anonymized data may form part of this publication that will be accessible globally. The lawful basis for carrying out this study under GDPR is Task in the Public Interest, (Article 6,1e) as research is cited as part of the University's duties. The lawful basis for processing any special categories of personal data is Scientific Research (Article 9,2j).

The user has rights to access, change or move information are limited, as Newcastle University need to manage the information in specific ways in order for the research to be reliable and accurate.

This study was approved for UK 22/05/2019 ref 13358/2018 "

6.2 Experiment Methodology

The Experiment was designed using PsychoPy (PsychoPy is an free cross-platform package allowing researchers to run a wide range of tests in the behavioral sciences (neuroscience, psychology, psychophysics, linguistics...)). The user is presented with the experiment which takes up to 10 minutes, including completing these forms and performing the tasks involved. The user will be asked to look at a series of images and decide if the road node is saturated by looking at the roads segments (links) or at the table to infer the same. There will be two types of images the one on left of Figure 6.1 is the rendered visualization and the one on right of Figure 6.1 is the UTC system display. The user is to use the scales provided in the rendered visualization to infer if it is saturated and similarly %SAT column on the UTC system display to infer for saturation.

6.2.1 Testing The Design Effectiveness

The design is tested by the user's response and accuracy to it. If there is a significant difference in both the response times and accuracy, it will be feasible to ascertain that the rendered images are an effective form of visualizing saturation for road links and also that prototyping effective and practical 3D visualizations are possible using Aloka

6.2.1.1 Experts

The experts are users who have used the UTC system or are using the UTC system. Participants were found within Siemens and the Urban Traffic Control Management for Newcastle Upon Tyne.

6.2.1.2 Non-Experts

Participants for Non-Expert category were users who never used the UTC system so have no point of reference but only information for which they should be looking. This makes this group a form of control group where no training of the existing system was given.

6.2.2 PsychoPy

Psychopy was used to setup the experiment pipeline where first a demo set of images will run which help participants get familiar with the experiment and what is expected of them before the real test begins. The user is shown a random image either UTC or Rendered and is not given a time limit, The image only changes once the user inputs his/her response after which a few seconds later the next image is presented. Response times are very accurate to below 0.1 ms on the software-side in psychopy. But on many keyboards there's a buffer which causes up to 30 ms in jitter [Bridges et al., 2020b]. This would apply to all stimulus software. This is taken into account when processing data. The monitor for the laptop used was 120Hz, which means there is a limit to the possible response time each user can respond within (at 60Hz that would be 16.66ms) in this case it would be 8.3 ms as it is a 120 Hz monitor. But as suggested in the paper below this is largely irrelevant as most users take 300ms to process visual input and then respond [Bridges et al., 2020a].

6.2.3 Statistical Analysis

In this experiment we have two sets of measurements from the participants, response time using the two visualisation tools. In testing the difference between the rendering time and UTC time, a two sided T-Test with alpha of 5% is used. Confusion matrix is also generated which shows how the true or false identification is done by the participants.

6.2.3.1 Confusion Matrices

The results are analysed using confusion matrices to get the best performance review and also if the rendered 3D visualization improved the user experience in understanding context. Confusion matrices are a popular measure used while solving classification problems. It can be applied to binary classification as well as multi-class classification problems. Confusion matrices represent counts from predicted and actual values.

6.2.3.2 Sensitivity and Bias/Specificity

Sensitivity, also known as the true positive rate, measures the proportion of actual positives that are correctly identified. Specificity, also known as the true negative rate, measures the proportion of actual negatives that are correctly identified. These metrics evaluate how well a test can identify true positives and true negatives, respectively.
Sensitivity and specificity mathematically describe the accuracy of a test in determining the presence or absence of a condition. In this analysis, sensitivity represents the percentage of subjects who correctly identify the condition using the developed tool. It is calculated using the formula: Sensitivity = True Positives / (True Positives + False Negatives).

Two key metrics were evaluated:

- Aprime (A'): A non-parametric estimate of discriminability, where a value near 1.0 indicates good discriminability and a value near 0.5 indicates chance performance.
- B"D (Beta"): A non-parametric estimate of bias, where B"D = 0.0 indicates no bias. Positive values represent a conservative bias (tendency to answer "no"), and negative values represent a liberal bias (tendency to answer "yes"). The maximum absolute value is 1.0

6.2.3.3 T-test

t-tests can help to determine whether or not the difference between an expected set of values and a given set of values is significant. It is usually first to define a null hypothesis in which it states that there is no effective difference between the observed sample mean and the hypothesized mean and that any measured difference is due only to chance. Overall, a *t-test* maybe either two-sided meaning simply that the means are not equivalent, or one-sided which specifies whether the observed mean is larger or smaller than the hypothesized mean. The test statistic t is then calculated. If the observed *t-statistic* is more than the critical value determined by the reference distribution, the null hypothesis is rejected. The reference distribution for the *t-statistic* is the t distribution.

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(s^2(\frac{1}{n_1} + \frac{1}{n_2}))}}$$

Figure 6.4: The formulae for calculating the *t-statistic* is given.

t is calculated as the ratio of difference in the means of two groups divided by the square root of the pooled standard deviation as provided in the equation 6.4. *n1* and *n2* are the number of observations in each groups and Ji's [Ji et al., 2013] are the observations as shown in Figure 6.4. The reference distribution for the *t-statistic* is the *t-distribution*. A large t value shows that difference between the means of two groups is greater the pooled standard deviation which indicates more difference between the mean values. The critical value depends on the significance level of the test which is the probability of erroneously rejecting the null hypothesis [Ji et al., 2013].

6.3 Experiment results

The experiment results are summarized into two sections for experts and non-experts. Below is data that has been summarized with mean response time for each and percentage of correct answers for both UTC and Rendered images for users.

6.3.1 Experts

The experts are participants who have utilized the UTC system or are currently using it as part of their job. These are also people who are familiar with this system. However they are not familiar with the new rendered images that express the same data as the UTC image. All the expert participants who are familiar with UTC could identify traffic saturation quickly with visual display than UTC. The bar chart shows the rendered time is lower compared to UTC for majority of the participants.

participant_id	rendered_mean_response_time	utc_mean_response_time	rendered_accuracy	utc_accuracy
participant_11	1.2644845615053800	1.4860058207821600	95.0	100.0
participant_18	2.4984293578229000	2.5108646262939100	95.0	100.0
participant_06	0.6614677523859430	1.0761581706101400	90.0	90.0
participant_09	0.8131277081964070	0.8692493403636040	90.0	95.0
participant_19	0.9731193355786670	1.1043519220198500	100.0	100.0
participant_07	0.6275551418104440	0.7962086606348750	100.0	100.0
participant_13	1.0592007042723700	1.488432226626900	100.0	100.0
participant_08	0.7600758785192740	1.0384134231237100	100.0	95.0
participant_15	1.0681074056949	1.189091028549590	100.0	100.0
participant_05	0.5941411447580320	0.6736185095825930	100.0	100.0
participant_12	1.4373808356904200	1.6154196954972600	100.0	100.0
participant_14	1.098519847888380	1.4319229517394000	100.0	95.0
participant_04	1.631781075980690	1.883119239071680	100.0	100.0
participant_17	2.1838550935106500	2.261899443408040	100.0	100.0
participant_10	1.1771424466613100	1.2735626465975700	100.0	100.0
participant_16	1.6851023753995800	2.4642604379405400	100.0	100.0
participant_02	1.2848843698418900	1.5808104494601100	100.0	100.0
participant_03	1.039624549362630	1.0793950523380800	100.0	100.0

Table 6.1: Experiment data for experts

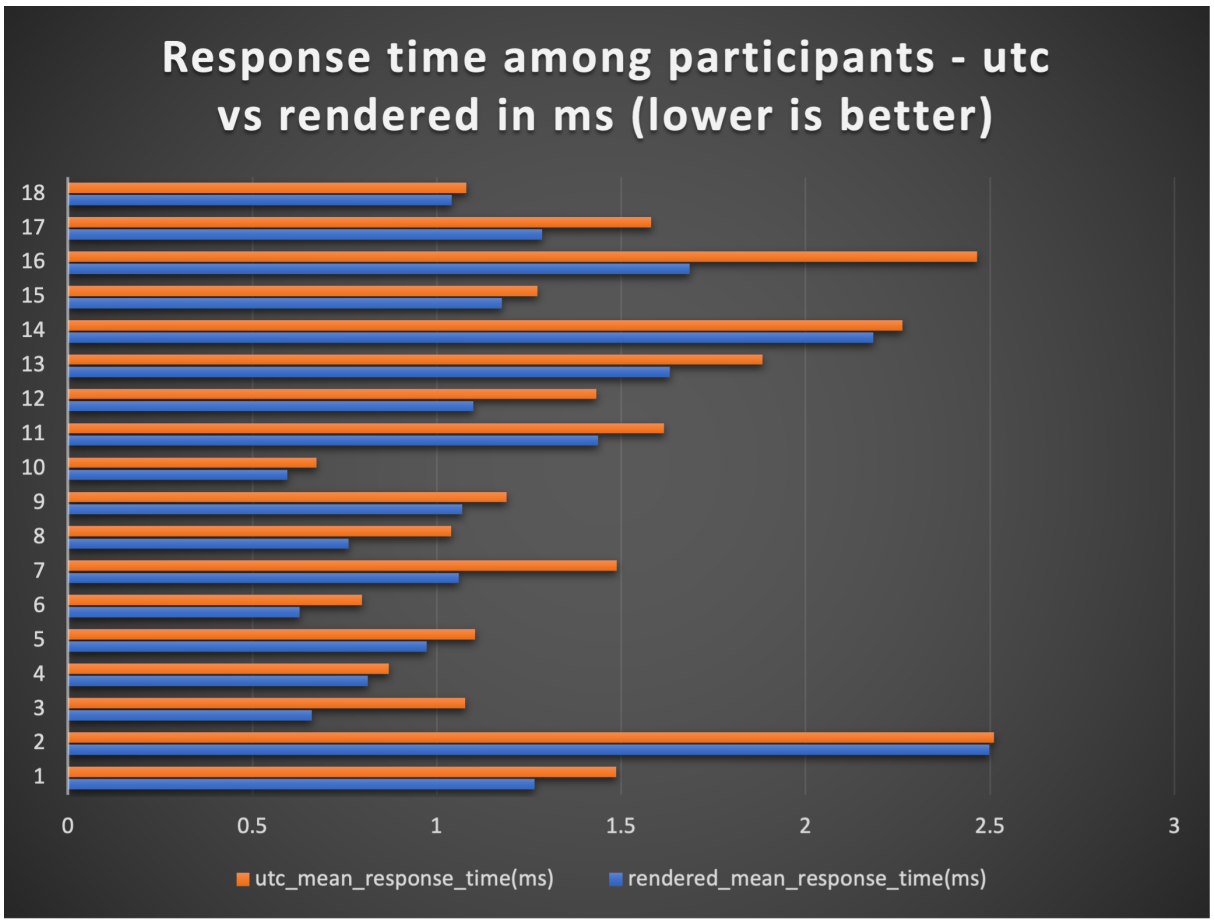


Figure 6.5: The graph above is visualizing the comparison between response times for each user in this category

6.3.2 Non-Experts

The non-experts have no previous knowledge on either of the two methods, similar results observed in the case of non-experts as well. The rendered time is lower for majority of the participants even with non-experts indicating the developed visualisation tool can accurately and quickly identify the traffic saturation points more efficiently.

participant_id	rendered_mean_response_time	utc_mean_response_time	rendered_accuracy	utc_accuracy
participant_11	1.2530315604846700	1.6012760729179700	100.0	100.0
participant_10	0.6864951341151030	0.8888499183289240	100.0	100.0
participant_06	0.8737111082082270	1.5908064678136700	100.0	90.0
participant_14	1.335652259932250	1.4476019827532600	100.0	100.0
participant_13	0.9095039415755310	1.0012967266026000	100.0	100.0
participant_05	1.4434695700000200	1.1314874100000000	90.0	100.0
participant_08	0.9930776555876950	0.8894282214692790	100.0	100.0
participant_03	1.1705742514575800	1.2390355601906800	100.0	100.0
participant_15	0.6863328223406940	0.8458882587485500	100.0	100.0
participant_07	1.3740236320445500	1.734143068481350	100.0	100.0
participant_12	1.347366447697280	1.8662851084285600	100.0	90.0
participant_02	0.9984265070874240	2.229926722845990	100.0	90.0
participant_01	2.5183958173147400	2.093771047773770	100.0	90.0

Table 6.2: Experiment data for non-experts

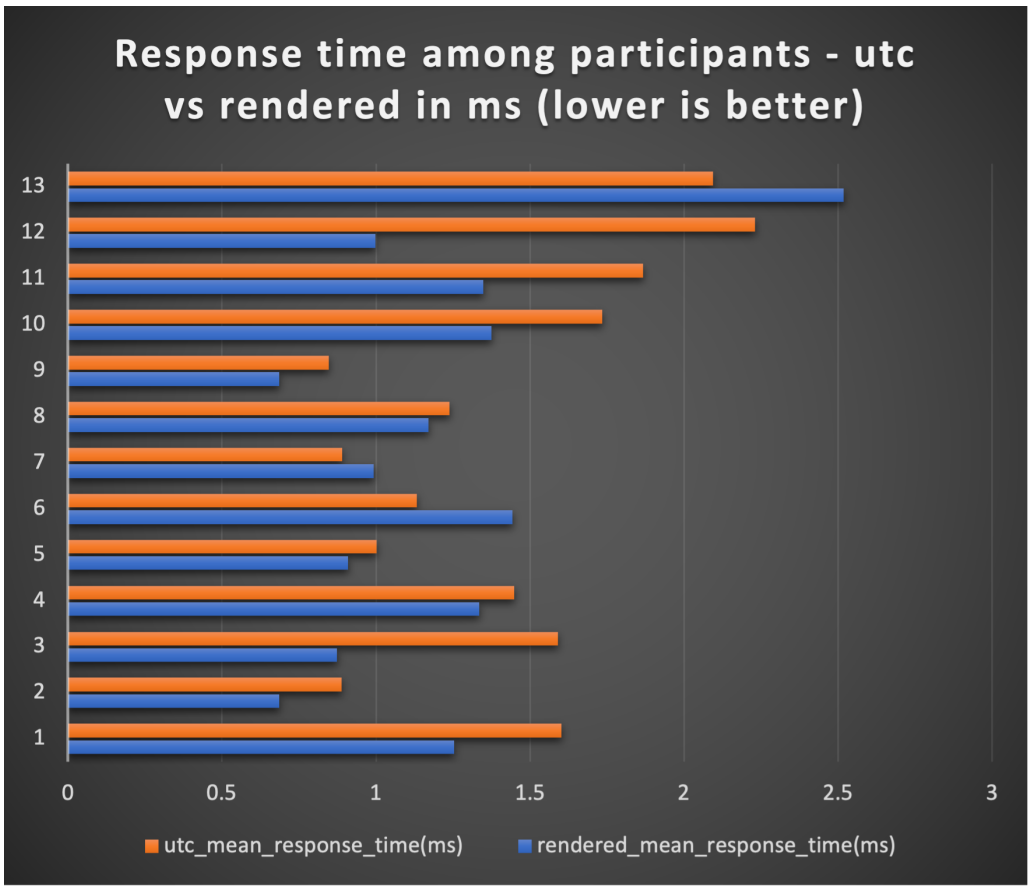


Figure 6.6: The graph above is visualizing the comparison between response times for each user in this category

6.4 Analysed Results

The results were analyzed to generate a confusion matrix for each dataset to assess the performance of the classification model. Both text and rendered images exhibited strong sensitivity, enabling participants to effectively determine whether a link was saturated. However, the study found slight biases: a small tendency for false alarms (identifying links as saturated when they are not) in rendered images, and a small tendency for misses (failing to identify actually saturated links) in text images. Further research with a larger study is needed to conclude if these biases are significant or merely due to chance, as the numbers are small.

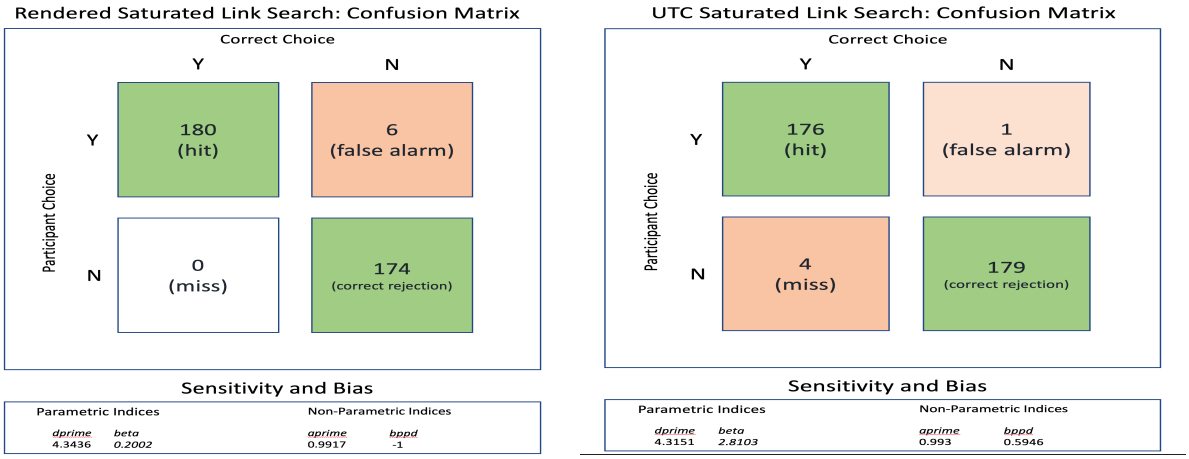
An interesting finding is the response time: participants made decisions about rendered images approximately 20% faster than with text images, and this held true regardless of whether the signal was present. Both expert and non-expert participants could identify false positives (incorrectly identifying links as saturated) and false negatives (missing actually saturated links) very effectively. This underscores the importance of effective visualization in identifying traffic situations in any city.

The confusion matrix showed a correct identification rate of 99%, demonstrating the high reliability of the tool. For rendered saturated links, the study observed high sensitivity (high aprime) and a bias towards false alarms, differing from non-experts. For text saturated links, there was also high sensitivity (high aprime) but with a bias towards misses. The false positives or false negatives also could be identified very effectively by expert and non-expert participants. Therefore, the study emphasizes the success of prototyping an effective Aloka visualisation in helping both experts and non-experts identifying the traffic situation.

In summary, the study found that rendered images allow for faster decision-making for saturated links, with high accuracy and effective identification of false positives and negatives. While there are small biases towards false alarms in rendered images and misses in text images, further research is needed to determine the significance of these biases. Overall, the tool is reliable and shows minimal bias.

6.4.1 Confusion Matrices and Statistical Analysis of Results

6.4.1.1 Results for Experts



(a) Confusion Matrix for rendered dataset for experts

(b) Confusion Matrix for UTC dataset for experts

Figure 6.7: Confusion Matrix generated for both rendered and UTC dataset for experts

	UTC Text	Rendered	pvalue
saturated present (y)	1.4049	1.2170	0.024718
saturated absent (n)	1.4643	1.2117	0.001196
pvalue	0.4534	0.9481	NA

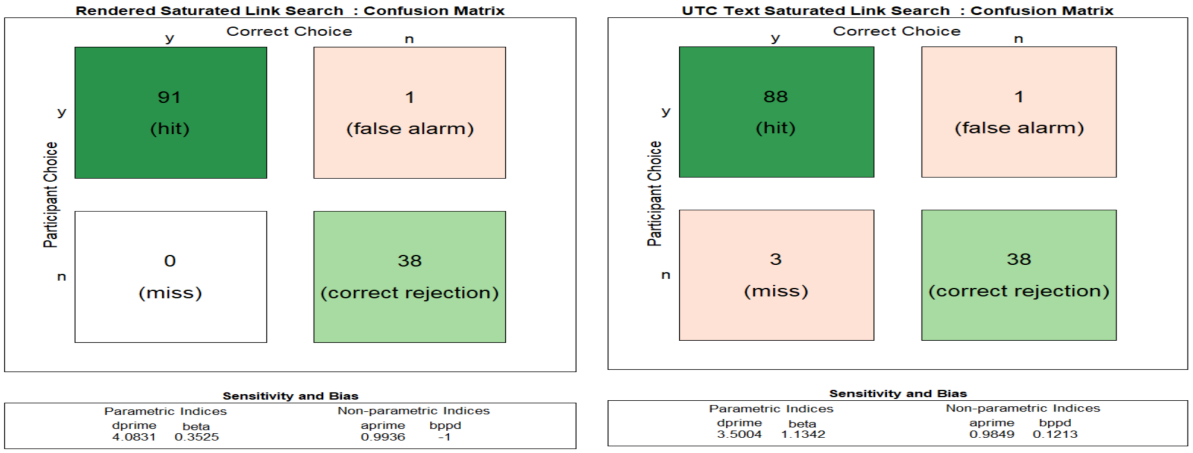
Figure 6.8: T-test result for experts

- Accuracy and Sensitivity:
  - Both datasets show high sensitivity and accuracy in detecting true positives and true negatives.
  - Rendered dataset showed a perfect sensitivity with no misses, indicating very high reliability in identifying true positives.
- Bias Analysis:

- Rendered dataset has a slight liberal bias (B" = -1.0) indicating a tendency to answer "yes".
- UTC dataset shows a conservative bias (B" = 0.5946) indicating a tendency to answer "no".

- Statistical Significance:
  - The p-values indicate that the differences in detecting saturated conditions between the UTC Text and Rendered datasets are statistically significant, particularly for the saturated absent condition (p-value = 0.024718).
  - Rendered dataset decisions are more consistent (lower p-values) compared to UTC Text.

6.4.1.2 Results for Non-Experts



(a) Confusion Matrix for rendered dataset for non-experts

(b) Confusion Matrix for UTC dataset for non-experts

Figure 6.9: Confusion Matrix generated for both rendered and UTC dataset for non-experts

	UTC Text	Rendered	pvalue
saturated present (y)	1.4780	1.1425	0.01061
saturated absent (n)	1.3101	1.3316	0.87183
pvalue	0.2264	0.1315	NA

Figure 6.10: T-test result for non-experts

- Accuracy and Sensitivity:
  - Both datasets show high sensitivity and accuracy in detecting true positives and true negatives.
  - The rendered dataset showed perfect sensitivity with no misses, indicating very high reliability in identifying true positives.
- Bias Analysis:
  - The rendered dataset has a slight liberal bias (B" = -1.0), indicating a tendency to answer 'yes'.
  - The UTC dataset shows a slight conservative bias (B" = 0.1213), indicating a slight tendency to answer 'no'.
- Statistical Significance:
  - The p-values indicate that the differences in detecting saturated conditions between the UTC Text and Rendered datasets are statistically significant for the saturated present condition (p-value = 0.01061).
  - The rendered dataset decisions are more consistent (lower p-values) compared to the UTC Text dataset for saturated absent condition.

6.4.1.3 Analysis Summary

Overall, the analysis shows that for both experts and non-experts the rendered dataset is highly reliable and efficient for identifying saturated links, with minimal bias and high

accuracy. The UTC dataset (for both experts and non-experts) also performs well but has a slight conservative bias and slightly lower consistency compared to the rendered dataset.

Chapter 7

Discussion

The original research goals focused on determining whether the visualization grammar for 3D visualizations is effective in prototyping designs for decision-making scenarios.

1. Can we define a visualization grammar that allows us to encode a 3D visualization?
2. To implement and evaluate the visualization grammar;
3. To design a 3D visualization of traffic saturation to better help operators detect road links that are saturated compared with existing UTC (Urban Traffic Control) systems;
4. Can we use the visualization grammar to prototype this 3D visualization design for traffic saturation?

In order to understand the results we frame and structure our questions as defined below:

- The Interpretations of the result: what are the key findings and what do the results mean?
- The Implications: do the results matter in this research and broader area?
- Limitations: what the results can and cannot tell us?
- A summary and recommendations for this research and future work: What doors does this research open and what can be done to further the field of study?

7.1 Key Findings

The key finds in this study can be summarized below:

- The results show that there is a significant improvement in response times
  - With approximately a 33% improvement in response times for non-experts.
  - And nearly 20% improvement in response times for experts.
- The study demonstrates a correlation between good visual design and improvement in understanding context of the data presented
- The analysis that was run on the data suggests the following
  - The confusion matrix shows users are less susceptible to miss and accuracy is quite good for both sets of users.
  - The T-test run on each data-set suggests rendered presents an improved outcome.
- The data suggests we have created an effective visualization to present traffic saturation for operators to detect such events.
- The ability to prototype effective visualizations is also demonstrated through the rendering of all visualizations by Aloka.

7.2 Interpreting the results

An effective visualization is one that enhances the user's understanding of the data; in this case study, it can also be one that improves an operator's ability to respond to events more quickly and accurately, while using the same data stream. As can be noted from the results a few participants were excluded due to color blindness. Since the rendered visualization relies on user's ability to detect colour it would make those data points unusable. There was another participant in non-experts whose data had technical issues

causing a massive skew in data. This was excluded as well to maintain integrity in the analysis.

Having raw numbers alone is not an effective way to understand if users or operators are understanding the context of the data. In our case we are trying to convey the region and design of the specific road junction to operator. As this can convey more information such as reasons for congestion due to the road shape or length etc. It can also let the operator know if it is in a busy city centre and this could also be grounds for high saturation. Overall the context of the data is conveyed more effectively to the operator. This is why exploring the user's or operator's opinion is crucial in understanding if it did improve context.

7.2.1 Exploring User Feedback

Understanding user feedback from non-experts revealed an interesting point that they found it easier to understand the rendered glyphs used for visualization as they were not familiar with the UTC table visualization:

- Result is obvious and the color is faster;
- Picture is faster, straightforward if colorblind is harder, contrast lower for viz image to make red stand out more;
- Arrows are quicker, columns likely to make a mistake over time, arrows more accurate;
- The picture is much easier than going through the column to find the saturation;
- It was very quick to identify saturated links in the table as the threshold was 100, and so I just had to look to see if there was a digit in the 100s column. It would have taken much longer if the threshold had been a 2 digit number;
- Easy experiment;

Exploring the user feedback from experts revealed a great detail in experiment and effectiveness of the rendered glyph. Most operators found the context the visualization provided

useful and also had more detailed information about the experiment and improvements they felt.

- Focus required for the UTC image is lower;
- A useful exercise;
- At first when reading consent form it seemed like the longer arrows had some sort of significance over the saturation even though they were not red. Once explained that this was just the size of the link then this was easy to understand;
- Whats more confusing/disorientating is the flicking between types of images. All the ones I got wrong are when one set of images change to another. Numbers would be more useful in context where just wanting to look for saturated links, Rendered would be more useful where geographic context is needed;
- Image of graph is at the top so eyes take longer to find the figures when previously looking at the map;
- Mind wandered a bit due to the receptive nature of the task;
- Table color can improve response time for the table. The red color is usually indicates faulty equipment in UTC and purple is used to indicate what red usually does; the experiment was slightly artificial in that it was possible to focus on the same column every time rather than switching from different images;
- I found the images easier than the tables;
- The table might be marginally faster due to being used to it. But feel its better for people at entry level to use the graphic;
- Very impressed with the straight forward graphics;
- In the instructions you should have greater or equal than 100%. It took me more time to find numbers in the columns;

- I felt much more comfortable looking at rendered images rather than UTC table view;
- The centering of the images is not consistent;

7.3 Limitations

7.3.1 Limitations in study

There are a few limitations in this study:

- It is beyond the scope of this study to try and evaluate a grammar. However we have explored the scope of the grammar and how it was used to construct visualizations in 3D with data binding to the materials used in ray traced rendering.
- The degree with which the effectiveness of the design of the visualization is measured is relative to the UTC table, not to other designs, which may or may not be a factor in understanding the design itself.
- An operator's response time which has improved significantly compared to the existing system may not be an indicator on effective decisions made by the individual operator. The design of the visualization helps the operator understand context of the region he is looking at and also detect saturated traffic links faster.

7.3.2 Limitations of Aloka

Although Aloka is a very flexible and extendable declarative 3D visualization grammar, in its **current state** there are a few limitations.

- Lack of Detailed Animation Support:
  - No Animation Definitions: The grammar does not include specific rules for defining animations or handling time-based changes within the 3D scene.
- Limited Interactivity Specifications:

  - Basic Event Handling: While "device" includes event types and sources, the grammar lacks detailed specifications for complex user interactions and real-time event handling.

- Simplified Shader and Material Definitions:
  - Basic Shader Attributes: The shader definitions are relatively basic, lacking detailed parameters for advanced shading techniques such as normal mapping, bump mapping, or custom shader programming.
  - Limited Material Properties: Material definitions are straightforward and may not cover complex material properties like subsurface scattering or multi-layered materials.

- Basic Lighting Models:
  - Simple Light Properties: The grammar defines basic light properties but does not account for advanced lighting techniques like global illumination, ambient occlusion, or volumetric lighting in its current state.

- Sparse Camera Configurations:
  - Limited Camera Settings: The camera definitions cover essential properties but do not include advanced features like depth of field, motion blur, or multiple camera views in its current state.

In summary its limitations include the lack of support for advanced animations, detailed interactivity, complex shaders and materials, sophisticated lighting models and comprehensive camera settings. Addressing these limitations would require extending the grammar to incorporate these advanced features and capabilities. The grammar and API was however designed to be able to easily extend this and was in consideration during the design phase. Hence, every one of these limitations are only limitations in its current state and can be overcome with future iterations.

Chapter 8

Summary and Conclusions

To summarize this thesis, a new declarative 3D visualization system was designed and developed and named Aloka. For helping improve context and response times in detecting issues in road links for operators a new 3D visualization design was prototyped using Aloka. Aloka represents a pioneering effort in the field of 3D visualization, addressing specific challenges associated with data representation in three dimensions, which traditional 2D systems could not adequately handle.

8.1 Thesis Conclusion

Aloka stands as a great example in how declarative programming can be designed to solve problems in 3D visualization where such a feat has not been attempted before. This abstraction can help data scientists and data visualization experts craft visualizations where the data required the third dimension to provide context or show more information. This expands the scope of data visualization as a topic as it opens up possibilities in how data visualization is defined and created. This is the novelty the thesis brings to practice. Aloka also facilitated the design of a new visualization that helped operators improve response times significantly and get better context and understanding of their data. The further proof that the response times in detecting traffic saturation for both experts and non-experts were improved irrespective of their differences albeit the experts showed a smaller difference in both, signifies that a 3D design in the context of traffic visualization

can impact positively on operator's ability to make decisions.

This summarizes a list of contributions that introduce novelty to the field of visualization.

- Introducing a declarative grammar specifically for creating multidimensional visualizations, potentially extending into the fourth dimension—time.
- Developing a unique visual method and design for representing traffic saturation.
- Implementing a cloud-based architecture to facilitate the modern rendering of visualizations, utilizing Infrastructure as a Service (IaaS), which enhances accessibility and scalability.

8.2 Future Work and Direction

Building upon the foundations laid by Aloka, the future directions of this research could aim to significantly expand and deepen the integration of 3D visualization technologies with real-time data processing and varied application domains. Key areas for further exploration include:

- Complex Materials in Visualization: Exploring the possibility of more 3D visualizations and complex material mapping based on stream data would be a new form and standard of future visualizations. This will also be used by many more scientists/data visualization experts to create and prototype unique 3D visualizations with more data types and variations in scale and scope.
- Enhanced Data Mapping: Future efforts will focus on refining the ability to integrate complex data streams into 3D visualizations. By developing methods to dynamically map data such as traffic flows, weather patterns, or population changes onto 3D models, these visualizations could provide invaluable insights for urban planning, environmental monitoring, and other critical fields. This advancement will make 3D visualizations a more practical tool for real-world applications.
- Time Integration and Performance Optimization: Research will also explore incorporating time as a fourth dimension in visualizations, allowing users to track changes

over time within the 3D space. Alongside this, there will be a focus on improving the performance of the Aloka system, optimizing it to handle larger datasets and enhancing its rendering capabilities. Utilizing cloud technologies could play a crucial role in achieving greater scalability and real-time processing capabilities.

- User-Friendly Design and Diverse Applications: Making Aloka more user-friendly will be a priority, ensuring that it is accessible to a broader audience, including those without technical expertise. Additionally, testing its application across various fields such as biomedical imaging for complex anatomical data or detailed component modeling in engineering will help in understanding its full potential and identifying areas for improvement.

- Cloud Accessibility and Collaborative Features: Expanding Aloka's cloud-based deployment will make the system more accessible to users around the globe and facilitate collaborative opportunities across different sectors. This approach includes exploring software-as-a-service (SaaS) models to lower the barriers for organizations and individuals needing advanced visualization tools.

- Educational Use and Resource Development: Developing educational resources and training programs around Aloka will be essential for teaching both academic and professional audiences the principles of 3D visualization. These resources will help cultivate a new generation of data scientists and visualization experts equipped to employ advanced visualization techniques in their work.

By addressing these areas, the Aloka project can not only enhance its current capabilities but also set a new standard for what can be achieved with declarative 3D visualization systems in various scientific, industrial, and educational contexts.

Bibliography

[Akram Hassan et al., 2019] Akram Hassan, K., Rönnberg, N., Forsell, C., Cooper, M., and Johansson, J. (2019). A study on 2d and 3d parallel coordinates for pattern identification in temporal multivariate data. In *2019 23rd International Conference Information Visualisation (IV)*, pages 145–150.

[Bariic et al., 2012] Bariic, A., Amaral, V., and Goulão, M. (2012). Usability evaluation of domain-specific languages. *Eighth International Conference on the Quality of Information and Communications Technology, Lisbon, Portugal*, 2012:342–347.

[Bertin, 1967] Bertin, J. (1967). *Sémiologie graphique (1st Edition)*. Gauthier-Villar.

[Bittner, 2020] Bittner, K. (2020). *The Current State of the Art in Real-Time Cloud Rendering With Raymarching*. Researchgate.

[Blake et al., 2020] Blake, A. J., Hahn, G. S., Grey, H., Kwok, S. A., McIntosh, D., and Gries, G. (2020). Polarized light sensitivity in pieris rapae is dependent on both color and intensity. *Journal of Experimental Biology*, 223:13.

[Borgo et al., 2013] Borgo, R., Kehrer, J., Chung, D. H. S., Maguire, E., Laramee, R. S., Hauser, H., Ward, M., and Chen, M. (2013). Glyph-based visualization: Foundations, design guidelines, techniques, and applications. *Eurographics State of the Art Reports*, 39.

[Bostock and Heer, 2009] Bostock, M. and Heer, J. (2009). Protovis: A graphical toolkit for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 6(1121–1128):2009.

[Bostock et al., 2011] Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309.

[Bridges et al., 2020a] Bridges, D., Pitiot, A., MacAskill, M. R., and Peirce, J. W. (2020a). The timing mega-study: comparing a range of experiment generators, both lab-based and online. *PeerJ*, 8:e9414.

[Bridges et al., 2020b] Bridges, D., Pitiot, A., MacAskill, M. R., and Peirce, J. W. (2020b). The timing mega-study: comparing a range of experiment generators, both lab-based and online. *PeerJ*, 8(e9414):e9414.

[Brychtová and Çöltekin, 2017] Brychtová, A. and Çöltekin, A. (2017). The effect of spatial distance on the discriminability of colors in maps. *Cartography and Geographic Information Science*, 44(229–245):3.

[Carnall, 2019] Carnall, C. (2019). *Managing change*. Routledge.

[Damiano and Walther, 2019] Damiano, C. and Walther, D. B. (2019). Distinct roles of eye movements during memory encoding and retrieval. *Cognition*, 184:119–129.

[Deering, 2000] Deering, M. (2000). The limits of human vision. pages 1–6.

[Demian and Fruchter, 2009] Demian, P. and Fruchter, R. (2009). Effective visualisation of design versions: Visual storytelling for design reuse. *Research in Engineering Design*, 19.

[Dhieb, 2019] Dhieb, M. (2019). Translating bertin into arabic today: new hidden facets of semiology of graphics. *Cartography and Geographic Information Science*, 46(2):163–175.

[Doersch et al., 2015] Doersch, C., Gupta, A., and Efros, A. A. (2015). Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430.

[Du et al., 2016] Du, J., Sheng, W., and Liu, M. (2016). Human-guided robot 3d mapping using virtual reality technology. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Daejeon, South Korea*, 2016:4624–4629.

[Eastman, 1986] Eastman, R. (1986). Economic geography. *Taylor & Francis, Ltd.*, 62(1):104–105.

[Entezari et al., 2018] Entezari, A., Roohani, I., Li, G., Dunstan, C., Rognon, P., Li, Q., Jiang, X., and Zreiqat, H. (2018). Architectural design of 3d printed scaffolds controls the volume and functionality of newly formed bone. *Advanced Healthcare Materials*, 8:18013.

[Ferreira et al., 2013] Ferreira, N., Poco, J., Vo, H. T., Freire, J., and Silva, C. T. (2013). Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158.

[Figueroa et al., 2018] Figueroa, J. C., Arellano, R., and Calinisan, J. (2018). A comparative study of virtual reality and 2d display methods in visual search in real scenes. (366-377):319–60591.

[Forsell and Johansson Westberg, 2007] Forsell, C. and Johansson Westberg, J. (2007). Task-based evaluation of multi-relational 3d and standard 2d parallel coordinates. *Proceedings of SPIE - The International Society for Optical Engineering.*

[Frank et al., 2019] Frank, A. G., Dalenogare, L. S., and Ayala, N. F. (2019). Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210:15–26.

[Frankel and Laby, 2015] Frankel, T. and Laby, A. B. (2015). *The regulation of money managers: mutual funds and advisers (Vol. 3).* Wolters Kluwer Law & Business.

[Gao et al., 2016] Gao, H. Y., Yao, Q. X., Liu, P., Zheng, Z. Q., Liu, J. C., and Zheng, H. D. ... & Zeng, Z. (2016). The latest development of display technologies. *Chinese Physics B*, 25:9.

[Gardony et al., 2018] Gardony, A. L., Martis, S. B., Taylor, H. A., and Brunyé, T. T. (2018). Interaction strategies for effective augmented reality geo-visualization: Insights from spatial cognition. *Human-Computer Interaction*, pages 1–43.

[Goode et al., 2015] Goode, S., Lin, C., Tsai, J. C., and Jiang, J. J. (2015). Rethinking the role of security in client satisfaction with software-as-a-service (saas) providers. *Decision Support Systems*, 70:73–85.

[Hafi et al., 2017] Hafi, L. E., Ding, M., Takamatsu, J., and Ogasawara, T. (2017). Gaze tracking and object recognition from eye images. *First IEEE International Conference on Robotic Computing (IRC), Taichung*, 2017:310–315.

[Hao and Wang, 2013] Hao, W. and Wang, Y. (2013). Saliency-guided Luminance Enhancement for 3D Shape Depiction. pages 9–14.

[Hassall et al., 2017] Hassall, M. M., Barnard, A. R., and MacLaren, R. E. (2017). Gene therapy for color blindness. *The Yale journal of biology and medicine*, 90(543–551):4.

[Heer and Bostock, 2010] Heer, J. and Bostock, M. (2010). Declarative language design for interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(1149–1156):6.

[Holliman and Watson, 2015] Holliman, N. and Watson, P. (2015). Scalable real-time visualization using the cloud. *IEEE Cloud Computing*, 2(6):90–96.

[Holliman et al., 2019] Holliman, N. S., Antony, M., Charlton, J., Dowsland, S., James, P., and Turner, M. (2019). Petascale cloud supercomputing for terapixel visualization of a digital twin. *IEEE Transactions on Cloud Computing*, pages 1–1.

[Hou and Zhang, 2007] Hou, X. and Zhang, L. (2007). Saliency detection: A spectral residual approach. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

[Häussler et al., 2018] Häussler, J., Stein, M., Seebacher, D., Janetzko, H., Schreck, T., and Keim, D. (2018). Visual analysis of urban traffic data based on high-resolution and high-dimensional environmental sensor data.

[Ji et al., 2013] Ji, X. S., Kang, S. R., Yu, Y. L., and Chien, W. K. (2013). A study on the statistical comparison methods for engineering applications. In *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 576–580.

[Johansson et al., 2014] Johansson, J., Forsell, C., and Cooper, M. (2014). On the usability of three-dimensional display in parallel coordinates: Evaluating the efficiency of identifying two-dimensional relationships. *Information Visualization*, 13(1):29–41.

[Jordan, 2016] Jordan, B. (2016). *Fundamentals of investments*. McGraw-Hill, Higher Education.

[Kaiser et al., 2019] Kaiser, D., Turini, J., and Cichy, R. M. (2019). A neural mechanism for contextualizing fragmented inputs during naturalistic vision. *eLife*, 8:e48182.

[Lavoué et al., 2018] Lavoué, G., Cordier, F., Seo, H., and Larabi, M.-C. (2018). Visual attention for rendered 3d shapes. *Computer Graphics Forum*, 37(191–203):2.

[Lee et al., 2020] Lee, C., Kim, Y., Jin, S., Kim, D., Maciejewski, R., Ebert, D., and Ko, S. (2020). A visual analytics system for exploring, monitoring, and forecasting road traffic congestion. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3133–3146.

[Leer et al., 2017] Leer, A., Engelhard, I. M., Lenaert, B., Struyf, D., Vervliet, B., and Hermans, D. (2017). Eye movement during recall reduces objective memory performance: An extended replication. *Behavior research and therapy*, 92:94–105.

[Li et al., 2017] Li, L., Yu, F., Shi, D., Shi, J., Tian, Z., Yang, J., Wang, X., and Jiang, Q. (2017). Application of virtual reality technology in clinical medicine. *American journal of translational research*, 9:3867.

[Matković et al., 2010] Matković, K., Lež, A., Gračanin, D., Ammer, A., and Purgathofer, W. (2010). Event Line View: Interactive Visual Analysis of Irregular Time-Dependent Data. In Taylor, R., Boulanger, P., Krüger, A., and Olivier, P., editors, *Smart Graphics*, pages 208–219, Berlin, Heidelberg. Springer Berlin Heidelberg.

[McCracken and Reilly, 2003] McCracken, D. D. and Reilly, E. D. (2003). Backus-naur form (bnf). In *Backus-Naur form (BNF)*.

[McDonald et al., 2011] McDonald, J., Powell Davies, G., Jayasuriya, R., and Fort Harris, M. (2011). Collaboration across private and public sector primary health care services: benefits, costs and policy implications. *J. Interprof. Care*, 25(4):258–264.

[Medwetz, 2019] Medwetz, A. (2019). *The Effects of Color on Visual Perception and Visual Clutter (Doctoral dissertation.* PhD thesis, Kent State University).

[Neuliep, 2017] Neuliep, J. W. (2017). *Intercultural communication: A contextual approach.* Sage Publications.

[Neuville et al., 2018] Neuville, R., Pouliot, J., Poux, F., De Rudder, L., and Billen, R. (2018). A formalized 3d geovisualization illustrated to selectivity purpose of the virtual 3d city model. *ISPRS International Journal of Geo-Information*, 7:5.

[Noordegraaf, 2016] Noordegraaf, M. (2016). Reconfiguring professional work: Changing forms of professionalism in public services. *Administration & Society*, 48(7):783–810.

[Oscar et al., 2017] Oscar, N., Mejía, S., Metoyer, R., and Hooker, K. (2017). Towards personalized visualization: Information granularity, situation, and personality. *DIS - Proceedings of the 2017 ACM Conference on Designing Interactive Systems,*, 2017:811–819.

[Oyster, 1999] Oyster, C. W. (1999). The human eye: structure and function. *Nature medicine*, 5 11:1229.

[Pandit and Dhakal, 2020] Pandit, R. and Dhakal, R. (2020). Assessment of color vision among health science students. *Nepal Medical College Journal*, 22:49–53.

[Pham et al., 2012] Pham, T., Mejía, S., Metoyer, R., and Hooker, K. (2012). The Effects of Visualization Feedback on Promoting Health Goal Progress in Older Adults. In Meyer, M. and Weinkaufs, T., editors, *EuroVis - Short Papers*. The Eurographics Association.

[ping Zhang and Tsingan, 2014] ping Zhang, R. and Tsingan, L. (2014). Extraversion and neuroticism mediate associations between openness, conscientiousness, and agreeableness and affective well-being. *Journal of Happiness Studies*, 15:1377–1388.

[Pinker and Feedle, 1990] Pinker, S. and Feedle, R. (1990). *A theory of graph comprehension*, pages 73–126.

[Piri, 2017] Piri, H. (2017). *Flow visualization in 3D printed PEM fuel cell bipolar plates (Doctoral dissertation*. University of British Columbia.

[Roettl and Terlutter, 2018] Roettl, J. and Terlutter, R. (2018). The same video game in 2d, 3d, or virtual reality–how do technology impact game evaluation and brand placements? *PloS one*, 13:7.

[Sage, 2016] Sage, M. (2016). Creating augmented reality experiences for enterprise: Good practices, lessons learned, and technological insights. *IEEE Consumer Electronics Magazine*, 6(1):42–44.

[Sandy, 2018] Sandy, M. (2018). 'announcing microsoft directx raytracing! *Microsoft, March*.

[Saputro et al., 2018] Saputro, A., Sumpeno, S., and Hariadi, M. (2018). Performance of the ieee 802.15.4 protocol as the marker of augmented reality in museum. *Journal of Physics: Conference Series*, 2020(1742-6596).

[Satyanarayan and Heer, 2014] Satyanarayan, A. and Heer, J. (2014). Lyra : An interactive visualization design environment. 33:3.

[Satyanarayan et al., 2017] Satyanarayan, A., Moritz, D., Wongsuphasawat, K., and Heer, J. (2017). Vega-lite: A grammar of interactive graphics. *IEEE Transactions on Visualization and Computer Graphics*, 23(341–350):1.

[Satyanarayan et al., 2015] Satyanarayan, A., Russell, R., Hoffswell, J., and Heer, J. (2015). Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22:10.

[Schiffer et al., 2010] Schiffer, T., Riffnaller-Schiefer, A., Berndt, R., Ullrich, T., Settgast, V., and Fellner, D. (2010). Enlightened by the web – a service-oriented architecture for real-time photorealistic rendering.

[Scowen, 1998] Scowen, R. S. (1998). Extended bnf — a generic base standard. In *Extended BNF — A generic base standard*.

[Seethamraju, 2015] Seethamraju, R. (2015). Adoption of software as a service (saas) enterprise resource planning (erp) systems in small and medium-sized enterprises (smes). *Information systems frontiers*, 17(3):475–492.

[Shahrbabaki, 2015] Shahrbabaki, S. (2015). Contribution of colour in guiding visual attention and in a computational model of visual saliency.

[Sicat et al., 2019] Sicat, R., Li, J., Choi, J., Cordeil, M., Jeong, W. K., Bach, B., and Pfister, H. (2019). Dxr: A toolkit for building immersive data visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 25(715–725):1.

[Terence, 2010] Terence, P. (2010). *Language Implementation Patterns*.

[van Dyck and Gruber, 2020] van Dyck, L. and Gruber, W. (2020). Seeing eye-to-eye? a comparison of object recognition performance in humans and deep convolutional neural networks under image manipulation.

[Velez et al., 2005] Velez, M. C., Silver, D., and Tremaine, M. (2005). Understanding visualization through spatial ability differences. *IEEE Visualization, 2005*, 5(511–518).

[Wang and Dudek, 2014] Wang, B. and Dudek, P. (2014). A fast self-tuning background subtraction algorithm. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 401–404.

[Wang et al., 2008] Wang, L., Giesen, J., McDonnell, K. T., Zolliker, P., and Mueller, K. (2008). Color design for illustrative visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(1739–1746):6.

[Wibble et al., 2020] Wibble, T., Södergård, U., Träisk, F., and Pansell, T. (2020). Intensified visual clutter induces increased sympathetic signaling, poorer postural control, and faster torsional eye movements during visual rotation. *PloS one*, 15:1.

[Wilkinson, 1999] Wilkinson, L. (1999). *Statistics and Computing- The Grammar of Graphics (1 Edition)*. Springer-Verlag New York, 1 edition.

[Witzel and Gegenfurtner, 2018] Witzel, C. and Gegenfurtner, K. R. (2018). *Color perception: Objects, constancy, and categories*. Annual Review of Vision Science.

[Witzel et al., 2021] Witzel, D. D., Turner, S. G., and Hooker, K. (2021). Self-Perceptions of Aging Moderate Associations of Within- and Between-Persons Perceived Stress and Physical Health Symptoms. *The Journals of Gerontology: Series B*, 77(4):641–651.

[Wood et al., 1994] Wood, K., Palmer, J., and Bretherton, R. (1994). Congestion analysis and diagnosis in utc networks. In *Seventh International Conference on Road Traffic Monitoring and Control, 1994.*, pages 172–176.

[Yost and North, 2006] Yost, B. and North, C. (2006). The perceptual scalability of visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(837–844):5.

[Yuan, 2016] Yuan, Y. (2016). Changing the world with virtualaugmented reality technologies. *IEEE Consumer Electronics Magazine*, 6(1):40–41.

[Zhonghua and Lingda, 2016] Zhonghua, Y. and Lingda, W. (2016). 3d-parallel coordinates: Visualization for time varying multidimensional data. *7th IEEE International*

Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 7:655–658.

[Ziemkiewicz et al., 2012] Ziemkiewicz, C., Ottley, A., Crouser, R. J., Chauncey, K., Su, S. L., and Chang, R. (2012). Understanding visualization by understanding individual users. *IEEE Computer Graphics and Applications*, 32(88–94):6.

[Ziemkiewicz et al., 2013] Ziemkiewicz, C., Ottley, A., Crouser, R. J., Yauilla, A. R., Su, S., Ribarsky, W., and Chang, R. (2013). How visualization layout relates to locus of control and other personality factors. *IEEE Transactions on Visualization and Computer Graphics*, 19:1109–1121.

Appendix A

Grammar Definition

The user manual for Aloka Grammar:

A.1 Resource Class

The resource class holds a diverse set of resources used by the view such as Data, Camera, Material, Lights, Scales and Fonts. The purpose of the resource class is to organize a common set of resources in a super class. This is by design, and helps Aloka give a structural way of defining and accessing resources.

A.1.1 Data Class

The properties of the Data class:

A.1.1.1 Name

This property defines the name of the data class. No two data sets can have the same name. The names of the data sets are unique and will be used as a reference in the grammar for mapping or other data analytics. The tag is defined as "name".

A.1.1.2 Source

Source refers to the source location of data or the data itself (JSON data only). The tag is defined as "source".

A.1.1.3 Type

The Type defines if the data is DECLARED (The data is going to be defined in the source as a JSON array or object) or LOCAL_STORAGE (The data is in local storage with reference to the API) or LIVE (The data is loaded externally, either an API or web access URL). The tag is defined as "type".

A.1.1.4 Format

The supported formats for data are BLEND (This is a blender file which can be used to load entire models to the view), TEXT (Simple text data), JSON (This is JSON format data). The tag is defined as "format".

A.1.1.5 Parameters

This section can contain authentication information if the data is being accessed from a secure API or file storage. The tag is defined as "params".

A.1.2 Camera Class

The properties of the Camera class: [https://docs.blender.org/manual/en/latest/render/cameras.html]

A.1.2.1 Name

This property defines the name of the camera class. No two cameras can have the same name. The names of the cameras are unique and will be used as a reference in the grammar for different views. The tag is defined as "name".

A.1.2.2 Type

Different camera types such as ORTHOGONAL(Orthographic viewing is useful because it provides a technical insight into the view/scene , making it easier to model and judge proportions, with Orthographic camera objects always appear at their actual size, regardless of distance), PANORAMA (This is intended for future VR integration into Aloka)

and PERSPECTIVE (The default 3D view, this is setup like how objects are viewed in the real world) are supported. The tag is defined as "type".

A.1.2.3 Sensor Size

This setting is a way to control the focal length of the camera. The tag is defined as "sensor_size".

A.1.2.4 Orthographic Scale

This controls the apparent size of objects projected on the image. The tag is defined as "orthographic_scale".

A.1.2.5 Resolution

Camera resolution in width and height to override the display width or height. This property has sub-properties "width" and "height"

A.1.3 Shader Class

The properties of the Shader class:

A.1.3.1 Name

This property defines the name of the shader class. No two shaders can have the same name. The names of the shaders are unique and will be used as a reference in the material class for combining shaders to create unique materials. The tag is defined as "name".

A.1.3.2 Type

The types of shaders supported are DIFFUSE (For rough materials), GLOSS (for reflective materials) and EMISSION (for emissive materials and light sources) . The tag is defined as "type".

A.1.3.3 Property : Roughness, Strength

Depending on the type of shader defined, the shader class has properties defining the roughness and strength of the shader. The types GLOSS and DIFFUSE will need roughness defined and EMISSIVE will require strength. Both roughness and strength can be overridden and mapped based on data binding in the grammar. The tags defined are "roughness" and "strength".

A.1.3.4 Color

A default color for the shader is defined here, following format [r (Red Value), g (Green Value), b (Blue Value)]. The color can be overridden or mapped in the grammar view. The color value ranges from 0.0 to 1.0. The tag is defined as "color".

A.1.4 Material Class

The properties of the Material class:

A.1.4.1 Name

This property defines the name of the material class. No two materials can have the same name. The names of the materials are unique and will be used as a reference in the geometry class mapping materials to geometric objects. The tag is defined as "name".

A.1.4.2 Surface

The surface property defines how the shaders are combined to create a material. Two shaders are referenced and imported as "shader_a" and "shader_b". Fac is a raw number from 0.0 to 1.0 the balance in the ratio defines how the shaders are mixed. For example if the fac is 0.0 then the shader_a is dominant and shader_b becomes transparent(null, it means that shader_a will be the only shader in the material) and vice versa if the fac is 1.0 , if the fac number is 0.5 the shaders are mixed equally.

A.1.5 Lights Class

The properties of the Lights class:

A.1.5.1 Name

This property defines the name of the lights class. No two lights can have the same name. The names of the lights are unique and will be used as a reference in the layers when importing lights to the scene. The tag is defined as "name".

A.1.5.2 Type

The types of the light supported are [https://docs.blender.org/manual/en/latest/render/lights/light_obj

- 'POINT': The Point light is an omni-directional point of light, that is, a point radiating the same amount of light in all directions. It's visualized by a plain, circled dot. Being a point light source, the direction of the light hitting an object's surface is determined by the line joining the light and the point on the surface of the object itself. It can be used as simple model of e.g. a light bulb. Light intensity/energy decays based on (among other variables) distance from the Point light to the object. In other words, surfaces that are further away will be rendered darker.

- 'DIRECTIONAL': A sun light provides light of constant intensity emitted in a single direction from infinitely far away. A sun light can be very handy for a uniform clear daylight open-space illumination. In the 3D View, the Sun light is represented by an encircled black dot with rays emitting from it, plus a dashed line indicating the direction of the light.

- 'SPOTLIGHT': A Spot light emits a cone-shaped beam of light from the tip of the cone, in a given direction.

- 'AREA': The Area light simulates light originating from a surface (or surface-like) emitter. For example, a TV screen, office neon lights, a window, or a cloudy sky are

just a few types of area light. The area light produces shadows with soft borders by sampling a light along a grid the size of which is defined by the user. This is in direct contrast to point-like artificial lights which produce sharp borders.

A.1.5.3 Color

A default color for the light is defined here, following format [r (Red Value), g (Green Value), b (Blue Value)]. The color can be overridden or mapped in the grammar view layer. The color value ranges from 0.0 to 1.0. The tag is defined as "color".

A.1.5.4 Strength

Strength defines the intensity of the light. The tag is defined as "strength".

A.1.5.5 Size

The size of the light object is determined by this property. The outcomes will vary depending on the type of light used. For example for the light type POINT ,when larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows and specular highlights. For light type SPOTLIGHT, when larger than zero, light will be emitted from a spherical surfaces with the specified radius. Lights with larger size have softer shadows and specular highlights. For light type DIRECTIONAL the size is not used as it is a light of constant intensity emitted in a single direction from infinitely far away. For light type AREA will be mapped to a geometric shape and size will increase it proportionally. The tag is defined as "size".

A.1.6 Scales Class

The properties of the Scales class:

A.1.6.1 Name

This property defines the name of the lights class. No two lights can have the same name. The names of the lights are unique and will be used as a reference in the layers when

importing lights to the scene. The tag is defined as "name".

A.1.6.2 Type

[https://www.questionpro.com/blog/nominal-ordinal-interval-ratio/]

- 'NOMINAL' : Nominal Scale, also called the categorical variable scale, is defined as a scale used for labeling variables into distinct classifications and doesn't involve a quantitative value or order. This scale is the simplest of the four variable measurement scales. Calculations done on these variables will be futile as there is no numerical value of the options. There are cases where this scale is used for the purpose of classification – the numbers associated with variables of this scale are only tags for categorization or division. Calculations done on these numbers will be futile as they have no quantitative significance. Nominal scale is often used in research surveys and questionnaires where only variable labels hold significance. For instance, a customer survey asking "Which brand of smartphones do you prefer?" Options : "Apple"- 1 , "Samsung"-2, "OnePlus"-3.

- 'ORDINAL': Ordinal Scale is defined as a variable measurement scale used to simply depict the order of variables and not the difference between each of the variables. These scales are generally used to depict non-mathematical ideas such as frequency, satisfaction, happiness, a degree of pain etc. It is quite straightforward to remember the implementation of this scale as 'Ordinal' sounds similar to 'Order', which is exactly the purpose of this scale. Ordinal Scale maintains descriptional qualities along with an intrinsic order but is void of an origin of scale and thus, the distance between variables can't be calculated. Descriptional qualities indicate tagging properties similar to the nominal scale, in addition to which, ordinal scale also has a relative position of variables. Origin of this scale is absent due to which there is no fixed start or "true zero".For example, a semantic differential scale question such as: How satisfied are you with our services? Very Unsatisfied – 1, Unsatisfied – 2, Neutral – 3, Satisfied – 4, Very Satisfied – 5 Here, the order of variables is of prime importance and so is the labeling. Very unsatisfied will always be worse than

unsatisfied and satisfied will be worse than very satisfied. This is where ordinal scale is a step above nominal scale – the order is relevant to the results and so is their naming. Analyzing results based on the order along with the name becomes a convenient process for the researcher. If they intend to obtain more information than what they would collect using nominal scale, they can use ordinal scale.

- 'INTERVAL': Interval Scale is defined as a numerical scale where the order of the variables is known as well as the difference between these variables. Variables which have familiar, constant and computable differences are classified using the Interval scale. It is easy to remember the primary role of this scale too, 'Interval' indicates 'distance between two entities', which is what Interval scale helps in achieving. These scales are effective as they open doors for the statistical analysis of provided data. Mean, median or mode can be used to calculate the central tendency in this scale. The only drawback of this scale is that there no pre-decided starting point or a true zero value. Interval scale contains all the properties of ordinal scale, in addition to which, it offers a calculation of the difference between variables. The main characteristic of this scale is the equidistant difference between objects. For instance, consider a Celsius/Fahrenheit temperature scale – 80 degrees is always higher than 50 degrees and the difference between these two temperatures is the same as the difference between 70 degrees and 40 degrees. Also, the value of 0 is arbitrary because negative values of temperature do exist – which makes Celsius/Fahrenheit temperature scale a classic example of interval scale. Interval scale is often chosen in research cases where the difference between variables is a mandate – which can't be achieved using nominal or ordinal scale. Interval scale quantifies the difference between two variables whereas the other two scales are solely capable of associating qualitative values with variables. The mean and median values in an ordinal scale can be evaluated, unlike the previous two scales. In statistics, interval scale is frequently used as a numerical value can not only be assigned to variables but calculation on the basis of those values can also be carried out. Even if interval scales are amazing, they do not calculate the "true zero" value which is why the

next scale comes into the picture.

- 'RATIO': Ratio Scale is defined as a variable measurement scale that not only produces the order of variables but also makes the difference between variables known along with information on the value of true zero. It is calculated by assuming that the variables have an option for zero, the difference between the two variables is the same and there is a specific order between the options. With the option of true zero, varied inferential and descriptive analysis techniques can be applied to the variables. In addition to the fact that the ratio scale does everything that a nominal, ordinal and interval scale can do, it can also establish the value of absolute zero. Best examples of ratio scales are weight and height. In market research, a ratio scale is used to calculate market share, annual sales, the price of an upcoming product, number of consumers etc. Ratio scale provides the most detailed information as researchers and statisticians can calculate the central tendency using statistical techniques such as mean, median, mode and methods such as geometric mean, the coefficient of variation or harmonic mean can also be used on this scale. Ratio scale accommodates the characteristic of three other variable measurement scales, i.e. labeling the variables, the significance of the order of variables and a calculable difference between variables (which are usually equidistant). Because of the existence of true zero value, the ratio scale doesn't have negative values. To decide when to use a ratio scale, the researcher must observe whether the variables have all the characteristic of an interval scale along with the presence of the absolute zero value. Mean, mode and median can be calculated using the ratio scale. The tag is defined as "type".

A.1.6.3 Data

Data refers to the data of the scale such as a list of values in a scale. This references the data object. The tag is defined as "data".

A.1.6.4 Range

This property will be used for the types RATIO and INTERVAL. The property uses 3 sub-classes 'MAXIMUM' (The maximum value in a scale), 'MINIMUM' (The minimum value in a scale) and 'ITERATION' (The iteration value in a scale). The tag is defined as "range".

A.2 Renderer Class

The properties of the Renderer class:

A.2.1 Name

This property defines the name of the renderer class. No two renderers can have the same name. The names of the renderers are unique and will be used as a reference in the view settings to select which renderer to use for the scene. The tag is defined as "name".

A.2.2 Platform

The platform defines what software package is being used to render the scene, For example: BLENDER, UNITY, UNREAL. Only BLENDER 2.79 is currently supported. The tag is defined as "platform".

A.2.3 Type

The type determines if the visualization view is STATIC (single image or frame) or real-time (a streamed render of frames). The tag is defined as "type".

A.2.4 Engine

This property which render engine within the platform is to be used. For example BLENDER has primary render engines as CYCLES , EEVEE and BLENDER GAME. the tag is defined as "engine"

A.2.5 Settings

The setting tag allows for setting render specific settings.

A.2.5.1 samples

[https://docs.blender.org/manual/en/latest/render/cycles/render_settings/sampling.html] The integrator is the rendering algorithm used to compute the lighting. Cycles currently supports a path tracing integrator with direct light sampling. It works well for various lighting setups, but is not as suitable for caustics and some other complex lighting situations. Rays are traced from the camera into the scene, bouncing around until they find a light source such as a light, an object emitting light, or the world background. To find lights and surfaces emitting light, both indirect light sampling (letting the ray follow the surface BSDF) and direct light sampling (picking a light source and tracing a ray towards it) are used. The tag is defined as "samples".

A.2.5.2 denoising

[https://docs.blender.org/manual/en/latest/render/layers/denoising.html] Denoising filters the resulting image using information (known as feature passes) gathered during rendering to get rid of noise, while preserving visual detail as well as possible. The tag is defined as "denoising".

A.2.6 Output

The output setup for the render is defined in its sub-sections. The tag is defined as "output".

A.2.6.1 Type

If the output is an IMAGE or other data type. The tag is defined as "type".

A.2.6.2 Format

The output format if defined here, If the output type is an IMAGE the possible formats include JPEG and PNG. The tag is defined as "format".

A.2.6.3 Compressed

This is a boolean that sets the output compression. The tag is defined as "compressed".

A.3 Filter Class

The filter class is a list of filters that are applied if defined. There is no name property on this class as it is a standard set of queries to be executed. The properties of the Filter class:

A.3.1 Data

Data refers to the data source for the filter set. This references the data object and the primary source for the scripts to run its queries on. The tag is defined as "data".

A.3.2 Scripts

The scripts here are are all JsonPath queries to be executed using the data as the source. The script key appends to the available list of data sources as a new filtered data. For example: [!!!] [https://support.smartbear.com/alertsite/docs/monitors/api/endpoint/jsonpath.html]

Expression	Description
`$`	The root object or array.
`.property`	Selects the specified property in a parent object.
`['property']`	Selects the specified property in a parent object. Be sure to put single quotes around the property name. **Tip:** Use this notation if the property name contains special characters such as spaces, or begins with a character other than `A..Za..z_`.
`[n]`	Selects the *n*-th element from an array. Indexes are 0-based.
`[index1,index2,…]`	Selects array elements with the specified indexes. Returns a list.
`..property`	Recursive descent: Searches for the specified property name recursively and returns an array of all values with this property name. Always returns a list, even if just one property is found.
`*`	Wildcard selects all elements in an object or an array, regardless of their names or indexes. For example, `address.*` means all properties of the `address` object, and `book[*]` means all items of the `book` array.
`[start:end]` `[start:]`	Selects array elements from the *start* index and up to, but not including, *end* index. If *end* is omitted, selects all elements from *start* until the end of the array. Returns a list.
`[:n]`	Selects the first *n* elements of the array. Returns a list.
`[-n:]`	Selects the last *n* elements of the array. Returns a list.
`[?(expression)]`	Filter expression. Selects all elements in an object or array that match the specified filter. Returns a list.
`[(expression)]`	Script expressions can be used instead of explicit property names or indexes. An example is `[(@.length-1)]` which selects the last item in an array. Here, `length` refers to the length of the current array rather than a JSON field named `length`.
`@`	Used in filter expressions to refer to the current node being processed.

A.4 Transforms Class

The transforms class is used to transform data by applying functions on the data. It can also export the function for later processing example: dynamic data in streaming. The properties of the Transforms class:

A.4.1 Name

This property defines the name of the transforms class. No two transforms can have the same name. The names of the transforms are unique and will be used as a data sources in the geometric mapping or geometric transforms to place objects in the view. The tag is defined as ”name”.

A.4.2 Type

The two types allowed are STATIC and DYNAMIC where the STATIC has access to data and the function is applied to the data and the data source is available soon after, in DYNAMIC types the function is exported as a function without access to data source and used in the view on a required and interactive basis.

A.4.3 Functions

The functions define available in-built functions used by Aloka. Every function has a unique parameter limit. eg: reverse_array takes a single array variable as input "reversed_values": "reverse_array input_array_variable", others take more than one variable as input "added_values": "add value_1, value_2". The list of supported functions by aloka are given below with a definition of what it does. [!!!]

- 'rgb_to_value'
- 'last_of_array'
- 'reverse_array',
- 'divide_array',
- 'subtract_array',
- 'set_value_array',
- 'WGS84toOSGB36_easting_array',
- 'WGS84toOSGB36_northing_array',
- 'set_value',
- 'average',
- 'multiply',
- 'add',

- 'subtract',
- 'divide',
- 'exp',
- 'sqrt',
- 'sin',
- 'cos',
- 'log',
- 'dot',
- 'sum',
- 'min',
- 'max',
- 'mean',
- 'cumsum',
- 'median',
- 'sort',
- 'transpose',
- 'ravel',
- 'reshape',
- 'insert',
- 'append',
- 'delete',

- 'concatenate',
- 'vstack',
- 'hstack',
- 'vsplit',
- 'hsplit',
- 'column_stack',
- 'WGS84toOSGB36'

A.5 Geometric transforms Class

The properties of the Geometric Transforms class:

A.5.1 Name

This property defines the name of the geometric transform class. No two geometric transforms can have the same name. The names are unique and will be used as a reference in the view settings to position geometries, lights and cameras in the scene. The tag is defined as "name".

A.5.2 Type

The two types of supported geometric transforms are STATIC and DYNAMIC. In STATIC geometric transforms the position or rotation and scale is a fixed number and won't change. Whereas a DYNAMIC type will have a set of data values or streamed values which are loaded as a list. They can also be used to dynamically load glyphs for a list of data points. The tag is defined as "type".

A.5.3 Position

This property denotes the position of the object in the scene in "x","y","z" coordinates. Currently Aloka uses Blender's coordinate system as default. The tag is defined as "position".

A.5.4 Rotation

This property denotes the rotation of the object in the scene in "x","y","z" coordinates. Currently Aloka uses Blender's coordinate system as default. The tag is defined as "rotation".

A.5.5 Scaled

This property denotes the scale of the object in the scene in "x","y","z" scales. Currently Aloka uses Blender's scaling system as default. The tag is defined as "scaled".

A.6 View Class

A view class defines a scene and a visualization, it describes how the scene should be constructed and what visualization techniques to use to generate a render. The properties of the View class:

A.6.1 Name

This property defines the name of the view class. No two views can have the same name. The names of the views are unique and will be used as a reference in the Start Point property as an entry point to the initial view. The tag is defined as "name".

A.6.2 Layers

Each view can hold many layers. Each of the layer class is rendered separately as an EXR file with depth information and stitched together to create a final render. If the process

is streamed/interactive only the layer which needs updating needs to be rendered, saving precious compute costs. Organizing the layers into the static layers and layers which are susceptible to change is by design. The Layer class is defined below:

A.6.2.1 Name

This property defines the name of the layer class. No two layers can have the same name. The names of the layers are unique and will be used by the renderer to identify which layer needs updating and which does not. The tag is defined as "name".

A.6.2.2 Geometries

Each layer holds a list of geometries. Each geometry has some property which define its purpose(if its a static object or a mapped object(this means the data is bound to the geometry and Aloka has to generate the geometries to reflect the data)) and positioning and materials. To explain each property:

- "name": this property defines the name of the geometry.
- "type":[!!!] The types of geometries supported by Aloka are DATA(creates a dynamic list of geometries that map to a "source" data that is an array of data), MODEL(a static 3D model file), DATA_POINT("source" contains a singular data object to map to the geometry), TEXT(contains a text geometry), PRIMITIVE(this defines a primitive geometry such as cube, plane, sphere, cylinder, cone etc) and SCALE (This is a very unique and custom type in which it creates a custom scale for the scene visualization).
- "source": This points to a data source or defined geometry type in Aloka depending on the type of geometry.
- "geometric_transform": The positioning of the geometry is defined here.
- "material": [!!!]: [Example of the material class overriding and mapping]Material of the geometry can be mapped to data by overriding the material hierarchy and defining which property should be mapped to which data.

- "events": Each geometry should have a events list that defines which device and which input of the device affects the change of the entire view.

A.6.2.3 Lights

Each layer also holds a list of lights. Each light has two properties "source": which points to the light object defined in the resources class and "geometric_transform" which points to the geometric transforms list to pick how the light should be positioned, rotated and scaled in the scene.

A.6.3 Settings

The view settings hold general setup information of the scene.

A.6.3.1 Render

The render tag defines which renderer to use for this particular view. Different renderers can be used for different views.

A.6.3.2 Display

This defines the display the view is targeting.

A.6.3.3 Optimization

The optimization property defines what kind of optimization Aloka should carry out to personalize the visualization to a certain user or display.

A.6.3.4 Camera

Each view can only have one camera and the camera is defined here. the camera property has two sub properties "source": This refers to the camera defines in the resource for easy access, "geometric_transform" : the property defines camera positioning for the view.

A.7 Start Point

This property in the grammar denotes a start point and is a reference to a view name. Essentially it points to the view which should be rendered first.